# Appendix A: Oracle Setup for Model Repository in Opsware SAS 5.5

**IN THIS APPENDIX**

This section discusses the following topics:

- Supported Oracle Versions
- Setting Up the Database
- Database Monitoring for the Model Repository

Document Updated: Sept. 1, 2006

The Opsware Model Repository stores information in an Oracle database. Before running the Opsware Installer, you must create an Oracle database on the server where you will install the Model Repository.

## Supported Oracle Versions

Support for the Model Repository is limited to specific versions of Oracle running on specific versions of operating systems. Table A-1 lists the supported Oracle versions.

*Table A-1: Supported Oracle Versions for Model Repository*

| ORACLE EDITION | VERSIONS |
|---|---|
| Oracle Standard Edition | 9.2.0.4<br>9.2.0.6<br>9.2.0.7 |
| Oracle Enterprise Edition | 9.2.0.4<br>9.2.0.6<br>9.2.0.7 |

Oracle version 9.2.0.5.0 is not supported with this release of Opsware SAS.

To be supported on the Model Repository, the Oracle versions listed in Table A-1 are limited to the operating systems listed in Table A-2.

*Table A-2: Supported Operating Systems for Model Repository*

| SUPPORTED OPERATING SYSTEMS FOR MODEL REPOSITORY | VERSIONS | ARCHITECTURE |
|---|---|---|
| Sun Solaris | Solaris 8<br>Solaris 9 | Sun SPARC<br>Sun SPARC |
| Red Hat Linux | Red Hat Enterprise Linux 3 AS | 32 bit x86 |

## Setting Up the Database

To set up the database, perform the following steps:

1. On the server where you will also install the Model Repository, install one of the supported versions of Oracle. Before installing Oracle, make sure that the operating system is supported by Opsware SAS.

   If you want the Model Repository to access a remote Oracle database, install Oracle Client on the server where you will install the Model Repository. The `truth.orahome` parameter (ORACLE_HOME) you specify during the installation of the Model Repository must point to the location of the Oracle Client.

2. Create an Oracle database with the UTF8 character set.

   The following clauses in the `CREATE DATABASE` statement include recommended sizes. (Your organization's guidelines might specify different sizes.)

   ```
   DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE SIZE 1000M
   AUTOEXTEND ON NEXT 64M MAXSIZE UNLIMITED

   UNDO TABLESPACE "UNDO" DATAFILE SIZE 1000M AUTOEXTEND ON NEXT
   64M MAXSIZE UNLIMITED
   ```

 

Storage requirements for the database grow as the number of managed servers grows. As a benchmark figure, you should allow an additional 3.1 GB database storage for every 1000 servers in the facility that Opsware SAS manages.

3. Specify initialization parameters in the `init.ora` file. Opsware SAS requires the following parameter settings. (All other settings can follow your organization's guidelines, or you can use the default settings.)

```
optimizer_mode = choose
query_rewrite_enabled = true
query_rewrite_integrity = trusted
open_cursors >= 2000
shared_pool >= 200000000
sort_area_size >= 1048576
nls_sort = generic_m
job_queue_processes >= 2
processes > 1000
db_block_size >= 8192
java_pool_size >= 50000000
workarea_size_policy = auto
sessions > 1000
cursor_sharing = similar
pga_aggregate_target >= 268435456
nls_length_semantics = char
```

Set the `nls_length_semantics` parameter to `char` for a standalone core installation. When you are adding a core to a multimaster mesh, set `nls_length_semantics` to the same value that the other Opsware cores are using. If you use different settings in the cores, Opsware SAS will not function correctly. Contact Opsware Professional Services for assistance upgrading the setting for an `nls_length_semantics` parameter in a core.

4. Set up the `tnsnames.ora` file.

   The `tnsnames.ora` file enables resolution of database names used internally by Opsware SAS. A `tnsnames.ora` file is required on the core servers running the following components: Model Repository, Data Access Engine, Web Services Data Access Engine, and Model Repository Multimaster Component.

   In a standalone core, the `tnsnames.ora` file must contain an entry for the Model Repository. For example:

```
truth =
(DESCRIPTION=
(ADDRESS=(HOST=magenta.opsware.com)(PORT=1521)
```

```
(PROTOCOL=tcp))
(CONNECT_DATA=(SERVICE_NAME=truth)))
```

In a multimaster mesh, the `tnsnames.ora` file of the central (master) core must contain an entry for its own Model Repository. The file must also have entries for the Model Repositories of the other cores in the mesh. For the entries of the other (non-central) cores, the host specifies the central core Gateway, and the port number is derived from this formula: (20000) + (facility ID of the non-central core).

