Abstract

This white paper provides a description of a solution for backing up large SAP Oracle databases with EMC® Avamar® by using multiple channels with RMAN. It describes how to configure such a solution and how to integrate it with existing BR*TOOLS processes used in an SAP environment.

The purpose of this paper is to describe the proposed solution and to provide an in-depth review of the capabilities of the EMC Avamar Oracle plug-in that meets the challenges of backing up large SAP Oracle databases. General guidance for performance-tuning Oracle backups, including methodologies and sizing rules, are provided throughout as well.

December 2009
Table of Contents

Executive summary ................................................................. 4
Introduction ............................................................................. 5
Overview of EMC Avamar ....................................................... 6
Configuration ........................................................................... 11
Additional considerations ....................................................... 19
Conclusion ............................................................................... 27
References ............................................................................... 28
Appendix A: Avamar Oracle plug-in backup and recovery scripts ........................................ 29
Appendix B: Avamar Administrator Oracle GUI backups ................................................ 33
Appendix C: Avamar and Oracle theory of operations .................................................... 36
Appendix D: Troubleshooting, hints, and tips ......................................................... 39
Executive summary

Business case
This white paper covers concepts and procedures to enhance the performance and scalability of the EMC® Avamar® Oracle plug-in for large SAP systems running on Oracle databases. It also addresses the benefits of source-based deduplication for large SAP environments using EMC Avamar.

The best practices presented in this paper are based on testing done by EMC on large Oracle databases (both for SAP and non-SAP environments.) The main goal of the testing was to demonstrate that the Avamar Oracle plug-in addresses the data protection needs of these larger databases. For the results discussed in this paper, the test sample Oracle database was approximately 5 terabytes in size.

Business environment
Traditionally, large SAP environments have not been good candidates for source-based deduplication because of tight backup windows and the inherent CPU requirements that hashing algorithms consume on the production host. Busy, mission-critical SAP production servers simply cannot afford to allocate their computing power (and memory) to backup processes since these servers are exclusively tuned for SAP usage.

Key results
Multiple streams (allocated channels) provided by Oracle RMAN and its backup interfaces are used to reduce the length of the Avamar backup job to meet the backup window and achieve recovery time objectives. This approach is applicable when using SAP’s BR*TOOLS for performing backups as well as when using only Oracle RMAN to manage the backup job directly.

This approach enables SAP to benefit greatly from Avamar’s source-based deduplication capabilities while still performing these backups within their scheduled windows.
Introduction

Purpose
The solution described in this paper explains backup and recovery options for Oracle using Avamar in concert with Oracle's Recovery Manager (RMAN), leveraging the SBT_TAPE interface supplied by Avamar. The SBT interface is a channel that Oracle can use to stream data to a backup device. This interface is, from an Oracle perspective, provided as a third-party media manager where Avamar is the third party.

Although the approach described here does not use specific functionality or direct BR*TOOLS integration, its use of Oracle RMAN ensures that existing BR*TOOLS processes are viable and thus remain unchanged and supported by SAP.

Scope
The primary focus of this paper is to describe how to protect large SAP environments running on Oracle using the Avamar Oracle plug-in that leverages multiple streams support. The approach described in this paper follows the same guidelines described in the white paper Oracle Backup, Recovery, and Performance Tuning using EMC Avamar and Oracle RMAN from EMC. Refer to this paper for more details on the Oracle database architecture and how it affects the backup strategy for large Oracle databases in general.

It should be noted that Avamar complies with the standards that are set forth by Oracle and SAP. A shared library is installed into the Oracle environment allowing Oracle RMAN to invoke one or more SBT interfaces. In addition, there are several field-tested samples of scripts and configurations available.

Audience
This white paper is intended for SAP Basis managers, technicians, and database administrators (DBAs) who want to understand how to utilize Avamar for performing source-based deduplication of SAP Oracle databases. This group includes customers, backup administrators, DBAs, technical consultants, delivery engineers, technical support engineers, SAP Basis managers, and EMC internal professionals.

It is assumed that the reader has a high-level understanding of the architectural components of Oracle databases and available tools and approaches for making backups. This includes:

- Understanding what datafiles, control files, and redo logs are
  - Their role in performing a recovery of the database

- Oracle RMAN functionality

- Understanding Archive Log mode
  - How hot backups use archive logging to provide data consistency and how this differs from a cold backup

- RMAN integration/interaction with BR*TOOLS
Overview of EMC Avamar

Benefits of using EMC Avamar

The Avamar Oracle plug-in utilizes Avamar’s powerful source-based, global data deduplication technology.

Unlike traditional backup solutions, EMC Avamar identifies redundant data segments at the source (production server, and so on) — before they are transferred across the network. By moving only new, unique subfile segments found within the database, Avamar delivers fast daily full backups while reducing the required daily network bandwidth consumed by up to 500 times.

This capability allows companies to maximize existing network bandwidth for backup and disaster recovery of remote offices and data centers despite typically slow, congested, or underprovisioned networks and infrastructure. Data can be encrypted both in flight and at rest for added security. Avamar's centralized management also makes protecting hundreds of remote offices easy and efficient.

By storing just a single instance of each subfile data segment globally on the Grid, Avamar also reduces the total back-end storage footprint by up to 50 times, enabling cost-effective disk-based recovery over extended retention periods. Although Avamar is a backup to disk (B2D) technology, it also works with existing tape and traditional backup software such as EMC NetWorker.

In addition, EMC Avamar’s Grid architecture provides online scalability and uses a patented Redundant Array of Independent Nodes (RAIN) technology that delivers high availability while ensuring that a single storage node failure does not impact the availability of the Avamar Grid.

The value of deduplication is particularly evident for large SAP environments where there is not only a lot of redundancy within each database instance, but also between separate copies of the same systems (for example, Test, Quality Assurance, reporting copies). This typical redundancy, designed expressly for this diverse re-purposed usage, means that multiple non-production copies of a single production SAP system are superb candidates for source-based deduplication.

Avamar and large SAP databases

As mentioned earlier, using a source-based deduplication backup program such as Avamar for backing up large databases has traditionally been perceived as a challenge.

SAP environments pose quite a backup challenge, not only because of their large sizes, but also because of a number of other characteristics that make backing them up a fairly demanding task.

Some of these characteristics are explained in the table below.

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Details</th>
<th>Why it complicates the backup/recovery process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business data distributed over several systems</td>
<td>Different components of the SAP landscape (ERP, CRM, SCM, legacy systems) hold interrelated information that is relevant for a single business process.</td>
<td>A true recovery requires the synchronization of different databases and systems or additional “business recovery” efforts.</td>
</tr>
<tr>
<td>Redundant data</td>
<td>Non-normalized data structures across separate SAP components can result in repeated information that needs to be maintained by the different applications.</td>
<td>Same as above. Consistency is important. (However, this characteristic also increases the value of deduplicated backup approaches.)</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Different database vendors</td>
<td>Although this is not common, it’s possible to have different database platforms for the different components of an SAP system.</td>
<td>The approaches for each type of database might be different and incompatible to each other. Cannot usually manage in a centralized manner.</td>
</tr>
<tr>
<td>No common checkpoint</td>
<td>Unless federated consistency is achieved (through storage-level protection, for example), there is no common checkpoint at the application level to determine when/where all components of the landscape are in-sync.</td>
<td>Cannot guarantee federated consistency by backing up each database separately.</td>
</tr>
<tr>
<td>Data in file systems</td>
<td>Important data is present not only in the databases, but also in the file system (binaries, java file system, and so on)</td>
<td>A database backup by itself doesn’t protect the whole system. Also challenging to achieve consistency between a database and a file system.</td>
</tr>
</tbody>
</table>

Not all of the challenges above are addressed with an independent database-level backup strategy, but the backup/restore process for a single system is still the most important backup building block in an SAP environment.

The solution described in this paper addresses the concerns traditionally encountered when applying source-based deduplication to back up large, distributed SAP database environments. This paper does not address the requirement for federated backup nor does it cover applying consistency across different database instances, but the coverage here enables customers to take advantage of the benefits of deduplication, which alleviates some of the complexity described above.

Although multiple database instances in the SAP landscape will be protected separately by Avamar, deduplication will be realized throughout the entirety of the Avamar Grid and therefore throughout the whole SAP landscape. Each unique subfile will only be backed up to the Avamar Grid once. All SAP instances being backed up to the same Avamar Grid will derive the dedupe benefit. Only a fraction of net new data found in the second, third, and nth copies of each SAP instance will be backed up to Grid.
SAP BR*TOOLS and the SAP Oracle backup

BR*TOOLS is a set of tools available for SAP customers running on Oracle databases for backing up, restoring, and managing the storage requirements of their database. Any backup and recovery solution for SAP applications running on Oracle databases must adhere to the compatibility requirements for using the standard BR*TOOLS processes.

