Oracle® Intelligent Agent

User's Guide

Release 8.1.5

February, 1999

Part No. A67825-02

ORACLE

Oracle® Intelligent Agent User's Guide, Release 8.1.5

Part No. A67825-02

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the Programs.

Contents

1 Intelligent Agent Overview

Oracle Intelligent Agent: An Overview 1-2
Characteristics 1-3
Job and Event Support 1-3
Simple Network Management Protocol Support 1-3
Service Discovery Process 1-4
Agent Discovery Process for NT 1-4
Agent Discovery Process for UNIX 1-5

2 Installation and Configuration

Installing the Intelligent Agent on Windows NT 2-2
Creating a Windows NT User Account for Running Jobs 2-2
Configuring a Domain User as Your Agent User 2-3
Controlling Operations of the NT Agent 2-4
Installing the Oracle Intelligent Agent on UNIX 2-5
Running the root.sh Shell Script 2-6
Controlling Operations of the UNIX Agent 2-7
Configuring SNMP for UNIX or Windows NT 2-7
Oracle Intelligent Agent and Oracle Names 2-8
Roles and Users Required by the Agent 2-9
Auto-Discovery 2-10
Pre-requisites for Auto-Discovery 2-11

3 Troubleshooting

Troubleshooting the Intelligent Agent 3-2
Quick Checks 3-2
Quick Checks for the Windows NT Agent 3-2
Quick Checks for UNIX Agents 3-4
Questions and Answers 3-5
Is TCP/IP configured and running correctly? 3-6
Do the DNS Name and the Computer Name Match? (Windows NT) 3-8
Are the Net8 configuration files correct? 3-9
Is Net8 functioning properly? 3-10
Did the Agent startup successfully? 3-11
Did the Agent connect to ALL instances on its node? 3-13
Is the Agent running with the correct permissions? (UNIX) 3-13
Does the OS user exist and does it have the correct permissions? (Windows NT) 3-13
Are you still using a 7.3.3 or earlier Agent? 3-14
Are there errors? 3-14
Why doesn't the Agent send status notifications back to the Enterprise Manager Console even though the jobs have run? 3-14
Intelligent Agent Startup Problems and Solutions 3-15
Generic Agent 3-15
NT Agent 3-20
Unix Agent 3-23
Intelligent Agent Error Messages and Resolutions 3-24
Generic Agent 3-24
NT Agent 3-26
UNIX Agent 3-27
Tracing the Agent 3-29
Tracing TCL 3-30

4 The Agent Data Gatherer

Configuring the Agent Data Gatherer 4-2
Description of the Data Gathering Service 4-3
How the Data Gathering Service is Installed 4-3
Understanding Where to Configure the Oracle Data Gatherer to Collect Database and Operating System Data 4-3

Running the Oracle Data Gatherer on an Intermediate Host 4-4
How to Configure the Data Gatherer 4-5
Upgrading From a Previous Version of the Data Gatherer when You Install the New Data Gatherer into a Different Oracle Home 4-5
Controlling Operations of the NT and UNIX Data Gathering Service 4-6
Additional Methods for Controlling Operations on Windows NT 4-7
Verify Data Gathering Service is Running 4-8
Troubleshooting the Configuration of the Oracle Data Gatherer 4-9
Obtaining Trace Information on the Oracle Data Gatherer 4-10

5 Job and Event Scripts

Scripting Language 5-2
Tcl Language Description 5-2
OraTcl Description 5-4
Server Message and Error Information 5-6
oramsg Elements 5-6
Use of Tcl with the Intelligent Agent 5-10
NLS Issues and Error Messages 5-11
OraTcl Functions and Parameters 5-12
Common Parameters 5-13
catfile 5-13
concatname 5-14
convertin 5-14
convertout 5-15
diskusage 5-15
echofile 5-16
export 5-16
import 5-16
loader 5-17
msgtxt 5-17
msgtxt1 5-18
mvfile 5-18
oraautocom 5-19
oracancel 5-19
oraclose 5-20

oracols 5-20
oracommit 5-20
oradbsnmp 5-21
orafail 5-21
orafetch 5-22
oragetfile 5-23
orainfo 5-24
orajobstat 5-25
oralogoff 5-25
oralogon 5-26
oraopen 5-26
oraplexec 5-27
orareadlong 5-27
orareportevent 5-28
oraroll 5-30
orasleep 5-30
orasnmp 5-31
orasql 5-32
orastart 5-33
orastop 5-33
oratime 5-34
orawritelong 5-34
rmfile 5-34
tempdir 5-35
tempfile 5-35

A Configuration Files

Configuration Files 2
snmp_ro.ora 2
snmp_rw.ora 2
services.ora 2
Parameters for snmp_ro.ora and snmp_rw.ora Files 3

Send Us Your Comments

Oracle Intelligent Agent User's Guide, Release 8.1.5

Part No. A67825-02

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways

- electronic mail - infosmp@us.oracle.com
- FAX - (650) 506-7200. Attn: Oracle System Management Products
- postal service
  Oracle Corporation
  Oracle System Management Products Documentation Manager
  500 Oracle Parkway
  Redwood Shores, CA 94065 U.S.A.

If you would like a reply, please give your name, address, and telephone number below.

Preface

Purpose of this Guide

This manual provides configuration information and answers to crucial troubleshooting questions pertaining to the Oracle Intelligent Agent. The Intelligent Agent User's Guide is directed towards users of Oracle Enterprise Manager Version 1 and Version 2, as well as other supported system management frameworks that communicate with the Oracle database through the Intelligent Agent.

Important: Enterprise Manager Version 2 uses a middle-tier Management Server. Users of Enterprise Manager Version 1 should ignore references to this component.

Intended Audience

This manual is intended for anyone installing, configuring, or troubleshooting the Oracle Intelligent Agent on UNIX or Windows NT platforms. Under most circumstances, the Agent requires little in the way of configuration and maintenance. For this reason, the Oracle Intelligent Agent User's Guide should be used as a reference, rather than read sequentially.

How this Manual is Oraganized

This manual contains five chapters and one appendix:

Chapter 1 Provides a functional overview to the Intelligent Agent and discovery services.

Chapter 2	Describes Agent installation and configuration procedures.
Chapter 3	Provides Intelligent Agent troubleshooting guidelines and procedures.
Chapter 4	Describes Data Gatherer installation and configuration.
Chapter 5	Describes Job and Event scripting using the Tool Command Language (Tcl).
Appendix A	Describes requisite configuration files used by Oracle Enterprise Manager.

Related Documents

For more information, see the following documents:

- *Oracle Enterprise Manager Configuration Guide*
- *Oracle SNMP Reference Guide*

1

Intelligent Agent Overview

This chapter provides a brief overview of the Intelligent Agent. The following topics are covered:

- Oracle Intelligent Agent: An Overview
- Service Discovery Process

Oracle Intelligent Agent: An Overview

The Oracle Intelligent Agent is an autonomous process running on a remote node in the network. The Agent resides on the same node as the service it supports. However, the Agent can support more than one service on a particular node. For example, if two databases are installed on one machine, a single Agent can support both databases. The Agent is responsible for:

- Providing local services or calling operating system dependent services to interact locally with the managed targets.
- Accepting jobs or events from the Oracle Management Server or other third-party applications.
- Running Oracle Enterprise Manager jobs, collecting their results and output, and/or queuing the results as required.
- Checking for events, and queueing the resulting event reports for Oracle Enterprise Manager.
- Canceling jobs or events as directed by the Console or other applications.
- Handling Simple Network Management Protocol (SNMP) requests, if SNMP is supported on the Agent's platform.

For information on configuring the Agent, see the Oracle server platform-specific installation documentation for your system.

Note: Enterprise Manager version 1 does not use version 2's middle-tier Management Server. Because this manual is geared towards users of both version 1 and version 2, users of Enterprise Manager version 1 should ignore references to the Management Server.

Characteristics

Intelligent Agents are autonomous because they function without requiring that the Console or Management Server be running. An Agent that services a database can run when the database is down, allowing the Agent to start up or shut down the database. The Intelligent Agents can independently perform administrative job tasks at any time, without active participation by the administrator. Similarly, the Agents can autonomously detect and react to events, allowing them to monitor the system and execute a fixit job to correct problems without the intervention of the administrator.

The Agents operate independently of the Console and Management Server and are able to execute jobs and monitor events when the administrator has logged out of the Console. The Agents queue any job or event messages destined for that administrator, and deliver them to the Management Server. When the administrator logs in to a Console again, the Management Server delivers pending messages to the administrator who is currently logged in. Information about the state of jobs and events are stored in files on the Agent's node. These files have a ".q" extension and are stored in the `$ORACLE_HOME/network/agent` directory.

Job and Event Support

Jobs and events are implemented as Tcl scripts. When the Agent executes a job or tests for an event, it runs the appropriate Tcl script.

When the Management Server sends a message to an Agent on behalf of an administrator logged into the Console, it also sends the information about the administrator's language and character set environment. The Agent uses the NLS environment information when it performs database administration tasks on behalf of the administrator. This allows administrators to manage databases in their native languages. For example, an administrator in France can administer a database in Germany and receive messages in French.

Simple Network Management Protocol Support

The Agent supports Simple Network Management Protocol (SNMP), allowing third-party systems management frameworks to use SNMP to receive SNMP traps directly from the Agent. The Agent provides access to Oracle's database Management Information Base (MIB) variables. You can submit jobs or events that access Oracle MIB variables even when the database resides on a platform that does not support SNMP. For more information on SNMP, see the *Oracle SNMP Support Reference Guide*.

Service Discovery Process

When you start the Agent, the first operation it must perform is to discover what services exist on the node that it monitors. The following "discovery" algorithms document the service discovery process for the two most common platforms on which the Agent runs.

Agent Discovery Process for NT

At Agent startup, a script is executed which reads configuration parameters from the Windows NT registry, the `listener.ora` file, and the `tnsnames.ora` file (if it exists).

The Agent discovers new services on the machine where it is installed and creates / rewrites / appends to its configuration files: `snmp_ro.ora`, `snmp_rw.ora`, and `services.ora`.

To determine what services are available on its machine (services that the Agent will manage), the Agent uses the following discovery algorithm:

1. The Agent records the ORACLE_SID and ORACLE_HOME information for each database service found in the Windows NT Registry.
2. Based on the values found in the Windows NT registry, the Agent reads the `listener.ora` files to determine which listeners service which databases. The location of the `listener.ora` configuration files is based on the SQL*Net configuration file locations. For example, the `TNS_ADMIN` environment variable and the location of the `$ORACLE_HOME/network/admin` directory are based on the ORACLE_HOME information found in the Windows NT registry.
3. The name of the discovered databases is based on the `GLOBAL_DBNAME` parameter defined in the `listener.ora` file for that database.
4. If `GLOBAL_DBNAME` parameters are not found in `listener.ora`, the Agent searches for a `tnsnames.ora` file using the same search methodology used to find the `listener.ora` file.
5. If the `tnsnames.ora` file is not found, the database alias, `<SID>_<hostnames>`, is assigned to a database service. The service will be known to the Agent by this alias, and it will be visible as such at the Oracle Enterprise Manager Console.

Note: If multiple aliases exist for the same instance in the TNSNAMES.ORA file, the Agent will use the one listed first. If you prefer to use a different alias, re-order the TNSNAMES.ORA file entries and restart the Agent.

If a database or any other new service is installed on the node where the Agent resides, the Agent must be restarted to add the new service to the Agent configuration files. This procedure also applies to UNIX versions of the Intelligent Agent.

Agent Discovery Process for UNIX

At startup, the Agent discovers new services on the machine where it is installed and creates its configuration files: snmp_ro.ora, snmp_rw.ora, and services.ora.

To determine what services are available on its machine (services that the Agent will manage), the Agent uses the following discovery algorithm

1. The Agent reads the oratab file for values of all the Oracle Homes and SIDs. Depending on the platform, the oratab file can be located in either of the following locations:
   - /etc
   - /var/opt/oracle
2. Based on the Oracle Homes values found in oratab, the Agent searches for the listener.ora files to determine which databases are serviced by which listeners.
3. The name of the discovered databases is based on the GLOBAL_DBNAME parameter defined in the listener.ora file for that database.
4. If GLOBAL_DBNAME parameters are not found in listener.ora, the Agent searches for a tnsnames.ora file using the same search methodology used to find the listener.ora file.
5. If the tnsnames.ora file is not found, the database alias, <SID>_<hostnames>, is assigned to a database service. The service will be known to the Agent by this alias, and it will be visible as such at the Oracle Enterprise Manager Console.

2

Installation and Configuration

This chapter covers generic setup and configuration procedures for the Intelligent Agent. The following topics are discussed:

- Installing the Intelligent Agent on Windows NT
- Controlling Operations of the NT Agent
- Installing the Oracle Intelligent Agent on UNIX
- Configuring SNMP for UNIX or Windows NT
- Oracle Intelligent Agent and Oracle Names
- Roles and Users Required by the Agent
- Auto-Discovery

Installing the Intelligent Agent on Windows NT

Intelligent Agents are shipped with the database and installed on remote, managed machines under an ORACLE_HOME environment. The Agent runs as a service.

The convention used to construct the service name is:

`Oracle<ORACLE_HOME_NAME>Agent`

where ORACLE_HOME_NAME is the name of the ORACLE_HOME you specified during installation.

For more information on installing the Intelligent Agent, please refer to the *Oracle Enterprise Manager Installation* (CD-ROM insert).

Creating a Windows NT User Account for Running Jobs

In order for the Agent to execute jobs on a managed node

- an NT user account must exist that has the advanced user privilege, "logon as batch job." The privilege can be assigned to an existing local or domain user, or a new NT user.
- the preferred credentials for the node must be set for that user in the Oracle Enterprise Manager console. For more information on setting preferred credentials, see the *Oracle Enterprise Manager Administrator's Guide*.
- the user that starts the Agent must have read/write permissions to `ORACLE_HOME\network` directory as well as write permissions to the `TEMP` directory or the `ORACLE_HOME` directory.

> **Note:** If you do not set up the "logon as batch job" privilege, you will receive the "Failed to authenticate user" message when you run jobs on the node.

- the user must have administrator privileges in order to start up and shut down Windows NT services, such as databases and listeners.

Please follow one of the procedures listed below.

Creating a New NT User Account

To create a new Windows NT user account on the local NT machine and grant the "log in as batch jobs" privilege to this user, perform the procedure below.

1. Select the User Manager from the Administrative Tools program group. See the Windows NT documentation for information on this tool.
2. Select New User from the User menu and check for the following:
   - The "User Must Change Password at the Next Logon" option box is not checked
   - "SYSTEM" or "system" is not used for the user name.
3. Under the Policies menu of the User Manager NT utility, select the User Rights option.
4. Check the "Show Advanced User Rights" box.
5. Select "Logon as a batch job" from the list of privileges.
6. Give the selected user this privilege.

Assigning Privileges to an Existing NT User Account

To assign privileges to an existing local user account, perform the following steps.

1. Choose the user on the User Manager panel and check for the following:
   - The "User Must Change Password at the Next Logon" option box should not be checked
   - "SYSTEM" or "system" is not used for the user name.
2. Under the Policies menu of the User Manager NT utility, select the User Rights option.
3. Check the "Show Advanced User Rights" box.
4. Select "Logon as a batch job" from the list of privileges.
5. Add the advanced user right to this user.

Configuring a Domain User as Your Agent User

Note: Domain users are not supported with 7.3.3 and earlier versions of the Agent.

To configure a repository user as your Agent user, perform the following steps.

1. Under the Policies menu of the User Manager NT utility, select the User Rights option.
2. Check the "Show Advanced User Rights" box.
3. Select "Logon as a batch job" from the list of privileges.
4. Click the Add button.
   a. Fill in the "List Names From" field: (choose your domain)
   b. Click Show Users button.
   c. In the listbox, choose the domain user.
   d. Click Add.
   e. Click OK.
5. In the User Rights Policy window, click OK.

Note: If you have both a local and a domain user with the same name, the local user takes precedence.

Controlling Operations of the NT Agent

This section contains information on controlling the Agent through Windows NT and the DOS prompt. It also contains a section on troubleshooting the Agent.

Note: Oracle Enterprise Manager and the Agent use Net8 to communicate with the databases in question. Please verify that Net8 can connect to every SID in question before attempting to use the Agent.

Starting the Intelligent Agent on Windows NT

To start the Agent on Windows NT, perform the following steps:

1. Double-click the Services icon in the Control Panel folder.
2. Select the Oracle<ORACLE_HOME_NAME>Agent service.

   The Startup Type is set to Manual, which allows the Agent to be started by a user. If you want the Agent to start automatically whenever you start the system, set the Startup Type to Automatic.

a. Click the Startup push-button. A Service Startup dialog box appears.

b. Choose Automatic under the Startup Type.

c. Click OK on the Service Startup dialog box.

3. Click the Start push-button to start the Agent.

Note: You can also start the Agent from the command line by typing the following:

```
net start oracle<ORACLE_HOME_NAME>agent
```

Stopping the Intelligent Agent on Windows NT

To stop the Agent on Windows NT, perform the following steps:

1. Double-click the Services icon in the Control Panel folder.
2. Select the OracleAgent service.
3. Click the Stop push-button to stop the Agent.

Note: You can also stop the Agent from the command line by typing the following:

```
net stop oracle<ORACLE_HOME_NAME>agent
```

Verifying that the Agent is Running

To verify that the Agent is running, look for its status in the control panel services or type `net start` at a command prompt. OracleAgent should appear in the list of running services.

