hp OpenView
Service Quality Manager

Reference Guide for Oracle Use

Edition: 1.2

for the HP-UX Operating System

January 2005

Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company

United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Netscape is a U.S. trademark of Netscape Communications Corporation.

NMOS™ is a trademark of RiverSoft Technologies Limited.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

Oracle7™ and Oracle7 Server™ are trademarks of Oracle Corporation, Redwood City, California.

PostScript® is a trademark of Adobe Systems Incorporated.

Riversoft™ is a trademark of RiverSoft Technologies Limited.

UNIX® is a registered trademark of The Open Group.

Windows® and Windows NT® are U.S. registered trademarks of Microsoft Corporation.

X/Open® is a registered trademark, and the X device is a trademark of X/Open Company Ltd. in the UK and other countries.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

Contents

Preface

This document describes the global Oracle approach implemented across OpenView SQM applications. It helps to understand the SQM/Oracle environment and describes the adopted standards. It also shows how to set up databases and adapt them to customer specific requirements and constraints.

The *OpenView SQM* databases addressed by this document are the following:

- Service Repository Manager database
- Service Performance Data Manager database
- Logger database
- Datamart staging database
- Datamart production database
- *Business Object* repository database. Only a ready to use oracle instance is created by *OpenView SQM* tools. For detailed information concerning the storage structure please refer to the relevant *Business Object* documentation.

Oracle Software installation is out of the scope of this reference guide. Information about the best database location for the standard SQM environment is part of the *HP OpenView SQM Planning Guide*.

Intended Audience

This document is aimed at the following personnel:

- Solution Architects
- System Managers
- Database Administrators
- OpenView SQM Administrators
- System Integrators

Prior knowledge of Oracle, preferably 9i Enterprise Edition, is a pre-requisite to fully appreciate the contents of this document.

Associated Documents

The following documents contain essential reference information:

- *HP OpenView SQM Planning Guide*

Chapter 1

Oracle Architecture Fundamentals

This chapter introduces several Oracle fundamentals and how they are implemented within SQM platforms. It gives in particular information on the physical layout of the database, where they will be installed and how they use the available disks. The predefined database naming plan is also described and shows the default instance identification rules. In the latest sections the Oracle backup principles are introduced. In application specific chapter these backup capabilities will be referenced according to the needs and constraints of each application.

1.1 Server Configuration Principle

OpenView SQM implements a global approach to the Oracle product. Oracle use is normalized in order to fit Oracle standards as well as internal *OpenView SQM* guidelines.

The next sections introduce these *OpenView SQM* standards. A dedicated chapter will describe the tools associated with this global Oracle approach and the way of customizing the standard deployments to best fit internal customer standards or map available hardware configurations (disk usage rationalization and optimization.

1.1.1 OFA Compliance

The *OpenView SQM* database layout is compliant with the Optimal Flexible Architecture (OFA) described in the Oracle documentation. The main benefits are the improvement of database maintainability and recoverability.

In particular it:

- Organizes large amounts of complicated software and data on disk to avoid device bottlenecks and poor performance.
- Facilitates routine administrative tasks, such as software and data backup functions, which are often vulnerable to data corruption.
- Alleviates switching between multiple Oracle databases.
- Manages and administers database growth.
- Helps to eliminate fragmentation of free space in the data dictionary, isolate other fragmentation and minimize resource conflicts.

The main difference with OFA is that the Oracle ‘applicative’ and ‘user data’ mount points are not directly created under the root directory but grouped under /usr/ORACLE. This location will now be called the ORACLE_ROOT (it is settable if needed and can even be reduced to /).

This leads to the next mount point definitions (using Oracle terminology):

- Software mount point: /usr/ORACLE/u01

- User data mount points: /usr/ORACLE/u02….u10 (9 data mount points are pre-defined)

And according to OFA:

- ORACLE_BASE is mandatory and refers to: Software_mount_point/app/oracle.
- ORACLE_HOME is defined as $ORACLE_BASE/product/release

The ORACLE_HOME for the 9i release 2 server would be:

```
/usr/ORACLE/u01/app/oracle/product/9.2.0
```

The database name (*dbname*) used to describe the next directory structure is equal to the database SID in the *OpenView SQM* environment where no multiple instance databases are used.

The Database admin location will be created with the following subdirectories:

ORACLE_BASE/admin/dbname/adhoc	Ad hoc SQL scripts
/adump	Audit files
/arch	Archived re-do log files
/bdump	Background process trace files
/cdump	Core dump files
/create	Programs used to create the database
/exp	Database export files
/pfile	Initialization parameter files
/udump	User SQL trace files
/logbook	Files recording the status and history of the database

The pure database datafiles are spread over the user data mount points (u02 to 10). The only difference with OFA is that the ‘dbname’ directories are split into sub-directories that organize the database files by type:

user_data_mount_point/oradata/dbname/DATA	Any User data files
/RBS	Any rollback segments
/TEMP	Any Temporary tablespace files
/LOB	Available for separate LOB storage
/IDX	Any index
/SYSTEM	System tablespace files
/TOOLS	For any tools tablespace datafile
/REDO	Any redo log file
/CRT	Any control file

This subdivision helps to identify file types at a glance and can be used to map additional mount points if needed.

1.1.2 Database Layout

Disk repartition is a key aspect for performance and reliability. This section describes the default mapping of the database files over the pre-defined user data mount points.

The following table details the nine user data mount points and their default allocations.

Table 1 Database Mount Point Layout Table

Data mount point Value (default)	**Default mount point allocation for all *OpenView SQM* databases**
/usr/ORACLE/u02	System tablespace files First database control file
/usr/ORACLE/u03	Rollback tablespace files Second database control file
/usr/ORACLE/u04	Temporary tablespace files Third database control file
/usr/ORACLE/u05	Tools tablespace files
/usr/ORACLE/u06	First data files location
/usr/ORACLE/u07	Second data files location (preferred for LOBS)
/usr/ORACLE/u08	Index files location
/usr/ORACLE/u09	First redo group files location
/usr/ORACLE/u10	Second redo group files location

Flexibility is provided to let you best match your available hardware configuration. In order to help you decide how to spread mount points over your disks some hints are given concerning the various RAID possibilities and disk caching mechanisms.

1.1.2.1 Disk Sharing Rules

When merging several mount points to fit a limited number of disks you may consider the next rules but also take into account the recommendation concerning RAID and caching use (see following sections). If you plan to use RAID or caching on your file system, never merge files with incompatible constraints.

The most important rules are:

- Separate data from indexes
- Separate log files from data and indexes
- Separate archived log files from data, log and indexes

A strictly OFA compliant disk repartition (4 mount points) is sufficient to fulfill these rules.

To improve performance, reliability and maintainability the following additional rules should be considered:

- Spread data over several disks
- Spread index over several disks
- Spread log files over disks dedicated to a specific log group
- Replication log files are separated from the data and log files
- LOBs can be isolated
- Control files are copied to several disks

1.1.2.2 Disk Selection

When you have multiple RAID and disk caching solutions you should carefully decide which software mount point to place on which disk. This is an open topic even for Oracle consultants: OFA leads to specialize the use you make of all disks but in some cases placing all data over a stepped and mirrored set of disks (RAID 0+1) provides very good performance but is an expensive solution.

Internal tests showed that specialized disks running under separate controllers usually provides better high end performance and scalability than fully relying on stripped data. However, if only one poor performing disk is introduced in the configuration (One redo group with slower I/O potential) you could suffer a significant impact on the performance, which quickly becomes inferior to the fully stripped and mirrored alternative.

Information about RAID and disk caching use is given in Appendix B.

1.1.3 Database Naming Plan

According to the number of databases involved in a platform and in order to ease future replication deployment it is important to follow a strict database naming convention. This is also helpful if tools like Oracle Enterprise Manager (OEM) are used to administer a whole platform. A well-defined naming plan helps to quickly identify databases.

Moreover, to enhance platform wide visibility Oracle global naming is automatically enabled for each database when it is installed. This is of the highest importance to understand a deployed replicated (out of the V1.2 scope) or distributed Oracle environment. Global naming insures that a database link is always identified by the target global database name.

The defined naming plan insures network wide unique and easy database identification. A database is identified by its SID (pre-defined by product) the hostname where it is installed and the host domain.

The table below summarizes the pre-defined naming plan for the *OpenView SQM* products. How this plan can be adapted or where the common information will be stored (to reduce the number of questions during deployment) is detailed in section 7.5

Table 2 Database SID Table

OpenView SQM Database	Predefined SID
SRM	srm
SPDM	spdm
Logger	logger
DM Staging	dmstag
DM Production	dmprod
BO Repository	borepos

The global database name is composed as shown below.

Global database name = database name. 'server_host'.'server_domain'.

The most immediate benefit is the simplification of deployment: you will not be asked for master global names or even give free SIDs and database domains. This will limit the problems of non-unique global database names or heterogonous naming for various databases belonging to the same *OpenView SQM* platform. The server host and domain are stored in a shared configuration file (Refer to section 7.4.).

It is recommended not to change the default naming schema in order to deploy homogeneous platforms.

1.1.4 Service Naming Use

1.1.4.1 Service Naming benefits

From the 8i database family onwards, Oracle recommends using Service Naming instead of SID to identify the database(s) clients want to connect with. This naming scheme enables high availability features and introduces a new level in database identification. A physical database can provide multiple services and one service can be provided by multiple databases. The service identification becomes stronger than the system identification as one service is useful if it relies on several systems to achieve high availability. From a client point of view, service naming introduces flexibility and allows advanced backup configuration.

One or several database services can be defined in the database init file (ORACLE_HOME/dbs/init*sid*.ora) by the 'service_names' parameter. Clients will then request the defined service on connection. One important mechanism is the Oracle service registration: at startup and on regular intervals (not settable and up to one minute) the database will register its service(s) by its listener(s) with additional information concerning its current load.

This implies that **the listener should always be started before the database** to be immediately informed of service availability at database startup. If the listener is started after the DB it will only accept connections after a given interval, based on the next registration time.

To check the service registration and the current service state you can run the "services" or "status" command within the lsnrctl utility.

Note

Static service information is still configured in the listener.ora file to allow management tool use like OEM and enable heterogeneous services or external procedures.

Each *OpenView SQM* product will use a pre-defined service. It is of the highest importance to have services that cannot be confused with the database SID. This is mandatory to avoid dedicated server process spawning by the listener even if the database is not available, as this would disturb high availability mechanisms.

Moreover, it is important to append a domain definition to the service to avoid automatic concatenation with the database domain by Oracle, which would lead to a variety of services (due to the host part of the database domain) that are not suitable.

The table below summarizes the pre-defined services that are built by adding "srv" to the default SID and appending the server domain.

Table 3 Database Service Table

***OpenView SQM* Product**	**Pre-defined Database service**
SRM	**srmsrv**.*server_domain*
SPDM	**spdmsrv**.*server_domain*
Logger	**loggersrv**.*server_domain*
DM Staging	**dmstagsrv**.*server_domain*
DM Production	**dmprodsrv**.*server_domain*
BO Repository	**borepossrv**.*server_domain*

Note

Even if you change the default SID, this will not affect the service name.