Some methods available for integrating third-party backup products with BR*TOOLS require the certification of the backup vendor’s implementation of certain SAP-specified interfaces. For the approach presented in this paper (where RMAN is used to call Avamar and provide multiple streams to aggregate channel throughput to perform the backup), there is no need for such certification.

Although more detail will be offered in the “Configuring BR*TOOLS” section, this approach uses a BR*TOOLS backup device type that allows Avamar to use RMAN to send/receive (restores) to an SBT backup library without necessitating the use of SAP’s BACKINT interface to complete the backup job.

A note on using RMAN’s recovery catalog

The BR*TOOLS process as explained in this paper does not use (or need) the RMAN recovery catalog (RCAT). BR*BACKUP automatically backs up the control file, profile, and log files after backing up the database files. It is important to include these files in the backup set defined for the SAP backup so they are protected along with their corresponding database files.

Note: For more information see SAP note #420698.

Considerations for backing up large databases

Any data protection discussion should focus on the customer’s business requirements. Chief among these are desired backup and recovery time objectives. Environmental factors also need to be considered, especially for databases larger than 500 GB. Key factors and questions to consider are the following:

- Database size and version
- Available CPU and memory (for the planned backup schedule)
- What type of disk subsystem is available?
- What type of Avamar environment is in place and how large is it?
- Was the Oracle database installed on the storage system using Oracle’s best practices?

By allocating multiple channels, and having the suitable environmental factors in place, Avamar is an excellent fit for large SAP/Oracle environments.

In order to preserve the CPU and memory utilization of the production server for productive processes, this solution can be combined with storage-based snapshot and replication functionality to perform the backup on a mount host dedicated for this task. For more information on this approach, please refer to EMC’s Proven Solutions documentation related to Backup and Recovery of SAP systems appropriate to your environment.

The typical range for this testing has been conducted on 4 TB and 5 TB Oracle databases. Avamar was deployed to back up the database within a typical 10-12 hour nightly backup window.
Following are two real-world SAP customer examples and their viability for Avamar backup:

**Scenario No. 1**

<table>
<thead>
<tr>
<th>Database Size</th>
<th>5 TB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operating System</strong></td>
<td>Sun Solaris</td>
</tr>
<tr>
<td><strong>Server Memory</strong></td>
<td>128 GB RAM (64 GB free)</td>
</tr>
<tr>
<td><strong>Server processors</strong></td>
<td>12 cores</td>
</tr>
<tr>
<td><strong>Storage System</strong></td>
<td>EMC Symmetrix DMX™</td>
</tr>
<tr>
<td><strong>Recovery Time Objective</strong></td>
<td>24 hours</td>
</tr>
<tr>
<td><strong>Backup Window</strong></td>
<td>12 hours</td>
</tr>
</tbody>
</table>

**Question:** Can the Avamar Oracle plug-in be leveraged to protect this configuration?

**Answer:** Avamar would be a good fit here. Allocate multiple RMAN channels to get the backups and restores done within their windows. This is the best play for this scenario.

**Scenario No. 2**

<table>
<thead>
<tr>
<th>Database Size</th>
<th>30 TB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAP System</strong></td>
<td>BI data warehouse</td>
</tr>
<tr>
<td><strong>Operating System</strong></td>
<td>IBM AIX</td>
</tr>
<tr>
<td><strong>Server Memory/processors</strong></td>
<td>Plenty</td>
</tr>
<tr>
<td><strong>Storage System</strong></td>
<td>Very fast</td>
</tr>
<tr>
<td><strong>Backup Window</strong></td>
<td>10-12 hours</td>
</tr>
</tbody>
</table>

**Question:** Can the Avamar Oracle plug-in be leveraged to protect this configuration?

**Answer:** Even with the approach presented in this paper, it would be very difficult to complete this backup within the standard 10-12 hour backup window because of the very large database size. Avamar is likely to not be a good fit here.

---

**Key features using RMAN with the Avamar Oracle plug-in**

The Avamar Oracle plug-in (with its SBT_TAPE library implementation) is a fully integrated third-party media management module that is designed to leverage Oracle’s RMAN transport of data to and from the third-party’s (Avamar’s) backup devices. By using RMAN, all of the key features of Oracle’s RMAN tool are available to Avamar.

RMAN is Oracle’s preferred method to back up and restore Oracle databases and benefits from many of Oracle’s latest technology extensions such as change block tracking (CBT). For SAP environments, this usually means configuring BR*TOOLS as a front end to the RMAN backup/restore job.

**Note:** For more information about using BR*TOOLS with RMAN, refer to the SAP white paper *BR*TOOLS with Oracle Recovery Manager.*
RMAN integrates with Avamar through an SBT device type ensuring that RMAN provides a consistent data stream(s) to (and from for restores) a third-party backup/SBT provider (Avamar).

Following is a short list of key RMAN features:

- Online backups – Full and Incremental
- Offline backups – Full and Incremental
- Archived redo log backups
- Dedicated Oracle database to track backups – recovery catalog (RCAT - not used in this solution)
- Oracle block-level corruption detection
- Restore capabilities
  - Full system restore
  - Datafile restore
  - Tablespace level restore
  - Point-in-time restore
  - Archived redo log restores

In more recent versions of Oracle, RMAN continues to introduce new advanced functionality. Please refer to Oracle documentation for a complete listing.
Configuration

Summary of steps

The following table summarizes the steps necessary to configure this solution.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prepare the target database</td>
</tr>
<tr>
<td>2</td>
<td>Install the Avamar operating system file system agent</td>
</tr>
<tr>
<td>3</td>
<td>Install the SBT library on the target database</td>
</tr>
<tr>
<td>4</td>
<td>Create users in the Avamar Grid and target database</td>
</tr>
<tr>
<td>5</td>
<td>Configure BR*TOOLS</td>
</tr>
<tr>
<td>6</td>
<td>Create the RMAN backup script</td>
</tr>
</tbody>
</table>

Step 1: Prepare the target database

The following are the steps involved in getting the target database ready for an Avamar backup using RMAN. The target database is the database to be backed up.

**Note:** Refer to the *EMC Avamar Oracle Client Installation Guide and Reference Manual* for more information about preparing the target database.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ensure that the Avamar File System client is installed and the client is registered to the Avamar Utility Node.</td>
</tr>
<tr>
<td>2</td>
<td>Install the Avamar SBT_TAPE library to the target database. This is outlined in detail in the <em>EMC Avamar Oracle Client Installation Guide and Reference Manual</em>.</td>
</tr>
<tr>
<td>3</td>
<td>Create a user within the target database and grant that user appropriate privileges. This same user must also be created on the Avamar utility node/Avamar Grid. Refer to “Step 4: Create users on the Avamar Grid and target database” for detailed information.</td>
</tr>
<tr>
<td>4</td>
<td>Ensure that the target database is currently in Archive Log mode. This is required to provide data consistency for hot backups.</td>
</tr>
<tr>
<td>5</td>
<td>Create an Oracle password file on the target database.</td>
</tr>
</tbody>
</table>
Step 2: Install the Avamar operating system file system agent

This key step ensures the Avamar client and SAP server/Oracle database are communicating.

The best practice is to test a small backup of the file system before beginning to test full-scale Oracle backups. It is also recommended to take a backup of the operating system data (file or file system) before attempting a backup and restore of Oracle data itself.

Note: The EMC Avamar Oracle Client Installation Guide and Reference Manual has more details on this step.

Step 3: Install the SBT library on the target database

Although very similar for each supported operating system, the common steps are documented below.

The installer installs the Avamar SBT_TAPE library into a directory such as /usr/local/avamar/lib on the client.

Note: Refer to the EMC Avamar Oracle Client Installation Guide and Reference Manual for specific installation instructions.

Depending on the operating system, it might be necessary to shut down the target database and then restart it to bind the SBT_TAPE library to the Oracle kernel.

Once this library (typically named libobk_avamar.so), is successfully installed, the Avamar RMAN script can then make the appropriate channel and stream calls.

Examples:
The following are example commands performed on an Oracle 10g database using Avamar version 4.x.

- Example of the installation on Red Hat.

  \$ rpm -hi AVAMARRMAN.rpm

- After installing this RPM, one can then log in to the database with SQL*Plus to start up the database.

  \$ oradba@oratest1# sqlplus /nolog
      connect / as sysdba
      shutdown immediate
      startup
      exit

This step (restarting the database) may be necessary to bind the Avamar SBT library to the Oracle kernel.
Step 4: Create users on the Avamar Grid and target database

This step creates users within Avamar and the target database and then grants correct privileges to these users.