You may also view the NT Task Manager to see the `dbsnmp` process information.

Installing the Oracle Intelligent Agent on UNIX

Install the Oracle Intelligent Agent from the Oracle CD-ROM as per the *Oracle Enterprise Manager Installation Guide*. The Intelligent Agent is a separate component to select.

Running the root.sh Shell Script

After you have successfully installed the Agent, the Oracle Installer prompts you to run `root.sh`.

`root.sh`, which is a shell script, updates/creates an `oratab` file. The `oratab` file is the file where the user will place references to all databases to be discovered by the Agent and controlled by the Oracle Enterprise Manager. For each database created, the entry is of the form: `<SID>:<$ORACLE_HOME>:[Y/N]`

The Agent is normally configured by `root.sh` as a `setuid` program. If `root.sh` was successful, the Agent will have been installed as `setuid root` so that the Agent can run jobs as the users whose name and password are given in the Preferred Credentials for that host.

If the Agent is not a `setuid` program, all Enterprise Manager jobs are run with the permissions of the user who started the Agent.

Note: The Agent being set to `setuid` root does not have the same effect as having the `root` user start the Agent. Having the `root` user start the Agent may cause security problems. Consult your platform documentation for exact details on `setuid` programs.

The user who submits node jobs to the UNIX Agent should have read/write access to the Agent's ORACLE_HOME. If the `root.sh` does not have `setuid` set, then any job submitted to the Agent will run with the privileges of the user who owns the Agent executable (dbsnmp.exe). `root.sh` will force the user to set the preferred credentials at the Oracle Enterprise Manager Console for any job on the Agent.

Note: Please be aware that only one intelligent Agent can be run on one UNIX machine although more than one Oracle Home can exist. The Agent communicates with the Management server on a dedicated port (1748).

Verifying that root.sh has been run successfully

To verify that `root.sh` had been run successfully, check the file permissions on dbsnmp.

1. Enter `cd $ORACLE_HOME/bin`

This changes the directory to the $ORACLE_HOME/bin directory where the Agent executable resides.

2. Enter `ls -al dbsnmp`

   This lists all relevant details about `dbsnmp`.

The output of the `ls -al` command for dbsnmp should be in the form

```
-rwsr-xr-x   1 root     g651     1497980 Jun 12 21:04 dbsnmp
```

`root` is the owner. `dbsnmp` is the Agent executable. In this example, the name of the group is `g651`. If `root` is the owner and `-rwsr-xr-x` are the permissions, then `root.sh` had been run successfully.

Controlling Operations of the UNIX Agent

On UNIX, the `lsnrctl command` is used to start and stop the Agent. The relevant `lsnrctl` commands are listed in the table below.

If you want to...	Enter the command...
Start the Agent on UNIX platforms	lsnrctl dbsnmp_start
Stop the Agent on the UNIX platform	lsnrctl dbsnmp_stop
Verify status of the Agent	lsnrctl dbsnmp_status

For additional information or restrictions for your platform, see the Intelligent Agent readme in `ORACLE_HOME/network/agent/doc/readme.wri`.

Configuring SNMP for UNIX or Windows NT

Third-party systems management applications use the SNMP Master Agent to communicate with the Intelligent Agent. The SNMP Master Agent and the Oracle Intelligent Agent must be configured correctly before the Oracle Intelligent Agent can communicate over SNMP to the Master Agent.

For the general procedures for configuring SNMP for Oracle databases and the Management SErver, refer to the *Oracle SNMP Support Reference Guide*.

For more comprehensive configuration information, see the installation or configuration guide specific to your platform since the SNMP configuration differs from platform to platform.

Oracle Intelligent Agent and Oracle Names

If you are running Oracle Names on a machine managed by an Oracle Intelligent Agent, it is assumed that the databases have already been registered with a Names Server and their aliases are defined by the GLOBAL_DBNAME parameters in the `listener.ora` files.

The Intelligent Agent 8.0.4 does not use Oracle Names to discover services it manages. It uses GLOBAL_DBNAME parameters in `listener.ora` files to determine which databases that listener services. This name appears in the Enterprise Manager Console Navigator as the database name to be managed.

The GLOBAL_DBNAME parameter typically describes the name of the database as it is registered with the Names Server, for example, the name and domain of the database as given in the database initialization parameter file. Values of the GLOBAL_DBNAME parameters must be unique.

When running jobs or monitoring events in this environment, the Intelligent Agent does not resolve database aliases via Oracle Names, the Agent will generate its own TNS connect string using the Bequeath Protocol.

Note: If you are planning to manage two or more Oracle databases on the same node, make sure the GLOBAL_DBNAME parameter in your `listener.ora` file is different for each database.

Roles and Users Required by the Agent

The necessary dbsnmp user account (with password "dbsnmp") and the SNMPAGENT role for the Intelligent Agent is contained in `catsnmp.sql`. The `catsnmp.sql` script is installed with the database. When an Oracle database is installed, the `catsnmp.sql` script is automatically run by `catalog.sql`

For security reasons, the customer may need to change the user / password for the Intelligent Agent's database logon. The default account is dbsnmp and the default password is dbsnmp. To change the user name and password to something other than dbsnmp / dbsnmp, you need to open, edit, and rerun `catsnmp.sql` for your own user and password. You will then need to edit `snmp_rw.ora`, adding the following parameters:

```
SNMP.CONNECT.<svcname>.NAME = <USERNAME>
SNMP.CONNECT.<svcname>.PASSWORD = <password>
```

To determine whether the `SNMPAGENT` role exists in a database, enter the following SQL command:

```
SELECT * FROM dba_roles;
```

If the `SNMPAGENT` role does not appear, run the `catsnmp.sql` script on the database.

If you already have several versions of the database running, you must run the `catsnmp.sql` script on each of these database in order to have the correct setup for all the grants and views the Agent needs to contact.

To run the script, you must log in as SYS or INTERNAL.

Note: The location of `catsnmp.sql` varies based on the database version you are running and the platform. For example, on NT for an Oracle 8.x database, the script is located at

`ORACLE_HOME\rdbms8x\admin`. Please note that there is no harm in running the `catsnmp.sql` script more than once.

Auto-Discovery

Beginning with 7.3.3, the Intelligent Agent has a built-in auto-discovery feature that automatically generates the needed configuration files containing information about services to be managed, each time the process is started. The following three files are created/appended during the discovery process:

- `$ORACLE_HOME/network/admin/snmp_ro.ora`

  The snmp_ro.ora file is a read-only file created by the Agent and contains information on services monitored by the Agent.

- `$ORACLE_HOME/network/admin/snmp_rw.ora`

  The snmp_rw.ora file contains index information of the managed services used internally by the Agent and it also allows users to specify variables, such as tracing.

- `$ORACLE_HOME/network/agent/services.ora`

  The services.ora file contains aliases for all services the Agent has to monitor. Only services listed in this file are monitored by the Agent. The content of this file are then sent to the console during discovery.

Note: Please refer to Appendix A, "Configuration Files" for more information on parameters used in these files.

When the Agent is started, the auto-discovery process reads configuration parameters from the following sources:

- `oratab` (on Unix nodes)
- Windows NT Registry (on Windows NT nodes)
- `listener.ora`
- `tnsnames.ora` (if one exists)

The discovery process extracts the services installed on that node and compiles the configuration files listed previously.

Beginning with 7.3.4 and 8.0.3, the Agent compiles SID information for each ORACLE_HOME, either from the `ORATAB` file(UNIX) or the NT registry. The Agent then parses the listener.ora files for related SID and listener information. If the `listener.ora` contains a GLOBAL_DBNAME section, the Agent sets the

database service name to the GLOBAL_DBNAME variable. If the variable does not exist, the Agent looks for a tnsnames.ora that contains a valid service name for the SIDs on that machine. If the Agent cannot find one, a service name called <SID>_<HOSTNAME> is created for each SID.

Note: In 7.3.3 and earlier versions, the Agent does not use the GLOBAL_DBNAME.

Note: If multiple aliases exist for the same instance in the tnsnames.ora, the Agent uses the one listed first. If you prefer to use a different alias, reorder the tnsnames.ora entries and restart the Agent.

Note: If you have more than one database instance on a machine and you are using GLOBAL_DBNAME parameter in the `listener.ora` file, these instances need to have a unique GLOBAL_DBNAME in the listener.ora. You may have to do edit the listener.ora manually.

Pre-requisites for Auto-Discovery

- SQL*Net V2 or Net80 TCP/IP must be present, and the necessary files must be created, prior to launching the Intelligent Agent. The only required SQL*Net (or Net8) file is `listener.ora`, but `tnsnames.ora` and `sqlnet.ora` should be configured correctly for particular service discovery. The Agent searches for these files in the `$ORACLE_HOME/network/admin` directory.
- TNS_ADMIN variable usage during Agent Discovery:

  (UNIX) All versions of the Unix discovery script allow the use of the TNS_ADMIN variable to locate input configuration files (`listener.ora` and `tnsnames.ora`). Only post-8.0.3/7.3.4 versions correctly write the output files (`snmp_ro.ora` and `snmp_rw.ora`) into TNS_ADMIN, if this environment

variable is set. If the TNS_ADMIN variable is not set, then the Agent will write the output files to its `$ORACLE_HOME/network/admin` directory.

(NT) In addition to the above, beginning with 8.0.5, the discovery script also reads the TNS_ADMIN value from the NT Registry. This variable is located as follows:

- (NT) TNS_ADMIN variable in Control Panel -> System -> Environment
- (NT) TNS_ADMIN key in the NT Registry

3

Troubleshooting

This chapter covers generic troubleshooting strategies in the event your Intelligent Agent does not function properly. The following topics are discussed:

- Troubleshooting the Intelligent Agent
- Quick Checks
- Questions and Answers
- Intelligent Agent Startup Problems and Solutions
- Intelligent Agent Error Messages and Resolutions
- Tracing the Agent
- Tracing TCL

Troubleshooting the Intelligent Agent

Under most circumstances, the Intelligent Agent itself requires very little in the way of configuration. In order to function properly, however, the Agent must be able to communicate with the managing host and managed services. If you are familiar with Oracle and your operating system, using the following abbreviated checklists will likely solve problems that can interfere with Agent operation.

Important: Because the Agent is continuously being improved from one release to the next, it is **strongly recommended** that you upgrade to the latest Agent available for your particular server release. Oftentimes, this will resolve problems you may encounter with earlier versions of the Agent.

Quick Checks

The following checklists cover the areas most likely to affect Agent operation. Agent troubleshooting checklists have been divided according to the two most common platforms on which the Agent is run: Windows NT and UNIX. The checklists are abbreviated and assume knowledge of both Oracle, the operating system, and related communication protocols. Specific troubleshooting procedures are covered in detail later in this chapter.

Quick Checks for the Windows NT Agent

If you are running an Agent on a Windows NT system, use the following checklist.

1. Make sure the Agent service is up by checking the OracleAgent service in your control panel. If the Agent did not start up, use any of the following hints listed below.
2. Check for messages written to the NT Event Viewer (under Administrative Tools) since this is where the NT Agent writes any problems associated with startup.
3. Check if `snmp_ro.ora`, `snmp_rw.ora`, and `services.ora` are created by the Agent on startup. `snmp_ro.ora` and `snmp_rw.ora` are in the `ORACLE_HOME\NET80\admin` directory, and `services.ora` is in the `ORACLE_HOME\NET80\agent` directory.

Compare the services listed with the services which are available on the machine. Please refer to Appendix A, "Configuration Files" for valid sample files.

If services are missing, check the following files for inconsistency or corruption:

- `listener.ora`
- `tnsnames.ora`

4. Check that you do not have a system path set to external drives.

   The Agent is a service and runs by default as SYSTEM. It also needs DLLs from the `ORACLE_HOME/BIN` directory. If you need mapped drives in your path, you MUST NOT set them in the SYSTEM path.

   To set your own path:

   a. Move mapped drive paths out of SYSTEM path variables and into your own.

   b. Reboot to "unset" the systems path.

5. Check if you have TCP / IP installed. TCP / IP is a requirement.

6. If you still do not know why the Agent did not start, trace the Agent.

   a. Set the following variables in `snmp_rw.ora`:

      `nmi.trace_level=admin` (or 16 if you want maximum information)

      `nmi.trace_directory=`<any directory in which the Oracle user has write privileges>

      `nmi.trace_file=`<name of the trace output file>

   b. Restart the Agent.

   c. Check the log files located in the `ORACLE_HOME/NET80/LOG` directory.

      `NMI.LOG` should show general Agent problems.

      `NMICONFIG.LOG` should show problems with auto-discovery.

7. Ensure that the DNS Host entry is set to the node name in the listener.ora and tnsnames.ora files.

   a. Run the start button-> settings-> control panel-> network-> protocol-> TCP / IP properties.

   b. Check the DNS Host entry.

8. Turn on tracing for the daemon.
   a. Open `$ORACLE_HOME/net80/admin/sqlnet.ora` and add the lines `daemon.trace_level=13` and `daemon.trace_directory=e:\orant\net80\trace`.
   b. Close the console to stop the daemon.
   c. Open the console to restart the daemon in trace mode.
   d. Submit a job and view the `daemon.trc` file for daemon and console problems.

Quick Checks for UNIX Agents

If you are running an Agent on a UNIX system, use the following checklist.

1. Make sure Agent listener is working. Enter the command:

   ```
   lsnrctl dbsnmp_status
   ```

   If your Agent is running, you should see something similar to the following:

   ```
   LSNRCTL for Solaris: Version 8.1.3.0.0 - Production on 04-NOV-98
   18:44:15

   (c) Copyright 1997 Oracle Corporation.  All rights reserved.

   The db subagent is already running.
   ```

2. Check the `ORACLE_HOME/NETWORK/log/dbsnmp*.log` file for errors on UNIX.
3. Check that the Oracle user has write permissions to `ORACLE_HOME/AGENT/LOG` as well as `ORACLE_HOME/NETWORK/AGENT`.
4. Check `snmp_ro.ora`, `snmp_rw.ora`, and `services.ora` for the entries created by the Agent. `snmp_ro.ora` and `snmp_rw.ora` are in the `ORACLE_HOME/NETWORK/ADMIN` directory, and `services.ora` is in the `ORACLE_HOME/NETWORK/AGENT` directory.

   Compare the services listed with the services which are available on the machine. Please refer to Appendix A, "Configuration Files" for valid sample files.

   If services are missing, check the following files for inconsistency or corruption:

   - `listener.ora`

- tnsnames.ora
- oratab

5. If you still do not know why the Agent did not start, trace the Agent by setting the following variables in snmp_rw.ora:
   - nmi.trace_level=admin (or 16 if you want more information)
   - nmi.trace_directory=<any directory which the Oracle user can write to>
   - nmi.trace_file=agent
6. If you have upgraded the database software and one of your machines is having problems with the generated snmp_ro.ora, snmp_rw.ora or services.ora file, follow the instructions below:
   a. Run catsnmp.sql under the INTERNAL or SYS account (NOT the dbsnmp account). Normally the catsnmp.sql script is run from catalog.sql upon database creation but since this is an upgrade, you may not have run this script yet. If the necessary scripts have not been run, the dbsnmp account is not created.
   b. If you have more than one SID or older SIDs referenced in the oratab file, run catsnmp.sql against each of the databases.
   c. The snmp_ro.ra file is a read only file which means that all changes to the file will be overwritten each time the Agent is started. You can make changes (if needed) to the snmp_rw.ora file.

   If you are trying to do backups, you must run backupts.sql with the dbsnmp/dbsnmp account.

Warning: Please do not modify the Tcl scripts (job and events scripts written in Tool Command Language) that come with the Agent. If you want to submit a job different from the ones that are predefined with the Agent, use the TCL Job where you are allowed to pass in arbitrary scripts and have the Agent execute them.

Questions and Answers

If after going through the troubleshooting checklists your Agent still is not functioning correctly, use the following section to cover other areas of Agent

operation that are less probable causes of Agent operating problems. In addition, many of the steps in the checklists are covered in greater detail for those users who may be less familiar with Oracle and/or the operating system on which the Agent is running. The following questions are covered in this section:

- Is TCP/IP configured and running correctly? on page 3-6
- Do the DNS Name and the Computer Name Match? (Windows NT) on page 3-8
- Are the Net8 configuration files correct? on page 3-9
- Is Net8 functioning properly? on page 3-10
- Did the Agent startup successfully? on page 3-11
- Did the Agent connect to ALL instances on its node? on page 3-13
- Did the Agent connect to ALL instances on its node? on page 3-13
- Is the Agent running with the correct permissions? (UNIX) on page 3-13
- Does the OS user exist and does it have the correct permissions? (Windows NT) on page 3-13
- Are you still using a 7.3.3 or earlier Agent? on page 3-14
- Why doesn't the Agent send status notifications back to the Enterprise Manager Console even though the jobs have run? on page 3-14

Note: You do not need to remove all ".q" files from the `$ORACLE_HOME/network/agent` directory in order to debug the Agent. Although this approach was recommended in the past, troubleshooting more recent versions of the Intelligent Agent no longer requires this action. There are exceptions to this rule, which will be pointed out later in the chapter.

Is TCP/IP configured and running correctly?

One of the most common problems that prevents the Agent from starting is TCP/IP configuration. To check whether your TCP/IP setup is configured correctly, issue the following commands at the command line:

- Determine if the host machine (machine on which the Agent runs) and the specified network IP address refer to the same machine. Type the following at the command line.

