

# ORACLE9I: KISS YOUR INIT.ORA GOODBYE!

*Dan Norris, norris@celeritas.com, Celeritas Technologies, LLC*

## INTRODUCTION AND OVERVIEW

Oracle9i has certainly changed the RDBMS world with all of its new features and enhancements. In past major releases involving new features, you had to configure the database to use the new features. If you didn't, it still functioned the same way it always had. In the Oracle9i release, one new feature is likely to cause problems for DBAs since it is automatically “enabled” and performing a default installation will employ this new feature. That new feature is the server parameter file.

The server parameter file is an important new feature and one that will likely cause many all DBAs trouble since the init.ora file is still present, but the server parameter file is used by default. I know that my out-of-the-box experience was confusing since I kept changing the good ol' init.ora file and my changes were ignored by the server.

In this paper, you'll learn about what a server parameter file is, what you can do with it, what you cannot do with it, why Oracle decided to create it, and other useful tidbits about managing these files.

## SERVER PARAMETER FILE

### **WHAT IS IT?**

The server parameter file (usually referred to as the "spfile" in many documents) is a binary file used by the Oracle9i server to maintain its instance parameters. With prior releases, many of the instance parameters were dynamically modifiable, but unless you also edited the init.ora file, your new value would be reset to the init.ora value (or the default if not present in the init.ora) upon an instance restart. This means that every time you use the convenient `ALTER SYSTEM` syntax to change an instance parameter on the fly, you are forced to modify the init.ora file manually, thus creating a maintenance nightmare.

If the same scenario occurred with a 9i database that was using a server parameter file, your one `ALTER SYSTEM` command would modify the parameter on the fly as well as make it persistent across instance restarts. That is because when you run the `ALTER SYSTEM` command, the default action is to also update the spfile for the instance.

Another benefit of the server parameter file is that it can be used when starting a database remotely via Oracle Enterprise Manager (OEM). The old days of having to synchronize a local copy of the init.ora file on your OEM system with the version on the database server system are over! Now you can just keep the one copy of the server parameter file on the server and it will be used when starting up the database remotely.

### **WHERE IS IT?**

Just like the init.ora file, the spfile has some default locations. You can choose to put the spfile anywhere you like as long as the instance has access to it.

### *WHERE IS ORACLE9I LOOKING BY DEFAULT?*

By default, the 9i instance will follow this sequence to determine its initialization parameters on most operating systems:

1. \$ORACLE\_HOME/dbs/spfile\$ORACLE\_SID.ora
2. \$ORACLE\_HOME/dbs/spfile.ora
3. \$ORACLE\_HOME/dbs/init\$ORACLE\_SID.ora

Once it finds one of these files present, the search is stopped—there is no "combining" of multiple files automatically.

If one of those 3 files is not found, the startup will fail with an error (in my tests, it returned an ORA-3113, but that's probably not the correct error message).

### *HOW TO OVERRIDE THE DEFAULT LOCATION*

With the init.ora file, when you wanted to override the default location, you could specify the location of the init.ora file in the startup command like:

```
SQL> startup pfile='/path/to/my/initSID.ora'
```

but there is no "spfile=" option to the startup command. However, there is a new init.ora parameter in 9i that allows you to specify the location of your spfile. To override the default location for the spfile, you create a one-line init.ora file that has the parameter "spfile=/path/to/spfile.ora" in it. Then, since your spfile.ora is not in the default location, you will need to place the init.ora file in the default location or use the "startup pfile=..." syntax to start the instance.

Seems like a long way to have to go to reach the ultimate goal, but as of right now, that's the only option you have. Wouldn't it have been nice to have a similar parameter to specify the location of the password file?

Depending on the platform you're running the server on, you could use symbolic links to "trick" the operating system (and therefore Oracle) into thinking that the file is in the default location when it is actually elsewhere. That solution is probably easier and works well on standalone nodes, but in a cluster environment, it may not be such an attractive option.

## WHAT IS IN IT?

The server parameter file contains any non-default parameters you've set explicitly and any parameters that the 9i server has self-tuned. If you'd like to view the contents of the server parameter file, you have a couple of options. One option is to use SQL and query the V\$SPPARAMETER view. The other option is to run the `CREATE PFILE='/path/to/create/init.ora' FROM SPFILE;` command in a SQL prompt to generate a text-based init.ora file from the spfile and then view it.