You may change the default SID for any reason but this wouldn't affect the services that keep the value listed in the above table. Services have to be unique for all DBs of a given application to ease the implementation of future high availability solutions

1.1.4.2 Service Naming restriction

Service Naming relies on automatic service registration. That process unfortunately overwrites some connect string options like Session Data Unit (SDU) value that can be helpful to tune client server communication over TCP/IP protocol. This is a known restriction identified as a bug by Oracle support. When you need to change the SDU value you must use SID naming and disable service registration.

1.1.4.3 Retaining naming method.

The database instances created by *OpenView SQM* tools are fully compliant with service naming and use the default services listed above. However, In V1.2 as neither TAF nor CTF are implemented, the **client will still rely on SID naming** to describe the target database instance they want to connect with.

To put in place SDU tuning the naming schema will remain unchanged but as service registration overwrites the static listener configuration it must be disabled by using for instance a dummy listening address to preserve the real listener's configuration. This is detailed in B.1 Oracle Net SDU Tuning**.**

1.2 Oracle Net

This section describes the Oracle Net configuration.

For any usual *OpenView SQM* platform no manual oracle networking configuration is needed. The connection strings used by client application to identify their DB is stored by default in the Tibco repository that hosts all other public and settable application attributes. That way, even the usual tnsnames.ora file becomes optional. It is however possible to use the tnsnames.ora file to describe the connection strings if you find it more convenient especially for tuning and client failover capabilities with more complex syntax.

1.2.1 Client Side

1.2.1.1 Driver type selection

For V1.2, only thin driver can be used.

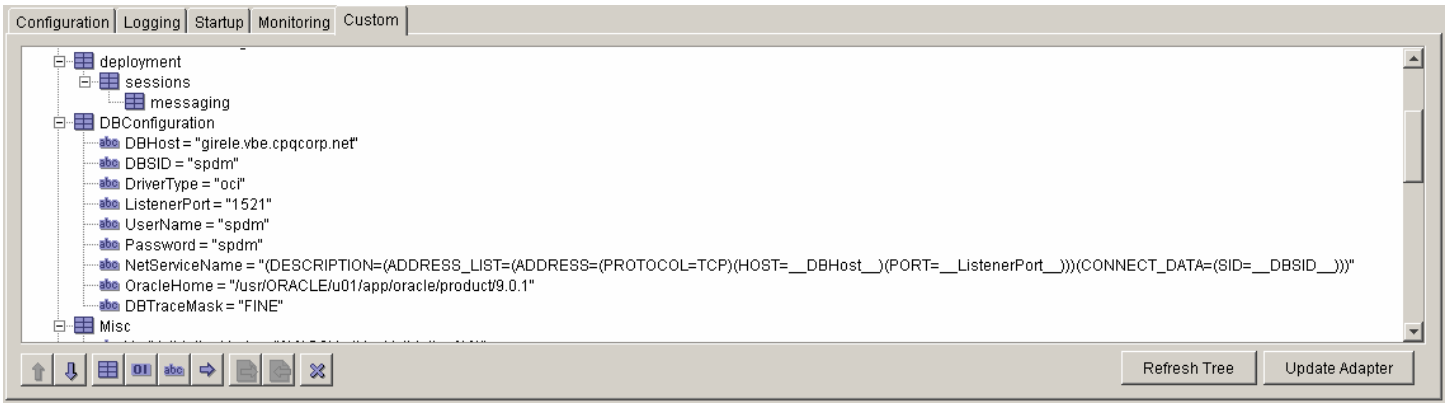
1.2.1.2 Connect string definition

As introduced earlier using tnsnames.ora file to define net service names is optional all the needed information is retrieved by the application from the centralized tibco repository.

The tibco repository is automatically completed with appropriate information during the application setup phase.

The next figure shows a description of a SPDM connection:

Figure 1 Default database connection description



1.2.1.3 Oracle Net File Location

The connection string stored in the tibco repository can be replaced by a net service name value. In that case oracle retrieves the connection information from the usual tnsnames.ora file. This can be configured to manage advanced oracle setup with the syntax and resources the oracle administrators are familiar with.

If relying on tnsnames.ora files, *OpenView SQM* uses a dedicated file in order to avoid conflicts with other existing applications that would need an Oracle Net connection. This *OpenView SQM* specific file is *TEMIP_SC_VAR_HOME*/config/tnsnames.ora.

The *OpenView SQM* runtime environment file (*TEMIP_SC_VAR_HOME/temip_sc_env.sh*) will point to that file location and define the TNS_ADMIN variable as "*TEMIP_SC_VAR_HOME*/config".

1.2.2 Server Side

On the server side the Oracle Net files are located in $ORACLE_HOME/network/admin.

The oratab file is stored in /etc

The server tnsnames.ora file isn't used to define the database links connect string. This string is directly passed to the link creation statements during DB deployment or dynamically created at application startup. This way, no entries are to be added in the tnsnames.ora files.

1.2.3 Protocol Considerations

Oracle Net connections will always be configured with the TCP protocol. In some cases, the client and server may run on the same host and the protocol should be manually set to IPC to maximize performance. This explicit protocol specification is needed as Oracle no longer supports the 'automatic IPC' feature.

1.2.3.1 Switching to IPC with the default configuration

When using the default connection string stored in the tibco repository this means updating "NetServiceName" to switch from TCP to IPC protocol and identify the instance by the "KEY" value instead of "HOST" and "PORT.

The IPC alternative is illustrated hereunder.

Figure 2 IPC database connection description

1.2.3.2 Switching to IPC when using Oracle Net configuration files.

When using the tnsnames.ora alternative the temip_oracle_client_configure tool would detect the local database and prompt for manual modification:

```
INFO :
     The "SRM" net service you are configuring
     refers to a local target database (host= target host)
     It is recommended to switch to IPC protocol :
     replace "(ADDRESS = (PROTOCOL =TCP)(HOST=target
host)(Port=1521))"
     by "(ADDRESS = (PROTOCOL = IPC)(KEY=TargetDatabaseSID))"
     in /var/opt/OV/SQM/config/tnsnames.ora
```

To improve TCP connection throughput, some tuning is possible on the packet size: Oracle provides flexibility by setting the Session Data Unit (SDU) size setting.

This tuning capability is described in B.1 Oracle Net SDU Tuning but may also be part of the product Release Notes as it is still subject to variations due to an Oracle bug related to service registration while SDU tuning is configured.

1.3 Security

1.3.1 User Authentication

Each *OpenView SQM* product uses its own user to connect to the database. No operating system authentication is used. The table below lists the created usernames within the product specific databases.

Table 4 OpenView SQM Database User Names

***OpenView SQM* database**	**Dedicated user name**
SRM	srm

SPDM	spdm
Logger	logger
DM Staging	staging
DM Production	production
BO Repository	borepos

1.3.2 User Rights

None of the above listed users will be granted 'DBA' rights or system privileges. Each user has a limited set of roles and privileges. For a detailed user rights description please refer to the product specific chapter.

Any new database will be created with a default password (the name of the database). You can modify it using Oracle specific commands. In that case, the password must also be changed in the Tibco Central repository for the corresponding component. As passwords are encrypted, use the ***temip_sc_encrypt*** command before. Refer to the *HP OpenView Administration Guide* for more details.

1.4 Database Backup and Recovery

OpenView SQM engineering does not provide a tool that encapsulates Oracle backup solutions. This section gives advice to avoid data loss and to be able to recover in case of severe database file corruption or disk loss. **Users should be familiar with the Oracle backup and recovery methodology to best anticipate and manage possible failures** (refer to "Oracle9i Backup and Recovery Concepts").

Database backup is not always sufficient to recover to the last point in time before a crash. The recovery methodology is at least as important as the backup and full recovery is only possible most of the time if some preventive actions have been taken.

Oracle provides a tool called RMAN (Recovery Manager) to assist backup and recovery operations. Oracle strongly recommends using these tools instead of self-developed procedures such as scheduled database file copies.

Users should be familiar with the RMAN command line interface or rely on the backup and recovery GUI accessible from OEM.

This entire chapter should be considered as an introduction to backup and recovery, it is a summary of Oracle's recommendations.

1.4.1 Backup Principles

This section is an overview of the related Oracle Documentation.

1.4.1.1 Physical and Logical Backups

Backups can be physical or logical:

A physical backup consists in copying (by OS or RMAN commands) the database data files and control file. If the database runs in archive mode you can also backup the archived redo logs.

Logical backups are exports of database schema objects into binary files. To retrieve schema information from the binary file you must use the IMPORT utility.

The RMAN COPY command generates file copies and validates the blocks in the files while the BACKUP command generates a logical backup set that can only be restored by RMAN (it consists of multiple backup pieces in a proprietary binary format).

1.4.1.2 Whole and Partial Database Backups

A whole database backup consists of a backup of the current control files along with all datafiles. This backup can be achieved by OS copy commands on all files, **RMAN BACKUP DATABASE**, or RMAN COPY against all individual files.

Instead of doing whole backups you can do partial database backups with various granularities. Even if OS copies are often supported, *OpenView SQM* engineering recommends using RMAN as suggested by Oracle. In the next examples only the RMAN alternative will be given, refer to the Oracle documentation for a complete description.

- **Tablespace backup** is only valid if the database is running in ARCHIVELOG mode, the redo logs are needed to make the restored tablespace consistent with others. To backup a tablespace you should use the RMAN BACKUP TABLESPACE command. In NOARCHIVELOG mode only read only and offline-normal tablespaces can be backed up.
- **Datafile backup** should be done with the RMAN BACKUP DATAFILE command. The same restriction relative to the ARCHILELOG mode applies as for tablespace backups, note also that all the files of a tablespace should be backed up to enable recovery. Due to this restriction it is recommended to prefer tablespace backups to file backups.
- **Controlfile backup** can be automated and done at each RMAN BACKUP operation; this is called controlfile autobackup (refer to the Oracle Documentation). Manual backups can be done with the RMAN BACKUP CURRENT CONTROLFILE command.
- **Archived Log file** backup is useful to secure these files that are needed to recover an inconsistent backup. The RMAN BACKUP ARCHIVELOG or RMAN BACKUP ….. PLUS ARCHIVELOG are available to achieve archived log backups.

1.4.1.3 Consistent and Inconsistent Backups

A consistent backup is composed of data files and control files that have all been checkpointed with the same System Change Number (SCN). To make a consistent backup the database needs to be closed with a clean shutdown. Whole consistent backups do not need any recovery after restore.

For databases that need to be open and available all the time you cannot perform consistent backups but need to do inconsistent ones. In this case you will obtain backup pieces with different SCNs. During the recovery phase the changes recorded in the redo logs **and archived redo logs** are needed to achieve consistency. Therefore, the database must run in ARCHIVELOG mode.