1. Create a user called “backup” in the Avamar GUI

The following table provides information on creating a user within Avamar Administrator.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start Avamar Administrator. Choose <strong>Navigation &gt; Administration</strong> or Click the <strong>Administration launcher</strong> button.</td>
<td>The Administration window appears</td>
</tr>
<tr>
<td>2</td>
<td>Click the <strong>Account Management</strong> tab. In the tree, select the client you want to add the new user to.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Choose <strong>Actions &gt; Account Management &gt; New User(s)</strong>… or Click the toolbar icon shown immediately to the left. Note: Use the Axion Authentication system option; this is the default selection.</td>
<td>The New User dialog box appears.</td>
</tr>
<tr>
<td>4</td>
<td>Create a user, with the following credentials: <strong>User</strong>: backup <strong>Password</strong>: backup. (Type the password <strong>backup</strong> to confirm)</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If you need help with these steps, consult the *EMC Avamar System Administration Guide* for further examples.

2. Create a user in the target DB

Create a user named backup in the target database; grant this user the following privileges. The samples below show the proper syntax.

**Note:** This procedure may vary a little depending on which version of Oracle you are running. However, these examples document the high-level steps. Please check with your Oracle DBA if you find issues.

Continuing with UNIX examples, these command lines are run from the target system.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in to the Oracle database using SQL<em>Plus oradba@oratest1# sqlplus /nolog SQL</em>Plus: Release 9.2.0.1.0 - Production on Wed Apr 23 21:33:25 2003 Copyright (c) 1982, 2002, Oracle</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Action</td>
<td>SQL Command</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>Create a user</td>
<td><code>create user backup identified by backup;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>User created.</td>
</tr>
<tr>
<td>3</td>
<td>Unlock the user account</td>
<td><code>alter user backup account unlock;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oracle 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User altered</td>
</tr>
<tr>
<td>4</td>
<td>If necessary, identify the user’s default</td>
<td><code>alter user backup identified by backup DEFAULT TABLESPACE users;</code></td>
</tr>
<tr>
<td></td>
<td>default tablespace</td>
<td>User altered</td>
</tr>
<tr>
<td>5</td>
<td>Grant the sysdba privilege to the user</td>
<td><code>grant sysdba to backup;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Grant succeeded</td>
</tr>
<tr>
<td>6</td>
<td>Exit SQL*Plus</td>
<td><code>exit</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disconnected from Oracle9i Enterprise Edition Release 9.2.0.1.0 - Production</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With the Partitioning, OLAP and Oracle Data Mining options</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JServer Release 9.2.0.1.0 - Production</td>
</tr>
</tbody>
</table>
Step 5: Configure BR*TOOLS

In order for BR*TOOLS to use RMAN (with Avamar) to perform backups and restores, parameters in the init<SID>.sap initialization file (typically found in $ORACLE_HOME/dbs) also need to be configured. These parameters specify the method through which BR*TOOLS calls RMAN without the need to use the BACKINT interface for integration with third-party backup tools.

Note: If integration with other SAP-certified backup tools such as EMC NetWorker becomes necessary to manage a second-phase backup (with or without Avamar), then additional configuration of BR*TOOLS to use the BACKINT interface is required. This method still takes advantage of RMAN's leveraging of multiple streams.

init<SID>.sap parameters and required values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Use for</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup_dev_type</td>
<td>rman_disk</td>
<td>RMAN backup with an external SBT backup library without BACKINT with local disk</td>
</tr>
<tr>
<td></td>
<td>rman_stage</td>
<td>RMAN backup with an external SBT backup library without BACKINT with remote disk</td>
</tr>
<tr>
<td>rman_channels</td>
<td>&lt;number of channels&gt;</td>
<td>(Optional) BRBACKUP will pass this parameter to RMAN to determine the number of channels to be used for the backup. This value was set to 4 for the tests described in this document. See further discussion under the &quot;Performance tuning Oracle backups&quot; section.</td>
</tr>
</tbody>
</table>

Note: For more details on BR*TOOLS configuration for RMAN, see the SAP white paper BR*TOOLS with Oracle Recovery Manager.

Note: These settings copy the control file, profile, and log files to a local or remote disk. The best practice is to include these files in the backup job to ensure these vital files for recovery are transferred to a final backup medium. For more information see SAP note 420698.
Step 6: Create the RMAN backup script

The RMAN backup script needs to specify the SBT library used by RMAN to call Avamar backup.

The following is an example RMAN backup script. Note that the Avamar-specific lines are in blue.

```sql
run
{
allocate channel T1 type 'SBT_TAPE'
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,
ENV=(PATH=/bin:/usr/bin)"
send '"--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt"';
backup database;
release channel T1;
}
```

The Avamar-specific lines in the RMAN script do the following:

This line directs RMAN to the correct location for the libobk sbt library.

```
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,
ENV=(PATH=/bin:/usr/bin)"
```

This call should be done on a single line as shown above.

The "ENV=(PATH=/bin:/usr/bin)" sets the environment variable so that the Avamar script can find its "avtar" daemon and accompanying "uname" credentials.

This line provides with the correct Avamar avtar flagfile directives. The my-avtar-flags.txt at the end of this line contains additional Avamar backup calls:

```
send '"--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt"';
```

Universal parts of the RMAN script:

```
run
{
allocate channel T1 type 'SBT_TAPE'
backup database;
release channel T1;
}
```

A number of other backup and recovery options can be added to this script; in fact, most DBAs have their own RMAN scripts already. The only net new action required when using legacy scripts is to ensure the PARMS= line makes the correct calls and that the send line for the flagfile directives is appended to their script. The actual my-avtar-flags.txt file will also need to be created in its location - /usr/local/avamar/lib for this example (described further in the section below).
RMAN backup script execution and sample output

These samples are performed on an Oracle 9i system with EMC NetWorker for demonstration purposes. In this example, RMAN is being called from the command line directly (BR*BACKUP is not being used in this example).

```
oracle@legato1% rman cmdfile backup_database.rman
Recovery Manager: Release 9.2.0.1.0 - Production
Copyright (c) 1995, 2002, Oracle Corporation. All rights reserved.
RMAN> connect target rman/rman@eval
2> connect catalog rman/rman@rman
3> run {
4> allocate channel T1 type 'SBT_TAPE';
5> backup database plus archivelog delete input;
6> }
7>
connected to target database: EVAL (DBID=1928152189)
connected to recovery catalog database
allocated channel: T1
channel T1: sid=12 devtype=SBT_TAPE
channel T1: NMO v4.0.0.0
Starting backup at 23-APR-03
current log archived
channel T1: starting archive log backupset
channel T1: specifying archive log(s) in backup set
input archive log thread=1 sequence=6 recid=1 stamp=492128357
channel T1: starting piece 1 at 23-APR-03
channel T1: finished piece 1 at 23-APR-03
piece handle=04elai38_1_1 comment=API Version 2.0,MMS Version 4.0.0.0
channel T1: backup set complete, elapsed time: 00:00:25
channel T1: deleting archive log(s)
archive log
filename=/space/home/oracle/product/9.2.0/dbs/arch1_6.db recid=1
stamp=492128357
Finished backup at 23-APR-03
Starting backup at 23-APR-03
channel T1: starting full datafile backupset
channel T1: specifying datafile(s) in backupset
including current SPFILE in backupset
including current controlfile in backupset
input datafile fno=00001
name=/space/home/oracle/oradata/eval/system01.dbf
input datafile fno=00002
name=/space/home/oracle/oradata/eval/undotbs01.dbf
input datafile fno=00005
name=/space/home/oracle/oradata/eval/example01.dbf
input datafile fno=00011
name=/space/home/oracle/oradata/eval/test1a.dbf
input datafile fno=00012
```
name=/space/home/oracle/oradata/eval/test1b.dbf
channel T1: starting piece 1 at 23-APR-03
channel T1: finished piece 1 at 23-APR-03
piece handle=05elai44_1_1 comment=API Version 2.0,MMS Version 4.0.0.0
channel T1: backup set complete, elapsed time: 00:01:26
Finished backup at 23-APR-03

Document Number: 1-1033-01 NetWorker Module for Oracle 8i and 9i on Solaris Evaluation Guide 15
channel T1: backup set complete, elapsed time: 00:00:07
channel T1: deleting archive log(s)
archive log
filename=/space/home/oracle/product/9.2.0/dbs/arch1_7.dbf recid=2 stamp=492128476
Finished backup at 23-APR-03
released channel: T1
Recovery Manager complete.

My-Avtar-Flags.txt

This file contains the “flags” that will be passed to the Avamar backup process when the RMAN backup script gets executed.