The following `tnsnames.ora` example is for the central core of a multimaster mesh. In this example, the TNS service name of the central core is `orange_truth`, which runs on the host `orange.opsware.com`. The TNS name of the non-central core is `cyan_truth`, which has a facility ID of 556. Note that the entry for `cyan_truth` specifies `orange.opsware.com`, the host running the central core's Gateway.

```
orange_truth =
(DESCRIPTION=
(ADDRESS=(HOST=orange.opsware.com)(PORT=1521)(
PROTOCOL=tcp))
(CONNECT_DATA=(SERVICE_NAME=truth)))
cyan_truth =
(DESCRIPTION=(ADDRESS=(HOST=orange.opsware.com)(PORT=20556)
(PROTOCOL=tcp))
(CONNECT_DATA=(SERVICE_NAME=truth)))
```

In a multimaster mesh, the `tnsnames.ora` file of a non-central (non-master) core must contain an entry for its own Model Repository, but does not require entries for other cores in the mesh. In the following `tnsnames.ora` example, the TNS service name of the non-central core is cyan_truth, and the core runs on the host, cyan.opsware.com.

```
cyan_truth =
(DESCRIPTION=
(ADDRESS=(HOST=cyan.opsware.com)(PORT=1521)
(PROTOCOL=tcp))
(CONNECT_DATA=(SERVICE_NAME=truth)))
```

If you install the Opsware core on multiple servers, the `tnsnames.ora` file with the same directory path must exist on the servers where the following Opsware components are installed: Model Repository, Data Access Engine, Opsware Command Center, Opsware Global File System, Model Repository Multimaster Component.

**5** Start the Oracle listener.

 

6 Initialize the Oracle JVM. The Oracle Installer provides an option for this, but you can also use the following script in the Oracle product directory:

```
$ORACLE_HOME/javavm/install/initjvm.sql
```

7 Create the following tablespaces:

```
AAA_DATA
AAA_INDX
LCREP_DATA
LCREP_INDX
TRUTH_DATA
TRUTH_INDX
```

When you create the `DATA` tablespaces, you should use the sizes shown in the following example:

```
CREATE TABLESPACE "LCREP_DATA" LOGGING DATAFILE SIZE 1000M
AUTOEXTEND ON NEXT 64M MAXSIZE UNLIMITED EXTENT MANAGEMENT
LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

When sizing the tablespaces, follow the general guidelines shown in the following table. If you need to determine a more precise tablespace sizing, contact your Opsware, Inc. Support Representative.

*Table A-3: Tablespace Sizes*

| TABLESPACE | MB/1000 SERVERS | MINIMUM SIZE |
|---|---|---|
| AAA_DATA | 256 MB | 256 MB |
| AAA_INDX | 256 MB | 256 MB |
| LCREP_DATA | 3,000 MB | 1,500 MB |
| LCREP_INDX | 1,600 MB | 800 MB |
| TRUTH_DATA | 1,300 MB | 700 MB |
| TRUTH_INDX | 300 MB | 400 MB |

8 Create the `opsware_admin` database user.

Opsware SAS uses the `opsware_admin` user to install and manage the Model Repository. Use the `TRUTH_DATA` tablespace with unlimited quota as the default tablespace for the `opsware_admin` user. Set the temporary tablespace according to your organization's guidelines.

9 Grant privileges to the `opsware_admin` user as shown in the following SQL statements:

```
grant alter session to opsware_admin with admin option;
grant create procedure to opsware_admin with admin option;
grant create public synonym to opsware_admin with admin
option;
grant create sequence to opsware_admin with admin option;
grant create session to opsware_admin with admin option;
grant create table to opsware_admin with admin option;
grant create trigger to opsware_admin with admin option;
grant create type to opsware_admin with admin option;
grant create view to opsware_admin with admin option;
grant delete any table to opsware_admin with admin option;
grant drop public synonym to opsware_admin with admin option;
grant select any table to opsware_admin with admin option;
grant select_catalog_role to opsware_admin with admin
option;
grant query rewrite to opsware_admin with admin option;
grant restricted session to opsware_admin with admin option;
grant execute on dbms_utility to opsware_admin with grant
option;
grant analyze any to opsware_admin;
grant select, insert, update, delete on sys.aux_stats$ to
opsware_admin;
grant alter system to opsware_admin;
grant create role to opsware_admin;
grant create user to opsware_admin;
grant alter user to opsware_admin;
grant drop user to opsware_admin;
grant create profile to opsware_admin;
grant alter profile to opsware_admin;
grant drop profile to opsware_admin;
```

**10** Set the `NLS_LANG` environment variable for the `oracle` Unix user.

This environment variable is required for the export and import operations when installing a multimaster core. The syntax of `NLS_LANG` follows:

```
NLS_LANG=<languague>_<territory>.<client_characterset>
```

For example, in the United States you might set `NLS_LANG` to the following value:

```
NLS_LANG=AMERICAN_AMERICA.UTF8
```

The value of `NLS_LANG` must match the character set used by the database, which can be determined by the following query:

```
sql> select value from nls_database_parameters where
parameter='NLS_CHARACTERSET';
```

If the export (source) and import (target) databases have different character sets, then for both set `NLS_LANG` to the character set of the export database.

 

11 Set up database monitoring. (See the following section.)

## Database Monitoring for the Model Repository

For the Oracle instance that the Opsware Model Repository uses, you should set up monitoring for the following key diagnostics:

- The availability of the Oracle instance, database, and listener process.
- The availability of space for the Model Repository (truth) schema growth

Additionally, Opsware Inc. recommends that you monitor key Oracle log files, including the `alert.log` and background and user trace files.

### Instance and Database Availability

In this topic, the examples for basic monitoring assume that the Oracle instance name is `truth`.

Opsware SAS becomes unavailable when Oracle becomes unavailable. Therefore, to ensure that Opsware SAS has access to the Oracle database, you must ensure that the Oracle instance is running, the Oracle database is open, and the listener is monitoring for connections.

#### *Checking the Instance*

To check the Oracle instance, perform the following steps:

1 To check for the status, login as the `oracle` Unix user and use the `ps` command to look for the processes with names starting with `ora_`. For example:

```
oracle$ ps -ef | grep ora_
oracle 14239     1  0   Mar 19 ?        0:08 ora_lgwr_truth
oracle 14245     1  0   Mar 19 ?        0:00 ora_reco_truth
oracle 14241     1  0   Mar 19 ?        0:16 ora_ckpt_truth
oracle 14237     1  0   Mar 19 ?        0:04 ora_dbw0_truth
oracle 14243     1  0   Mar 19 ?        0:16 ora_smon_truth
oracle 14235     1  0   Mar 19 ?        0:00 ora_pmon_truth
oracle 14247     1  0   Mar 19 ?        0:00 ora_cjq0_truth
```

2 Confirm that the instance is running by connecting to the database as `sysdba`. (Be sure to set your `ORACLE_HOME` and `ORACLE_SID` environment variables appropriately.)

```
oracle$ sqlplus "/ as sysdba"
. . .
Connected to:
```

```
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
JServer Release 9.2.0.4.0 - Production
```

The "Connected to:" message confirms that the instance is available.

### Checking the Database

Opsware SAS needs the database to be mounted and open for general use in order to function. To check the database, perform the following steps:

1. To check the status of the database, connect to the instance as sysdba and issue the following query:

   ```
   sql> select database_status from v$instance;
   ```

   The result should be ACTIVE.

2. To check the mode in which the database was opened, issue the following query:

   ```
   sql> select open_mode from v$database;
   ```

   The result should be READ WRITE.

### Checking the Listener

To check the Oracle listener (tnslsnr), perform the following steps:

1. Check the status of the listener with the lsnrctl command:

   ```
   oracle$ lsnrctl status
   . . .
   Service "truth" has 1 instance(s).
     Instance "truth", status READY, has 1 handler(s) for this
   service...
   ```

   The status should be READY.

2. Test connectivity to the instance from the Data Access Engine (spin) and Web Services Data Access Engine (twist) hosts by running the tnsping utility (or by connecting with SQL*Plus with a net-service name identifier):

   ```
   oracle$ tnsping truth
   . . .
   Attempting to contact
   (DESCRIPTION=(ADDRESS=(HOST=localhost)(PORT=1521)(PROTOCOL=t
   cp))(CONNECT_DATA=(SERVICE_NAME=truth)))
   OK (0 msec)
   ```

   The OK statement confirms that the listener is up and can connect to the instance.

 

### *Checking for Datafile Space Availability*

Opsware SAS stores its data in a series of size tablespaces, each consisting of one or more datafiles. For the size of the data set to grow, you must ensure that each tablespace has enough space for the allocation of new rows.

You can verify the auto-extensibility of tablespaces with the following query:

```
sql> select d.file_name, d.tablespace_name, d.status,
d.autoextensible,
d.bytes / 1024 / 1024,
nvl(d.bytes - sum(s.bytes), d.bytes) / 1024 / 1024,
nvl(d.bytes - sum(s.bytes), d.bytes) / d.bytes * 100
from sys.dba_data_files d, sys.dba_free_space s
where (s.file_id (+)= d.file_id)
and d.bytes is not null
group by d.tablespace_name, d.file_name, d.status,
d.autoextensible, d.bytes;
```

You can also monitor tablespace usage by running a test for the System Diagnosis feature of the Opsware Command Center. The test is named Oracle Tablespaces and is listed under the Data Access Engine component. This test checks to see if manually-extended tablespaces are less than 85% full.

### Monitoring Oracle Log Files

Monitor the following Oracle log files:

- The Oracle `alert.log` file. (Check this file for `ORA-` errors because some of the errors will not be displayed.)

  `$ORACLE_BASE/admin/truth/bdump/alert_truth.log`

  `$ORACLE_BASE/admin/truth/[bcu]dump/*.trc`

Configure a `cron` job to perform the following actions:

- Periodically poll for changes to files, for the creation of files, or for the presence of `ORA-` errors.
- Report these errors by e-mail or another way to a DBA.

 