1.4.1.4 Online and Offline Backups

While backing up an online tablespace or data files some inconsistencies can occur on blocks, which are then called fractured blocks. These inconsistencies are due to simultaneous read and write operations. RMAN manages this situation by detecting it and then retries the block backup until consistency is achieved. OS copies are vulnerable to fractured blocks and need to stop database file checkpointing during backup with ALTER TABLESPACE BEGIN BACKUP. After backup, ALTER TABLESPACE END BACKUP reverts to the normal checkpointing process.

Backing up offline tablespaces or data files prevents any recovery phase reverting to an online state after having restored the backup.

1.4.2 Recovery Principles

This section is an overview of the related Oracle Documentation.

1.4.2.1 Types of Oracle Recovery

After an Oracle instance abnormal stop, **crash recovery** is automatically performed on startup. The goal is to restore the data block changes stored in the cache. Note that no Archive logs are needed.

Media recovery refers to recovery of a lost or damaged current data file or control file. Media recovery is not automatic and can require archived log files. Media recovery time can be significant according to the volume of data to restore and the number of changes to apply during recovery.

1.4.2.2 Redo Application during Recovery

The first recovery phase is called rolling forward (or cache recovery). It consists in re-applying all the changes recorded in the redo logs to the data files. As rollback information is stored in the logs, cache recovery also rebuilds the rollback segments. Note that in case of media recovery, this phase can need to apply archived log file information to the restored files.

The first phase may apply uncommitted changes that will have to be undone in the second phase called rolling back (or transaction recovery). This is done by applying the undo blocks.

1.4.2.3 Complete and Incomplete Media Recovery

Complete recovery means recovering the database from the restored backup time to the most current point in time. This means using a full or incremental backup(s) and applying all required logs (online and archived if needed).

An incomplete recovery means a recovery that does not revert to the most current point in time, as it does not apply all the required logs. This is for example the case when archived logs are missing or when you explicitly want to revert to a specific point in time to undo a recent mistake.

1.4.3 Backup and Recovery Strategies

This section gives guidelines and strategies according to your expected security level and your database availability constraints.

1.4.3.1 Backup Strategies

This section describes backup solutions according to your database configuration.

The set of files needed to recover from a failure is called the redundancy set. This set stores all the required backup information and must be stored on a separate physical media than the current database files. Not respecting this rule leads to useless backups in case of media failure. Note that backup of online files like redo logs and control files can be achieved by Oracle multiplexing, disk mirroring or both. Refer to the Oracle documentation for a full description of the recommended disk configuration.

Choosing the Database Archiving Mode

By default the deployed *OpenView SQM* Databases run in NOARCHIVELOG mode. This is done to avoid additional administrative tasks. **If you have adequate disk resources and management availability it is recommended to archive the logs in order to be able to recover up to the latest committed transaction even in cases of media failure.**

If one of the next constraints is true for you then you will have to run your database in ARCHIVELOG mode:

- If you want to be able to revert your data to any prior point in time (incomplete recovery) then you need archived logs. This could be useful if you plan to undo some possible mistakes.
- If you want to protect against any data lost in case of media failure then you need archived logs. Note that if your data is static or if its update rate is compatible with your backup period you can avoid any data loss just by using your periodic backup. An example would be a daily alarm archive followed by a daily backup (can be incremental). You should balance the cost of maintaining archived logs with the risk of media failure. If you multiplex your logs and control files and protect you data files by mirroring, the risk of media failure is reduced.
- If your database needs to be available all the time then you need archived logs as you will have to backup open files.

Backing up a NOARCHIVELOG Database

This mode does not prevent data loss in case of media failure. It is recommended to use disk mirroring to handle single disk failure.

- You should always perform a whole database backup just after database creation.
- Make regular database backups in accordance with the interval of work you could accept to lose in case of severe media failure.
- Take a whole consistent backup immediately after each database physical structure modification.

Backing up an ARCHIVELOG Database

This mode provides complete recovery capability from a disk failure.

- You should always perform a whole database backup just after database creation and once you have turned on the archive mode.
- Backup extensively used tablespaces frequently to reduce the rolling forward time in case of recovery.
- Backup the control file after each structural change to the database. Use the RMAN utility to do that.
- Archived logs are vital for recovery. If you miss only one when you try to recover, you lose the whole benefit of having archived them. You should back them up as soon as possible and keep them as long as possible, especially for point in time recovery between any available incremental backup.

Protect Important Files

Besides the potential data file mirroring, pay special attention to control files, online redo logs and archived redo logs (if any).

- You should keep at least two copies of the control file on separate disks under separate controllers. Use Oracle multiplexing and mirroring.
- Online redo logs should be multiplexed, (several members in the same group on separate disks). They are needed for crash and media recovery.
- Keep at least two copies of the archived logs.

Note

You should not backup online redo logs: they should only be multiplexed. In NOARCHIVELOG mode their backup would be useless as you take only closed backups. In ARCHIVELOG mode the filled logs are already backed up.

If you accidentally restore online logs you may corrupt the database.

Backup Frequency

Backup frequency depends on the database activity.

If database structural or physical changes occur you should take a backup before and after modification. In NOARCHIVE mode a consistent whole database backup is needed while a control file backup is sufficient in ARCHIVELOG mode.

If you open a database with RESETLOG option you should also take a full database backup.

The regular backup frequency depends on the data creation/deletion/update rate.

Note that even if you do regular database backups it can greatly reduce the recovery time if you individually backup intensively used tablespaces or data files at a higher rate.

1.4.3.2 Restore and Recovery Strategies

Restore and recovery operations are often performed in crisis situations. It is important to be familiar with performing these tasks to avoid any mistake that could generate more problems than it solves (like online log overwriting).

It is recommended to simulate several crashes and media failures on various files in test environments.

Managing Media Failures

In case of media failure:

- Identify first the files to recover.
- According to your database running mode and your backup strategy, determine if you can perform complete or incomplete recovery and if the database is open or closed.
- Restore all required backups (or file copies).
- Apply available redo logs to the latest available backup.
- Re-open the database (in case of incomplete recovery or after having restored a control file do not forget the RESETLOGS option.
- Refer to the Oracle documentation for some typical media failure recoveries.

1.4.3.3 Backup and Recovery Examples

The above sections have introduced the backup and recovery principles. The RMAN utility will not be detailed here. Refer to the RMAN users guide for a complete description.

Note that advanced RMAN use can greatly alleviate the backup and recovery maintenance tasks, as RMAN can store and manage all the existing backups and automatically clean obsolete ones. You can also create scripts to group RMAN commands and store them in the recovery catalog. Note that the control file autobackup option is useful to secure the backup operation; you will no longer risk missing a control file version in line with your backups.

Oracle provides some useful backup and recovery template scripts that address usual use-cases. These scripts are stored in $ORACLE_HOME/rdbms/demo.

- case1.rcv applies to a database running in NOARCHIVELOG mode.

- case2.rcv applies to a database running in ARCHIVELOG mode.
- case3.rcv shows how to perform tablespace point-in-time recovery.
- case4.rcv shows how to create a duplicate database that you can use to practice your backup and recovery strategy.

Chapter 2

SRM Database

This chapter presents the Service Repository Manager component regarding the Oracle Database implementation.

2.1 Overview

This database is aimed to store service models.

2.2 Database Layout and Sizing

2.2.1 Database Sizing

The Available pre-defined sizes are:

Table 5 SRM Database Predefined Database Size

Pre-defined database size	Available size
SMALL	820 MB
MEDIUM	1.2 GB
LARGE	1.77 GB

2.2.2 Database Backup

- It is recommended to backup the tablespaces, as specified in the table below.
- Backup is especially important after model updates.

2.2.3 Database Physical Layout

The following table lists the different tablespaces created in the database:

Table 6 SRM Physical Layout

Tablespace name	Corresponding file	Additional information
data	data01.dbf	SQM Model data. Backup required.

Tablespace name	Corresponding file	Additional information
com_sd	comsd01.dbf	SQM Model partitioned common table data. Backup required.
com_scd	comscd01.dbf	SQM Model partitioned common table data. Backup required.
com_scd_scd	comscdscd01.dbf	SQM Model partitioned common table data. Backup required.
com_spd	comspd01.dbf	SQM Model partitioned common table data. Backup required.
com_sptyd	comsptyd01.dbf	SQM Model partitioned common table data. Backup required.
com_dfd	comdfd01.dbf	SQM Model partitioned common table data. Backup required.
com_dfpd	comdfpd01.dbf	SQM Model partitioned common table data. Backup required.
com_dfptyd	comdfptyd01.dbf	SQM Model partitioned common table data. Backup required.
com_si	comsi01.dbf	SQM Model partitioned common table data. Backup required.
com_sig	comsig01.dbf	SQM Model partitioned common table data. Backup required.
com_sci	comsci01.dbf	SQM Model partitioned common table data. Backup required.
com_sptyi	comsptyi01.dbf	SQM Model partitioned common table data. Backup required.
com_dfi	comdfi01.dbf	SQM Model partitioned common table data. Backup required.
com_dfptyi	comdfptyi01.dbf	SQM Model partitioned common table data.

Tablespace name	Corresponding file	Additional information
		Backup required.
com_bind1	combind101.dbf	SQM Model partitioned common table data. Backup required.
com_bind2	combind201.dbf	SQM Model partitioned common table data. Backup required.
com_expr	comexpr01.dbf	SQM Model partitioned common table data. Backup required.
com_sla	comsla01.dbf	SQM Model partitioned common table data. Backup required.
com_cust	comcust01.dbf	SQM Model partitioned common table data. Backup required.
com_slo	comslo01.dbf	SQM Model partitioned common table data. Backup required.
com_sco	comsco01.dbf	SQM Model partitioned common table data. Backup required.
com_slpo	comslpo01.dbf	SQM Model partitioned common table data. Backup required.
com_scpo	comscpo01.dbf	SQM Model partitioned common table data. Backup required.
com_ot	comot01.dbf	SQM Model partitioned common table data. Backup required.
com_scd_scd	comscdscd01.dbf	SQM Model partitioned common table data. Backup required.
com_pd_bind2	compdbind201.dbf	SQM Model partitioned common table data. Backup required.
com_sci_sci	comscisci01.dbf	SQM Model partitioned common table data. Backup required.

Tablespace name	Corresponding file	Additional information
com_sig_si	comsigsi01.dbf	SQM Model partitioned common table data. Backup required.
com_sci_dfi	comscidfi01.dbf	SQM Model partitioned common table data. Backup required.
com_default	comdefault01.dbf	SQM Model partitioned common table data. Backup required.
all_indexes	index01.dbf	SQM Model indexes. Backup optional.

Chapter 3

SPDM Database

This chapter presents Service Performance Data Manager regarding the Oracle Database implementation.