  1. ping <hostname>
  2. ping <IP address>
  3. Check to see if the above commands return the same information (Windows NT). For UNIX systems, you should see "<hostname> is alive" and "<IP address> is alive" respectively.

- Determine if the host machine is reachable by issuing the following command.

  telnet <hostname>

- Validate whether the host machine is available on the local network and whether the machine on which you are running the Console (Management Server for the case of V2) can access the host machine.

  1. ping the machine running the Agent using its IP address from itself .
  2. ping the machine running the Agent using its IP address from the machine running the Console.

- Determine if the machine running the Console is available on the local network and whether the machine running the Agent can communicate with the Console machine.

  1. ping <IP address of the console machine>

     Ping the machine running the Console from itself.

  2. ping <IP address of the console machine>

     Ping the Console machine from the machine running the Agent

  3. Check to see if the steps above return the same information.

Note: To determine the hostname of a Windows NT system, type "hostname" at a command prompt.

Correcting TCP/IP configuration problems

1. (Windows NT) Edit the `WINNT\system32\drivers\etc\hosts` and `lmhosts` files.

If these files have never been used, only sample files will exist in the directory. Either rename or copy the `.sam` files to just the file name with no extension.

(UNIX) Log in as root and edit the `/etc/hosts` file.

2. Verify that the IP address and host information for each system are correct.

   Example: (Windows NT)

   (Replace the information in brackets with the actual host information for that system.)

```
HOSTS file:
        <122.111.111.111>   <hostname>

LMHOSTS file:
        <122.111.111.111>   <netbios name or hostname>  #PRE
```

   Note: You can also verify this information through the Windows NT Control Panel -> Network property sheet.

3. Delete the `$ORACLE_HOME\network\agent\*.q` and services.ora files.

   Note: The *.q files contain information about current jobs and events. Do not delete these files without first removing all jobs and events registered against this Agent.

4. Delete the `$ORACLE_HOME\network\admin\snmp_ro.ora` and `$ORACLE_HOME\network\admin\snmp_rw.ora files`.

5. Restart the Agent.

Do the DNS Name and the Computer Name Match? (Windows NT)

Before Release 8.0.4 of the Agent, the NT Agent required the DNS Hostname and the Computer Name to be identical. These parameters can be checked/changed from the following Windows NT Control Panel property sheets.

To verify the computer name:

- Control Panel --> Network --> Identification --> Computer Name

To verify the DNS Name:

- Control Panel --> Network --> Protocols --> TCP/IP Protocol --> Properties-->DNS --> Hostname

Are the Net8 configuration files correct?

In addition to proper network configuration, which allows nodes in your network to communicate, components of your Oracle environment must also be able to communicate with each other. Net8 provides the session and data communication medium between client machines and Oracle servers, or between Oracle servers. For this reason, proper Net8 configuration is a prerequisite for Agent communication. This section covers the most common problems that can occur when Agent communication fails.

Net8 configuration files are found in `$ORACLE_HOME/Net80/admin`, or $TNS_ADMIN (Windows NT) or `$ORACLE_HOME/network/admin` (UNIX).

Primary configuration files are:

- `listener.ora`
- `sqlnet.ora`
- `tnsnames.ora`

See Appendix A, "Configuration Files" for information and examples of the above files.

TNS_ADMIN variable usage during Agent Discovery

(UNIX) All versions of the Unix discovery script allow the use of the TNS_ADMIN variable to locate input files (`listener.ora` and `tnsnames.ora`). Only Agent versions 7.3.4 and above correctly write the output files (`snmp_ro.ora` and `snmp_rw.ora`) into TNS_ADMIN, if set.

(Windows NT) Beginning with version 8.0.5, the discovery script also reads the TNS_ADMIN value from the NT Registry.

The Agent also uses the TNS alias information found in the `listener.ora` file. The Agent does so even within an Oracle names environment. This behavior is intentional since an Oracle Names server may be temporarily unavailable and the Agent needs to be able to resolve names at all times. Check the following to make sure the local translation of the TNS alias takes place:

1. Verify that the `listener.ora` file contains the following for each instance:
   - Two IPC entries
   - One TCP entry

   Do not activate the listener on port 1748, since Agent is listening on this port. (This is the reason you can use TNSPING against the Agent; TNSPING cannot differentiate between a listener and an Agent)

   The Agent requires IPC entries and TNS alias definitions on the server, in addition to alias definitions from the Console, to perform alias translations. This correct IPC entries and TNS alias definitions are essential for correct Agent / Console (V1) or Agent / Management Server (V2) communications.
2. Ensure that the DNS Host entry is set to the node name in the `listener.ora` and tnsnames.ora files.
   1. From the Windows NT menu bar, click Start -> Settings -> Control Panel
   2. Double-click on the Network icon
   3. Click on the Protocols tab
   4. Select TCP / IP Protocol and click Properties.
   5. Check the DNS Host entry.

Note: When using the 7.3.3 Oracle Intelligent Agent on a Windows NT system that has 2 NIC cards, create only one service descriptor in the `tnsnames.ora` containing the IP address of only one of the NIC cards. Do not create separate service descriptors for each NIC card and do not put both IP addresses in the `address_list` of the single service descriptor.

Is Net8 functioning properly?

If your Net8 configuration is correct and you are still unable to contact the Agent, the next step is to determine whether services in your Net8 network can be reached. You can use the TNSPING utility on each database you want to access by entering the following at the command prompt:

```
tnsping <network service name>
```

If you can connect successfully from a client to a server (or from a server to a server) using TNSPING, the command will return an estimate of the round trip time (in milliseconds) it takes to reach the Net8 service. This indicates Net8 is functioning properly.

Next, add the following alias (Agent debug entry) to the Console's `tnsnames.ora` file:

```
agent_<sid>.world=
   (DESCRIPTION =
       (ADDRESS_LIST =
           (ADDRESS =
               (COMMUNITY =TCP.world)
               (PROTOCOL = TCP)
               (Host = <your-agent-hostname>)
               (Port = 1748)
           )
       )
   )
```

Then ping the Agent from the OEM console using:

```
tnsping agent_<sid>
```

or

```
tnsping80 agent_<sid>
```

If the TNSPING command does not work, add the above alias to the Agent machine's `tnsnames.ora` file and try using TNSPING from the machine on which the Agent resides. Every Agent must be TNSPING-able using this alias.

Did the Agent startup successfully?

Check whether the Agent process is running:

UNIX Agents

From a command prompt type:

```
lsnrctl dbsnmp_status
```

The status returned should read:

```
The db subagent is already running
```

Windows NT Agents

1. From the Start menu, select Settings-->Control Panel
2. Double-click on Services
3. Verify that the *OracleAgent* service has been started.

If the Agent did not start up, use any of the hints listed in the following table:

Table 3–1 Troubleshooting an Agent that Will Not Start

UNIX	Windows NT
Check the `$ORACLE_HOME/network/log/dbsnmp*.log` file for errors	Check for messages written to the NT Event Viewer (under Administrative Tools) since this is where the NT Agent writes any problems associated with startup.
Check the `$ORACLE_HOME/network/log/nmiconf.log` file for errors.	Check the `$ORACLE_HOME/network/log/nmiconf.log` file for errors.

Check that the Oracle user has write permissions to the following directories:
`$ORACLE_HOME/agent/log`
`$ORACLE_HOME/network/agent`
	Check the properties of the Agent Service to verify the OS account used by the Agent (default is 'System') Check that the Agent user has write permissions to the following directories:
	`$ORACLE_HOME/agent/log`
	`$ORACLE_HOME/Net8/agent`

Check snmp_ro.ora, snmp_rw.ora, and services.ora for the entries created by the Agent. The snmp_ro and snmp_rw.ora files are located in the $ORACLE_HOME/network/admin directory, and services.ora is in the $ORACLE_HOME/network/agent directory.	Check if snmp_ro.ora, snmp_rw.ora, and services.ora are created by the Agent on startup.The snmp_ro and snmp_rw.ora files are located in the $ORACLE_HOME\network\admin directory, and services.ora is located in the $ORACLE_HOME\network\agent directory.
Compare the services listed with the services which are available on the machine. See Appendix A for valid sample files. If services are missing, check the following files for inconsistency or corruption: ■ listener.ora ■ tnsnames.ora ■ oratab	Compare the services listed with the services which are available on the machine. See Appendix A for valid sample files. If services are missing, check the following files for inconsistency or corruption: ■ listener.ora ■ tnsnames.ora
Check if you have TCP/IP installed. TCP/IP is a requirement. See Is TCP/IP configured and running correctly?	Check if you have TCP/IP installed. TCP/IP is a requirement. See Is TCP/IP configured and running correctly?

Table 3–1 Troubleshooting an Agent that Will Not Start

UNIX	Windows NT
If you still do not know why the Agent did not start, turn on tracing. (see Tracing the Intelligent Agent)	Check that you DO NOT have a systems path variable containing external drives. The Agent is a service and runs by default as SYSTEM. It also needs DLLs from the $ORACLE_HOME/bin directory. If you need external mapped drives in your path, you MUST NOT set them in the SYSTEM path. To set your own path: 1. Move external mapped drive paths out of systems path variable and into your own. 2. Reboot to "unset" the systems path.

If you still do not know why the Agent did not start, turn on tracing. For more information on setting up Agent tracing, see "Tracing the Agent" on page 3-29)

Did the Agent connect to ALL instances on its node?

To test whether an Agent can connect to the database(s) it monitors on a given node, try connecting to each database with the following connect string:

```
dbsnmp/dbsnmp@address_list
```

You must perform this test on the node where the Agent resides.

Note: Agents prior to 7.3.3 maintain two permanent connections to its local databases. Post 7.3.3 Agents maintain only one permanent connection.

Is the Agent running with the correct permissions? (UNIX)

To verify whether the Agent has the correct user permissions, see "Installing the Oracle Intelligent Agent on UNIX" on page 2-2.

Does the OS user exist and does it have the correct permissions? (Windows NT)

An OS user needs to be specified for the node and must have the following permissions:

- read/write permissions to the `$ORACLE_HOME\Net80` or `$ORACLE_HOME/network` directory and all of its sub directories

- read / write permissions to the $TEMP directory ($TEMP can be found on NT by selecting Control Panel --> System). If no $TEMP is defined, the OS user must have read / write permissions to the Oracle Home directory where it creates a directory called "work".
- The user above needs to have write permissions to the `$ORACLE_HOME/network/agent` directory.

Are you still using a 7.3.3 or earlier Agent?

Proper operation of the Oracle Enterprise Manager Job and Event systems requires you run version 7.3.4 or later of the Intelligent Agent. Running a 7.3.3 or earlier version of the Agent will limit available Job and Event system functionality.

Important: It is highly recommended that you upgrade 7.3.3 or earlier Agents to 7.3.4 or later versions.

Are there errors?

(Windows NT) Check the NT EVENT VIEWER -> APPLICATIONS -> LOG for any errors starting the DBSNMP process.

(Windows NT and UNIX) Check the `$ORACLE_HOME/network/log/nmiconf.log` file for discovery errors.

Why doesn't the Agent send status notifications back to the Enterprise Manager Console even though the jobs have run?

Most likely the job does actually run, but the Agent is unable to contact the console to send back notifications. Verify that hostname resolution can occur. Verify that the IP and hostname of the Windows NT machine running the console is in the /etc/hosts file on the Unix box or the hostname can be resolved via DNS/NIS. Retry the job.

To test the TCP/IP resolution, perform the following tests from a command prompt:

```
ping <hostname>
ping <IPaddress>
```

If the server is running telnet or ftp services(UNIX):

```
telnet <hostname>
ftp <hostname>
```

Since PING uses IP and not TCP, it is a good way of determining if the problem is in the packet routing.

To determine if the problem is actually with TCP, use the telnet or ftp utilities.

Alternatively, you can perform the following:

1. Be sure the name and IP address of the OEM console machine is in the /etc/hosts file on the Sun server, otherwise the Agent is not able to return messages to the console because it can not resolve the name of the machine to an IPADDRESS.
2. Open the Oracle Enterprise Manager's Daemon Manager, under CONFIGURATION PARAMETERS and specify the LISTENING ADDRESS parameter to contain the full IP address of the console machine. This forces the Agent to use the IP address in order to contact the console machine.

The default listening address (TNS format) is:

```
LISTENING ADDRESS = (ADDRESS=(PROTOCOL= TCP)(Host=machine_name)(Port=7770))
```

If a job stays in the scheduled status, repeatedly delete it using the DEL key. Restart the job. Sometimes it takes several submits until it starts up a delay of up to a minute until a job starts is common, especially the first time an Agent tries to sync with the OEM console with old Agents (7.3.2)

Intelligent Agent Startup Problems and Solutions

The following section covers specific problems, situations, and errors that may be encountered while trying to start the Intelligent Agent.

Generic Agent

This section covers problems that are common to both UNIX and Windows NT versions of the Intelligent Agent. Problems are prioritized according to their likelihood of occurrence.

'Failed to authenticate user' error when running a job In order for the Agent to execute jobs on a managed node, the following conditions must be met:

- An NT user account must exist that has the advanced user right, "logon as batch job." (Windows NT). The privilege can be assigned to an existing local or domain user (starting with 7.3.3), or a new NT user. Refer to Windows NT Specific Instructions in the Configuring the Intelligent Agent section.
- The preferred credentials for the node must be set for that user in the Oracle Enterprise Manager Console. Refer to "Setting Preferences" in the Oracle Enterprise Manager Configuration Guide.
- The user must have permissions to write to `$ORACLE_HOME/network` or `$ORACLE_HOME\Net80` directory.

'Login denied', 'Invalid username/password' messages in trace files This usually happens if you have a databases prior to 7.3.3 on the machine. From V7.3.3 onwards, a script called `CATSNMP.SQL` is included in the `CATALOG.SQL` dictionary script. This script is responsible for creating the DBSNMP user the Agent needs to connect. Older databases did not have this script yet.

Verify if the user 'DBSNMP' exists. If not, run the `catsnmp.sql` script.

'Listener not found for SID' when starting Agent When attempting to start the Intelligent Agent, the following error occurs: Listener Not Found for SID. The SID listed is always the last SID in the Oratab file. The `listener.ora` and `tnsnames.ora` files contains valid TNS descriptors for the SIDs. The oratab file does not have invalid SIDs and all SIDs have a dbsnmp account. This has been fixed for Agent versions 7.3.4 and later.

The 7.3.3 `nmiconf.tcl` script parses the `listener.ora` file looking for uppercase: ADDRESS, SID_LIST_, SID_DESCRIPTION and SID_NAME. Change the parameters listed above to uppercase, and discovery works.

'ORACLE_HOME does not exist' when starting the Agent This message comes from the discovery script, `nmiconf.tcl`. Make sure you have $ORACLE_HOME environment variable set to the ORACLE_HOME of the Agent and re-start the Agent.

The Agent is only finding one database on a certain node If you have more than one database on a single node, then you need to make sure that each instance has a unique GLOBAL_DBNAME in the listener.ora. You may have to define this manually in the listener.ora.

No snmp_ro.ora and snmp_rw.ora are generated. This error can occur if the Agent cannot write to `$ORACLE_HOME\network\admin`. Refer to the

$ORACLE_HOME\netowrkiog\nmiconf.log for errors. For more information on Agent startup problems, see "Did the Agent startup successfully?" on page 3-11.

Not all services are discovered. Check the services.ora file to determine which services have been discovered.

All the services the Agent finds on a machine, must be defined in the relevant SQL*Net / Net8 configuration files. If the service(s) are not defined, service discovery will fail and, in the worst case, the Agent will hang or return errors.

> **Windows NT**: Beginning with version 8.0.4, the Agent searches for service names that begin with 'OracleService' or 'OracleService<SID>'. Every entry beginning with 'OracleService' is considered to be a database running on this machine. Every SID encountered by the Agent must be defined in the relevant SQL*Net / Net8 files.
>
> **UNIX**: The oratab file is used to determine which SIDs are present. For 7.3.3 Agents and earlier, discovery fails if it encounters a SID that is not accurate (like in a Developer 2000 environment). To work around this problem, the environment variable $ORATAB can be used to access an alternate oratab file which contains only the databases you wish the Agent to see.

For the remaining databases, check the oratab file, and the SQL*Net / Net8 files to see if these files exist and that all definitions are present. Make sure that all of the databases are listed in the listener.ora file. For more information, see "Are the Net8 configuration files correct?" on page 3-9 and "Is Net8 functioning properly?" on page 3-10.

The Agent doesn't start correctly anymore If the Agent already started previously, and now refuses to start correctly, it may be that something has changed in the environment. Usually, a good thing to try is to let the Agent completely rediscover all its services again.

Delete the files snmp_ro.ora, snmp_rw.ora, and services.ora and restart the Agent. If that does not fix the problem, remove those files and also delete the $ORACLE_HOME/network/agent/*.q files.

Warning: This deletes all of your jobs and events. Make sure and delete these from the Console first.

Multiple Listeners The Intelligent Agent does not support multiple listeners for a single database on one machine. The services.ora contains the information that is used to communicate discovery information from Agent to daemon. It does not support multiple listeners for a single database. Due to a limitation in the discovery

of the services, there can be only one listener present on the machine per database you wish to monitor. If two listeners are listening for the same database, the Agent returns errors, or refuses the discovery.

For Agent versions 8.0.5 and above, if multiple listeners exists on a node, only the first listener the Agent encounters is used.