## HOW TO MODIFY IT

Although the spfile appears quite readable, it is not a plain text file and should not be modified directly. To change values stored in the server parameter file, you must use SQL. Of course, reading the Oracle documentation is always (well, almost always) the best source for finding the correct syntax, but here's the abridged form:

```
ALTER SYSTEM SET parameter_name = value [ SCOPE = SPFILE | MEMORY | BOTH ];
```

The `ALTER SYSTEM` statement is examined later in this paper.

## HOW TO CREATE IT

If you use the database creation assistant (dbca on UNIX platforms), you have the option of creating the spfile (it is enabled by default). If you create a database from scratch using your own scripts, here's all you need to know to create the spfile:

```
CREATE SPFILE FROM PFILE;
```

There is nothing you need to do to "enable" the spfile, if Oracle finds one, it will be used. If you want to use a particular init.ora file as the "seed" for your spfile, you can specify the init.ora file you wish to use by using this syntax:

```
CREATE SPFILE FROM PFILE='/path/to/particular/init.ora';
```

and finally, if you wish to have your spfile created in a particular location (not the default location), then you can add that path to the `CREATE` statement as follows:

```
CREATE SPFILE='/path/to/spfile.ora' FROM PFILE='/path/to/particular/init.ora';
```

## HOW TO BACK IT UP AND RESTORE IT

Like most things having to do with Oracle backup and recovery there is a procedure for backing up the spfile. You cannot simply back up the spfile itself since it is always open for writing from the instance should someone issue an `ALTER SYSTEM` statement or if the instance has self-tuned a parameter that needs to be persistent.

So, in order to back up this file, you generate the init.ora file using a command similar to the one used to create the spfile. The syntax is:

```
CREATE PFILE['/path/to/init.ora'] FROM SPFILE['/path/to/spfile.ora'];
```

If you leave out the paths, it will read from the default spfile location and write to the default init.ora location and filename.

Restoring an spfile is not rocket science. Restoration is no more difficult than the backup method. The procedure is to restore the text init.ora file you created as the backup and then follow the steps to create a new spfile from the init.ora file as shown above in "How to create it".

## **HOW THE SERVER PARAMETER FILE IS USED**

### **IF IT IS SHARED**

If your server parameter file is shared (it is named `spfile.ora`), then it may contain parameters for multiple instances. This can be very useful, but also somewhat confusing. The next section discusses the shared server parameter file in detail.

### **IF IT IS NOT SHARED**

If the parameter file is not shared (it is named `spfile$ORACLE_SID.ora`), then it is only being used by one instance. Any parameters for other instances found in that `spfile` are ignored. Usually, the instance identifier is `*` (asterisk) which means that particular parameter will be used by any instance that reads that `spfile`. The ability to specify global parameters in this way is especially useful parameters when using a shared server parameter file.

### **WHAT IF I DON'T WANT ONE?**

Oracle is not currently forcing you to have an `spfile` if you absolutely do not want one. However, popular opinion seems to be that it is a useful new feature and, furthermore, it enables even more useful new features such as self-tuning persistence.

If you still wish to go without a server parameter file, remove the `spfile.ora` and `spfile$ORACLE_SID.ora` files from your `$ORACLE_HOME/dbs` directory and the server will resort to using the `init$ORACLE_SID.ora` just as it has in previous releases.

## **SHARED SERVER PARAMETER FILE(S)**

### **INSTANCE-SPECIFIC PARAMETERS**

Instance-specific parameters allow a single `spfile` to serve multiple instances (and/or databases) by creating a prefix (the instance name) for every parameter. The special prefix `*` is reserved to mean all instances. To create a shared server parameter file from “scratch,” you would create an `init.ora` file that looks like this:

```
*.db_block_size=8192
*.sort_area_size=1048576
erpprd.db_cache_size=163840000
crmprd.db_cache_size=81920000
*.shared_pool_size=52428800
erpprd.db_name=erpprd
crmprd.db_name=crmprd
erpprd.control_files=("/u01/oradata/erpprd/control01.ctl")
crmprd.control_files=("/u02/oradata/crmprd/control01.ctl")
erpprd.log_archive_start=true
erpprd.log_archive_dest_1= ("LOCATION=/u04/oradata/archive/erpprd/arch REOPEN=300")
erpprd.log_archive_format="_%s.arc"
```

and then issue the `CREATE SPFILE='/path/to/O_H/dbs/spfile.ora' FROM PFILE='/path/to/above/init.ora/file'`; command to create the shared `spfile`. Because of the name it is given by the command, it will be used unless there is a file named `spfile$ORACLE_SID.ora` present in the `$ORACLE_HOME/dbs` directory for a given instance (see previous section titled “Where is it?”). Note that in the above shared server parameter file, both instances have the same `shared_pool_size`, `db_block_size`, and `sort_area_size`, but different `db_cache_size` parameters.