3.1 Overview

This database is aimed to store performances values and status

3.2 Database Layout and Sizing

3.2.1 Database Sizing

The Available pre-defined sizes are:

Table 7 SPDM Database Predefined Database Size

Pre-defined database size	Available size
SMALL	2.7 GB
MEDIUM	4.1 GB
LARGE	14.9 GB

3.2.2 Database Backup

It is not necessary to backup the SPDM database. One of the main features of the SPDM is to compute secondary parameter values at regular periods, from Service Component parameter values stored previously (also know as primary parameters). The primary parameter values change too frequently to put in place a database backup. Moreover, all parameter values computed by the SPDM are stored within the Logger and the Datamart components, which have a policy database backup in place.

3.2.3 Database Physical Layout

The following table lists the different tablespaces created in the database:

Table 8 SPDM Physical Layout

Tablespace name	Corresponding file	Additional information

Tablespace name	Corresponding file	Additional information
Static_data_tbs_01	Static_data01.dbf	Tablespace used to store definitions of Service, Service instance, etc…(information extracted from the SRM)
Static_idx_tbs_01	Static_idx01.dbf	Tablespace used to store static model indexes used in the Static data tablespace.
Dynamic_data_tbs_01	Dyn_data01.dbf Dyn_data02.dbf Dyn_data03.dbf Dyn_data04.dbf Dyn_data05.dbf Dyn_data06.dbf Dyn_data07.dbf Dyn_data08.dbf Dyn_data09.dbf Dyn_data10.dbf	Tablesplace used to store primary parameter values, computed parameter values and statuses.
Dynamic_idx_tbs_01	Dynamic_idx01.dbf Dynamic_idx02.dbf Dynamic_idx03.dbf Dynamic_idx04.dbf Dynamic_idx05.dbf Dynamic_idx06.dbf Dynamic_idx07.dbf Dynamic_idx08.dbf Dynamic_idx09.dbf Dynamic_idx10.dbf	Tablespace used to store dynamic model indexes used in the Dynamic data tablespace.
Purge_rbs	Purge_rbs01.dbf	Specific rollback tablespace used to purge dynamic and static data.

Chapter 4

Logger Database

This chapter presents the Logger component regarding the Oracle Database implementation.

4.1 Overview

This database is aimed to store any events on services, performances values and status. These events are loaded into the Datamart staging area a regular interval.

4.2 Database Layout and Sizing

4.2.1 Database Sizing

The Available pre-defined sizes are:

Table 9 Logger Database Predefined Database Size

Pre-defined database size	Available size
SMALL	570 MB
MEDIUM	1.62 GB
LARGE	2.2 GB

4.2.2 Database Backup

Data stored in the logger are very transient. No Backup is needed as the stored information is loaded into the staging database each ‘Datamart staging granularity’ interval (refer to Datamart config guide, default value is 15 minutes)

4.2.3 Database Physical Layout

The following table lists the different tablespaces created in the database:

Table 10 Logger Physical Layout

Tablespace name	Corresponding file	Additional information
Logger	Logger01	Stores all received messages.

Tablespace name	Corresponding file	Additional information
Loggerlob	Index01	Stores the indexes
loggerindex	Lob01	Stores all LOB element when exceeding the max inline storage length

Chapter 5

DM Staging Database

This chapter presents the Datamart (Staging part) component regarding the Oracle Database implementation.

5.1 Overview

This database is aimed to store and organize any events, performances values, status.

5.2 Database Layout and Sizing

5.2.1 Database Sizing

The Available pre-defined sizes are:

Table 11 DM Staging Database Predefined Database Size

Pre-defined database size	Available size
SMALL	2.2 GB
MEDIUM	3.2 GB
LARGE	4.5 GB

5.2.2 Database Backup

It is recommended to backup that database after model updates. The Data stored in the staging area are normalized there but will become persistent in the production area.

5.2.3 Database Physical Layout

The following table lists the different tablespaces created in the database:

Table 12 DM Staging Physical Layout

Tablespace name	Corresponding file	Additional information
staging	stag01.dbf	Tablespace used to store static raw data and export table
tbsindex	index01.dbf	Tablespace used to store

Tablespace name	Corresponding file	Additional information
		static indexes used in the staging static data tablespace.
dynamic_staging	dynstag01.dbf	Tablespace used to store dynamic raw data
ca_dynamic_tbsindex	caind01.dbf	Tablespace used to store dynamic indexes used in the dynamic_staging dynamic data tablespace but for the CAView performance data.
sci_dynamic_tbsindex	sciind01.dbf	Tablespace used to store dynamic indexes used in the dynamic_staging dynamic data tablespace but for the SCI performance data.
cvi_dynamic_tbsindex	cviind01.dbf	Tablespace used to store dynamic indexes used in the dynamic_staging dynamic data tablespace but for the Compliance performance data.

Chapter 6

DM Production Database

This chapter presents the Datamart (Production part) component regarding the Oracle Database implementation.

6.1 Overview

This database is aimed to store and organize for Reporting usage historical events, performances values and status.

6.2 Database Layout and Sizing

6.2.1 Database Sizing

The Available pre-defined sizes are:

Table 13 DM Production Database Predefined Database Size

Pre-defined database size	Available size
SMALL	5.9 GB
MEDIUM	8.5 GB
LARGE	14.3 GB

6.2.2 Database Backup

It is highly recommended to backup the production area after model update. As this database is the warehouse production database regular offline backups should be performed. To avoid any data loss each backup must contain all summarized tables and records stored in raw data tables that have not yet been flagged as summarized (column “is_summarized” set to “F”).

6.2.3 Database Physical Layout

The following table lists the different tablespaces created in the database:

Table 14 DM Production Physical Layout

Tablespace name	Corresponding file	Additional information
production	prod01.dbf	Tablespace used to store static raw data and dimensions data.
dynamic_production	dynprod01.dbf	Tablespace used to store dynamic raw data.
hourly_summ_production	prod02.dbf	Tablespace used to store hourly summarized data.
daily_summ_production	prod03.dbf	Tablespace used to store daily summarized data.
weekly_summ_production	prod04.dbf	Tablespace used to store weekly summarized data.
monthly_summ_production	prod05.dbf	Tablespace used to store monthly summarized data.
quarterly_summ_production	prod06.dbf	Tablespace used to store quarterly summarized data.
yearly_summ_production	prod07.dbf	Tablespace used to store yearly summarized data.
tbsindex	index01.dbf	Tablespace used to store static indexes used in the production static data tablespace.
ca_dynamic_tbsindex	caind01.dbf	Tablespace used to store dynamic indexes used in the dynamic_production dynamic data tablespace but for the CAView performance data.
sci_dynamic_tbsindex	sciind01.dbf	Tablespace used to store dynamic indexes used in the dynamic_production dynamic data tablespace but for the SCI performance data.
cvi_dynamic_tbsindex	cviind01.dbf	Tablespace used to store dynamic indexes used in the dynamic_production dynamic data tablespace but for the Compliance performance data.
hourly_summ_tbsindex	index02.dbf	Tablespace used to store indexes used in the hourly_summ_production hourly summarized data tablespace.
daily_summ_tbsindex	index03.dbf	Tablespace used to store

Tablespace name	Corresponding file	Additional information
		indexes used in the daily_summ_production daily summarized data tablespace.
weekly_summ_tbsindex	index04.dbf	Tablespace used to store indexes used in the weekly_summ_production weekly summarized data tablespace.
monthly_summ_tbsindex	index05.dbf	Tablespace used to store indexes used in the monthly_summ_production monthly summarized data tablespace.
quarterly_summ_tbsindex	index06.dbf	Tablespace used to store indexes used in the quarterly_summ_production quarterly summarized data tablespace.
yearly_summ_tbsindex	index07.dbf	Tablespace used to store indexes used in the yearly_summ_production yearly summarized data tablespace.
shsa	shsa01.dbf	Only default tablespace no data stored there.

Chapter 7

Oracle Tools

7.1 Introduction

To implement the new global approach, a set of unified tools is provided to manage database deployment and deletion as well as Oracle Net file configuration for all *OpenView SQM* applications using Oracle databases.

The user interface is similar for all databases and only a few parameters will be asked for a default installation. Despite the interface simplification flexibility is provided to best match your sizing and hardware configuration. The possible adaptations of the default database layout and sizing are handled in a unique manner for all databases.

The following sections will introduce the tools; describe their functionality and how they perform. A particular focus will be put on adaptation capabilities to help you map your databases over your hardware.

Note that additional product specific tools can be provided to perform statistics on given tables or perform specific database administrative tasks. These tools are described in the product specific sections.

7.2 Oracle Script Installation

In *OpenView SQM*, a dedicated oracle component is intended to deliver the Oracle configuration scripts. This particular subset can be installed in standalone mode (without *OpenView SQM* Framework). For more information concerning the installation procedure please refer to the OpenView SQM Installation Guide.

7.3 Integration with Install Anywhere (IA) Setup Process

The tools described in this section can be used at the command line and are also called by the graphical setup. The IA graphical setup is strictly identical to the command line mode in its processing and exactly the same scripts are involved. Only the interface differs in the sense that the command line asks for input parameters while IA provides all the required information in configuration files.

The IA directives that encapsulate the shared tools provide the same level of flexibility as the command line (refer to the tool interface description sections). To setup advanced parameters the only way provided is to edit their values in the related configuration files.

The next sections describe how to use the command line without the graphical interfaces. To see the graphical interfaces please refer to the SQM Installation Guide.

7.4 Tool Organization

$TEMIP_SC_HOME/oracle is the main directory that stores all the required information and utilities. This location is overwritten on release update. The configuration information is duplicated under ***$TEMIP_SC_VAR_HOME*/oracle**.

***$TEMIP_SC_HOME*/oracle** is organized in subdirectories as shown below:

- **scripts**: The shared tools
- **utils**: Common utilities used by the tools to perform basic tasks (they are never used directly)
- Product specific subdirectories and configuration files that should never be edited

***$TEMIP_SC_VAR_HOME*/oracle** is used to store generated logs and configuration inputs:

- **conf**: directory that stores configuration files relative to the server
- **srm**: configuration and logs relative to the *SRM database*
- **spdm**: configuration and logs relative to the SPDM datgabase
- **logger**: configuration and logs relative to the Logger database
- **dmstag**: configuration and logs relative to the Datamart staging database
- **dmprod**: configuration and logs relative to the Datamart production database
- **dmarch**: configuration and logs relative to the Datamart archive database
- **borepos**: configuration and logs relative to the BO repository database

Note

Configuration files (*.cfg) should never be edited in the ***$TEMIP_SC_HOME***/oracle sub tree. The tools will discard these files and rely on the copies available in the *$TEMIP_SC_VAR_HOME*/oracle.

The next sections detail all the tools available under ***$TEMIP_SC_HOME***/oracle/scripts:

- **temip_create_database**: unique tool responsible for any database creation
- **temip_delete_database**: unique tool to assist the database deletion
- **temip_oracle_client_configure**: can be used manually to configure net service descriptors in a unified manner.

Each tool can be called without parameters and will display a help message or guide you through the available choices.

7.5 temip_create_database

This tool relies on the product specific information. It will deploy all databases in a unified way by using a set of common utilities.