DBSNMP ERROR: Multiple Listeners found The fact that only one listener is running does not mean that other listeners do not exist! The Agent finds all listeners that are configured (via listener.ora files), not just the ones running. Also, the Agent does not restrict its search to the directory pointed by $TNS_ADMIN. It checks all possible locations for listener.ora files.

Note: For UNIX machines, if more than one listener appears to be configured to serve the same SID (located in the `oratab` file), the Agent picks the first one it finds, but also displays the message so the user is aware of this.

If the listener picked up by the Agent is not what you intended, correct the problem and restart the Agent.

'Invalid service name' or 'File operation error' while registering a job or event. This error is usually seen when the services on the console and the services discovered by the Agent are out of sync. For example, if you have an event registered against TESTDB and someone changes the name of the database to PRODDB, that Agent and Console are out of sync.

To fix this start by removing all job and event registrations from this service and dropping the node where the services exist from the console. Rediscover the node from the console using the auto-discovery wizard.

NOTE: With 7.3.2 the alias are case sensitive.

If you have a NT Agent please refer to 'Invalid service name' while registering a job or event.

Agent consuming too much memory Prior to Versions 7.3.4, 8.0.3.1.1 on NT, and 8.0.3 on Solaris, the Agent had a memory leak, causing it to use more and more memory. This leak occurs if an alias / service is specified which the Agent cannot contact (i.e.: database is down, listener is not started, etc.). Each time the Agent tries to contact

this service, the memory associated with this request is not freed. It also loses handles with the events dbprobe and listenerupdown.

Upgrade the Agent. If this is not possible, make sure only running services are monitored. Verify the memory usage of the Agent. If it becomes too high, stop and start the Agent.

Warning: Do not modify the supplied TCL scripts.

The Tcl (Tool Command Language) scripts supplied with the Intelligent Agent are used with the Oracle Enterprise Manager Job and Events system. If you want to submit a job different than the ones that are predefined with the Agent, use the TCL Job where you are allowed to pass in arbitrary scripts and have the Agent run them.

The DBSNMP.EXE utilizes 100% of CPU on Windows NT Enterprise Edition 4.0. The problem may occur when the NT Service OracleServiceORCL is not running or not there (because the customer creates his own database, with a SID other than ORCL) and the Oracle Performance Utility (not the Oracle Enterprise Manager Performance Pack but the Oracle8 performance utility) is installed.

If you do not have a database on your machine with SID = ORCL you will experience this problem, as the Oracle Performance Utility has hard coded a BEQ connect descriptor referencing the ORCL SID, in the NT registry during installations. The location of the connect descriptor is (HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Oracle80\Performance). Check the value of Username, Password and Hostname.

Solution is to deinstall the Oracle8 Performance utility.

'Transport read error' or 'Transport write error' messages This indicates a problem with the TCP/IP layer. Most obvious cause for this is that the IP address and the hostname do not reference the same physical machine.

Verify that TCP/IP is configured and running correctly. (See Is TCP/IP Installed and Running Correctly)

'Oralogin failed in orlon' You may receive this error while executing a TCL script using the oratcl verb oralogon through the Software Developer's Kit. "Oralogin

failed in orlon" means that the connect string is either wrong or for some reason, the account used cannot logon to the database.

Oracle 8 database switches Redo-Log every minute when the version 8.x Agent is running If you have jobs or events scheduled at a very low interval (30 seconds), this causes activity on your system. For example, if you have the event 'USER BLOCKS' registered, the Agent checks concurrent/waiting locks by building temporary table, deleting the tables, building them again - for every check.

Solution: Lower your interval or disable logging on the underlying table.

NT Agent

This section covers Intelligent Agent errors that can occur with the Windows NT version of the intelligent Agent. As with the previous section, errors are prioritized according to the likelihood of occurrence.

'Failed to connect to Agent' error. (Jobs that remain in submitted status)

There are in fact two hostname definitions on NT: One NETBios one, used for the NT's internal Named Pipes protocol, which is always installed. The other is the TCP/IP hostname, which is only configurable when you install TCP/IP on NT.

To find the NT NetBios hostname:

- Start Control Panel / Network
- The Computer Name in the dialog box is the NetBios hostname.

To find the TCP/IP hostname:

- Start Control Panel / Network / Protocols / TCP-IP / Properties / DNS
- This is the TCP/IP hostname.

On an NT server, you can 'ping' the two names, even if they are configured differently. Other clients, however, only 'ping' real TCP/IP hostnames. If the Agent is using local IPC connections, it uses Named Pipes. Therefore the NetBios name, while all external connections will use the TCP/IP name.

A mismatch in these names leads to 'unable to contact Agent', or forever pending jobs in the console. Therefore, make sure that the NetBios and the TCP/IP hostname are identical.

Receive the error failed -> 'output from job lost' while running job. The Windows NT user that you created for the Agent (see Agent Configuration, Configuration Guide) needs read / write permissions to the `$ORACLE_HOME\net80\agent` directory (and TEMP directory, for some applications) and read permissions to the SYSTEM32 directory

Verify that the NT user has these permissions.

Agent finds no services after discovery This problem has been fixed for Agent versions 7.3.4 and higher. For Agent versions 7.3.3 and lower, the following workaround can be used.

Check the `listener.ora` file, and make sure that no `$ORACLE_HOME` parameter is specified in the `SID_LIST` section. Specifying an `$ORACLE_HOME` in the `SID_LIST` section prevents the Agent from finding the requisite files for service discovery.

'Invalid service name' while registering a job or event. If you have a 8.0.4 Agent, you may experience this problem. If you have a default domain other than ".world". The Agent tries to append a ".world" to the database name during discovery. For example, if your default domain is nl.oracle.com and you define your `GLOBAL_DBNAME = database.nl.oracle.com`, the Agent defines the database name to be `database.nl.oracle.com.world`. This problem only occurs when the Agent and Console reside on the same machine (they share the some configuration files).

The workaround is to append "`.world`" to all services that do not currently have a specified domain.

The OracleAgent service hung on starting. This may occur if there are externally mapped drives on the system path variable. For more information see Did the Agent startup successfully? on page 3-11.

You many also be encountering a problem that is specific to Intelligent Agent 7.3.3 and other versions of the Oracle database (i.e. 7.3.2) You may identify that this is happening by checking the Windows NT Control Panel -> Services dialog. The Oracle Agent shows that the status is started, but when you highlight it to shut it down, the Stop button becomes disabled, along with the Start, Pause, and Continue buttons.

You cannot install IA 7.3.3 with a 7.3.2 db on Windows NT because these two products use two different versions of the RSFs (7.3.3 and 7.3.2). Both these RSFs can not be installed together.

Please upgrade your database to 7.3.3 with SQL*Net 2.3.3. If you cannot do this, then you have to remove OEM 1.3.5 & IA 7.3.3, make sure you have SQL*Net 2.3.2 client/server/adaptors installed. Install IA 8.0.3 and use OEM 1.4. IA 8.0.3 uses the RSF 803 and NET8, so there ia no conflict with Oracle 7.3.x.

Agent appears to have stopped working after I run a job frequently If you run a scheduled job every minute and receive the error on NT console "Event ID 2009, Number of sessions exceeded 2048" Verify that there are 2048 users logged in by viewing, NT-PROGRAMS-ADMIN TOOLS-WIN NT DIAGNOSTICS, CHOOSE THE NETWORK TAB. The Intelligent Agent is not releasing sessions when a scheduled job is finished running.

The workaround is to stop and start Agent every couple of hours.

or

Upgrade to the 8.0.4 Agent

Agent crashes with Dr. Watson error

The problem may occur when the NT Service OracleServiceORCL is not running or not there (you may have created a database, with a SID other than ORCL) and the Oracle8 Performance Utility is installed.

If you do not have a database on your machine with SID = ORCL you will experience this problem, as the Oracle8 Performance Utility has hard coded a BEQ connect descriptor referencing the ORCL SID, in the NT registry during installations. The location of the connect descriptor is (HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Oracle80\Performance). Check the value of Username, Password and Hostname.

The solution is to deinstall the Oracle8 Performance utility.

If this does not fix the problem, try setting the variable `automatic_ipc = off` in the `sqlnet.ora` file, comment out all IPC addresses in the `listener.ora` file, and restart the Agent.

Dr Watson : access violation in the application PLUS80.EXE If you get this error while trying to run a SQL*Plus job, then you have encountered a SQL*Plus bug. If you start SQL*Plus on the command line and connect using the connect string INSTEAD of the alias as described in the tnsnames.ora file, you see the same thing--a crash of SQL*Plus. It had to do with the connect string size exceeding the allocated connect string buffer size. For more information, see ORA-12163: 'TNS:connect descriptor is too long' on page 3-24.

Define a connect string for your database in your `tnsnames.ora` file that is less than 255 characters. Then re-start your Agent, and re-discover this node from the console.

Agent gives fatal NT error and service is uncontrollable This happens if Oracle7 and Oracle8 are installed in the same ORACLE_HOME on NT. This is NOT a supported configuration. The Agent tries to start, gives some sort of time-out error, saying that it failed to start, and then goes into a state where you are unable to stop or start the Agent service anymore.

First start the Agent, then start the Oracle8 services

OS and ORA errors after upgrading to OEM 1.5 This occurs when you install OEM 1.5 on an NT with a 8.0.3 database. The required support files are upgraded from 8.0.3 to 8.0.4 during the install.

If you then try to start the Agent, you will receive an error stack. If you reboot the NT machine, there will be aberrant behavior: no database, an OEM console, no Agent.

This is clear and documented: On NT, only the first two numbers are supported. Trying to install two version different in the third digit does not work. Only, in the number OEM 1.5, one is not aware that you are also installing 8.0.4 RSF's.

Unix Agent

Discovery fails with no services at all

First check that all of the SQL*Net files are present and correctly defined. You can then debug discovery by editing your oratab file contains only a valid SID with a listener running. After you get this working, you can add the remaining entries in the `oratab` file to see which entry is causing the problem.

Check the `$ORACLE_HOME/network/log/nmiconf.log` files for errors.

Auto-discovery will not work 'Failed to get service information for node <nodename>'

This is a well-known bug of Solaris 7.3.3 and has already been fixed in 7.3.4 and 8.0.3. You get this error if there are non-database entries in the oratab file. i.e. client side installs

You should also have received the following message when starting the Agent:

```
No listener found for SID <sid>
```

There are two workarounds for this:

1. Comment out these entries in the oratab when starting the Agent.
2. Make a copy of the oratab file and remove these entries. Then, set the $ORATAB environment variable of the user who starts the Agent to this file.

When running lsnrctl dbsnmp_start you get: NO LISTENER FOUND FOR <XXX>

This is problem with the Intelligent Agent 7.3.3. Your `services.ora` file is empty. When looking at the `snmp_ro.ora` file--the SNMP.VISIBLESERVICES is empty.

The "cannot find listener for XXXX" usually comes from a SID in the oratab file that does not have a listing in the listener.ora file. This means that the listener is not listening on this specific SID. Unless this is a SID that you are trying to connect the Agent to, this is a warning and should not stop the discovery process for the other SIDs.

Please check if the case sensitivity is respected in the `oratab` and the `listener.ora` for all the SIDs, and make sure that the oratab file does not have SIDs with a * or any other prefix. An example of this problem is when an SID used for some other Oracle product that starts with a * and this stopped discovery.

Intelligent Agent Error Messages and Resolutions

Generic Agent

ORA-12163: 'TNS:connect descriptor is too long' Copy the snmp.address.<host_name> parameter from your `$ORACLE_HOME\network\admin\snmp_ro.ora` file. Paste this address and parameter into your `$ORACLE_HOME\network\admin\snmp_rw.ora` file. In snmp_rw.ora, reduce the size of this connect string by removing the address entries for IPC. (NMP and SPX may also be removed.)

Shutdown/restart the Agent. See examples below.

Note: The parameter snmp.address in no longer found in `snmp_ro.ora` starting with the 7.3.4/8.0.3 Agents. Therefore, you will have to use this example to add a new variable to your `snmp_rw.ora`.

EXAMPLES:

Entry to be copied out of snmp_ro.ora:

```
snmp.address.ORCL_MACHINE-PC = (DESCRIPTION=(ADDRESS_LIST
=(ADDRESS=(PROTOCOL=IPC)(KEY=oracle.world))(ADDRESS=(PROTOCOL=IPC)(KEY=ORCL))(AD
DRESS=(COMMUNITY= TCP.world)(Host=machine-pc)
(PROTOCOL=TCP)(Port=1521))(ADDRESS=(COMMUNITY=TCP.world)(Host=machine-pc)
(PROTOCOL=TCP)(Port= 1526)))(CONNECT_DATA=(SID=ORCL)(SERVER=DEDICATED)))
```

Modified entry in snmp_rw.ora:

```
snmp.address.ORCL_machine-PC = (DESCRIPTION=(ADDRESS_LIST
=(ADDRESS=(COMMUNITY=TCP.world)(Host = machine-pc)(PROTOCOL= TCP)(Port=
1521))(ADDRESS=(COMMUNITY= TCP.world)(Host = machine-pc)(PROTOCOL=
TCP)(Port=1526)))(CONNECT_DATA=(SID=ORCL)(SERVER=DEDICATED)))
```

TNS-12542: 'TNS:address already in use'

This is actually a Net8 Listener error.

The following is documented in the 8.0.3.0.0 Intel NT release notes for the Net8 Listener. When a client connects to an Oracle8 server in dedicated server mode, WINSOCK2 Shared Sockets feature is used so that the client connection is routed from the listener to the database server. This feature improves the connection time, because the client does not need to close the socket connection with the listener and establish a new connection with the database server.

With the use of Shared Sockets, threads also use the same port as the listener. If you shut down the listener and try to start it up again for the same port, the listener does not start up if the port is in use due to any open connections with the database. Ensure that no client is connected to the database before starting up the listener. Note that if you are using a listener with a different port number you are able to start it up.

> **Warning:** Do not bring down the listener when any clients are connected to the database. If you need to listen for a new database, modify the listener.ora configuration file, and issue the reload command from the Listener Control Utility LSNRCTL80.

See *Oracle Networking Products Getting Started for Windows Platforms* for more information about the `listener.ora` file and the LSNRCTL80 utility. Oracle

Corporation attempted to overcome the restriction by using the WINSOCK2 option to allow the re-use of a port, but the option does not work reliably. Oracle Corporation is currently working with Microsoft Corporation to resolve this issue.

For additional information about the `reload` command, see the *Net8 Administrator's Guide.*

VOC-04816 'Invalid Destination' While submitting a job, validation fails with "failed to find address for Agent_node". And then the VOC-04816 Invalid Destination. This might also be caused by an invalid address in the tnsnames.ora located on the console.

Upgrade your Agent to at least 7.3.3. or later.

Verify that your SQL*Net configuration files are correct?

NT Agent

Any NT Operating System Error when starting the Agent

If you see an OS error when starting the Agent, check to see whether it is an actual Agent error as described in snmimsg.mc. Due to one of the Windows APIs not working as documented, the Agent fails to print out the real cause of the error.

Use the Event Viewer in the Administrative tools group of Windows NT. You should find the true cause of the problem documented. The source for the Agent errors are under the service name "dbsnmp". Highlight the most recent dbsnmp entry in the list. Double click on the event to get the actual results.

In order to debug the Agent after you have received an OS error, follow the following steps:

- Launch the EVENT VIEWER. (This is in the ADMINISTRATIVE TOOLS icon group.) Click on LOG from the main menu, then choose APPLICATION. The source for the Agent errors are under the service name "dbsnmp". Highlight the most recent dbsnmp entry in the list. Double click on the event to get the actual results.
- NMI.LOG and NMICONF.LOG should contain more information about the specific error that has occurred.
- Verify that `snmp_ro.ora` and `snmp_ro.ora` are in the `$ORACLE_HOME\network\admin` directory and they are not zero length.
- Verify that user that started the Agent can read / write the queue files.

- Make sure that the Machine Name and the DNS Hostname are the same.
- See Verifying the DNS Name and the Computer Name on NT. Verify that SQL*Net is running.
- Verify that TCP/IP is configured and running correctly. (See Is TCP/IP Configured and Running?)
- Remove all non-essential files from the `$ORACLE_HOME/network/agent/MIB` directory.
- Make sure that you have not installed OEM 1.5 on NT in the same ORACLE_HOME as a 8.0.3 database.
- If you are pre-Oracle 8.0.3, verify that you have only one Agent running. This is possible starting from Oracle 8.0.4, which creates a new Agent service(OracleAgent80). This allows a machine to have two Agent services.
- Upgrade to the 8.0.4 Agent

UNIX Agent

NMS-004 When starting Agent If snmpd is running on the unix box, set the following in the server's /etc/snmpd.conf:

```
smux 0.0 "" <ipaddress of server>
```

If the nms-4 error remains, turn on logging in the snmpd.conf file with the following parameter:

```
logging   file=/usr/tmp/snmpd.log   enabled
```

The log file gives more information about what could be happening. For example, a space between the double quotes in the smux line can cause the application to misinterpret the space as a password. The double quotes by themselves - mean no password.

NMS-0308 : 'Failed to listen on address : another Agent may be running'.

There are two possible causes for this error:

1. If two Agents are installed on a machine, in two different ORACLE_HOME, then you see this message if you try to start the second Agent. This is because both Agents try to listen the same default port #1748

Only have one Agent on a machine.

2. The port 1748 where the Agent listens is being used by someone else, or is not being released by dead process that were formerly using it (unfortunately common problem on SUN) .