The following is a sample copy of the My-Avtar-Flags file with flag descriptions and values.

```
--debug
--pidname=Oracle
--pidnum=1002
--logfile=/usr/local/avamar/var/avtar.log
--vardir=/usr/local/avamar/var
--id=backup
--ap=backup
--path=/clients/vaerp-db4.va.company.org
```

Where these commands do the following:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>--id</td>
<td>Username within the Avamar utility node</td>
</tr>
<tr>
<td>--ap</td>
<td>Password for the Avamar user</td>
</tr>
<tr>
<td>--pidnum</td>
<td>Number for the OS of the target database</td>
</tr>
<tr>
<td></td>
<td>Linux 1002  &lt;--  line 3</td>
</tr>
<tr>
<td></td>
<td>Solaris 2002</td>
</tr>
<tr>
<td></td>
<td>Windows 3002</td>
</tr>
<tr>
<td></td>
<td>HP-UX 4002</td>
</tr>
<tr>
<td></td>
<td>AIX 5002</td>
</tr>
<tr>
<td>--path</td>
<td>Avamar domain and client of the target database</td>
</tr>
</tbody>
</table>

Back Up Large SAP Oracle Databases Using EMC Avamar and Oracle RMAN
A Detailed Review
Additional considerations

Introduction
This section provides further guidance on performance tuning and configuration of the Avamar architecture for performing Oracle backups. No SAP-specific issues are discussed in this section.

Performance tuning Oracle backups

Identifying the client-side physical system bottleneck
When architecting a scalable and robust Oracle backup and recovery solution, an identification of the bottlenecks within that Oracle database, particularly one supporting an SAP environment, is of critical importance. These are likely to exist with typical dependencies on the module mix, daily/month-end/quarter-end load, workload, and a number of other variables. A clear identification of the environmental limitations that exist in each backup target is essential to sizing, and more importantly, not having backup operations impact SAP operations in any way, shape, or form.

A number of variables are set and managed inside the database itself. One of the main determinants of database performance is the System Global Area (SGA) size. SGA size is where the main RMAN I/O originates.

Note: SGA tuning is typically the job of the DBA and is beyond the scope of this paper. A classic reference book that covers this tuning in detail is Oracle Database 10G RMAN Backup & Recovery by Matthew Hart and Robert Freeman.

Physical bottleneck
The focus here is the Avamar backup client server hosting the Oracle database.

Every system built has a slowest component that translates into a bottleneck or constraint. Typical among these are CPU, RAM, and disk I/O to name just a few. The goal is to identify bottleneck(s) to maximize the efficiency of the backup and recovery jobs while minimizing the impact to the Oracle database. It is common to expose and address one bottleneck, only to encounter another.

The identification process is outlined in the following sections. The following table shows some examples of physical server bottlenecks. The backups and recoveries will go only as fast as that bottleneck permits, in effect, acting as a hard ceiling.

<table>
<thead>
<tr>
<th>Operation type</th>
<th>Physical component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup</td>
<td>Memory</td>
</tr>
<tr>
<td>Backup</td>
<td>CPU</td>
</tr>
<tr>
<td>Backup</td>
<td>I/O Subsystem</td>
</tr>
<tr>
<td>Backup</td>
<td>Network – (First Backup)</td>
</tr>
<tr>
<td>Restores</td>
<td>Network</td>
</tr>
<tr>
<td>Restores</td>
<td>I/O Subsystem</td>
</tr>
</tbody>
</table>

Avamar processes Oracle backups as fast as memory and CPU permit. Note that Avamar’s initial first backup will appear to utilize more network and server resources since the full effect of deduplication is not realized until subsequent day-over-day backups. These latter backups process faster and thus utilize fewer resources as the local Avamar cache files allocated to file system hashing and chunking become fully populated, decreasing client workload as each hash and chunk content address is stored.
During backups, one critical I/O subsystem metric is read performance. If this metric is exposed as the bottleneck, there are a number of remedial actions to improve read and seek performance. The combinations of bottleneck/remedies are many and thus beyond the scope of this paper, but EMC’s vast experience at customer sites can be leveraged to address these scenarios, not to mention the ever-intrepid DBA, server, and network teams that work in concert to address just such performance issues.

With that said, typical database backup performance ranges from approximately 60 to 100 GB/hour per backup stream.

Backup throughput has been observed in excess of 200 GB/hour per stream. EMC has, in numerous cases, observed a linear increase in backup performance as multiple parallel backup channels/streams are added to back up the database.

**Using multiple RMAN channels**

Allocating multiple RMAN channels to boost the overall backup performance is the strategy to pursue, under the following conditions:

- If the I/O subsystem is well architected and can support multiple concurrent backup streams (each capable of streaming backup data in excess of 100 GB/hour)

  **Note:** "Well architected" in this context means that backing up the data through multiple streams should not create contention in the I/O subsystem, which would slow down overall backup performance.

- If the host has multiple processors available during the backup window.
  By default, the Avamar client will distribute the SHA-1 hashing and compression out evenly to all processors on the client. However, the main avtar thread is mapped to a single processor, and on some clients with slower individual processors, this could be a potential bottleneck.

  In this case of multiple processors, though, allocating multiple RMAN channels creates multiple avtar processes, each of which will have a main thread that is mapped to a different processor, thus boosting overall performance on a one avtar-to-one-cpu basis.

- As described in the later section “Avamar cache sizing for databases,” the maximum amount of memory consumed by the Avamar process to optimally perform the Avamar backups of the database data will be approximately 0.001 (1/1,000) of the total size of the database.

  For example, for Avamar to back up a 5 TB Oracle database, the maximum amount of memory consumed will be approximately 5 GB total for all concurrent streams.

Although the examples cited later in this paper are based on using four RMAN channels to back up the Oracle database data, the optimum number of required RMAN channels needs to be determined based on the following factors:

- The optimum number of separate backup streams that can be supported by the I/O subsystem before excessive contention impacts performance or diminishing returns are encountered.
• The number of processors available. CPU utilization will tend to continue to improve as more channels are added, until the number of channels equals the number of processors.

• As mentioned earlier, the amount of memory that needs to be allocated for the backup process is independent of the number of channels that are used to perform the backup.

• The goal is to allocate RMAN channels so that the average amount of data per channel does not exceed 1.5 TB.

Based on current lab testing and field observations, the Avamar/Oracle recovery should go as fast as the network allows. This throughput refers to the client-side network interface - testing to date has been confined to a single GigE interface. A second client-side network interface would reduce this bottleneck proportionally, meaning roughly a doubling of network throughput.

The case for multiple RMAN channels

Once bottlenecks have been identified, it is best to test a single channel RMAN script for the backup and recovery of the database. A single RMAN stream runs in the range of 60-100 GB per hour. Depending on your SAP environment, this single stream throughput may be adequate to satisfy the backup and recovery time objectives for the database.

For larger databases, they will require the backup and restores to run at faster speeds to satisfy the backup and recovery time objectives.

For example, a 2 TB database requires at least 21 hours to restore if using a single stream. In the same setting, a 4 TB database would require at least 42 hours.

Allocating multiple channels is the play here, with this methodology being equivalent to generating multiple streams/avtar processes.

Avamar cache sizing for databases

Note: For more background information about the client cache files, please refer to the EMC Avamar Operational Best Practices manual.

A correctly tuned hash cache file allows for backups to run much faster than a non-tuned file. It should be noted that this tuning does not impact restores (p_cache.dat).

For database backups, having a properly sized hash cache file, namely p_cache.dat, is key to maximizing backup throughput. The hash cache stores the SHA-1 hashes of the variable size chunks and hash composites that have been sent to the Avamar Grid. The hash cache is used to quickly identify which chunks or composites have previously been backed up and already reside on the Avamar Grid.

Therefore, for the 2 TB database example cited earlier, this means that only ~6 GB of new chunked data is sent from client to Grid on a daily basis. Compare this to the data change rate, say 10 percent, equating to ~200 GB that a traditional backup product's incremental would have to send over the wire.

Hash cache sizing is critical since undersizing is failure to maximize, but oversizing is the absolute practice to avoid since it can impact a database server powering a SAP environment:

- **Undersized** (Excessive random seeks) - With an undersized cache size, overall backup performance slows down. For instance, if the avtar process finds that a hash of a chunk is not contained in its local hash cache, then it queries the Avamar
Grid to see if it “is present.” Although these hash "lookups" on the Avamar Grid are only 20 bytes, it requires random seeks, which negatively impact overall backup performance.

- **Oversized** (Inefficient memory usage) - In this scenario, the hash caches, loaded into the client’s memory at the beginning of the backup, consume more memory than they require. This, in turn, could cause memory swapping on the client, thus slowing down backup performance. In a worst-case scenario, this condition could even cause backups (or the SAP application) to fail due to memory allocation failures.

**Note:** Optimal sizing of the hash caches is critical to backing up Oracle databases using multiple RMAN channels. If they are undersized, then the combined effect of multiple avtar processes querying the Avamar Grid for the presence of hashes of chunks previously backed up could cause the Avamar Grid to become the bottleneck due to excessive random seeks, slowing down overall performance.

Overallocation of memory for hash caches exposes a condition where the client could run out of memory resources to support all concurrent avtar streams, not to mention its application/database load.