To best understand how this tool works, its functional description is completed with some design hints to help you master advanced customization. For advanced configurations and customization it is highly recommended to fully read the detailed sections.

Note

temip_create_database must be run as root user. This is required to su the oracle user whenever needed.

7.5.1 Interface Description

You can display a help message with:

```
temip_create_database –h
```

Usually you should invoke temip_create_database without any parameter to enter an interactive mode that will ask you to choose the database you would like to create and guides you through installation by asking basic questions.

A non-interactive mode (option –NI) exists but is mainly designed for applications like graphical interfaces, which uses the tool without any interaction and first sets the typically asked inputs in the configuration files.

If you have pre-filled your configuration files (refer to detailed tool description) you could for example create a SRM database without any interaction by entering:

```
temip_create_database srm –NI
```

7.5.1.1 Questions

temip_create_database first suggests a choice of available databases:

```
temip_create_database will deploy a database for one of the
following OpenView SQM products:

        - TeSC logger               (logger)    [1]
        - TeSC logger's archive     (arch)      [2]
        - TeSC SPDM                 (spdm)      [3]
        - TeSC SRM                  (srm)       [4]
        - TeSC Datamart production  (dmprod)    [5]
        - TeSC Datamart staging     (dmstag)    [6]
        - TeSC Datamart archive     (dmarch)    [7]
        - TeSC BO repository        (borepos)   [8]
        - Exit    (terminate the session)       [e]

             Please enter the database you want to install:
```

Note

The options [2] and [7] are not supported in the current version of SQM. Do not use them.

- Once you have chosen the database you want to install, you will be asked for pre-defined sizes: SMALL, MEDIUM, LARGE or USER. The USER size is provided to allow you to size your database as you want. Note that it is defaulted to the SMALL configuration if you do not specify any specific value in the configuration files.
- In a non-clustered environment you will be asked if the new database should be restarted or not at machine boot time
- You will then be asked for product specific inputs, if any.

Note

You will never be asked for SID or database name as these inputs are defaulted to the values described in the server configuration section. Refer to the advanced use chapter for details on how to modify default values.

If you call the tool twice for the same product, it puts forward your previous replies as default values.

7.5.2 Functional Description

temip_create_database takes dynamic information from the prompt and relies on configuration files for more detailed tuning and layout information. Two main steps are distinguished during deployment:

- The “instance creation” phase that really creates the databases, with its classical tablespaces and resources. It also runs several Oracle scripts to create internal views and the usual packages.
- The “schema creation” phase that creates the product’s schema with associated tables and users. Note that this phase also creates the underlying tablespaces.

Note

The tool will run even if the specified database already exists. In this case it will skip the instance creation and only run schema creation, even the Oracle Net file update will be skipped.

The next list summarizes the main actions performed by temip_create_database and shows how the tool embeds the instance and schema creation.

- Validates the environment
- Checks if the database already exists
- Asks for parameters (they can also be read from the configuration files when using the non-interactive mode)
- Manages instance creation (only if the database instance does not already exist)
  - Builds “instance creation” script (from a template)
  - Runs “instance creation” script and generates logs
  - Stops and restarts the new database
  - Checks the database instance creation by log parsing
- Manages schema creation (only if successful instance creation or if the instance database was already existing)
  - Builds “schema creation” script
  - Runs “schema creation” script and generates logs
  - Checks the database schema creation
- Updates listener and oratab file (only in case of new instance creation)
- Stops and restarts the listener (only in non-clustered environments)

7.5.3 Log Files

The deployment phase generates several logs. All the generated log files are indicated at the interface level.

The instance and schema creation phases generate separate logs named create_instance.log and create_schema.log. These files are stored in the log subdirectory present under each product specific directory: *$TEMIP_SC_VAR_HOME*/oracle/*product*/log/.

Other more specific logs are spooled to the DB_ADMIN location (“create” subdirectory). That admin directory also stores backups of modified files.

7.5.4 Advanced Use

The simplified interface alleviates the database creation and helps to deploy databases that are fully standardized. In some cases you may want to modify the database layout, naming or sizing. Therefore flexibility is provided via configuration files. This section introduces the configuration files and shows how temip_create_database relies on template scripts to help you best understand what happens at run time and to customize your databases.

7.5.4.1 Configuration Files

- $TEMIP_SC_VAR_HOME/oracle/conf/temip_oracle_configuration.cfg

This configuration file stores information relating to the server settings. This file defines variables like ORACLE_HOME, ORACLE_BASE, server host and domain and default software and user data mount point mapping. It only needs to be updated if you install Oracle without using the tools provided by *OpenView SQM* engineering or if you already have an installed server that does not fit the described standards.

If temip_create_database detects that your server does not match your configuration files, it will prompt you for a manual update and list the parameters you should check. Usually you will need to give ORACLE_HOME and ORACLE_BASE and two other variables named SERVER_HOST and SERVER_DOMAIN that will be used for listener update and service definition. How to change the mount point mapping will be detailed as a use-case.

- $TEMIP_SC_VAR_HOME/oracle/product/conf/temip_database_mandatory_parameters.cfg

This product specific file stores typical database parameters that are used during instance creation. All products use these parameters but their values can differ from one database to the other (for example rollback tablespace sizing). The stored information is related to the database naming, its mapping over mount points and gives typical sizing inputs. For classical configurations you will never have to edit this file, the use case section gives examples of default value modifications in case of layout and naming modifications.

- $TEMIP_SC_VAR_HOME/oracle/product/conf/temip_database_product_configuration.cfg

This product specific file is used to manage potential product specific questions and define sizing information relative to product specific tablespaces.

7.5.4.2 Templates

The instance and schema creation phase use the variables provided in the configuration files to build runtime sql scripts based on templates. These templates are stored in *$TEMIP_SC_HOME*/oracle/*product*/scripts such as “create_instance.sql.tmpl” and “create_schema.sql.tmpl”. The latest template instantiation (if any) can also be found as “create_instance.sql” and “create_schema.sql”.

The database init and configuration files are also provided as templates in this directory.

7.5.5 Typical Use-Cases

7.5.5.1 My Oracle Mount points do not match the described standard

Ten default mount points are defined and you have fewer disks or would like to use a reduced set of mount points.

Edit:

$TEMIP_SC_VAR_HOME/oracle/conf/*temip_oracle_configuration*.cfg

and update the values of variables named ORACLE_MPx (x from 01 to 10) to fit your configuration.

Note

You should not delete mount point definitions but ONLY change their value. To reduce their number, give the same value to several mount points.

7.5.5.2 I would like to change my database file distribution over the defined Oracle mount points

This is a product specific modification. You should edit the appropriate:

$TEMIP_SC_VAR_HOME/oracle/*product*/conf/*temip_database_mandatory_parameters.cfg*' file.

In the file section named "DATABASE LAYOUT" you can modify the control file location and all other files grouped by type: (rollback, data, index, temporary, redo group 1 and 2). Note that the database file locations are relative to the server mount point definition.

Example of rollback tablespace's file location:

DB_RBS_LOC=${ORACLE_MP3}/oradata/${DB_NAME}/RBS

7.5.5.3 I need to change the default database SID

This could be required if you want to deploy a new database of a product and already have a previous version installed with same SID. You could also deploy multiple instances for the same product on a unique server for test purposes (test + operational instance or master/replica on the same server for functional tests).

This is a product specific modification. You should edit the appropriate:

$TEMIP_SC_VAR_HOME/oracle/*product*/conf/*temip_database_mandatory_parameters.cfg*' file.

Set the ORACLE_SID variable (defaulted to the product) to the appropriate value.

7.5.5.4 I want to run multiple instances of the same product database on a unique server

You will need to choose a different SID for the second instance (refer to the above use-case).

Note

Even if you change the SID, this will not impact the database service. If you have two instances with different SIDs running on the same host and use service naming, then client connection will be routed to one or the other according to service registration mechanisms.

To explicitly choose one instance you should specify (INSTANCE_NAME = *SID*) in the CONNECT_DATA section of the used net service name in the client's tnsnames.ora file.

7.5.6 Troubleshooting

7.5.6.1 Common Errors

Common errors are detected by temip_create_database during deployment: in particular, errors due to insufficient disk space or sizing errors (extent allocation problems), missing directories or already existing files. All connection problems that occurred during deployment are also considered as fatal.

Check the log files present in *$TEMIP_SC_VAR_HOME/*oracle/*product*/log if your deployment fails. You can use temip_delete_database to cleanup the partially created database and run temip_create_database again after having fixed the original problem (your previous replies will be provided as default values).

Note

To detect DB files temip_delete_database needs some database internal views. If they are missing you must delete the already created files manually (refer to temip_delete_database section).

7.5.6.2 Configuration Errors

Typing errors in the configuration files are another error source. When editing these files you should avoid useless blanks at line beginning or ending, special characters like tabulations should also be avoided.

To best troubleshoot these errors check the generated database init and configuration files and look for misspelled parameters. If these files seem to be correct, check the generated create_instance.sql and create_schema.sql files in *$TEMIP_SC_HOME*/oracle/*product*/scripts and look for invalid or missing sql statement arguments.

Note

temip_create_database uses ‘/usr/bin/su oracle’ to run the scripts. Your Oracle user’s environment should not set variables like ORACLE_HOME or ORACLE_SID when doing this. The tools detect this problem and ask for manual fixes if needed.

7.5.6.3 temip_create_database hangs

Check the currently created log file in *$TEMIP_SC_VAR_HOME/*oracle/*product*/log. If temip_create_database hangs it is most likely while running one of Oracle’s scripts. catproc.sql in particular hangs if connection problems (end of file on communication channel) occur at runtime. In this case it may wait for user inputs and hangs.

It is recommended to stop the deployment or let it continue by answering Oracle’s script question. Once the instance creation phase completes, temip_create_database will stop and ask for a manual log check.

After having exited the deployment phase use temip_delete_database to delete the instance and recreate it. (your previous inputs will be provided as default values).

7.6 temip_delete_database

This tools relies on the configuration files to get information related to the Oracle installation and the deployed instances (ORACLE_HOME , SID).

The next paragraphs describe the interface and explain how the tool detects and deletes *OpenView SQM* databases.

Note

temip_delete_database must be run as root user. This is required to su the oracle user whenever needed.

7.6.1 Interface Description

You can display a help message with:

```
temip_delete_database –h
```

Usually you should invoke temip_delete_database without any parameter to enter an interactive mode that will display a list of detected databases and let you chose the one you want to delete.

```
Building the list of available databases...

Please choose the database you want to delete.

     - TeSC logger               (logger)     [1]
     - TeSC SPDM                 (spdm)       [2]
     - TeSC SRM                  (srm)        [3]
     - TeSC Datamart staging     (dmstag)     [4]
     - Exit    (terminate the session)        [e]

          Enter your choice:
```

A non-interactive mode (option –NI) exists but is mainly designed for applications like graphical interfaces, which uses the tool by specifying the product to delete and disables confirmation requests.

```
temip_delete_database srm –NI
```

will delete the SRM database without any confirmation request.