To confirm port is being used by someone else

1. Use this command in UNIX

```
netstat -a | grep 1748
                          ^---- this is port #
```

   If any result shown on screen that ends in "LISTENING" then the port is in use.

2. If the following is true :

- netstat -a | grep 1748 ---> results in "LISTENING"
- LSNRCTL> dbsnmp_status (results in "The db subagent is not started.")

Then do this.

- ps -ef | grep dbsnmp
- kill -9 _____ (fill in process numbers)
- restart Agent with

LSNRCTL> dbsnmp_start

If it still fails to start the Agent, go through steps again, but before re-starting the AGENT, do this.

- cd $ORACLE_HOME/network/agent
- rm *.q, services.ora, snmp_ro.ora, and snmp_rw.ora
- restart Agent with

LSNRCTL> dbsnmp_start

This will re-start the Agent and remove all of the job and events queues it was using in the past.

If all else fails, re-booting the machine should definitely free up the port.

You may also have to relink the Agent to clear this problem. Please see the *Oracle Enterprise Manager Configuration Guide* and README for more information.

NMS-001 while starting the Agent This message indicates that the SNMP Master Agent (the process on UNIX that controls the SNMP protocol) could not be contacted. By default the Agent listens and works over SQL*Net, but the Agent can also work over SNMP on UNIX systems.

This message can safely be ignored unless you are trying to communicate with a Master Agent.

NMS-205 while starting the Agent The 'dbsnmp' user could not be located.

Run the catsnmp.sql script for that database with either the SYS or INTERNAL accounts.

NNL-018 while starting the Agent. The Agent tries to contact the Names Server, but can not get in contact with it. This can happen if a Names Server is indeed installed but not used.

Add a line in the file snmp_rw.ora

```
NMI.REGISTER_WITH_NAMES = FALSE
```

NMS-351 while starting the Agent This happens if there mismatches between the ID's in the '*.q' files in the $ORACLE_HOME/network/agent directory. Delete all the '*.q' in the $ORACLE_HOME/network/agent directory. Rebuild your repository. Restart the Agent.

Tracing the Agent

Beginning with 7.3.3, the Agent reads information from the `snmp_ro.ora` and `snmp_rw.ora` files in the $ORACLE_HOME\network\admin directory.

> **Note**: These files only exist after you have started the Agent the first time. If you want to trace the Agent the first time it is started, you can manually create a new file called `snmp_rw.ora` and add the trace parameters to this file. Otherwise, start the Agent and then modify the `snmp_rw.ora` file to add the trace information and restart the Agent.

Example of modifications of the `snmp_rw.ora` file:

```
NMI.TRACE_LEVEL = (OFF | USER | ADMIN | 16 )
```
The NMI.TRACLE_LEVEL settings mirror those used for SQL*Net.

Optional:

```
NMI.TRACE_FILE = agent        Default=dbsnmp.trc
NMI.TRACE_DIRECTORY = C:\TEMP  Default=$ORACLE_HOME/network/trace
```
(Any existing directory where the Agent has write permissions)

The log file, $ORACLE_HOME/network/log/nmi.log, is written by the Agent on every startup, even if tracing is not turned on. It contains the name and version of the Agent and the name and location of the Agent's configuration files. If tracing is turned on, it also contains problems encountered with the database and listener connections.

The log file, $ORACLE_HOME/network/log/nmiconf.log, is created on the first start up of the Agent and appended to every time after that. The auto discovery is done by the Tcl script, nmiconf.tcl (hence, the log file name). This file is written to only during startup. $ORACLE_HOME/agentbin/ORATCLSH is a special-purpose TCL shell that supports all standard TCL verbs (supported in TCL75.dll) plus a large subset (not all) of the ORATCL verbs supported by the OEM Agent. ORATCLSH is not a general purpose utility and may only be used in combination with the OEM Agent as it depends on files and data structures maintained by the OEM Agent.

There is no documentation of ORATCLSH and it has never been part of the supported feature set of the OEM Agent. It is provided strictly as a debugging tool to help Oracle customers and developers in developing OEM job and event scripts. The executable ORATCLSH is provided for debugging your TCL scripts. Before executing ORATCLSH, set the environment variable TCL_LIBRARY to point to $ORACLE_HOME/network/agent/tcl, the location of the init.tcl file.

Tracing TCL

You may also turn Tcl tracing on by setting the environment variable ORATCL_DEBUG and turning tracing on in the `snmp_rw.ora` file. The ORATCL_DEBUG must be set to the $ORACLE_HOME/network/trace directory. You must shut down and re-start the Agent for these parameters to take effect. TCL tracing creates a file, oratcl.trc in the above location. Every time an event is run an entry is added to the oratcl.trc file.

4

The Agent Data Gatherer

The Data Gatherer, which collects performance data used by the Oracle Capacity Planner and the new Java-based Oracle Performance Manager, is installed along with the Intelligent Agent.

Configuring the Agent Data Gatherer

> **Note:** If you are not using either the Oracle Capacity Planner or the Oracle Performance Manager, you do not need to configure or start the Oracle Data Gatherer.

This section contains the following topics:

- Description of the Data Gathering Service on page 4-3
- How the Data Gathering Service is Installed on page 4-3
- Understanding Where to Configure the Oracle Data Gatherer to Collect Database and Operating System Data on page 4-3
- Running the Oracle Data Gatherer on an Intermediate Host on page 4-4
- How to Configure the Data Gatherer on page 4-5
- Upgrading From a Previous Version of the Data Gatherer when You Install the New Data Gatherer into a Different Oracle Home on page 4-5
- Controlling Operations of the NT and UNIX Data Gathering Service on page 4-6
- Additional Methods for Controlling Operations on Windows NT on page 4-7
- Verify Data Gathering Service is Running on page 4-8
- Troubleshooting the Configuration of the Oracle Data Gatherer on page 4-9
- Obtaining Trace Information on the Oracle Data Gatherer on page 4-10

Description of the Data Gathering Service

The data gathering service (also known as Oracle Data Gatherer) is used to collect performance data.

The Oracle Data Gatherer is responsible for handling requests from client applications (for example, Oracle Capacity Planner and Oracle Performance Manager) that want to collect data. For each client application, you specify the performance data to collect (for example, file I/O or CPU usage data) and the time interval between data samples. The Oracle Data Gatherer then collects the requested data for the client at the specified interval.

How the Data Gathering Service is Installed

The Oracle Data Gatherer is part of the Oracle Intelligent Agent, and it is automatically installed when the Agent is installed on a managed host. Therefore, in the following sections, any reference to the Oracle Data Gatherer being installed on a host means that the Oracle Data Gatherer service was installed on the host when the Agent was installed on the host.

Understanding Where to Configure the Oracle Data Gatherer to Collect Database and Operating System Data

The Oracle Data Gatherer collects performance data for:

- databases
- operating systems
- other supported sources

In general, the Oracle Data Gatherer should be configured on the host where the target (database or host) to be monitored is physically located. In other words, if you want the Oracle Data Gatherer to collect database data from the MEDICAL database on host BENEFITS, then the Oracle Data Gatherer should be installed and configured on host BENEFITS. Similarly, if you want the Oracle Data Gatherer to collect operating system data on host EMPLOYEES, then the Oracle Data Gatherer should be installed and configured on host EMPLOYEES.

In some cases, however, it may not be possible to install and configure the Oracle Data Gatherer on a particular host. The Oracle Data Gatherer can be installed on a host only when both the following requirements are met:

1. The host operating system must support the installation and configuration of the Oracle Data Gatherer.
2. An Oracle Home for Oracle 8.0.4 or later must exist on the host. Note that the host does not have to have an Oracle 8.0.4 or later database, just an Oracle Home directory for Oracle 8.0.4 or later.

If a host does not meet these requirements, the Oracle Data Gatherer cannot be installed on the host, which means that the Oracle Data Gatherer will not be able to collect operating system statistics from the host. However, you can collect database data from a host on which the Oracle Data Gatherer cannot be installed and configured.

Running the Oracle Data Gatherer on an Intermediate Host

To collect database data from a host on which the Oracle Data Gatherer cannot be installed and configured, install and configure the Oracle Data Gatherer on a different host that meets both of the requirements. You can then use the Oracle Data Gatherer on this host to collect database data remotely on hosts that do not have the Oracle Data Gatherer installed. This is called using an intermediate host Oracle Data Gatherer.

For example, suppose you want to collect database data for the CONSULTANTS database on host EMPLOYEES, but the Oracle Data Gatherer cannot be installed and configured on host EMPLOYEES. If the Oracle Data Gatherer is installed and configured on host BENEFITS, you can use the Oracle Data Gatherer on BENEFITS as an intermediate host Oracle Data Gatherer. The Oracle Data Gatherer on BENEFITS is able to collect database statistics from the CONSULTANTS database on host EMPLOYEES.

You might also want to use an intermediate host Oracle Data Gatherer if an additional process footprint/overhead cannot be tolerated on the host where you want to collect database data. Of course, in this situation the collection activity will still take place against the database. This minimal collection activity overhead will be present regardless of the host where the Oracle Data Gatherer is located.

How to Configure the Data Gatherer

You must configure the Oracle Data Gatherer after it is installed on a host.

The Oracle Data Gatherer, by default, tries to use the username/password account set up as the preferred credentials for the database to locate the Data Gatherer. If the preferred credentials are incorrect or if the Data Gatherer is not located on that host the client will reprompt you for the location of the Data Gatherer.

You may want to set up the preferred credentials for the database before starting the client applications (Performance Manager and Capacity Planner).

The TNSNAMES.ORA file on the host where the Oracle Data Gatherer is installed must include entries for:

- the target database
- the repository for Oracle Capacity Planner and Oracle Performance Manager. The repository location was defined using the Oracle Configuration Assistant.

Upgrading From a Previous Version of the Data Gatherer when You Install the New Data Gatherer into a Different Oracle Home

It is possible to install the new version of the Oracle Data Gatherer into a different Oracle Home than the previous version. If you plan to do this, follow these steps:

1. Stop the previous version of the Oracle Data Gatherer. See Controlling Operations of the NT and UNIX Data Gathering Service on page 4-6 for more information on stopping the Oracle Data Gatherer.
2. Install the new version of the Oracle Data Gatherer, but do not start it.
3. Move the Capacity Planning configuration files (state files) and data files associated with the previous version of Oracle Data Gatherer to the Oracle Home where you have installed the new version of Oracle Data Gatherer.

   The Oracle Data Gatherer state and data files are located in the `$ORACLE_HOME/odg/reco` directory. You need to copy the files into the new `$ORACLE_HOME/odg/reco` directory before you use Oracle Capacity Planner to connect to the new version of the Oracle Data Gatherer and set up any new collections.

If you do not move these files, the following problems will occur:

a. Binary data is not loaded.

Any binary data files created by Oracle Data Gatherer which have not yet been loaded into the Capacity Planner database will not be loaded.

b. Data collection definitions are not maintained.

You will need to redefine your Capacity Planner data collections.

If you have installed the new version of Oracle Data Gatherer into the same Oracle Home as the previous version or if you do not currently use the Oracle Capacity Planner, do not move the state and data files.

4. Start the new version of the Oracle Data Gatherer. See Controlling Operations of the NT and UNIX Data Gathering Service on page 4-6 for more information about starting the Oracle Data Gatherer.

Controlling Operations of the NT and UNIX Data Gathering Service

On UNIX and NT, Oracle Enterprise Manager uses the `vppcntl` command to manage the data gathering service. The `vppcntl` executable is located in `ORACLE_HOME/bin`.

Commands to control Oracle Data Gatherer are listed in the table below:

If you want to...	Enter the command...
Start Oracle Data Gatherer	`vppcntl -start`
Stop Oracle Data Gatherer	`vppcntl -stop`
Verify that Oracle Data Gatherer is running	`vppcntl -ping`
Identify the version of Oracle Data Gatherer	`vppcntl -version`

Note: You can only run one version of the Data Gatherer on a host at a time; therefore, you do not need more than one Data Gatherer on a host. It is recommended that you deinstall the previous version of the Data Gatherer. If you try to start more than one Data Gatherer on a host, you will get an error.

If you are running Oracle Enterprise Manager in a mixed environment, it is recommended that you upgrade to the latest client. Refer to the compatibility matrix below for more information.

Oracle Enterprise Manager Versions	Data Gatherer Version
Enterprise Manager 1.5.5 and 1.6.0 clients	Runs with 8.0.4 and 8.0.5 Data Gatherer Does not run with 8.1.5 Data Gatherer
Data Gatherer 2.0.4 clients	Runs with 8.0.4, 8.0.5, or 8.1.5 Data Gatherer

Additional Methods for Controlling Operations on Windows NT

This section contains information on controlling the Oracle Data Gatherer through Windows NT.

By default, you start the Oracle Data Gatherer manually on a host. To start Oracle Data Gatherer automatically through the Control Panel on Windows NT, perform the following steps:

1. Double-click the Services icon in the Control Panel folder.
2. Select the name of the Oracle Data Gatherer service you want to start.

   If the host has one Oracle Home, then the name of the Oracle Data Gatherer service is OracleDataGatherer.

   If the host has multiple Oracle Homes and the Oracle Data Gatherer has been installed into more than one Oracle Home, then multiple data gatherer services are displayed. When the Oracle Data Gatherer is installed into multiple Oracle Homes, the names of the data gathering services use the naming convention Oracle<Oracle_Home_name>DataGatherer. For example, suppose a host has two Oracle Homes, named 804 and 805, and the data gathering service has been installed both homes. The Oracle Data Gathering services for those Oracle Homes are named Oracle804DataGatherer and Oracle805DataGatherer, respectively.

   The Startup Type is set to Manual, which allows the data gathering service to be started by a user. If you want Oracle Data Gatherer to start automatically whenever you start the system, set the Startup Type for Automatic.

   a. Click the Startup push-button. A Service Startup dialog box appears.
   b. Choose Automatic under the Startup Type.

c. Click OK on the Service Startup dialog box.

3. Click the Start push-button to start the data gathering service.

To stop Oracle Data Gatherer through the Control Panel on Windows NT, perform the following steps:

1. Double-click the Services icon in the Control Panel folder.
2. Select the name of the Oracle Data Gatherer service that you want to stop.
3. Click the Stop push-button to stop the data gathering service.

Verify Data Gathering Service is Running

On Windows NT, there are several ways to determine if the Oracle Data Gatherer is running by checking the:

- Status returned by the `vppcntl -ping` command.
- Status in the NT control panel services.
- vppdc process information in the NT Task Manager.

Oracle recommends that you use the `vppcntl -ping` command because it tells you if the Oracle Data Gatherer is running and also performs a test to determine whether it is responsive and running properly.

The data gathering service's alert / warning log is `ORACLE_HOME\odg\bin\alert_dg.log`.

On UNIX, use `vppcntl -ping` to verify if Oracle Data Gatherer is running. The data gathering service's alert / warning log is `$ORACLE_HOME/odg/bin/alert_dg.log`.

Troubleshooting the Configuration of the Oracle Data Gatherer

This section describes possible configuration problems and how to solve them.

Table 4–1 Troubleshooting Oracle Data Gatherer Configuration Problems

Problem	Steps for FIxing the Problem
The client host (where Oracle Capacity Planner or Oracle Performance Manager is installed) cannot connect to the host where the Oracle Data Gatherer is installed.	■ From the client host, use the ping utility to ping the Oracle Data Gatherer host: `ping Oracle-data-gatherer-host-name` ■ If you have set preferred credentials for the database and a Data Gatherer exists on that host, you are not prompted for the Data Gatherer location. The Data Gatherer on that host is used as the default. If you have not set the preferred credentials for the database or if no Data Gatherer exists on that host, the Oracle Database Logon dialog box prompts you for the name of an Oracle Data Gatherer host. You must enter exactly the same host name as you entered in the ping command. For example, if you successfully pinged host DAVID.COMP.COM from the client host, you cannot specify DAVID as the name of the Oracle Data Gatherer host in the Oracle Database Logon dialog box. You must specify DAVID.COMP.COM as the name of the Oracle Data Gatherer host in the Oracle Database Logon dialog box.
The host where the Oracle Data Gatherer is located cannot connect to the target database.	■ From the host where the Oracle Data Gatherer is located, connect to the target database using SQL*Plus by specifying the username, password, and service of the target database: `SQLPLUS> connect username/password@service-name` ■ If you have set preferred credentials for the database and a Data Gatherer exists on that host, you are not prompted for the Data Gatherer location. The Data Gatherer on that host is used as the default. If you have not set the preferred credentials for the database or if no Data Gatherer exists on that host, the Oracle Database Logon dialog box prompts you for the username, password, and service of the target database. Be sure to enter exactly the same username, password, and service information that you entered in SQL*Plus.

If the steps for fixing the configuration problems fail, verify that your TNSNAMES.ORA entries are configured correctly, as described in How to Configure the Data Gatherer on page 4-5.

Obtaining Trace Information on the Oracle Data Gatherer

You can also obtain trace information on the Oracle Data Gatherer. To obtain this information, you must run the Oracle Data Gatherer from the Oracle Enterprise Manager console.

To view the Oracle Data Gatherer trace information on the screen, type the following command at the DOS prompt or UNIX command line:

```
vppdc -console -debug
```

To send the Oracle Data Gatherer trace information to a file, type the following command at the DOS prompt or UNIX command line:

```
vppdc -console -debug > trace.txt
```

If the above command is used, the trace file is named trace.txt. If you prefer, you can specify a different name for the trace file.