---

**Rules for sizing hash caches properly**

To size the hash caches appropriately, the following rules apply:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>By default, the hash cache could consume up to 1/16th of the physical RAM on the Avamar client.</td>
</tr>
<tr>
<td>2</td>
<td>If the hash cache files already exist and are larger than the required size, the cache files (p_cache.dat) need to be deleted and then re-created for resizing to a correct smaller size.</td>
</tr>
</tbody>
</table>
| 3    | The hash cache doubles in size each time it needs to grow.  
The current hash cache sizes are:  
- 24 MB  
- 48 MB  
- 96 MB  
- 192 MB  
- 384 MB  
- 768 MB  
- 1,536 MB  
At this time, the maximum hash cache size that is supported is 1,536 MB. |
| 4    | The hash cache includes only one 20-byte SHA-1 hash per variable size chunk (this is a hash of the contents of the chunk by scanning database records) plus an additional 4 bytes overhead per chunk.  
Therefore, each entry in the hash cache consumes 24 bytes per entry. |
| 5    | During Oracle backups, the average chunk size (before compression) is 24 KB per chunk. |
| 6    | To control the size of the hash cache used for a dataset, use the "--hashcachemax<>" option. The value needed for this attribute is a "not to exceed" size of the hash cache file (expressed in MB per rule 3). |
For example, setting "--hashcachemax=500" allows the hash cache file to grow to 384 MB (where 384 MB is the largest hash cache file that is less than the 500 hashcachemax above).

From rules 3,4,5, and 6, the following rule can be derived for setting the "hashcachemax" option:

\[ --hashcachemax = 2 \times \text{MB} \times \text{Size of database in GB} \]

For a database size of 700 GB, the formula would be:

\[ \text{Hash Cache Max should be less than } 2 \times 700 \text{ MB (GB to MB conversion)} = 1400 \text{ MB} \]

### Examples of properly sized hash cache files

The following are examples of how to optimize the "hashcachemax" option and the resulting size of each hash cache size:

#### Sample hash cache sizes based on stream size and hashcachemax

<table>
<thead>
<tr>
<th>Stream size</th>
<th>Hashcachemax</th>
<th>Hash cache size</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 GB DB Stream</td>
<td>--hashcachemax=20</td>
<td>24 MB</td>
</tr>
<tr>
<td>100 GB DB Stream</td>
<td>--hashcachemax=200</td>
<td>192 MB</td>
</tr>
<tr>
<td>500 GB DB Stream</td>
<td>--hashcachemax=1000</td>
<td>768 MB</td>
</tr>
<tr>
<td>700 GB DB Stream</td>
<td>--hashcachemax=1400</td>
<td>768 MB</td>
</tr>
<tr>
<td>1000 GB DB Stream</td>
<td>--hashcachemax=2000</td>
<td>1536 MB</td>
</tr>
<tr>
<td>1536 GB DB Stream</td>
<td>--hashcachemax=3072</td>
<td>1536 MB (MAXIMUM)</td>
</tr>
</tbody>
</table>

For larger dataset sizes, the maximum hash cache will likely be 1,536 - this is presently the maximum hash cache size.

This means that a dataset over 2 TB using a single hash cache will be undersized and thus slow down due to excessive random seeks on the Grid. A 2 TB dataset with a 1,536 MB hash cache size does not have a large enough hash cache to support this size. Any time cache of any type is not hit is the beginning of a performance problem.

Here is where Avamar's support for multiple streams becomes relevant.

To use this feature, specify the "--hashcachemax=<>>" parameter in the my-avtar-flags file.

Ensure that the original p_cache.dat file is renamed so that the next Avamar backup builds a new one.

When allocating multiple RMAN channels to back up the Oracle database, each channel processes roughly the same amount of data. This uniform distribution means that one can specify the same "--hashcachemax=<2 x amount of data per RMAN channel (GB)>>" in the my-avtar-flags.txt file.

When backing up large databases (>2 TB databases), it is important to minimize the amount of memory consumed on the host. In this case, the amount of data backed up per RMAN channel should be between ~750 GB to ~1.5 TB. This range ensures that Avamar can effectively utilize a 768 MB or 1,536 MB hash cache file with minimal wasted space in this file.
If the cache files are fairly close to these values, then this should suffice - major deviations from these rules carry the penalties of under- or oversized. As mentioned earlier, oversizing is to be avoided at all costs since this can impact the database host and its availability as a data source to the SAP environment.

Example scripts and multiple Avamar cache files

When architecting an Oracle backup and recovery leveraging multiple RMAN channels using Avamar, the use of multiple sets of Avamar cache files is best practice. This distribution is very similar to the method used when multiple cache files are invoked to spawn multiple avtar processes for very dense file system backups.

Allocation of multiple avtar processes will be covered later in this paper. Its use is greatly simplified by using the Avamar "--cacheprefix" flag, an avtar flag created expressly for this scenario.

Note: Read more about the "--cacheprefix" flag in EMC Avamar Operational Best Practices.

Using this option ensures that only a few changes are needed to allow for multiple cache files.

The cache files are f_cache and p_cache. These files reside in the directory /usr/local/avamar/var.

Multiple channel RMAN Avamar backup script

```bash
run {
    configure controlfile autobackup on;
    set controlfile autobackup format for device type sbt to "CONTROLFILE.%F";
    allocate channel c1 type sbt
    PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar64.so
            ENV=(PATH=/bin:/usr/local/avamar/bin)";
    allocate channel c2 type sbt
    PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar64.so
            ENV=(PATH=/bin:/usr/local/avamar/bin)";
    allocate channel c3 type sbt
    PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar64.so
            ENV=(PATH=/bin:/usr/local/avamar/bin)";
    allocate channel c4 type sbt
    PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar64.so
            ENV=(PATH=/bin:/usr/local/avamar/bin)";
    set controlfile autobackup format for device type sbt to "CONTROLFILE.%F";
    send channel='c1' '"--flagfile=/scripts/my-avtar-flags.txt" "--
cacheprefix=c1" "--
logfile=/usr/local/avamar/var/c1_avoracle.log"';
    send channel='c2' '"--flagfile=/scripts/my-avtar-flags.txt" "--
cacheprefix=c2" "--
logfile=/usr/local/avamar/var/c2_avoracle.log"';
    send channel='c3' '"--flagfile=/scripts/my-avtar-flags.txt" "--
cacheprefix=c3" "--
logfile=/usr/local/avamar/var/c3_avoracle.log"';
    send channel='c4' '"--flagfile=/scripts/my-avtar-flags.txt" "--
cacheprefix=c4" "--
logfile=/usr/local/avamar/var/c4_avoracle.log"';
    backup database plus archivelog;
    release channel c1;
    release channel c2;
    release channel c3;
    release channel c4;
}
```
The blue text outlines the unique parts of this script. This is essentially how to address the multiple channels and map them to the multiple cache files. The \texttt{-cacheprefix=c(n)} creates a unique set of cache files for that channel. There are four in this example.

The \texttt{-logfile=} creates a unique set of log files for each channel.

All of these files are stored in the standard \texttt{/usr/local/avamar/var} directory.

**Multiple channel RMAN restore script**

```
run {
configure controlfile autobackup on;
set controlfile autobackup format for device type sbt to "CONTROLFILE.%F";
allocate channel c1 type sbt
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar64.so
ENV=(PATH=/bin:/usr/local/avamar/bin)";
allocate channel c2 type sbt
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar64.so
ENV=(PATH=/bin:/usr/local/avamar/bin)";
allocate channel c3 type sbt
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar64.so
ENV=(PATH=/bin:/usr/local/avamar/bin)";
allocate channel c4 type sbt
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar64.so
ENV=(PATH=/bin:/usr/local/avamar/bin)";
set controlfile autobackup format for device type sbt to "CONTROLFILE.%F"
send channel='c1' '"--flagfile=/scripts/my-avtar-flags.txt "--logfile=/usr/local/avamar/var/c1_avoracle.log"';
send channel='c2' '"--flagfile=/scripts/my-avtar-flags.txt "--logfile=/usr/local/avamar/var/c2_avoracle.log"';
send channel='c3' '"--flagfile=/scripts/my-avtar-flags.txt "--logfile=/usr/local/avamar/var/c3_avoracle.log"';
send channel='c4' '"--flagfile=/scripts/my-avtar-flags.txt "--logfile=/usr/local/avamar/var/c4_avoracle.log"'';
restore database;
recover database;
release channel c1;
release channel c2;
release channel c3;
release channel c4;
}
```

This script is used for restore. The fact that it looks very much like the backup script is part of its design. When running any restore script, make sure the DBA is involved and understands the restore and recovery of the Oracle database. There are many restore scenarios and the DBA needs to be aware of the state of the database before beginning.

The main benefit of this recovery is that it leveraged multiple streams for the restore, much like how the backup was performed.