An '–ALL' option is available to delete all installed *OpenView SQM* databases. You can use this option even in non-interactive mode but this will delete all databases without any confirmation request.

```
temip_delete_database -ALL –NI
```

7.6.2 Functional Description

The tool first detects the deployed databases, it relies therefore on the database administrative location presence: $ORACLE_BASE/admin/*dbname* where *dbname* is one of the pre-defined *OpenView SQM* products.

Once you have chosen the product it will get additional information such as the SID from the configuration files.

If required, the tool will start the database in mount mode.

Then it queries V$DATAFILE, V$LOGFILE and V$CONTROLFILE, displays the found files and asks for deletion confirmation.

After file deletion it removes the database init and configuration files and the obsolete directories.

Note

You must manually edit the oratab and listener.ora files to remove static information relative to the destroyed database.

7.6.3 Manual Database Deletion

As seen above, the deletion tools rely on database view to get the list of files to delete. In case of severe DB corruption, or if the Database creation failed before these views were created, you may have to delete a database manually.

If the database instance is running try to shutdown it (even with shutdown abort).

1. First delete all the database files stored under the data mount points

rm –rf $ORACLE_ROOT/*/ORADATA/PRODUCT_ID

- ORACLE_ROOT is by default /usr/ORACLE
- PRODUCT_ID is one of product as listed in Table 2 Database SID Table (remains fixed even if you changed the default SID)

2. Then delete the init files links and lock files under $ORACLE_HOME/dbs

3. Remove all files having the SID in their name (lower and upper case)

4. Finally remove the Database admin location

rm –rf $ORACLE_BASE/admin/*PRODUCT_ID*

7.6.4 Typical Use-Cases

7.6.4.1 I need to delete multiple instances of the same product database

In this case you have changed the default SID before installing the second instance with temip_create_database.

temip_delete_database will first delete the latest created instance (SID still stored in *$TEMIP_SC_VAR_HOME*/oracle/*product*/*temip_database_mandatory_parameters.cfg*). After deletion you should set ORACLE_SID to the second instance's value in that file and run the deletion tool again.

7.6.5 Troubleshooting

7.6.5.1 temip_delete_database detects no file

The database has been proposed for deletion by the tool. But no file has been found. The views queried to retrieve the files could not exist or the connection and mount failed.

These conditions are detected by temip_delete_database that prompts for a manual check and indicates log files.

Note that if you do not change the default database layout all files can be seen with the next command:

```
ls /usr/ORACLE/*/oradata/dbname/*/*
```

Do not forget the link to the database init file in $ORACLE_HOME/dbs and the database admin directories.

Chapter 8

Oracle Troubleshooting

This chapter provides some basic Oracle commands to troubleshoot and manage your databases. It cannot address all the possible issues but gives hints and recommendations related to the current operations.

8.1 Managing the Oracle Environment

Before doing any management operation related to Oracle databases you should be logged into the system as user ‘oracle’, part of the ‘dba’ group.

In Oracle’s current environment the next variables should be set:

ORACLE_HOME: could be /usr/ORACLE/u01/app/oracle/product/`9.2.0`

LD_LIBRARY_PATH: $ORACLE_HOME/lib

ORACLE_SID: the SID of the database you would like to manage

Adding ORACLE_HOME/bin to the PATH is also helpful.

Note

For localization purpose the SQM databases use the larger time zone file provided by oracle thus you must set the next variable to be able to start an oracle instance

ORA_TZFILE=${ORACLE_HOME}/oracore/zoneinfo/timezlrg.dat

If the default time zone file fits your local need you can switch to the default file ($ORACLE_HOME/oracore/zoneinfo/timezone.dat)

ORA_TZFILE is gotten from $TEMIP_SC_HOME/oracle/conf/temip_oracle_configuration.cfg during database creation and is also set in the SQM environment file : $TEMIP_SC_VAR_HOME/temip_sc_env.sh

8.1.1 Listener Management

To administer listeners (defined in $ORACLE_HOME/network/admin/listener.ora) a dedicated tool is provided. $ORACLE_HOME/bin/lsnrctl.

You can omit specifying the *listener_name* in your commands if your listener has the default “LISTENER” name.

- To start a listener:

  `lsnrctl start` *`listener_name`*

- To stop a listener:

  `lsnrctl stop listener_name`

- To obtain a status:

  `lsnrctl status listener_name`

- To get information on the database services handled by the listener:

  `lsnrctl services listener_name`

Note

Listeners should be started before the database instances and stopped afterwards.

If you stop and restart a listener while a database is running it can take up to one minute before services are registered again.

8.1.2 Database Instance Management

To manage instances use $ORACLE_HOME/bin/sqlplus after having set ORACLE_SID to the appropriate instance's SID.

Note

From Oracle 9.0.1 onwards, "svrmgrl" and the "connect internal" connect string are obsolete. The following commands replace them.

1. Connect to the instance (get the SQL prompt)

   ```
   sqlplus /nolog
   ```

2. Login as 'sys' user:

   ```
   SQL > connect / as sysdba
   ```

3. To start the instance:

   ```
   SQL > startup
   ```

4. To stop a running instance:

   ```
   SQL > shutdown normal | immediate | transactional| abort
   ```

Normal

No new connections are allowed after the statement is issued.

Before the database is shut down, Oracle waits for all currently connected users to disconnect from the database.

The next startup of the database will not require any instance recovery procedures.

Immediate

No new connections are allowed, nor are new transactions allowed to be started, after the statement is issued.

Any uncommitted transactions are rolled back. (If long uncommitted transactions exist, this method of shutdown might not complete quickly, despite its name.)

Oracle does not wait for users currently connected to the database to disconnect. Oracle implicitly rolls back active transactions and disconnects all connected users.

The next startup of the database will not require any instance recovery procedures.

Transactional

No new connections are allowed, nor are new transactions allowed to be started, after the statement is issued.

After all transactions have completed, any client still connected to the instance is disconnected.

At this point, the instance shuts down just as it would when a SHUTDOWN IMMEDIATE statement is submitted.

The next startup of the database will not require any instance recovery procedures.

A transactional shutdown prevents clients from losing work, and at the same time, does not require all users to log off.

Abort

No new connections are allowed, nor are new transactions allowed to be started, after the statement is issued.

Current client SQL statements being processed by Oracle are immediately terminated.

Uncommitted transactions are not rolled back.

Oracle does not wait for users currently connected to the database to disconnect. Oracle disconnects all connected users.

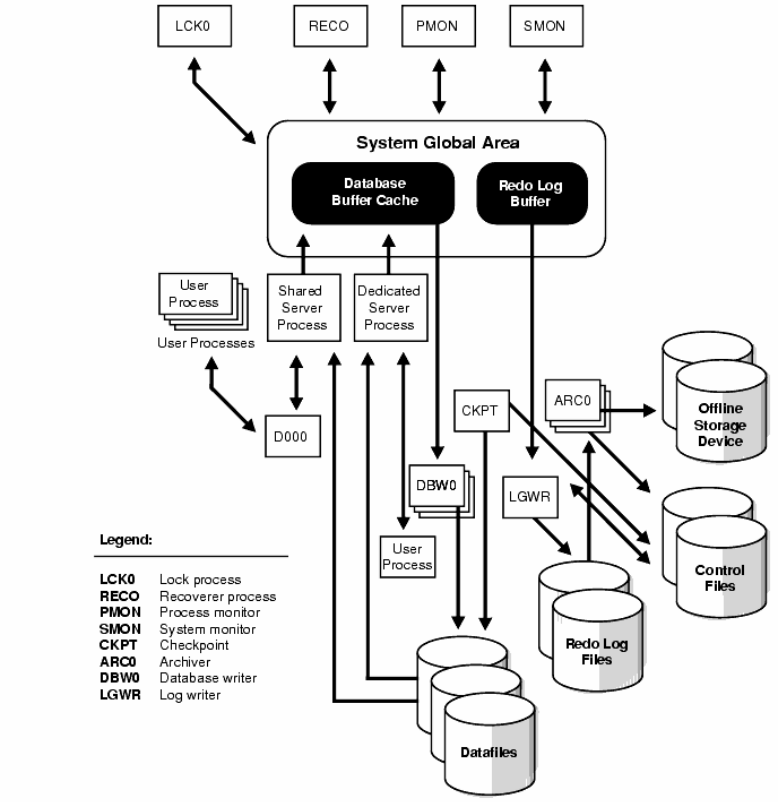
The next startup of the database **will** require instance recovery procedures.

8.2 Getting Basic Information from an Oracle Instance

8.2.1 Identifying the Main Database Processes

The next figure shows the main Oracle processes. For a detailed description refer to the Oracle documentation.

Figure 3 Oracle Processes and Architecture



8.2.2 Getting Current Database Parameters

The database initialization parameters are stored in the init*sid*.ora file. You will find a link to that file under $ORACLE_HOME/dbs/.

This file only shows explicitly defined parameters; to get full information on all existing database parameters you can query the V$PARAMETER view:

```
SQL > select name, value from V$PARAMETER;
```

8.2.3 Finding Database Files

To re-size files or to manually delete a database you may want to list all its files. To achieve this you can use Oracle views

.

```
SQL> select name from V$DATAFILE;
NAME
----------------------------------------------------------
/usr/ORACLE/u02/oradata/srm/SYSTEM/system01.dbf
/usr/ORACLE/u05/oradata/srm/TOOLS/tools01.dbf
/usr/ORACLE/u03/oradata/srm/RBS/rbs01.dbf
/usr/ORACLE/u04/oradata/srm/TEMP/temp01.dbf
/usr/ORACLE/u06/oradata/srm/DATA/users01.dbf
/usr/ORACLE/u06/oradata/srm/DATA/data01.dbf
/usr/ORACLE/u06/oradata/srm/DATA/index01.dbf

SQL > select member from V$LOGFILE;
MEMBER
----------------------------------------------------------
/usr/ORACLE/u09/oradata/srm/REDO/redo01.log
/usr/ORACLE/u10/oradata/srm/REDO/redo02.log
/usr/ORACLE/u09/oradata/srm/REDO/redo03.log

SQL > select name from V$CONTROLFILE;
NAME
----------------------------------------------------------
/usr/ORACLE/u02/oradata/srm/CRT/control01.ctl
/usr/ORACLE/u03/oradata/srm/CRT/control02.ctl
/usr/ORACLE/u04/oradata/srm/CRT/control03.ctl

In case of database deletion do not forget to delete the
database initialization file and its admin directory.
```

8.3 Fixing Common Oracle Error Messages

8.3.1 Common Oracle Network Errors

Refer to the *Oracle Net Services Administrator's Guide* for more information.