Note: If you want to run the Data Gatherer in debug mode and a Data Gatherer is already running, you must stop the Data Gatherer, and then run it from the command line as shown above using the -debug flag.

5

Job and Event Scripts

Topics covered in this document include:

- Scripting Language
- Server Message and Error Information
- Use of Tcl with the Intelligent Agent
- NLS Issues and Error Messages
- OraTcl Functions and Parameters

Scripting Language

The Tcl Language with OraTcl extensions is used to write the job and events scripts. Tcl is used for the scripts because it fulfills the necessary requirements, such as:

- Host system access for handling with files and devices, launching programs, and executing operating system functions.
- SQL and PL/SQL functions for accessing the RDBMS.
- RDBMS administration functions.
- SNMP accessing, both for the database MIB variables that the Agent itself supports, and for external MIBs, like the host's or other SNMP-enabled services.
- Communication with the Oracle Intelligent Agent and other Oracle software, such as Oracle Trace.
- A syntax for describing job and event scripts that:
  - Can be used to drive the user interface.
  - Provide information on the nature of the job or event, and any input or output.
  - Allow access to the Oracle message file system for NLS support.

Tcl Language Description

Tcl originated with Dr. John Ousterhout from the University of California, Berkeley, California. Tcl, current release version 7.5, stands for *Tool Command Language*.

Tcl is both a language and a library. Tcl is a simple textual language that is intended primarily for issuing commands to interactive programs, such as text editors, debuggers, illustrators, and shells. Tcl has a simple syntax and is programmable. Tcl users can write command procedures to provide more powerful commands than those in the built-in set.

Tcl is also a library package that can be embedded in application programs. The Tcl library consists of a parser for the Tcl language, routines to implement the Tcl built-in functions, and procedures that allow each application to extend Tcl with additional commands specific to that application. The application program generates Tcl commands and passes them to the Tcl parser for execution. Commands may be generated by reading characters from an input source, or by associating command strings with elements of the application's user interface, such as menu entries, buttons, or keystrokes. When the Tcl library receives commands it parses them into component fields and executes built-in commands directly. For

commands implemented by the application, Tcl calls back to the application to execute the commands. In many cases commands will invoke recursive invocations of the Tcl interpreter by passing in additional strings to execute. Procedures, looping commands, and conditional commands all work in this way.

An application program gains several advantages by using Tcl for its command language.

- Tcl provides a standard syntax. After you learn Tcl, you are able to issue commands easily to any Tcl-based application.
- Tcl provides programmability. All a Tcl application needs to do is to implement a few application-specific low-level commands. Tcl provides many utility commands plus a general programming interface for building up complex command procedures. By using Tcl, applications do not need to re-implement these features.
- Extensions to Tcl provide mechanisms for communicating between applications by sending Tcl commands back and forth. The common Tcl language framework makes it easier for applications to communicate.

Tcl was designed with the philosophy that one should actually use two or more languages when designing large software systems. One for manipulating complex internal data structures, or where performance is key, and another, such as Tcl, for writing small scripts that tie together the c programming pieces and provide hooks for others to extend. For the Tcl scripts, ease of learning, ease of programming and ease of integrating are more important than performance or facilities for complex data structures and algorithms. Tcl was designed to make it easy to drop into a lower language when you come across tasks that make more sense at a lower level. In this way, the basic core functionality can remain small and one need only bring along pieces that one particular wants or needs. For more information on Tcl/Tk, access the following web sites:

- http://sunscript.sun.com/
- http://www.neosoft.com/tcl
- ftp://ftp.smli.com/pub/tcl/

Note: World Wide Web site locations often change and the addresses may not be available in the future.

OraTcl Description

Agent jobs and event scripts require both host system access for handling files and devices, launching programs, executing operating system functions, and accessing Oracle databases. OraTcl was developed to extend Tcl for Oracle usage and SNMP accessing. The categories of OraTcl functions are:

- SQL and PL/SQL functions
- RDBMS administration functions
- SNMP accessing
- Communication with the intelligent Agent and other Oracle software
- Character set conversion and error handling verbs
- General purpose utility functions

For descriptions of the OraTcl functions and variables, see *OraTcl Functions and Parameters* on page 5-12.

Example: OraTcl Script

The following example illustrates the basic use of OraTcl.

```
#!/usr/local/bin/Tcl -f
#
# monthly_pay.Tcl
#
# usage: monthly_pay.Tcl [connect_string]
#  or    Tcl -f monthly_pay.Tcl [connect_string]
#
# sample program for OraTcl
# Tom Poindexter
#
# example of sql, pl/sql, multiple cursors
# uses Oracle demo table SCOTT.EMP
# uses id/pass from command line,
# or "scott/tiger" if not specified
#
# this example does not illustrate efficient sql!
# a simple report is produced of the monthly payroll
# for each jobclass
#
global oramsg
set find_jobs_sql { select distinct job from SCOTT.EMP }
set monthly_pay_pl {
```

```
begin
  select sum(sal) into :monthly
 from SCOTT.EMP
 where job like :jobclass;
end;
}
set idpass $argv
if {[string length $idpass] == 0} {
  set idpass "scott/tiger"
}
set lda [oralogon $idpass]
set cur1 [oraopen $lda]
set cur2 [oraopen $lda]
orasql $cur1 $find_jobs_sql
set job [orafetch $cur1]
while {$oramsg(rc) == 0} {
  oraplexec $cur2 $monthly_pay_pl :monthly "" :jobclass "$job"
  set total_for_job [lindex [orafetch $cur2] 0]
  puts stdout "Total monthly salary for job class $job = \$ $total_for_job"
  set job [orafetch $cur1]
}
oraclose $cur1
oraclose $cur2
oralogoff $lda
exit
```

Server Message and Error Information

OraTcl creates and maintains a Tcl global array `oramsg` to provide feedback of Oracle server messages. `oramsg` is also used to communicate with the OraTcl interface routines to specify NULL return values and LONG limits. In all cases except for `NULLVALUE` and `MAXLONG`, each element is reset to NULL upon invocation of any OraTcl command, and any element affected by the command is set. The `oramsg` array is shared among all open OraTcl handles.

Note: `oramsg` should be defined with the global statement in any Tcl procedure that needs it.

`oramsg` Elements

The following are `oramsg` elements.

oramsg (agent_characterset)
The character set of the Agent, such as US7ASCII. This is used with the `convertin` and `convertout` verbs to convert character sets. See *convertin* on page 5-14 and *convertout* on page 5-15.

oramsg (db_characterset)
The character set of the database, such as US7ASCII. This is used with the `convertin` and `convertout` verbs to convert character sets. See *convertin* on page 5-14 and *convertout* on page 5-15.

oramsg (collengths)
A Tcl list of the lengths of the columns returned by `oracols`. `collengths` is only set by `oracols`.

oramsg (colprecs)
A Tcl list of the precision of the numeric columns returned by `oracols`. `colprecs` is only set by `oracols`. For non-numeric columns, the list entry is a null string.

oramsg (colscales)
A Tcl list of the scale of the numeric columns returned by `oracols`. `Colscales` is only set by `oracols`. For non-numeric columns, the list entry is a null string.

oramsg (coltypes)
A Tcl list of the types of the columns returned by `oracols`. coltypes is only set by `oracols`. Possible types returned are: CHAR, VARCHAR2 (Version 7), NUMBER,

LONG, rowid, DATE, RAW, LONG_RAW, MLSLABEL, RAW_MLSLABEL, or unknown.

oramsg (errortxt)
The message text associated with `rc`. Because the `oraplexec` function may invoke several SQL statements, there is a possibility that several messages may be received from the server.

oramsg (handle)
Indicates the handle of the last OraTcl function. The handle, a mapping in memory used to track commands, is set on every OraTcl command except where an invalid handle is used.

oramsg (jobid)
The job Id of the current job. Defined for job scripts only.

oramsg (language)
The NLS language of the Console, such as AMERICAN_AMERICA.US7ASCII.

oramsg (maxlong)
Can be set by the programmer to limit the amount of `LONG` or `LONG RAW` data returned by `orafetch`. The default is 32K Bytes. The maximum is 64K (Version 6) or 2147483647 (Version 7) bytes. Any value less than or equal to zero is ignored. Any change to maxlong becomes effective on the next call to `orasql`. See notes on `MAXLONG` usage with `orafetch`.

oramsg (nullvalue)
Can be set by the programmer to indicate the string value returned for any NULL result. Setting `oramsg`(nullvalue) to DEFAULT will return 0 for numeric null data types, such as INTEGER, FLOAT, and MONEY, and a NULL string for all other data types. `NULLVALUE` is initially set to `default`.

oramsg (ocifunc)
The number OCI code of the last OCI function called by OraTcl. See the *Programmer's Guide to the Oracle Call Interface* for descriptions.

oramsg (oraobject)
Contains the object upon which this script is acting. Defined for event scripts only.

oramsg (orahome)
The ORACLE_HOME directory.

oramsg (oraindex)
A Tcl list of the SNMP index values from the `snmp.ora` configuration file.

oramsg (orainput)
A Tcl list that contains the names of the job's input files. Probably most jobs will not need input files, but a job which invokes SQL*Plus with a SQL script, or Export with a specification file, would use this feature. Defined for job scripts only.

oramsg (rc)
Indicates the results of the last SQL command and subsequent `orafetch` processing. `rc` is set by `orasql`, `orafetch`, `oraplexec`, and is the numeric return code from the last OCI library function called by an OraTcl command.

See the *Oracle Error Messages and Codes Manual* for detailed information.Typical values are listed in Table 5–1, "Error Messages".

Table 5–1 Error Messages

Error	Meaning
0000	Function completed normally, without error.
0900 - 0999	Invalid SQL statement, invalid sql statements, missing key-words, invalid column names, etc.
1000 - 1099	Program interface error. For example, no sql statement, logon denied, or insufficient privileges.
1400 - 1499	Execution errors or feedback.
1403	End of data was reached on an `orafetch` command.
1406	A column fetched by `orafetch` was truncated. Can occur when fetching a LONG or LONG RAW, and the maxlong value is smaller than the actual data size.

oramsg (rows)
The number of rows affected by an insert, update, or delete in an `orasql` command, or the cumulative number of rows fetched by `orafetch`.

oramsg (sqlfunc)
The numeric OCI code of the last SQL function performed. See the *Programmer's Guide to the Oracle Call Interface* for descriptions.

oramsg (starttime)
The time at which the job was scheduled to be started. Defined for jobs only.

Use of Tcl with the Intelligent Agent

Tcl scripts are used by the Intelligent Agent for jobs and events. While both are Tcl scripts, they are distinct in the Agent and in the user interface.

Jobs are scripts scheduled to run once or multiple times. They typically cause side-effects, such as starting up a database, performing a backup, or sending output to the screen via the puts command, and can potentially have long execution times. Jobs can have output files and input files, such as a SQL script, while event scripts do not. Note that output files on Unix, DOS, or OS/2 are `stdout` redirected.

Event scripts, on the other hand, are used uniquely for detecting exceptions. A Tcl event script can monitor databases, host systems, or SQL*Net services by using a variety of means. If the script determines that a certain condition has occurred, it can send a return code to the Agent that states the severity of the event. Event scripts tend to run more frequently than jobs and so they are expected to have relatively short execution times. Also, it is assumed that event scripts do not cause any side effects.

While both jobs and events use Tcl to accomplish their tasks, they are very different in nature and as such have different execution environments. Specifically, on UNIX systems, jobs are forked into a separate process, while events are usually executed in-line with the Agent code.

The Tcl interpreter state is saved between executions and the value of Tcl global variables is preserved, for inline event scripts only, to give the illusion of a virtual process. This allows an event script to maintain a history so that the event does not get raised over and over again. For example, after you have notified the console that a value has gone above 90, you can refrain from notifying it again until the value goes below 80 and then back above 90. Database connections using the `oralogon` function are cached across all inline event scripts, so that repeated event scripts that use the same connect string can utilize the same connection.

Not all commands and global variables are available to both jobs and events. Jobs will not have the `oraobject` global variable that tells an event what service it is running against. Events will not have the `orainput` global that jobs use for SQL*Plus scripts.

NLS Issues and Error Messages

When a user registers for an event or schedules a job, the user's language preference is available to the Agent. There is a special remote procedure call which reports the language and current address of each console user. The Agent proceeds to issue an ALTER SESSION command to the specified language every time the `oralogon` function is called. This means that any subsequent messages or output coming from the Oracle server will be in the user's language. In addition, character set conversion is explicitly not done on the Agent, so that the Console can do it on the user's side.

If an event script or a job script fails execution, an error message is sent back to the Console in the user's language. Typically this will be an Oracle message returned by one of the Oracle Tcl extensions, if the verb was given inadequate parameters. For example `oralogon` might return the error: "ERROR: ORA-01017: invalid username/password; logon denied" if it is given an incorrect connect string. However, the error message could also be a Tcl specific message, such as: "ERROR: Tcl-00456: division by zero error", which will be stored in a message file and thus can be returned in the user's preferred language. The default language used by the Agent will be American English if no user language preference is specified or if an error message text does not exist in the user's language.

OraTcl Functions and Parameters

This section lists the OraTcl functions and parameters. Functions or other words that appear in OraTcl syntax are shown in this font: `function`. Parameters in square brackets '`[option]`' are optional, and the '|' character means 'or'. All parameters are passed into the functions and are IN mode.

- SQL and PL/SQL functions

oraautocom	*oracancel*	*oraclose*	*oracols*	*oracommit*
orafetch	*oralogoff*	*oralogon*	*oraopen*	*oraplexec*
orareadlong	*oraroll*	*orasql*	*orawritelong*	

- RDBMS administration functions

orastart	*orastop*

- SNMP accessing functions

oradbsnmp	*orasnmp*

- Communication with the Intelligent Agent and other Oracle software functions

orafail	*oragetfile*	*orainfo*	*orajobstat*	*orareportevent*

- Character set conversion and error handling functions

convertin	*convertout*	*msgtxt*	*msgtxt1*

- General purpose utility functions

catfile	*concatname*	*diskusage*	*echofile*	*export*
import	*loader*	*mvfile*	*orasleep*	*oratime*
rmfile	*tempdir*	*tempfile*		

Common Parameters

The following parameters are used in multiple OraTcl functions and the descriptions are provided in this section.

column
The column name that is the LONG or LONG RAW column.

connect_string
A valid Oracle database connect string, in one of the forms:

```
name | name/password | name@n:dbname | name/password@n:dbname
```

destaddress
`destaddress` is the destination address of the Agent.

filename
The name of the file that contains the LONG or LONG RAW data to write into the column or the name of the file in which to write the LONG or LONG RAW data.

logon-handle
A valid *cursor-handle* previously opened with `oraopen`. The handle is a mapping in memory used to track functions.

rowid
The Oracle database `rowid` of an existing row, and must be in the format of an Oracle `rowid` datatype.

table
The Oracle database table name that contains the row and column.

catfile

Purpose This function returns the contents of a file.

Syntax `catfile filename`

Parameters **filename**
The file that you want to display.

Example `catfile /tmp/files1 or c:/orant/sysman/admin/vobmgr.log`

concatname

Purpose This function returns the full pathname for a file given a list of file name components.

Syntax `concatname components`

Parameters **components**
A list containing the filename and each directory name where the file is located.

Example `concatname [list $oramsg(orahome) network agent]`

convertin

Purpose This function converts the parameter string from the client's (Console) character set to the destination character set. The function returns the converted string.

Syntax `convertin dest_characterset string`

Parameters **dest_characterset**
Destination character set. For database specific jobs or events, use $oramsg(db_characterset). For node specific jobs or events, use $oramsg(agent_characterset). See *oramsg Elements* on page 5-6.

string
The string that is converted.

Comments The client and the Agent node may use different languages or character sets. It is the responsibility of the Tcl script developer to perform the character set conversion. In general, all the job or event input parameters should be converted unless they are guaranteed to be ASCII.

convertout

Purpose This function converts the parameter string from the destination character set to the client's (Console) character set. The function returns the converted string.

Syntax `convertout dest_characterset string`

Parameters **dest_characterset**
Destination character set. For database specific jobs or events, use $oramsg(db_characterset). For node specific jobs or events, use $oramsg(agent_characterset). See *oramsg Elements* on page 5-6.

string
The string that is converted.

Comments The client and the Agent node may use different languages or character sets. It is the Tcl script developers' responsibility to perform the character set conversion. In general all the job or event output should be converted unless they are guaranteed to be ASCII.

diskusage

Purpose This function returns disk usage information on a specified file. The output is four lists: file systems, total space, available space, and mount points.

Syntax `diskusage file`

Parameters **file**
A file name. You can omit the filename to display information on the entire file system.

Example `diskusage [list /tmp]`

echofile

Purpose This function writes a string to a file.

Syntax `echofile string filename`

Parameters **string**
The string you want to write to the file.

filename
The name of the file where you want to store the string.

Example `echofile asdf /tmp/temp`

export

Purpose This function executes the Oracle Export database tool.

Syntax `export arguments`

Parameters **arguments**
These are the command-line arguments that are used by the Export tool.

Comments For information on Export, see the *Oracle7 Server Utilities User's Guide.*

import

Purpose This function executes the Oracle Import database tool.

Syntax `import arguments`

Parameters **arguments**
These are the command-line arguments that are used by the Import tool.

Comments For information on Import, see the *Oracle7 Server Utilities User's Guide.*

loader

Purpose This function executes the Oracle SQL*Loader database tool.

Syntax `loader arguments`

Parameters **arguments**
These are the command-line arguments that are used by the SQL*Loader tool.