**Multiple channel Avamar My-Avtar-Flags file**

```
--hashcachemax=<2 x amount of data per RMAN channel (in GB)>
--expires=30
--server=avamar220.1ss.emc.com
--pidname=Oracle
--pidnum=1002
--logfile=/usr/local/avamar/var/avtar.log
```
Similar to our earlier examples, the `--hashcachemax` flag has been added to accommodate the large database files. The balance of the script is standard RMAN calls.

By using the `--cache prefix` option, the modularity of the `my-avtar-flags` file in the script ensures that each channel reads this flag file and processes it accordingly.
Conclusion

This white paper demonstrates how Avamar can be used as a viable solution for backing up and restoring large SAP environments.

Avamar’s ability to allocate and write backup data to multiple RMAN channels is the central method to meeting backup and recovery time objectives for suitable datasets.

Use of existing BR*TOOLS processes to run RMAN-configured jobs for SAP benefit greatly from Avamar’s source-based deduplication capabilities while still performing these backups within their scheduled windows while not impacting the host or its database availability.
References

White papers
For additional information, see the white papers listed below.

- SAP white paper: *BR*Tools with Oracle Recovery Manager* (available from SAP Service Marketplace)
- EMC Best Practices Planning white paper: *Oracle Backup, Recovery, and Performance Tuning using EMC Avamar and Oracle RMAN* (available on Powerlink)

Product documentation
For additional information, see the product document listed below.

- *EMC Avamar Oracle Client Installation Guide and Reference Manual*

Other documentation
For additional information, see the documents listed below.

- SAP Note #105047: Support for Oracle functions in the SAP environment.
- SAP Note #113677: Recovery catalog and backups with RMAN
- SAP Note #420698: Backups with MCNFS without BACKINT
- SAP Presentation: *Backup & Restore Concepts for mySAP System Landscapes* (this collateral plus other backup and recovery documentation is available from the Advance Technology Group at [http://service.sap.com/atg](http://service.sap.com/atg))
- *Oracle Database 10G RMAN Backup & Recovery* by Matthew Hart and Robert Freeman
Appendix A: Avamar Oracle plug-in backup and recovery scripts

Introduction
This section shows examples of scripts used for various backup and recovery scenarios.

Examples

**Back up all and delete the logs**
connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat

run
{
  allocate channel T1 type 'SBT_TAPE'
  PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/bin:/usr/bin);
  send "--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt'';
  backup database plus archivelog delete input;
  release channel T1;
}

**Crosscheck archive logs**
connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat

run
{
  allocate channel T1 type 'SBT_TAPE'
  PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/bin:/usr/bin);
  send "--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt'';
  crosscheck archivelog all;
  release channel T1;
}

**Back up the datafile**
connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat

{
  allocate channel T1 type 'SBT_TAPE'
  PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/bin:/usr/bin);
  send "--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt'';
  backup datafile '/d1/dev3data/usr01.dbf';
  release channel T1;
}

**Recover the datafile**
connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat

run
{
  allocate channel T1 type 'SBT_TAPE'
  PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/bin:/usr/bin);
  send "--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt'';
  recover datafile '/d1/dev3data/usr01.dbf';
  release channel T1;
}
bin:/usr/bin";
    send "--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt";
    restore datafile '/d1/dev3data/usr01.dbf' ;
    recover datafile '/d1/dev3data/usr01.dbf';
    release channel T1;
}

Back up the tablespace
connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat

run {
    allocate channel T1 type 'SBT_TAPE'
    PARM="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=
    (PATH=/bin:/usr/bin)");
    send "--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt";
    backup tablespace DEMO;
    release channel T1;
}

Recover the tablespace
connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat

run {
    allocate channel T1 type 'SBT_TAPE'
    PARM="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=
    (PATH=/bin:/usr/bin)");
    send "--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt";
    restore tablespace DEMO;
    recover tablespace DEMO;
    sql "alter tablespace DEMO online";
    release channel T1;
}

Restore archive log all
run
{
    allocate channel T1 type 'SBT_TAPE'
    PARM="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/
    bin:/usr/bin)");
    send "--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt";
    restore archivelog all ;
    release channel T1;
}

Disaster recovery using Autobacked up control files
connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat

    set DBID = 92755261
run {
allocate channel T1 type 'SBT_TAPE'
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/bin:/usr/bin)"
send '"--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt"';
#restore spfile;
restore spfile from autobackup;
release channel T1;
} shutdown immediate;
startup nomount;

Disaster recovery restoring a control file

run {
allocate channel T1 type 'SBT_TAPE'
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/bin:/usr/bin)"
send '"--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt"';
#restore controlfile;
restore controlfile from autobackup;
alter database mount;
recover database;
release channel T1;
}

Simpler DR example of incomplete recovery

connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat
connect target /
set DBID = 92755261
startup force nomount;
run {
allocate channel T1 type 'SBT_TAPE'
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/bin:/usr/bin)"
send '"--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt"';
#restore controlfile;
restore controlfile from autobackup;
alter database mount;
recover database;
alter database open resetlogs;
}

Perform cold backup using RMAN

connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat
run {

shutdown immediate;
startup mount;
allocate channel T1 type 'SBT_TAPE'
PARMS="SBT_LIBRARY=/usr/local/avamar/lib/libobk_avamar.so,ENV=(PATH=/bin:/usr/bin)"
send '"--flagfile=/usr/local/avamar/lib/my-avtar-flags.txt"'
backup
(Database format 'df_t$t_s$s_s$p')
release channel t1;
sql 'Alter Database Open';
}

Oracle on Windows example

connect target backup/backup@evaldb
connect catalog backup/backup@rcvcat

run {
  configure controlfile autobackup on;
  allocate channel c1 type sbt
  PARMS="SBT_LIBRARY=C:\PROGRA~1\avs\bin\orasbt.dll,ENV=(PATH=C:\PROGRA~1\avs\bin)"
  format '%d_%U';
  set controlfile autobackup format for device type sbt to "CONTROLFILE.DUBLIN.%F";
  send '"--flagfile=C:\TEMP\FlagFile.txt' --debug';
  backup database plus archivelog delete input;
}

Notice the short pathname for SBT_LIBRARY=C:\PROGRA~1\avs\bin\orasbt.dll

Flagfile located at C:\TEMP\FlagFile.txt:
--debug
--pidname=Oracle
--pidnum=3002
--logfile=C:\TEMP\avtar.log
--vardir="C:\Program Files\avs\var"
--id=root
--ap=8RtoTriz
--path=\clients\vm-qasys-02
### Appendix B: Avamar Administrator Oracle GUI backups

**Snapup options** The options in the following figure can be specified during on-demand Oracle database backup operations, or within the datasets. They are explained below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Oracle Instance Name</strong></td>
<td>specifies the Oracle server name. This field can normally be left blank because the Oracle instance name is automatically stored when the</td>
</tr>
<tr>
<td>User name</td>
<td>target client is browsed during on-demand snapup or when a dataset is applied to a member of a group during a scheduled snapup.</td>
</tr>
<tr>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>Delete Archive Logs after backup</td>
<td>if selected, automatically deletes Oracle archive logs immediately following a successful database backup.</td>
</tr>
<tr>
<td>Enhanced Commonality Factoring</td>
<td>enables or disables enhanced data deduplication. During snapups, enhanced data deduplication typically reduces the amount of client data that</td>
</tr>
<tr>
<td></td>
<td>must be sent to the server but requires additional client CPU resources. Choices are:</td>
</tr>
<tr>
<td></td>
<td>- Disabled</td>
</tr>
<tr>
<td></td>
<td>- Default (use the global enhanced data deduplication setting already set on the server)</td>
</tr>
<tr>
<td></td>
<td>- Enabled</td>
</tr>
<tr>
<td>Enable debugging messages</td>
<td>if selected, adds all available debugging output to both avtar and avoracle. This is useful when submitting a program.</td>
</tr>
</tbody>
</table>

- **Oracle Instance Name** specifies the Oracle server name. This field can normally be left blank because the Oracle instance name is automatically stored when the target client is browsed during on-demand snapup or when a dataset is applied to a member of a group during a scheduled snapup.

- **User name** specifies the username used to authenticate with the Oracle database. If this field is left blank, RMAN tries to log in with the same username and password that the Avamar client agent is running under, and attempts to assume SYSDBA privileges. Typically, this field should contain the special account name (backupuser) manually created as part of database preparation. Username and Password comprise a connection string to Oracle. The connection string must specify a user who has backup privileges for the database.

- **Password** specifies the password for the username account.

- The **Delete Archive Logs after backup** checkbox, if selected, automatically deletes Oracle archive logs immediately following a successful database backup.

- **Enhanced Commonality Factoring** enables or disables enhanced data deduplication. During snapups, enhanced data deduplication typically reduces the amount of client data that must be sent to the server but requires additional client CPU resources. Choices are:
  - Disabled
  - Default (use the global enhanced data deduplication setting already set on the server)
  - Enabled

- The **Enable debugging messages** checkbox, if selected, adds all available debugging output to both avtar and avoracle. This is useful when submitting a program.
problem to EMC Technical Support so that an issue can be more easily diagnosed.