Table 15 Common Oracle Network Errors

Error #: Message	Description/Troubleshooting Procedures
ORA-12154: "TNS:could not resolve service name"	Cause: Oracle Net could not locate the service name specified in the TNSNAMES.ORA configuration file. Actions: 1. Verify that a TNSNAMES.ORA file exists and that it is accessible. 2. Verify that there are no multiple copies of the TNSNAMES.ORA file. 3. In your TNSNAMES.ORA file, verify that the service name specified in your connect string is mapped to a connect descriptor in the TNSNAMES.ORA file. Also, verify that there are no syntax errors in the file. 4. Verify that there are no duplicate copies of the SQLNET.ORA file. 5. Verify that your SQLNET.ORA file does not contain a

Error #: Message	Description/Troubleshooting Procedures
	NAMES.DEFAULT_DOMAIN parameter. If this parameter exists you should remove it, otherwise you must specify the domain name in your connect string. 6. If you are connecting from a login box, verify that you are not placing an "@" symbol before your connect service name.
ORA-12198: ``TNS:could not find path to destination" and ORA-12203 ``TNS:unable to connect to destination"	Cause: The client is not able to find the desired database. Actions: 1. Verify that you have entered the service name you wish to reach correctly. 2. Verify that the service name ADDRESS parameters in the connect descriptor of your TNSNAMES.ORA file are correct. 3. Verify that your TNSNAMES.ORA file is stored in the correct directory. 4. Verify that the listener on the remote node has started and is running. If not, start the listener by using the Listener Control Utility. 5. If you are connecting from a login box, verify that you are not placing an "@" symbol before your connect service name.
ORA-12224:"TNS:no listener"	Cause: The connection request could not be completed because the listener is not running. Actions: 1. Ensure that the supplied destination address matches one of the addresses used by the listener. 2. Verify also that this is not a version compatibility problem.
ORA-12533: "TNS:illegal ADDRESS parameters"	Cause: The protocol specific parameters in the ADDRESS section of the designated connect descriptor in your TNSNAMES.ORA file are incorrect.
ORA-12545: "TNS:name lookup failure"	Cause: The listener on the remote node cannot be contacted. Actions: 1. Verify that the ADDRESS in the TNSNAMES.ORA file or the LISTENER.ORA file is correct. 2. Verify that the listener on the remote node has been started. You may check its status with the STATUS command of the Listener Control Utility, and start it with the START command if necessary.
ORA-3113: "TNS:End of file on communication channel"	Cause: An unexpected end of file was processed on the communication channel. This may be an indication that the communications link may have gone down at least temporarily; or it may indicate that the server has gone down.

8.3.2 Common Instance Error Messages

Table 16 Common Oracle Errors

Error #: Message	Description/Troubleshooting Procedures
ORA-1034: “Oracle not available”	Cause: The database instance was not started up. Possible causes include the following: • The SGA requires more space than was allocated for it.

	• The operating system variable pointing to the instance was improperly defined. Action: 1. Refer to accompanying messages for possible causes and correct the problem highlighted in the messages. 2. Verify the Oracle instance processes: ps –ef | grep ora | grep <*oracle_sid*> Stop and restart the database, if processes are not found.
ORA-01014: Oracle shutdown in progress	Cause: A user tried to log on to Oracle while an instance shutdown was in progress. Oracle logons are disabled while Oracle is being shut down. Action: Wait until Oracle is brought back up before attempting to log on.

8.3.3 Help with Error Information

An Oracle tool can help you to get information about an -xxxx error:

```
$ORACLE_HOME/bin/oerr ora xxxx
```

`Oerr` displays probable causes of the error and the action to take.

8.3.4 Trace Files

Oracle Server Products provide information about background processes in trace files. These files are located in the database admin directory:

$ORACLE_BASE/admin/*product*/bdump/
(Example, /usr/ORACLE/u01/app/oracle/admin/srm/bdump/alert_srm.log).

Identifying error information (ORA-xxxx:message) will help you to resolve problems.

8.4 Oracle Utilities

No exhaustive list and description is provided here. Only the main assistants and controllers are introduced with their basic functionalities.

8.4.1 Assistants

Oracle provides a set of graphical utilities to alleviate database management and configuration operation.

8.4.1.1 Database Configuration Assistant

This tool is launched with:

```
$ORACLE_HOME/bin/dbca
```

It can be used to create databases and provides pre-defined sizing choices. This tool is also able to delete or modify existing databases.

8.4.1.2 Oracle Net Configuration Assistant

This tool is launched with:

```
$ORACLE_HOME/bin/netca
```

It simplifies basic Oracle Net configuration and can be used to check or complete the tnsnames.ora and listener.ora files.

Note

CTF and TAF configuration are still manual tasks.

8.4.1.3 Oracle Enterprise Manager Configuration Assistant

This tool is launched with:

```
$ORACLE_HOME/bin/emca
```

It provides an easy way to configure a local management server. It is also responsible for all creation, upgrade and deletion operations on the associated repository database.

8.4.2 Oracle Controller

The controllers introduced in the next sections are non-graphical controllers used to manage resources. All these utilities are located under $ORACLE_HOME/bin/.

8.4.2.1 Oracle Intelligent Agent Controller

This controller manages the Oracle Intelligent agent also known as Oracle SNMP agent.

Use:

```
agentctl start|stop|status|restart [agent]
```

8.4.2.2 Oracle Enterprise Manager Controller

This controller is mainly used to manage the Oracle Management Server. To start the management server and be able to use the graphical OEM console over a repository you will for example use:

```
oemctl start oms
```

8.4.2.3 Listener Controller

This controller has been introduced in the “listener management” section.

8.4.3 Oracle Enterprise Manager

A common tool is provided to launch all the graphical managers integrated or related to the Oracle Enterprise Manager software. You can start most of these managers independently but to launch the main console you should use:

```
$ORACLE_HOME/bin/oemapp console
```

Appendix A

Raid Use

This section briefly describes the various RAID levels and their possible use on Oracle files.

A.1 Raid Use

A.1.1 RAID 0: Striping With No Parity

Raid 0 offers no protection against drive failures, it can provide a slight I/O performance improvement since disks are able to do some work in parallel.

It is useful in reducing disk hot spots for Oracle data files, but is generally not recommended for other Oracle files.

A.1.2 RAID 1: Shadowing

Raid 1 provides complete protection against single drive failure.

When all drives are functioning, reads complete slightly faster than a single disk read. Writes take slightly longer than a single disk write. During a single disk failure performance is equivalent to a single disk.

Raid 1 can be used for any Oracle file but is especially useful for redo log and control files.

Note

The DBA/system administrator must use the RAID controller utilities to keep up with failed disks since the shadowing of the file is hidden from Oracle.

A.1.3 RAID 0+1: Striping and Shadowing

Raid 0+1 offers hot spot reduction and redundancy.

Can be used with Oracle data files but should not be used with redo log files.

A.1.4 RAID 3: Striping with Static Parity

RAID 3 attempts to give the performance and redundancy of RAID 0+1 without the high cost associated with RAID 1's 1-for-1 drive redundancy.

Performance is affected by disk failure and the parity disk can become a bottleneck

Is useful for Oracle data files but not for redo log files.

A.1.5 RAID 5: Striping with Rotating Parity

RAID 5 is similar to RAID 3 but the parity is spread across all drives.

This eliminates the parity drive as a bottleneck.

Can be used with Oracle data files but should not be used with redo log files.

A.1.6 Summary

Table 17 RAID Use Table

Type of RAID	Control file	Database file	Redo Log file	Archive Log file
0 Stripping	Avoid	OK	Avoid	Avoid
1 Shadowing	Recommend ed	OK	Recommended	Recommended
0+1 Stripping +	OK	Recommended	Avoid	Avoid

Shadowing		(1)		
3 Stripping with static parity	OK	OK (2)	Avoid	Avoid
5 Stripping with rotating parity	OK	OK (2)	Avoid	Avoid

Note

- (1) RAID 0+1 is recommended for database files because this avoids hot spots and gives the best possible performance during a disk failure. The disadvantage of RAID 0+1 is that it is a costly configuration.
- (2) RAID 3 and 5 are recommended for database files if RAID 0+1 is too expensive. It should however be avoided for write intensive datafiles.

A.2 Disk Caching

Oracle behaves as a disk-caching engine; the caching mechanism is of the “write-back” type. The data buffer of the SGA is the cache portion (virtual memory) and the log files are needed to prevent data inconsistency.

In I/O intensive environments, other disk caching products can be added to the native architecture. This section describes their advantages for the various Oracle files and highlights reliability, memory waste and performance issues.

A.2.1 Reliability

Write-back cached database files can lead to problems when DB recovery is needed: Oracle knows at all times where the current copy of a database block is. It is either in the buffer cache or it is in the database file. When write-back caching is used on database disks and a system failure occurs, it is possible that Oracle's recovery mechanism may find a disk database block to be older than expected, and have insufficient information to bring the database block up-to-date.

Write-back cached redo log files can lead to transaction recovery problems: Any committed transaction is written to a persistent store -- the redo log file -- so that if the system crashes, Oracle can regenerate the transaction. With write-back caching this redo is no longer persistent and it is possible that Oracle may not recover transactions that it said were committed before the failure.

Some write-back caching, such as HSZ40, are battery-backed memory implemented as an I/O controller to avoid these limitations during system crashes.

A.2.2 Memory Waste

Software-based caching leads to a memory waste due to the double caching of the same information. This is not the case with hardware cache products. Note that two levels of caching is not always an efficient solution as the I/O caching may be under-used.

A.2.3 Performance Expectations

A.2.3.1 ORACLE DBWR and disk I/O caching

There are two notable problems with using disk caching with Oracle's current DBWR algorithm. The first is that caching disk writes may not make Oracle run faster. Second, if

caching does succeed in making the DBWR run faster, it may actually slow down your ability to perform read transactions.

A.2.3.2 ORACLE LGWR and disk I/O caching

Log writer process writes out batches of transactions. The batching factor decreases if disk latency does. This algorithm can lead to a separate write for each transaction, which can consume an entire CPU if caching is efficient.

A.2.3.3 System-wide performance

The performance improvements attributable to a caching product are highly dependent on the following factors:

- Whether the product is hardware or software based.
- Whether the product is doing write-back or write-through caching.
- The size of the cache.
- The read vs. write mix on the system (heavy writes favor write-back caching, heavy reads favor write-through caching.)
- The locality of reference of the I/Os. The performance requirements for different applications on the same system.

A.2.4 Summary

Table 18 Disk caching recommendation

Type of caching	Control file	Database file	Redo Log file	Archive Log file
Write-Through	OK	OK	Avoid	Avoid
Write-Back, unprotected	Never	Never	Never	Never
Write-Back, protected	OK (1)	OK (1)	Avoid (2)	Avoid

Note

- (1) Oracle does not recommend write-back caching unless you have tested the ability of the cache to recover from system crashes.
- (2) Write-back caching can cause the LGWR to work too hard and consume an entire CPU.

Appendix B

Oracle Net SDU Tuning

B.1 Oracle Net SDU Tuning

The remote client server connection can lead to performance drops due to TCP use. Oracle Net provides flexibility by Session Data Unit (SDU) tuning.