### **ADVANTAGES**

One advantage of using the shared server parameter file is that you can keep a set of global parameters that you want to apply to all instances in one place where all instances will access them. For instance, many of us set parameters like `nls_date_format` or `db_domain` that are generally the same for most if not all instances in your environment. While it isn't a huge chore to remember to set these parameters up correctly, consider the case where the domain name changes (your company was acquired, for instance) and you have to go about changing all those `init.ora` files. If you were using a shared server parameter file, you only need to update one source per system and then restart the instance(s).

Another potential advantage is the ability to change parameters for a given instance from another instance on the same machine. Although not the most popular way to use this feature, it is possible. More attractive is the ability to examine the instance parameters for every instance on a given system from one instance on that system (assuming that they are all sharing

the same spfile). Instance-specific parameters are shown in the `V$SPPARAMETER` view with the `SID` column set to the name of the instance that particular parameter applies to. In the example above, both the `erpprd` and `crmpnd` instances will have access to view and modify the instance-specific parameters for any instance sharing the same spfile they are using.

## DISADVANTAGES

As with all consolidated resources, the major disadvantage is that one little mistake affects many different things. These types of disadvantages plague every action a DBA takes since most instances are used by multiple people, applications, departments, or even companies. So, I wouldn't see this as a show-stopper—it just points out the need for careful planning and clear thinking when making changes that have a large impact.

Another potential disadvantage is the introduction of a single point of failure. Of course, as with any production database, files critical to the operation of that resource should reside on some sort of protected storage. For the spfile, that may be a RAID 5 volume since I/O to the spfile will be minimal compared to the I/O for database files.

## REAL APPLICATION CLUSTERS

Oracle Real Application Clusters (RAC) is the current generation of a product that was called Oracle Parallel Server (OPS) in the past. The RAC product was renamed in part to show the major feature changes and enhancements that are delivered with the 9i RAC solution.

For the purposes of this paper, I'll concentrate only on instance parameters, which is one of the areas that has made managing the Oracle Parallel Server product challenging in the past. Usually, in OPS configurations, there is a file containing global instance parameters and another containing instance-specific parameters. The instance-specific file was used to start each instance in the cluster and it, in turn, included the global parameter file using an `ifile` parameter. The global parameter file was sometimes located on a shared mount point (usually NFS until the recent evolution of cluster file systems) for access from multiple systems simultaneously.

With RAC, there are new options that will make the task of synchronizing instance parameters easy. Using a shared server parameter file, the global parameters for all instances as well as those specific to each instance can be put into the same repository. That repository, being a binary format, can be put onto a shared disk device such that you never need to copy the file from node to node to synchronize it. Each node in the cluster can read and write to the file when updates to parameters need to be made. Those updates can be from a DBA-initiated request (`ALTER SYSTEM` statement) or possibly from the server doing some self-tuning.

### *SHARED PARAMETER FILE ON SHARED STORAGE*

As mentioned previously, the shared parameter file would provide the most benefit if it was on some shared storage. Depending on the Oracle support for your platform, you may be able to use a cluster file system rather than a raw device for your shared spfile. This allows even easier management since sizing raw devices is difficult at times and resizing them is even more of a challenge for both the DBA and the system administrator. For instance, on Compaq Tru64 clusters, you can put the entire RAC database including the spfile on the cluster file system. From each system, you can see all database files on the shared file system and share the spfile as well. If you do not have an Oracle-supported cluster file system available in your cluster environment, the spfile can reside on a raw device and be accessible to each cluster instance simultaneously.

So, the shared server parameter file is a great benefit to those running parallel databases since it allows centralization of all parameters, both global and instance-specific, into one file.