- The **Use Recovery Catalog** checkbox is followed by three recovery catalog fields. The Recovery Catalog Server Name, Recovery Catalog Username and Recovery Catalog Password settings are used to specify a Recovery Catalog Server connection string for RMAN. By using a Recovery Catalog Server, many specialized features of RMAN may be used. A thorough discussion of these features is beyond the scope of this paper.

- **Recovery Catalog Server Name** specifies the recovery catalog server name.

- **Recovery Catalog Username** specifies the recovery catalog username.

- **Recovery Catalog password** specifies the recovery catalog password.

### Restore options

The options in the following figure can be specified during the restore operations. They are explained below.

- **Oracle Instance Name** specifies the Oracle server name. This field can normally be left blank because the Oracle instance name is automatically stored when the target client is browsed during restore operations.

- **User name** specifies the username used to authenticate with the Oracle database. If the field is left blank, RMAN tries to log in with the same username and password the Avamar client agent is running under, and attempts to assume SYSDBA privileges. Typically, this field should contain the special account name (backupuser) manually created as part of database preparation. Username and Password comprise a connection string to Oracle. The connection string must specify a user that has backup privileges for the database.

- **Password** specifies the password for the username account.
• **Recovery Until SCN** is for the recovery portion, so you can play back archive log files until the transaction with this SCN is reached. If this field is left blank, **avoracle** will scan through the known backups in the backup catalog (either in the system control file or the recovery catalog) and find the highest SCN associated with a backup file.

• The **Enable debugging messages** checkbox, if selected, adds all available debugging output to both **avtar** and **avoracle**. This is useful when submitting a problem to EMC Technical Support so that an issue can be more easily diagnosed.

• The **Only restore control file** checkbox, if selected, restores only the control file, and known backups in the backup catalog (either in the system control file or the recovery catalog) are listed. This is useful when the user wants to find an SCN for a database or archive log without actually performing the restore.

• **Restore Command Options** is for additional command line options that will be passed to the RMAN restore command.

• **Recover Command Options** is for additional command line options that will be passed to the RMAN **recover** command.


---

**Troubleshooting GUI operations**

If you are encountering issues with browsing your database(s) within the GUI, this section will help you identify what the issue is.

Avoracle looks at the oratab file for database names and then cross-references those names to the tnsnames file. Local databases that resolve this way are then presented in the browser window for selection for backups. Avoracle defaults to the `$ORACLE_HOME/network/admin` directory. So, make sure the tnsnames file is in the `$ORACLE_HOME/network/admin` directory.

If these instructions do not resolve the browse issue, do the following:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create the file <code>/opt/AVMRclnt/var/avoracle.cmd</code>.</td>
</tr>
</tbody>
</table>
| 2    | Add the following two lines to the file:  
  • `debug`  
  • `logfile=/opt/AVMRclnt/var/avoracle_browse.log` |
| 3    | Try to browse again. The browse attempt will be logged in `avoracle_browse.log`.  
| 4    | Look in the `avoracle_browse.log` file and you will be able to identify the error you have and take the necessary steps to correct it. |
Appendix C: Avamar and Oracle theory of operations

Avoracle theory of operations

Avoracle is normally invoked whenever any operations on the Oracle database are carried out through the Avamar Administrator. The three operations supported by Avoracle are:

- Browse
- Backups
- Restores

While the functionality of Backup and Restore should be fairly obvious, Browse is an intermediate and not-so-visible operation.

When an Avamar Oracle operation is selected in Avamar Administrator, it sends a message to the concerned client node. This message is called a workorder, and it contains all the information a client needs to perform an operation.

Communication between the Avamar Administrator and the client node is always initiated through Avagent. Avagent runs as a daemon (service) and listens for messages from the Administrator.

Avagent inspects the workorder received, and uses the information to decide which of the Avamar clients needs to be invoked. If the operation is meant for the Oracle plug-in, Avagent starts Avoracle and passes it the workorder.

Browse

The first step in starting a backup of an Oracle database is the Browse operation.

When the Oracle plug-in is selected in the Backup panel of Avamar Administrator, it sends a Browse request to Avoracle (through Avagent). This triggers a database discovery process in Avoracle.

Avoracle reads /etc/oratab to get a list of databases on the system and their home directories. Using the Oracle home directories it then looks for the tnsnames.ora file.

The special fields of interest in tnsnames.ora are HOST and SERVICE. Using this, Avoracle generates a list of databases that are local to the system. This list is then returned to Avamar Administrator in a message. Avamar Administrator then uses this list to populate the Backup panel.

Backup

When one or more databases are selected for backup, a new message is sent to Avoracle. This message contains the list of databases to be backed up, together with any options that the user specifies. The message also contains credentials that Avoracle can use to communicate with the Avamar Server.

In the current implementation Avoracle can only do backups and restores sequentially. So, even if multiple databases have been specified, they are done one by one. The following description is for one database. For multiple databases the
sequence is just repeated.

Avoracle uses RMAN and Oracle’s SBT feature for taking backups. For this, Avoracle specifies a loadable library that can communicate with the Avamar Server. This library implements certain entry points that have been written to Oracle’s specification. The library path and some other parameters are communicated to RMAN through a script that Avoracle creates.

In Oracle 10g and later, Oracle has a restriction that prohibits RMAN from running as “root”. Avoracle scripts are run as the default user “oracle” unless an alternate user id has been specified.

The RMAN script, credentials, and logfiles are created and stored in the Avamar “var” directory. Then avoracle creates another process and starts RMAN, telling it what script to use. Avoracle then goes into a loop waiting for a connection from avtar (more on this later) or for RMAN to complete its operations and exit.

Once RMAN starts, it communicates with Oracle and tells it which library needs to be loaded. Oracle loads this library (which is actually an Avoracle component) and uses it for backup. After this point the backup process is controlled by RMAN. RMAN, by default, backs up archive logs, the database file, and the control file. Based on certain factors RMAN splits up the entire backup into one of more backup pieces. For each backup piece (which is actually opaque to Avoracle), RMAN begins by communicating the piece name to libobk. It then performs the write using multiple calls to libobk, ending the operation with a close command.

When libobk receives the backup command it creates an avtar process. The name of the backup piece is communicated to avtar together with some other information.

Avtar is also told how to connect with the already running Avoracle. This last step is required so that avtar can get the workorder from Avoracle and be able to send it status updates.

Avtar establishes connection with the Avamar Server and then waits for data from libobk. Every time the write entry point of libobk is called by Oracle, libobk takes the data and send it to avtar.

Avtar will process the data (chunking and hashing) and store it on the server. The backup piece is stored on the server with the same name that RMAN had generated. This makes it easier to find the right file during restores. Once all the data for a backup piece has been received avtar terminates and libobk returns control to RMAN.

After the entire database has been backed, RMAN unloads libobk and exits, returning control to Avoracle.

**Restores**

The Restore process is similar to the Backup process except for the direction of data flow. The Restore browse panel is populated by the Avamar Administrator and it lets the user select one or more backups to be restored.

Avoracle receives this list and does preprocessing on it to extract the control file names. For every control file that has been selected for restore the following happens.
Avoracle first creates an RMAN script for restoring just the control file. It starts RMAN with the script (everything else is the same as the backup operation). RMAN loads libobk and restores the control file. If the user has not specified the SCN for restore, Avoracle examines the RMAN logfile and extracts the latest SCN corresponding to the control file selected.

Avoracle then creates a new script for recovery of the database files. At this point RMAN has mounted the database using the control files. Once again RMAN is started up with the proper script, and libobk is loaded. Based on the SCN specified and the backup history in the control file, RMAN decides which backup pieces need to be restored. These piece names are communicated to libobk, which in turn talks to avtar. Avtar gets these files from the Avamar Server and the data is sent back to RMAN through the corresponding entry point in libobk. Once all the required files have been restored, RMAN does a recover operation, opens the database, and exits.

Following is a graphical representation of the entire process.
Appendix D: Troubleshooting, hints, and tips

Introduction

The following are some important factors for diagnosing the Media Management Library:

- For 64-bit Oracle versus a 32-bit library, verify that the "bitness" is the same.
- Must pass the correct PATH, such that RMAN can initiate "avtar" and "uname".
- Must set the right environment variable such that the Avamar library can be located and loaded.
- Different platforms (Solaris, HP-UX, Linux, AIX, and Windows) sometimes use different environment variable names for the library.
- Different shells (cmd, bash, sh, csh) use different syntax for setting and exporting environment variables.
- Avamar software, config, and log location (--vardir, Library, logs) vary by platform.

The PATH and ENVIRONMENT are important because of the complex process structure involved:

1. Avagent receives a workorder, and invokes Avoracle.
2. Avoracle invokes (via script) "rman".
3. RMAN invokes "oracle".
4. Oracle invokes "libok_avamar" (Media Management Library)
5. The library invokes avtar.