Comments For information on SQL*Loader, see the *Oracle7 Server Utilities User's Guide*.

msgtxt

Purpose This function returns message text in the client's (Console) language for the given product name, facility and message number. The output is in the format of "FACILITY-ERROR : MESSAGE TEXT".

Syntax `msgtxt product facility error_no`

Parameters **product**
Product name. For example, `rdbms`.

facility
Facility name. For example, `ora`.

error_no
Error number. For example, `1101`.

Comments This function is used to put out error messages in the job output file. The message will be displayed in the client's (Console) language.

msgtxt1

Purpose This function returns a message in the client's (Console) language for the given product name, facility and message number. The output is in the format of "MESSAGE TEXT".

Syntax `msgtxt1 product facility error_no`

Parameters **product**
Product name. For example, `rdbms`.

facility
Facility name. For example, `ora`.

error_no
Error number. For example, `1101`.

Comments This function is used to put out confirmation messages in the job output file. The message will be displayed in the client's (Console) language.

mvfile

Purpose This function moves a file to a different location / name.

Syntax `mvfile filename destination`

Parameters **filename**
The name of the file you want to move or rename.

destination
The new destination / name.

Comments None

oraautocom

Purpose This function enables or disables automatic commit of SQL data manipulation statements using a cursor opened through the connection specified by `logon-handle`.

Syntax `oraautocom logon-handle {on | off}`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

Comments `oraautocom` raises a Tcl error if the `logon-handle` specified is not open.

Either `on` or `off` must be specified. The automatic commit feature defaults to off.

oracancel

Purpose This function cancels any pending results from a prior `orasql` function that use a cursor opened through the connection specified by `logon-handle`.

Syntax `oracancel logon-handle`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

Comments `oracancel` raises a Tcl error if the `logon-handle` specified is not open.

oraclose

Purpose This function closes the cursor associated with `logon-handle`.

Syntax `oraclose logon-handle`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

Comments `oraclose` raises a Tcl error if the `logon-handle` specified is not open.

oracols

Purpose This function returns the names of the columns from the last `orasql`, `orafetch`, or `oraplexec` function as a Tcl list. `oracols` may be used after `oraplexec`, in which case the bound variable names are returned.

Syntax `oracols logon-handle`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

Comments `oracols` raises a Tcl error if the `logon-handle` specified is not open.

The `oramsg` array index `collengths` is set to a Tcl list corresponding to the lengths of the columns; index `coltypes` is set to a Tcl list corresponding to the types of the columns; index `colprecs` is set to a Tcl list corresponding to the precision of the numeric columns, other corresponding non-numeric columns are a null string (Version 7 only); index `colscales` is set to a Tcl list corresponding to the scale of the numeric columns, other corresponding non-numeric columns are a null string (Version 7 only).

oracommit

Purpose This function commits any pending transactions from prior `orasql` functions using a cursor opened with the connection specified by `logon-handle`.

Syntax `oracommit logon-handle`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

Comments `oracommit` raises a Tcl error if the logon handle specified is not open.

oradbsnmp

Purpose This function retrieves SNMP MIB values.

Syntax `oradbsnmp get | getnext object_Id`

Parameters **object_Id**
`object_Id` can be either an actual MIB object Id, such as "1.3.6.1.2.1.1.1.0", or an object name with a possible index attached to it, such as "sysDescr" or "sysDescr.0".

Comments `oradbsnmp` is a function for retrieving SNMP MIB values maintained by the Agent, such as the RDBMS public MIB or the Oracle RDBMS private MIB. It does not write to the well-known UDP port for SNMP and obtains its values directly from the Agent's internal data structures. It works if the host does not have an SNMP master Agent running on it. See *orasnmp* on page 5-31 for more details on what `get` and `getnext` do. There are several reasons why `oradbsnmp` should be used instead of fetching the values from V$ tables with SQL commands:

- The Agent maintains a cache of MIB values fetched from the V$ tables to avoid burdening the RDBMS excessively. `oradbsnmp` is often faster than SQL and imposes less overhead on the system.
- When SGA access is implemented, it will be transparent to this function, for those MIB variables that are fetched directly from the SGA.
- In the case of `getnext`, the next `object_id` is the next `object_id` within the private and public RDBMS MIBs, and not one of another MIB. It is impossible to retrieve system-specific information using this function; use `orasnmp`.

orafail

Purpose This function forces a Tcl script to fail.

Syntax `orafail errormsg`

Parameters **errormsg**
`errormsg` can either be a quoted string of text or a string of the form: FAC-XXXXX where XXXXX is an Oracle message number for the given facility, such as VOC-99999.

Comments The error message will be used for display purposes on the client side.

orafetch

Purpose This function returns the next row from the last SQL statement executed with `orasql` as a Tcl list.

Syntax `orafetch logon-handle [commands]`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

commands
The optional commands allows `orafetch` to repeatedly fetch rows and execute commands for each row.

Comments `orafetch` raises a Tcl error if the `logon-handle` specified is not open.

All returned columns are converted to character strings. A null string is returned if there are no more rows in the current set of results. The Tcl list that is returned by `orafetch` contains the values of the selected columns in the order specified by select.

Substitutions are made on commands before passing it to Tcl_Eval() for each row. `orafetch` interprets @n in commands as a result column specification. For example, @1, @2, @3 refer to the first, second, and third columns in the result. @0 refers to the entire result row, as a Tcl list. Substitution columns may appear in any order, or more than once in the same command. Substituted columns are inserted into the commands string as proper list elements. For example, one space will be added before and after the substitution and column values with embedded spaces are enclosed by {} if needed.

A Tcl error is raised if a column substitution number is greater than the number of columns in the results. If the commands execute a break, `orafetch` execution is interrupted and returns with Tcl_OK. Remaining rows may be fetched with a subsequent `orafetch` function. If the commands execute return or continue, the remaining commands are skipped and `orafetch` execution continues with the next row. `orafetch` will raise a Tcl error if the commands return an error. Commands should be enclosed in "" or {}.

OraTcl performs conversions for all data types. Raw data is returned as a hexadecimal string, without a leading "0x". Use the SQL functions to force a specific conversion.

The `oramsg` array index `rc` is set with the return code of the fetch. 0 indicates the row was fetched successfully; 1403 indicates the end of data was reached. The index of rows is set to the cumulative number of rows fetched so far.

The `oramsg` array index maxlong limits the amount of long or long raw data returned for each column returned. The default is 32768 bytes. The `oramsg` array index nullvalue can be set to specify the value returned when a column is null. The default is "0" for numeric data, and "" for other datatypes.

oragetfile

Purpose This function is used by jobs to copy a remote file into a local file.

Syntax `oragetfile destaddress remotefile localfile [BIN]`

Parameters

destaddress
See *Common Parameters* on page 5-13.

remotefile
`remotefile` is the name of the file that is the source of the copy.

localfile
`localfile` is the name of the file that is the target of the copy.

Comments `oragetfile` fetches the file `remotefile` into the local file `localfile` from the Agent at `destaddress`. If the `BIN` argument is specified, the file is transferred in binary mode.

`destaddress` may be obtained from the `orainfo` function. Note that the address provided must be the spawn address of the Agent, the special address on which it listens for file transfer requests, and not the normal address used for all other RPCs.

Additional Information: For more information on the address of an Intelligent Agent, see the chapter on configuring the Agent in the *Oracle Enterprise Manager Installation Guide*.

orainfo

Purpose This function is used by jobs to get configuration information.

Syntax `orainfo destaddress`

Parameters

destaddress
See *Common Parameters* on page 5-13.

Comments `orainfo` fetches Agent configuration information from the Agent at `destaddress`. If `destaddress` is not present, then it is fetched from the Agent on the local machine. The Agent configuration is a Tcl list, as follows:

- A list of databases monitored by this Agent. The list includes the database name, `ORACLE_HOME`, and SID for each database.
- The Agent's normal RPC address, a tnsnames (TNS) string.
- The Agent's file transfer address, a TNS string.

orajobstat

Purpose This function is used by a job to send intermediate output back to the Console.

Syntax `orajobstat destaddress string`

Parameters

destaddress
See *Common Parameters* on page 5-13.

string
`string` can either be a quoted string of text or a string of the form: FAC-XXXXX where XXXXX is an Oracle message number for the given facility, such as VOC-99999. The string is used for display on the client side.

Comments `destaddress` is the address of the Agent, not the daemon. This function is issued from a job process, not from within an Agent process. The Agent's address can be obtained with `orainfo`.

oralogoff

Purpose This function logs off from the Oracle server connection associated with `logon-handle`.

Syntax `oralogoff logon-handle`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

Comments `oralogoff` raises a Tcl error if the logon handle specified is not open. `oralogoff` returns a null string.

oralogon

Purpose This function connects to an Oracle server using `connect_string`.

Syntax `oralogon connect_string`

Parameters

connect_string
See *Common Parameters* on page 5-13.

Comments A `logon-handle` is returned and should be used for all other OraTcl functions using this connection that require a `logon-handle`. Multiple connections to the same or different servers are allowed, up to a maximum of six total connections.

Additional Information: The connection limit is covered in the operating system-specific notes. When `oralogon` is used in an event script, it benefits from the connection cache. It will usually be able to reuse the connections opened by other event scripts against the same database. See *NLS Issues and Error Messages* on page 5-11 for details. `oralogon` raises a Tcl error if the connection is not made for any reason, such as login incorrect or network unavailable. If connect_string does not include a database specification, the value of the environment variable `ORACLE_SID` is used as the server.

oraopen

Purpose This function opens an SQL cursor to the server. `oraopen` returns a cursor to be used on subsequent OraTcl functions that require a `logon-handle`.

Syntax `oraopen logon-handle`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

Comments `oraopen` raises a Tcl error if the `logon-handle` specified is not open. Multiple cursors can be opened through the same or different logon handles, up to a maximum of 25 total cursors.

oraplexec

Purpose This function executes an anonymous PL block, optionally binding values to PL/SQL variables.

Syntax `oraplexec logon-handle pl_block [:varname value ...]`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

pl_block
`pl_block` may either be a complete PL/SQL procedure or a call to a stored procedure coded as an anonymous PL/SQL block.

:varname value
`:varname value` are optional pairs.

Comments `oraplexec` raises a Tcl error if the `logon-handle` specified is not open, or if the PL/SQL block is in error. `oraplexec` returns the contents of each :varname as a Tcl list upon the termination of PL/SQL block.

Optional :varname value pairs may follow `pl_block`. Varnames must be preceded by a colon, and match the substitution names used in the procedure. Any `:varname` that is not matched with a value is ignored. If a `:varname` is used for output, the value should be coded as a null string, "".

The `oramsg` array index rc contains the return code from the stored procedure.

orareadlong

Purpose This function reads the contents of a LONG or LONG RAW column and write results into a file.

Syntax `orareadlong logon-handle rowid table column filename`

Parameters

logon-handle rowid table column filename
See *Common Parameters* on page 5-13.

Comments orareadlong returns a decimal number upon successful completion of the number of bytes read from the LONG column.

orareadlong raises a Tcl error if the logon-handle specified is not open, or if rowid, table, or column are invalid, or if the row does not exist.

orareadlong composes and executes an SQL select statement based on the table, column, and rowid. A properly formatted Rowid may be obtained through a prior execution of orasql, such as "SELECT rowid FROM table WHERE ...".

orareportevent

Purpose This function is used by jobs to report an unsolicited event to the Agent and Event Management system in the Console. The oemevent executable can also be used.

Syntax orareportevent eventname object severity message [results]

Parameters eventname
eventname is the name of the event. This is the four-part name of the event in the form:

```
/vendor/product/category/name
```

You can enter any character string but all four parts and the forward slashes (/) are required.

The first two levels of name have special significance and have many predefined strings that Oracle script writers must use:

- Level one is the definer of this script, typically the integrating company name such as oracle, or user for unspecified customers.
- Level two is the name of the product to which this script is related, for example rdbms, office, agent, osgeneric, sqlnet, or hpux. All Oracle services have defined names which Oracle script writers must use.

The eventname is assumed to be in 7-bit ASCII, so that it never changes regardless of platform or language. See eventdef.tcl in the ORACLE_HOME\net8\admin directory (Oracle Enterprise Manager release 1.5.0 on a Windows NT platform) for a list of defined event names.

Note: The actual event script name may be shortened, upper-cased, or manipulated in other ways to make it a legal, unique filename on a given platform.

The format is operating system-specific. For example, `/oracle/rdbms/security/SecurityError` can be stored as `$oracle_home/network/agent/events/oracle/rdbms/security/securityerror.tcl` on a Unix system.

object
`object` is the name of the object that the event is monitoring, such as the database or service name listed in the `snmp.visibleservices` parameter in the `snmp.ora` file, or `$oramsg(nodename)`.

severity
`severity` is the level of severity of the event. For `orareportevent`, the value is `1` (warning), `2` (alert), or `-1` (clear). For `oemevent`, this is the literal text string `alert`, `warning`, or `clear`.

message
`message` is a quoted text string that is displayed in the Console, such as "File not found."

[results]
`results` is any results that may occur from the event. This is a Tcl list with the specific results for the event, such as the tablespace in error or the user who had a security violation.

Comments This is the method for any job to report an unsolicited event to the Agent, and back to the Console. . For information on the Event Management system, see the *Oracle Enterprise Manager Administrator's Guide*.

oraroll

Purpose This function rolls back any pending transactions from prior `orasql` functions that use a cursor opened through the connection specified by `logon-handle`.

Syntax `oraroll logon-handle`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

Comments `oraroll` raises a Tcl error if the logon handle specified is not open.

orasleep

Purpose This function causes the Tcl script to pause for a number of seconds.

Syntax `orasleep seconds`

Parameters **seconds**

Comments `orasleep` calls slcsleep() for the required number of seconds. There is no default, minimum, or maximum value.

orasnmp

Purpose This function performs either an SNMP `get` or `getnext` operation on the object specified by `object_id`.

Syntax `orasnmp get | getnext object_Id`

Parameters **object_Id**
The `object_Id` can be either an actual MIB object Id, such as "1.3.6.1.2.1.1.1.0", or an object name with an index attached to it, such as "sysDescr" or "sysDescr.0".

Comments Object names come from MIB text files. A full network manager, such as OpenView, has a MIB compiler that accepts MIB files and parses the ASN.1, creating a database of all objects in all the MIBs. The Agent needs to be simpler. There is a standard configuration directory which contains one or more two-column ASCII files of the format:

```
"rdbmsDbPrivateMibOID",   "1.3.6.1.2.1.39.1.1.1.2",
"rdbmsDbVendorName",      "1.3.6.1.2.1.39.1.1.1.3",
"rdbmsDbName",            "1.3.6.1.2.1.39.1.1.1.4",
"rdbmsDbContact",         "1.3.6.1.2.1.39.1.1.1.5",
....
```

The Tcl interpreter reads these files and does a binary search on them at runtime to resolve an object name to an `object_Id`.

The index values to use for Oracle services are configured via the `snmp.ora` file. These indices can also be obtained from the `oraindex` global variable. See *Server Message and Error Information* on page 5-6.

The result of `orasnmp` is a Tcl list of the form:

```
{object_id   value}
```

where `object_Id` is the object id associated with `value`. In the case of an `orasnmp get`, `object_Id` is the same as object, while for a `getnext`, it would be the next logical `object_Id`. It is assumed that the `orasnmp` operation applies to the local host only. The function actually sends out an SNMP query to the well-known SNMP port on the local host, so it is possible to query MIB variables other than Oracle's, such as those of the host or other applications that support SNMP. An SNMP Master Agent needs to be running on the local host for this function to work. See *oradbsnmp* on page 5-21 for an optimized way to retrieve the Oracle database MIB objects. If the Master Agent is not running, this function fails.

orasql

Purpose This function sends the Oracle SQL statement SQL statement to the server.

Syntax `orasql logon-handle sql_stmt`

Parameters

logon-handle
See *Common Parameters* on page 5-13.

sql_stmt
`sql_stmt` is a single, valid SQL statement.

Comments `logon-handle` must be a valid handle previously opened with `oraopen`. `orasql` raises a Tcl error if the `logon-handle` specified is not open, or if the SQL statement is syntactically incorrect.

`orasql` will return the numeric return code 0 on successful execution of the SQL statement. The `oramsg` array index rc is set with the return code; the rows index is set to the number of rows affected by the SQL statement in the case of insert, update, or delete. Only a single SQL statement may be specified in `sql_stmt`. `orafetch` allows retrieval of return rows generated. `orasql` performs an implicit `oracancel` if any results are still pending from the last execution of `orasql`.

Table inserts made with `orasql` should follow conversion rules in the Oracle SQL Reference manual.

orastart

Purpose This function starts an Oracle database instance.

Syntax `orastart connect_string [init_file] [SYSDBA|SYSOPER] [RESTRICT] [PARALLEL] [SHARED]`

Parameters

connect_string
See *Common Parameters* on page 5-13.

init_file
`init_file` is the path to the `init.ora` file to use.

Comments The default for `init_file` is:

```
ORACLE_HOME/dbs/init${ORACLE_SID}.ora
```

[SYSDBA | SYSOPER] are role flags for the user starting up the database. [RESTRICT] [PARALLEL] [SHARED] are database options. If [RESTRICT] is specified, database is started in restricted mode.

orastop

Purpose This function stops an Oracle database instance.

Syntax `orastop connect_string [SYSDBA|SYSOPER] [IMMEDIATE|ABORT]`

Parameters

connect_string
See *Common Parameters* on page 5-13.