## ALTER SYSTEM SYNTAX ADDITIONS

There are two clauses I feel are important to understand when dealing with server parameter files: the `SET` and `RESET` clauses of the `ALTER SYSTEM` statement. Here are the basic syntaxes for these two clauses:

```
ALTER SYSTEM SET parameter_name = value [COMMENT 'comment'] [SID = 'sid'] [SCOPE = SPFILE
| MEMORY | BOTH];
ALTER SYSTEM RESET parameter_name [SCOPE = SPFILE | MEMORY | BOTH] [SID = 'sid']
```

**COMMENT=COMMENT**

The `COMMENT` clause allows the DBA to add a note about the parameter change. If `SPFILE` or `BOTH` is specified, this note is written to the spfile as well for a permanent record. If `MEMORY` is specified, the note shows up as part of the parameter value in the `V$PARAMETER` view.

**SID='SID'**

The `SID` clause allows you to set instance-specific parameters. This is only necessary if you are using a shared server parameter file and wish to set a parameter for a specific instance. While the Oracle9i documentation states that this clause is only relevant in the Real Application Clusters environment, I believe it is relevant any time you are using a shared server parameter file. As mentioned earlier, it is possible to share a server parameter file between instances on the same system to realize certain benefits.

**SCOPE=MEMORY**

The `MEMORY` setting only changes the value being used by the running instance (but that value is lost when the instance restarts). If you started up the instance with a normal `init.ora` file, this is the only setting you can use. In order to use the spfile, you will have to restart the instance.

**SCOPE=SPFILE**

If it is set to `SPFILE` or `BOTH` then the spfile is updated. Using the `SPFILE` setting only changes the spfile (if you wanted the parameter to take effect on the next restart). If you are modifying a parameter that is static, you must specify `SCOPE=SPFILE` or you will receive an error.

**SCOPE=BOTH**

The `BOTH` setting updates the running instance as well as the spfile for future restarts. The default behavior is `BOTH`.

**RESET**

The purpose of this clause is to give you a way to remove an instance-specific version of a parameter. For example, say there was a setting for `sort_area_size` that had been set for all instances (`SID='*'`) in the spfile, but one instance had its own setting for `sort_area_size`, you would use the `RESET` clause specifying the `SID='specific_instance'` to remove the instance-specific `sort_area_size` parameter for that instance. At that point, the value of `sort_area_size` for that instance would assume the value specified in a previous or subsequent `ALTER SYSTEM SET ... SID='*'` statement.

**OVERVIEW OF SOME DYNAMIC INSTANCE PARAMETERS IN ORACLE9I**

To emphasize the effectiveness of this new feature, this section highlights a few of the parameters that are dynamically modifiable and would likely be important ones to persist across instance restarts.

**SHARED\_POOL\_SIZE**

That's right, in Oracle9i the shared pool size is modifiable on the fly. You can actually make the server allocate and deallocate memory while people are using the server. Of course, there are a number of things to consider when doing so, but that topic is beyond the scope of this paper.

**DB\_CACHE\_SIZE**

In Oracle9i, the parameter used to size the database block buffer cache in previous releases (`db_block_buffers`) has been replaced with the new and improved `db_cache_size`. The units for this parameter have also changed so that the new parameter is specified in bytes, not in database blocks. That change is partly due to the fact that we can have multiple block sizes in our 9i database. Obviously, if this parameter were changed, it is highly likely that we would want to have it persist across instance restarts.

**DYNAMIC PARAMETERS ABOUND**

Overall, the number of dynamically-modifiable parameters in 8.1.7 is nearly doubled in 9.0.1 to 110. With that many parameters being modified on the fly, you can start to see the huge advantage of the server taking care of updating a parameter file for you. The 9.0.1 server will also perform self-tuning as it runs regardless of whether or not you use a server parameter

file. If you do not use a server parameter file, all the self-tuning advances made while the instance is up are lost when you restart the instance.

## **CONCLUSION**

The server parameter file is a very important new feature that will assist with the maintainability of the Oracle system configuration. Like any new feature, I'm sure that it will require some time and practice before DBAs get used to using server parameter files, but the effort spent trying to understand them will be well-spent.

It is hard to tell at this early stage, but I anticipate that Oracle will eventually deprecate the `init.ora` file we know and love for the sporty new server parameter file in a future release of the database. In the meantime, DBAs should spend some time getting to know the server parameter file and what great things it can offer.