The first step can be short-cut, by invoking "rman" directly from a command line.
However, the PATH and ENVIRONMENT are still critical.

Diagnostic commands: A sample session

Locate avtar and uname. Their locations will need to be added to the PATH.

which avtar
   /usr/bin/avtar

which uname
   /usr/bin/uname

Ensure the flag file is set up correctly.

cat /opt/AVMRclnt/bin/my_avtar_flags.txt

--debug
--pidname=Oracle
--pidnum=2002
--logfile=/usr/local/avamar/var/avtar.log
--vardir=/usr/local/avamar/var
--id=backup
--ap=backup
--path=/clients/ux210.fairisaac.com

Note that on a Solaris platform, --vardir and --logfile must point to /opt/AVMRclnt/var, and "c:\program files\avs\var" or the equivalent for Windows. The /usr/local/avamar/var is used in the case of Linux systems.

Ensure that the Media Management Library exists and has the expected size, bitness, and version. (The following commands are for Solaris64.)

ls -la /opt/AVMRclnt/lib/

total 5684

 drwxr-xr-x  2 root  root  512 Jan 23 10:47 ./
 drwxr-xr-x  5 root  root  512 Jan 23 09:06 ../
-rwlr-xr-x  1 root  root  528320 Sep 13 13:47 libgcc_s.so.1*
-rwxlr-xr-x  1 root  root  55556 Sep 13 13:47 libobk_avamar.so*
-rwlr-xr-x  1 root  root  66280 Sep 13 13:47 libobk_avamar64.so*
-rwlr-xr-x  1 root  root  2232464 Sep 13 13:45 libstdc++64.so.2.10.0*

file /opt/AVMRclnt/lib/libobk_avamar64.so
/opt/AVMRclnt/lib/libobk_avamar64.so: ELF 64-bit MSB dynamic
lib SPARCV9 Version 1, dynamically linked, not stripped

file /usr/oracle/product/9.2.0.8.64bit/bin/oracle
/usr/oracle/product/9.2.0.8.64bit/bin/oracle: ELF 64-bit MSB executable SPARCV9 Version 1, dynamically linked, not stripped

ls -l $ORACLE_HOME/lib/libok.so
/usr/oracle/product/9.2.0.8.64bit/lib/libok.so: No such file or directory

Try loading the library using the oracle "SBTTEST" (again for Solaris64, use bash). The library path and command to set the appropriate variable have the following variations:

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Environment variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, Solaris</td>
<td>LD_LIBRARY_PATH</td>
</tr>
<tr>
<td>HP-UX</td>
<td>SHLIB_PATH</td>
</tr>
<tr>
<td>AIX</td>
<td>LIBPATH</td>
</tr>
</tbody>
</table>

The following are the commands to set the variable changes, depending on the shell:

<table>
<thead>
<tr>
<th>Shell</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>bash, sh</td>
<td>LD_LIBRARY_PATH=&quot;$LD_LIBRARY_PATH&quot;:/opt/AVMRcint/lib export LD_LIBRARY_PATH</td>
</tr>
<tr>
<td>csh</td>
<td>setenv LD_LIBRARY_PATH &quot;$LD_LIBRARY_PATH&quot;:/opt/AVMRcint/lib</td>
</tr>
</tbody>
</table>

```
echo Test file contents > testfile

set env LD_LIBRARY_PATH=/opt/AVMRcint/lib

echo $LD_LIBRARY_PATH

/usr/oracle/product/9.2.0.8.64bit/lib:/usr/lib:/usr/dt/lib:/usr/openv
/netbackup/bin

set env LD_LIBRARY_PATH=/opt/AVMRcint/lib

echo $LD_LIBRARY_PATH

/usr/oracle/product/9.2.0.8.64bit/lib:/usr/lib:/usr/dt/lib:/usr/openv
/netbackup/bin

sbttest testfile -libname /opt/AVMRcint/lib/libobk_avamar64.so

/opt/AVMRcint/lib/libobk_avamar64.so could not be loaded.
```

Check that it is installed properly, and that the LD_LIBRARY_PATH environment variable (or its equivalent on your platform) includes the directory where this file can
be found. Additional error information is listed below:

```
ld.so.1: sbttest: fatal: libgcc_s.so.1: open failed: No such file or directory
```

**Retry, with the right environment variable:**

```
export LD_LIBRARY_PATH=/opt/AVMRclnt/lib

echo $LD_LIBRARY_PATH

/opt/AVMRclnt/lib

sbttest testfile -libname /opt/AVMRclnt/lib/libobk_avamar64.so
```

The `sbt` function pointers are loaded from `/opt/AVMRclnt/lib/libobk_avamar64.so` library.

```
Avamar    : sbtinit V1

Init: Proxy disabled

-- sbtinit succeeded

Avamar    : sbtinit V1

Init: Proxy disabled

-- sbtinit (2nd time) succeeded

sbtinit: Media manager supports SBT API version 2.0

sbtinit: vendor description string=EMC|Avamar (avtar backup)

sbtinit: allocated sbt context area of 4672 bytes

sbtinit: Media manager is version 4.0.0.37

Avamar    : sbtinit2

-- sbtinit2 succeeded
```
-- regular_backup_restore starts ................................

Avamar    : sbtbackup

   : file=testfile

   : block_size=16384

Child process started

Avamar    : Waiting for avtar to start ... sleep 10

avtar Warning <0000>: Cannot create log /opt/AVMRclnt/var/avtar-20090128_074710-00001.log

avtar Info <5551>: Command Line: /opt/AVMRclnt/bin/avtar.bin --vardir=/opt/AVMRclnt/var --bindir=/opt/AVMRclnt/bin

   --sysdir=/opt/AVMRclnt/etc --

   label=testfile -c testfile --from-stdin --check-stdin- path=false

avtar FATAL <5086>: No server address specified. Reinstall software or add

   --server= address.

Avamar    : avtar exited

MMAPI error from sbtbackup: 7501, sbtbackup: avtar exited, trying to create
testfile

   -- sbtbackup failed

rm testfile

Set LD_LIBRARY_PATH (Or equivalent on HPUX/AIX)

export LD_LIBRARY_PATH=/opt/AVMRclnt/lib

Then re-running the RMAN command.

rman
Recovery Manager: Release 9.2.0.8.0 - 64bit Production

Copyright (c) 1995, 2002, Oracle Corporation. All rights reserved.

RMAN> connect target /

connected to target database: IVVINT0 (DBID=3840860340)

RMAN> connect catalog rman920/rman920@rmnp0

connected to recovery catalog database

RMAN> run

{ allocate channel T1 type 'SBT_TAPE'
  PARMS="SBT_LIBRARY=/opt/AVMRclnt/lib/libobk_avamar64.so,ENV=(PATH=/bin:/usr/bin:/opt/AVMRclnt/bin)";

  send '"--flagfile=/opt/AVMRclnt/bin/my_avtar_flags.txt"';

  backup database plus archivelog delete input;

  release channel T1;

}

Avamar : sbtinit V1

trace_file=/usr/oracle/admin/ivvint0/udump/sbtio.log

Init: Proxy disabled

Avamar : sbtinit2

allocated channel: T1

channel T1: sid=17 devtype=SBT_TAPE

channel T1: EMC|Avamar (avtar backup)
Avamar    : sbtcommand = "--flagfile=/opt/AVMRclnt/bin/my_avtar_flags.txt"

sent command to channel: T1 Starting backup at 2009-01-28:07:36:31

current log archived

channel T1: starting archive log backupset

channel T1: specifying archive log(s) in backup set

input archive log thread=1 sequence=1931 recid=1922 stamp=676951918

--Snip--

channel T1: starting piece 1 at 2009-01-28:07:36:35

Avamar    : sbtbackup

: file=83k5u2c2_1_1

: block_size=262144

Child process started

Avamar    : Waiting for avtar to start ... sleep 10

--Snip--

avtar Info <5551>: Command Line: /opt/AVMRclnt/bin/avtar.bin -- 
vardir=/opt/AVMRclnt/var --bindir=/opt/AVMRclnt/bin 

|--sysdir=/opt/AVMRclnt/etc --
flagfile=/opt/AVMRclnt/bin/my_avtar_flags.txt --debug --pidname=Oracle --

pidnum=2002

|--logfile=/usr/local/avamar/var/avtar.log --
vardir=/usr/local/avamar/var --id=backup --password=******

|--account=/clients/ux210.fairisaac.com --label=83k5u2c2_1_1 -c
83k5u2c2_1_1 --from-stdin --check-stdin-path=false

2009/01/28-13:36:35.85013 [avtar]  sock::libinit(enc=, ciphers=, verify=0)
socktype=sock_tcp

-- Snip --

Recovery Manager complete