Comments [SYSDBA | SYSOPER] are role flags for the user shutting down the database. [IMMEDIATE | ABORT] are the shutdown mode flags.

Note: Shutdown normal might be expected to fail every time, because the Agent maintains its own connection to the database, but we send a special RPC to the Agent when this is done, which causes it to disconnect from the database.

oratime

Purpose This function returns the current date and time.

Syntax `oratime`

Parameters None

Comments None

orawritelong

Purpose This function writes the contents of a file to a LONG or LONG RAW column.

Syntax `orawritelong logon-handle rowid table column filename`

Parameters

logon-handle rowid table column filename
See *Common Parameters* on page 5-13.

Comments `orawritelong` composes and executes an SQL update statement based on the table, column, and rowid. `orawritelong` returns a decimal number upon successful completion of the number of bytes written to the LONG column. A properly formatted ROWID may be obtained through a prior execution of the `orasql` function, such as "SELECT rowid FROM table WHERE ....".

`orawritelong` raises a Tcl error if the `logon-handle` specified is not open, or if rowid, table, or column are invalid, or if the row does not exist.

rmfile

Purpose This function removes a file.

Syntax `rmfile filename`

Parameters **filename**
This is the file that you want to remove.

Comments None

tempdir

Purpose This function returns a directory name that is used to store temporary files.

Syntax `tempdir`

Parameters **none**

Comments On Unix platforms, the `tempdir` is usually `/tmp`. On Windows NT systems, the tempdir is usually `\temp`.

tempfile

Purpose This function returns a temporary filename with the given extension. The filename is generated using the timestamp.

Syntax `tempfile extension`

Parameters **extension**
This is the file extension of the temporary file.

Example `tempfile dd`
returns the following on a Unix platform

```
/tmp/143153.dd
```

THE AGENT DOES NOT RETURN VERSION INFORMATION USING TCL VERB [DB_VERSION]

Certain advanced events must know the version of the database it is working on.

For the 8.1.3 Agent on NT

For the 8.1.3 Agent on NT, the version information is not returned using the Tcl verb [DB_VERSION] if you start the Agent service before starting the database or if you start the Agent service when the database or the listener servicing the database is down. Please make sure that the database and listener are up before starting the Agent service.

For the 8.1.5 Agent on Solaris

For the 8.1.5 Agent on Solaris, the version information is not returned using the Tcl verb [DB_VERSION] if you start the Agent service before starting the database or if you start the Agent service when the database or the listener servicing the database is down and you have not set your snmp_rw.ora file to recognize the state changes of the database.

To setup the 8.1 Solaris Agent to recognize the change in the state of the database, ensure that you have the following entry in the snmp_rw.ora file:

```
dbsnmp.noheuristic=1
```

To check the version of the database, type the following in the ORATCLSH, making sure that the oramsg (oraindex) is set to the correct database index of a running database:

```
ORATCLSH>oradbsnmp get applVersion.$oramsg(oraindex)
```

A

Configuration Files

This appendix discusses the configuration files that are generated by the intelligent agent and parameters that can be set to optimize agent operation for different system setups. The following topics are discussed:

- Configuration Files
- Parameters for snmp_ro.ora and snmp_rw.ora Files

Configuration Files

snmp_ro.ora

The `snmp_ro.ora` file is located in `$ORACLE_HOME\net80\admin` on Windows NT platforms. This file is located in `$ORACLE_HOME/network/admin` on UNIX. Do **NOT** update this read-only file.

snmp_rw.ora

The `snmp_rw.ora` is located in `$ORACLE_HOME\net80\admin` on Windows NT platforms. You can modify this read-write file, but this should be done carefully. It contains the following parameters:

```
SNMP.INDEX.service_name = unique_index_number
SNMP.CONTACT.service_name.world = "contact_info"
NMI.REGISTER_WITH_NAMES = FALSE
```

The following parameters are not automatically generated, but may be added to the file:

```
NMI.TRACE_LEVEL = OFF | USER | ADMIN | nn
SNMP.CONNECT.service_name.USER = user_name
SNMP.CONNECT.service_name.PASSWORD = password
SNMP.DBPOLLTIME = nn
DBSNMP.IPCTIME = nn
NMI.TRACE_DIRECTORY = directory
NMI.TRACE_FILE = filename
NMI.LOG_DIRECTORY = directory
NMI.LOG_FILE = filename
```

services.ora

The `services.ora` file is created when the agent starts and is located in `$ORACLE_HOME\network\agent` on the Windows NT platform and `$ORACLE_HOME/network/agent` on UNIX. This file contains a list of the services, such as Oracle databases and listeners, on the node where the agent resides. This file is retrieved from the agent by Oracle Enterprise Manager through the Navigator Discovery menu options.

Note: Do not manually edit the services.ora file. The agent rewrites the file on startup.

Parameters for snmp_ro.ora and snmp_rw.ora Files

These parameters are used in the `snmp_ro.ora` and `snmp_rw.ora` configuration files for the Intelligent Agent release. These parameters are also used in the `snmp.ora` file, the primary configuration file for Intelligent Agent releases prior to the 7.3.3 release.

In the following parameters, note these substitutions:

- *service_name* is the name of the service, database or listener, that you intend to monitor as it appears in `tnsnames.ora`.
- *host_name* is the host name of your machine.
- *world* is the name, if any, of your community.

Note: If *.world* is used in the `sqlnet.ora` file, then you must use *.world* in the `snmp.ora` and `tnsnames.ora` files. For example, *service_name.world* and *host_name_lsnr.world*.

SNMP.VISIBLESERVICES = (service_name1.world, service_name2.world, ...)
The name of the services that the agent is monitoring.

SNMP.INDEX.service_name.world = index_number
The unique index number of the service that the agent is monitoring. The index number can be any number. The only limitation is that if you have more than one index line, the index numbers must be unique. For example:

```
snmp.index.<service_name1>=10
snmp.index.<service_name2>=20
```

SNMP.SID.service_name.world = server_id
The server Id (SID) of the database service that the agent is monitoring.

SNMP.CONNECT.service_name.world.USER = user_name
The username that the subagent uses to connect to the database. The default is `dbsnmp`. This parameter is optional. The catsnmp.sql script should be edited and reexecuted if this parameter is not the default setting.

The "subagent" refers to the Intelligent Agent. Sometimes, the Intelligent Agent is called a subagent to the master SNMP agent when configuring SNMP on a server. However, SNMP does not have to be configured on the server before the Intelligent Agent will work (except for the Netware platform). For security reasons, the customers sometimes do not want to use the default Intelligent Agent database account / password of dbsnmp / dbsnmp. The example listed should only be used if they want to change the Intelligent Agent's database logon account.

SNMP.CONNECT.service_name.world.PASSWORD = password
The password for the username that is used by the subagent to connect to the database. The default is `dbsnmp`. This parameter is optional. The catsnmp.sql script should be edited and reexecuted if this parameter is not the default setting.

The "subagent" refers to the Intelligent Agent. Sometimes, the Intelligent Agent is called a subagent to the master SNMP agent when configuring SNMP on a server. However, SNMP does not have to be configured on the server before the Intelligent Agent will work (except for the Netware platform). For security reasons, the customers sometimes do not want to use the default Intelligent Agent database account / password of dbsnmp / dbsnmp. The example listed should only be used if they want to change the Intelligent Agent's database logon account.

SNMP.ORACLEHOME.service_name.world = *ORACLE_HOME_DIR*
The Oracle home directory of the database. A separate entry is required for each database even if `ORACLE_HOME_DIR` is the same for all services.

SNMP.CONTACT.service_name.world ="contact_info"
A string containing contact information, such as name, phone number, and email, of the administrator responsible for the service. This parameter is optional.

DBSNMP.POLLTIME = nn
The time interval (seconds) that the agent polls the database to check whether it is down. If the database has gone down or was never connected, this is the interval between retries. The default is 30 seconds.

Note: If the intelligent agent must monitor more than two instances, you should increase the value of DBSNMP.POLLTIME proportionally with the number of monitored instances.

For example:

The agent needs to monitor 10 instances. DBSNMP.POLLTIME should be set to 150. (10 / 2 * 30 = 150)

DBSNMP.IPCTIME = nn
The time interval (seconds) that the agent's Work process pings its Comm process to check whether it is down. If Comm is unable to respond to Work's ping within this time, Work will kill the old Comm process and spawn a new one. The default is 30 seconds.

The value of this parameter should be increased if false node up / down events, that trigger and clear within a short interval of time, regularly get reported to the Console. To check how frequently Comm gets restarted, execute the following command

```
ps -ef | grep dbsnmp
```

and note the start time of the child process.

This parameter does not pertain to the agent on Windows NT.

NMI.TRACE_LEVEL = OFF | USER | ADMIN | nn
Turns on tracing at the specified level. Oracle recommends that you set the trace level to 13. Level 16 produces a deluge of information, which is only useful if a bug is being investigated. With a level of 16, you can see actual TCP / IP packet contents. With a level of 15, I can only see that packets are being passed. This parameter is optional.

NMI.TRACE_DIRECTORY = *directory*
Directory where trace file is written. The setting is only relevant in conjunction with `nmi.trace_level`. If omitted, trace files are written to `$ORACLE_HOME\network\trace`. This parameter is optional.

NMI.TRACE_FILE = *filename*
Filename of the trace file. This parameter is optional.

NMI.LOG_DIRECTORY = *directory*
Directory where log file is written. This parameter is optional.

NMI.LOG_FILE = *filename*
Filename of the log file. This parameter is optional. On Windows NT, the filename defaults to `dbsnmp`.

Note: The following addresses are automatically set by the agent. Changing the addresses makes the agent undetectable by the Enterprise Manager Console and forces a manual configuration setup.

dbsnmp.address =(ADDRESS=(PROTOCOL=protocol) (HOST=host_ name)(PORT=port_no)))

The TNS address that the agent uses to listen for incoming requests. There should be no space or return characters in the address. This parameter is the address that the Agent listens on for network connections.

TCP/IP must be installed on the server since it is required to automatically discover services with the agent.

The agent requires PORT=1748. The port address 1748 is a registered TCP port granted to Oracle by the Internet Assigned Number Authority (IANA). The port address is automatically set. Changing this port makes the agent undetectable by the Enterprise Manager Console and forces a manual configuration setup.

For agent releases previous to the 7.3.3 release, this address must match exactly the entry for this agent in the tnsnames.ora file on the machine where the Oracle Enterprise Manager Console resides.

dbsnmp.spawnaddress =(ADDRESS= (PROTOCOL=protocol) (HOST=host_ name)(PORT=spnport_no)))

The TNS address which the agent can use to accept RPC's. This address is used for file transfers. The spnport_no used in this parameter is different than port_no used in the DBSNMP.ADDRESS parameter.

The agent PORT=1754. The port address 1754 is a registered TCP port granted to Oracle by the Internet Assigned Number Authority (IANA). Changing this port makes the agent undetectable by the Enterprise Manager Console and forces a manual configuration setup.

Glossary

MIB: Management Information Base.

A collection of SNMP Object ID's (OID) that are usually related

OID: SNMP Object ID

A period delimited sequence of numbers of the form `a.b.c...x.y.z`. It is a unique identifier for an item of information that is part of a MIB. Typically OIDs can have names associated with them. OIDs are hierarchical in nature. Hence `1.2.3` comes before `1.3` but after `1.2`. For example the OID that contains the number of physical reads an Oracle7 database has performed is:

```
oraDbSysPhysReads, 1.3.6.1.4.1.111.4.1.1.1.8
```

RDBMS Public MIB

A Standard MIB for relational databases agreed upon by the Internet Engineering Task Force (IETF). This MIB supports a variety of OIDs relating to relational databases in general such as the database name (eg. `rdbmsDbName, 1.3.6.1.3.55.1.2.1.4`)

Oracle Private MIB(s)

A MIB that is specific to Oracle products only.

SNMP: Simple Network Management Protocol.

A network protocol that manipulates OIDs. In the case of Oracle, only two primitive SNMP operations are supported: `get oid` which fetches the value of `oid` and `getnext oid` which gets the value of the next OID after `oid`.

Event

An event is a condition that can arise on either a database or a node monitored by an Intelligent Agent. For example, a database that suddenly goes down results in a DBDOWN event. Events can be detected in one of two ways: (1) By running Tcl scripts periodically that monitor for certain conditions or (2) By allowing a 3rd party to report the occurence of an event directly to the agent.

Job

A job is a Tcl script that can be executed once or on a re-occuring schedule. Unlike events which monitor for specific conditions, jobs are expected to accomplish a certain task. Example of jobs are: backup and start database.

Fixit Job

A special kind of job that is triggered by the occurence of an event. For example, if the tablespace full event detects that a tables space is over 90% full, the fixit job will be run automatically to add a datafile to the tablespace.

Index

A

agent
  intelligent, 1-2
  process, 1-2

C

catfile, 5-13
catsnmp.sql script, 2-9
character set
  agent, 5-6
  OraTcl conversion and error handling functions, 5-12
communication with agent functions
  OraTcl, 5-12
concatname, 5-14
configuration files
  services.ora, 2
  snmp_ro.ora, 2
  snmp_rw.ora, 2
convertin, 5-14
convertout, 5-15
creating
  Windows NT user account, 2-2

D

dbsnmp.address, 7
DBSNMP.POLLTIME, 4
dbsnmp.spawnaddress, 7
diskusage, 5-15

E

echofile, 5-16
export, 5-16

G

general purpose utility functions
  OraTcl, 5-12
GLOBAL_DBNAME parameter, 2-8

I

import, 5-16
installing
  Intelligent Agent, 1-2
Intelligent Agent, 1-2, 1-3
  configuring
    Windows NT, 2-4, 2-5
  installing
    UNIX, 2-5
    Windows NT, 2-2
  roles and users required, 2-9
  root permissions, 2-6
  TCP protocol, 7
  UNIX
    installation, 2-5
    installed as setuid root, 2-6
    starting and stopping, 2-7
  Windows NT
    creating user account, 2-2
    installation, 2-2
    starting and stopping, 2-4
Intelligent Agent data gathering service

Windows NT
starting and stopping, 4-6
Windows UNIX
starting and stopping, 4-6
Intelligent Agents
use of Tcl, 5-10

J

job and event scripts
Tcl Language, 5-2

L

language preference
NLS issues and error messages, 5-11
listener.ora files, 2-8
loader, 5-17
logon as batch job, 2-2

M

Management Information Base (MIB)
with SNMP, 1-3
msgtxt, 5-17
msgtxt1, 5-18
multiple database services
agent configuration, 4
snmp.ora parameters, 4
mvfile, 5-18

N

NLS environment, 1-3
NLS issues and error messages
about, 5-11
NMI.LOG_DIRECTORY, 6
NMI.LOG_FILE, 6
NMI.TRACE_DIRECTORY, 5
NMI.TRACE_FILE, 5
NMI.TRACE_LEVEL, 5

O

oemevent, 5-28
oraautocom, 5-19
oracancel, 5-19
Oracle Names, 2-8
Oracle server messages
about, 5-6
oramsg, 5-6
oraclose, 5-20
oracols, 5-20
oracommit, 5-20
oradbsnmp, 5-21
orafail, 5-21
orafetch, 5-22
oragetfile, 5-23
orainfo, 5-24
orajobstat, 5-25
oralogoff, 5-25
oralogon, 5-26
oramsg elements, 5-6
oraopen, 5-26
oraplexec, 5-27
orareadlong, 5-27
orareportevent, 5-28
oraroll, 5-30
orasleep, 5-30
orasnmp, 5-31
orasql, 5-32
orastart, 5-33
orastop, 5-33
oratab file, 1-5
OraTcl
character set conversion and error handling functions, 5-12
commands and parameters, 5-12
communication with agent functions, 5-12
description, 5-4
general purpose utility functions, 5-12
job and event scripts, 5-4
RDBMS administration functions, 5-12
SNMP accessing functions, 5-12
SQL and PL/SQL functions, 5-12
variables, 5-13
oratime, 5-34
orawritelong, 5-34

P

parameters
  OraTcl, 5-13
platform-specific installation documentation, 1-2
preface
  heading
    PH PrefaceHead, vii

R

RDBMS administration functions
  OraTcl, 5-12
rmfile, 5-34
root.sh, 2-6

S

script
  job and event, 5-2
services.ora, 1-4, 1-5, 2
setuid root, 2-6
SNMP
  and agents, 1-3
SNMP accessing functions
  OraTcl, 5-12
snmp*.ora file parameters, 3
snmp_ro.ora, 1-4, 1-5, 2
snmp_rw.ora, 1-4, 1-5, 2
SNMP.CONNECT.service_name.world.PASSWORD
  , 4
SNMP.CONNECT.service_name.world.USER, 4
SNMP.CONTACT.service_name.worl, 4
SNMP.INDEX.service_name.world, 3
snmp.ora
  parameters, 3
SNMP.ORACLEHOME.service_name.world, 4
SNMP.SID.service_name.world, 3
SNMP.VISIBLESERVICES, 3
SQL and PL/SQL functions
  OraTcl, 5-12
SQL*Net
  configuration files, 2

T

Tcl
  scripts executed by agents, 1-3
Tcl language
  description, 5-2
  job and event scripts, 5-2
  web sites, 5-3
Tcl scripting
  about, 5-2
  example, 5-4
TCP protocol, Intelligent Agent, 7
tempdir, 5-35
tempfile, 5-35
troubleshooting
  agent, 3-2

V

verifying root.sh run successfully, 2-6