This value represents the packet size at the service layer of the TCP model and affects the overall network traffic.

The SDU setting is however complicated due to an Oracle bug:

Bug.1113588 MAXIMUM SDU SIZE CANNOT BE SET FOR DEDICATED SERVER REGISTERED DYNAMICALLY.

This cookbook shows how to set the SDU and gives several bug workaround descriptions adapted to the *OpenView SQM* configurations.

B.1.1 Most Impacted Operation

Tests have shown that in *OpenView SQM* environments, the ‘purge’ operation is the most impacted due to the number of client server roundtrips and the volume of transferred data. Other operations can also be impacted but in far less significant proportions. **To optimize ‘purge’ operations SDU tuning is needed.**

B.1.2 How to Set SDU (ignoring the bug)

The default SDU value equals 2k and its maximum value is 32k. SDU is negotiated between the client and listener at connection time and the lowest configured value is retained. The SDU value needs to be set on the client side as well as at the listener level.

B.1.2.1 Client Side Configuration

The SDU parameter is part of the DESCRIPTION tag of a net service connection string:

The example below shows how to set SDU to 32k. This definition is stored in *$TEMIP_SC_VAR_HOME*/conf/tnsnames.ora

```
srm =
  (DESCRIPTION =
    (SDU=32767)
    (ADDRESS_LIST =
      (FAILOVER = OFF)
      (ADDRESS = (PROTOCOL = TCP)
                 (HOST=aigle.hp.net)(Port=1521))
    )
```

```
    (CONNECT_DATA =
      (SERVICE_NAME = srmsrv.hp.net)
    )
  )
```

B.1.2.2 Server side configuration

The SDU parameter is part of the static instance description stored in $ORACLE_HOME/network/admin/listener.ora.

The example below shows how to set SDU to 32k for a given instance.

```
(SID_DESC =
   (GLOBAL_DBNAME = srm.harrel.hp.net)
   (ORACLE_HOME = /usr/ORACLE/u01/app/oracle/product/9.2.0)
   (SDU=32767)
   (SID_NAME = srm)
 )
```

B.1.3 SDU Value Definition

Tests were performed to define the optimal SDU value, even if it depends on the application's behavior and on the network, a global conclusion is that bigger SDU values provide better performance. You should set SDU to its 32k maximum.

B.2 Bug Description

Once you have configured SDU on the client and server side, your setting should be effective (after listener restart and for new connections).

However, Oracle's service registration mechanism will overwrite your setting at the listener's level and revert to the default 2k value.

Several workarounds are described on Oracle's support web site.

Basically, two types of workarounds are suggested:

- Use MTS, the dispatcher can be declared with an additional SDU parameter that effectively works.
- Disable service registration.

B.3 Recommended Workarounds

- The MTS solution will not be described, only DPS should be implemented. It has been tested and works. Refer to Oracle resources for more details.

B.3.1 Workaround When no TAF is Needed

Write operations are always performed on a single instance master; *OpenView SQM* implements neither multi instance database nor multi-master replication. Therefore write operations never use TAF or CTF.

All net services describing a connection to a master for write operations can therefore specify an SDU and workaround the bug just by replacing the service name by the global database name as suggested above.

The workaround described is also applicable to read operations **if they are not configured to provide high availability with TAF or CTF.**

B.3.1.1 How to Proceed?

- Disabling service registration can be achieved by indicating a dummy listener to the database via the LOCAL_LISTENER parameter (in the DB init file) or by discarding the dynamically registered information from the client side.
- Discarding dynamic information is the simplest solution. You simply need to set the SERVICE_NAME tag value (in tnsname.ora) to the GLOBAL_DBNAME specified in the listener.ora file.
- With the example above, replacing:
- "(SERVICE_NAME = srmsrv.hp.net)" by
- "(SERVICE_NAME = srm.harrel.hp.net)" would be sufficient to take your SDU value into account.

```
srm =
  (DESCRIPTION =
    (SDU=32767)
    (ADDRESS_LIST =
      (FAILOVER = OFF)
      (ADDRESS = (PROTOCOL = TCP)
                 (HOST=harrel.hp.net)(Port=1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = srm.harrel.hp.net)
    )
  )
```

Appendix C

Automatic Listener and Instance start/stop

This appendix reproduces note: 1019790.6 available on Metalink.

This document discusses methods for automatically starting up and shutting down Oracle instances during UNIX

system startup or shutdown. Specific schemes for performing these actions on HP-UX are presented, along with a listing of the important scripts and directories that come into play in the process.
For more detailed explanations and alternatives, please refer to the System Administrator's Guide for HP-UX.
Search Words: dbstart, dbshut, HP
Solution Description:
==========
Files of Interest In Your Oracle Installation

The following list summarizes the functions performed by the different Oracle startup and shutdown scripts (which are invoked during system startup and shutdown, respectively).

Script	Description
$ORACLE_HOME/bin/dbstart	Ensures a clean startup of database instance(s), even after system failure
$ORACLE_HOME/bin/dbshut	Ensures a clean shutdown for data base instances
/etc/oratab or /var/opt/oracle/oratab	Contains a field that specifies whether a particular database instance should be brought up/down at system startup/shutdown time. By specifying "Y" in this field, the "dbstart" and "dbshut" scripts bring this database instance up or down.

System V Versus BSD UNIX Startup and Shutdown Procedures

Since the system re-boot procedure is unique for each Unix Operating System, but can be generalized based on whether the Operating System is System V-based or BSD-based, this section describes the generic approach taken by System V and BSD-based UNIX operating systems at system startup and shutdown time. For more specific information, refer to the appropriate operating system-specific section below.

System V Procedures
- - - - - - - - - -
System V initialization scripts are contained in /etc/rc<n>.d directories where "n" is the run-level value of the script. A run-level of 0 usually signifies power shutdown mode, while a run-level of 2 signifies multi-user mode.

The directories contain initialization scripts such as "S75cron" and "K30tcp".
These scripts are named using the following method:

[K or S][two-digit number][descriptive filename]

Names starting with "S" indicate scripts that are called at startup; names starting with "K" indicate scripts that are called at shutdown time.

Scripts containing larger numbers in their names are executed after those with lower numbers. Oracle startup scripts typically contain larger numbers in their names, such as "S99oracle", indicating that the script should be run after the system has been started up. Oracle shutdown script names, on the other hand, usually contain

smaller numbers, such as "K01oracle" indicating that the script should be run before system shutdown.

BSD Procedures
- - - - - - -
BSD-based systems, use /etc/rc*, files, such as /etc/rc, /etc/rc.local and so on, at system startup time.

During system shutdown, the /etc/shutdown command is invoked before any system or system-defined scripts. Some implementations invoke /etc/rc.shutdown at shutdown time.

HP-UX Version 9.0x:

Relevant Files
/etc/rc -- System startup file
/etc/shutdown -- System shutdown executable
/etc/shutdown.d -- System shutdown directory

All references to <oracle_owner> in this section should be replaced by the userid of the Oracle installation owner, and all references to <$ORACLE_HOME> should be replaced by the path to which $ORACLE_HOME points.

Unlike with Solaris and OSF/1, the HP/UX system startup and shutdown are relatively straightforward procedures. The /etc/rc script is executed at system startup. To provide automatic startup for Oracle, insert the following line immediately before the logical end of the script:

su <oracle_owner> -c <$ORACLE_HOME>/bin/dbstart

System shutdown is brought about by running the binary /etc/shutdown. This binary first executes all scripts and executables in the directory /etc/shutdown.d, then brings the system down normally.

Use any of the following methods to bring the Oracle instances down before the system is halted.

o Copy the $ORACLE_HOME/bin/dbshut script into /etc/shutdown.d.

o Make /etc/shutdown.d/dbshut a symbolic link to $ORACLE_HOME/bin/dbshut.

o Write a short script such as the one below:

#!/bin/sh

su <oracle_owner> -c <$ORACLE_HOME>/bin/dbshut

Make sure that the script is owned and executable only by the super user.

Note: if you are running Oracle instances from both Oracle version 6 and 7,
use the "dbshut" script included with the Oracle version 7.

HP-UX Version 10.x:

Relevant Files

/sbin/init.d/oracle -- contains the main script for doing db startup and shutdown

/etc/rc.config.d/oracle -- this file enables/disables automatic startup and shutdown of the databases

/sbin/rc1.d/K100oracle ->/sbin/init.d/oracle
-- this link tells the system when to execute the Oracle script with the
'stop' command in single user mode during system shutdown

/sbin/rc2.d/S990oracle ->/sbin/init.d/oracle
-- this link tells the system when to execute the Oracle script with the
'start' command in multi-user mode during system startup

All references to <oracle_owner> in this section should be replaced by the userid of the Oracle installation owner, and all references to <$ORACLE_HOME> should be replaced by the path to which $ORACLE_HOME points.

o Create an executable script /sbin/init.d/oracle for the startup
and shutdown of Oracle

Your script should attempt to start the database with the following command:

su <oracle_owner> -c <$ORACLE_HOME>/bin/dbstart

Your script should attempt to shut the database with the following command:

su <oracle_owner> -c <$ORACLE_HOME>/bin/dbshut

Make sure that the script is owned and executable only by the super user.

For an example of the complete script, refer to page 6-12 of the
7.2.2 Installation and Configuration Guide for HP 9000 series.

o Create the file /etc/rc.config.d/oracle

This file should contain:
ORACLE_START=1
export ORACLE_START

o Edit /etc/oratab

ORACLE_SID:ORACLE_HOME:Y|N

where Y or N indicates whether you want the dbstart and dbshut scripts to be run

o Make /sbin/rc1.d/K100oracle a symbolic link to /sbin/init.d/oracle
o Make /sbin/rc2.d/S990oracle a symbolic link to /sbin/init.d/oracle

For Oracle 7.3.2.1 there is a problem with dbstart and dbshut accessing ?/bin/sqldba instead of ?/bin/svrmgrl. To get around this problem simply do the following:

o Replace SQL*DBA with the following script:

```
---------- Start Of Script-------------------------
##########################################
# This script is provided to support the #
# 'dbstart' & 'dbshut' commands only.    #
##########################################
if [ $# = 1 ] && [ $1 = "command=exit" ] ; then
echo 'SQL\*DBA: Release 7.X.X.0.0 - Support Version \c' date
echo 'Copyright (c) Oracle Corporation 1996.  All rights reserved.'
exit 0
else
  #   Add LD_LIBRARY_PATH to work round other problems.
    LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
    export LD_LIBRARY_PATH
    exec $ORACLE_HOME/bin/svrmgrl $@
    fi

---------- End Of Script -------------------------
```

o Save this script as $ORACLE_HOME/bin/sqldba

o Add execute permission

```
$ chmod 755 sqldba
```

Note: Do not try to make a soft link. DBTART and DBSHUT will try to find the version of SQL*DBA which is provided by the above script.

Glossary

Metalink

Oracle’s web based knowledge and support database.

Tibco Repository

Central repository used to store the whole OpenView SQM platform configuration