Oracle® Intelligent Agent

User's Guide

Release 9.2.0.2

October 2002

Part No. A96676-02

ORACLE®

Oracle Intelligent Agent User's Guide, Release 9.2.0.2

Part No. A96676-02

Contents

3 Job and Event Scripts

B Troubleshooting

Send Us Your Comments

Oracle Intelligent Agent User's Guide, Release 9.2.0.2

Part No. A96676-02

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. 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?

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

- FAX: (650) 506-7200 Attn: Oracle System Management Products
- Postal service:
  Oracle Corporation
  Oracle System Management Products Documentation Manager
  500 Oracle Parkway, 50p5
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Purpose of this Guide

This manual provides configuration information and answers to crucial troubleshooting questions pertaining to the Oracle Intelligent Agent. The *Oracle Intelligent Agent User's Guide* is directed towards users of Oracle Enterprise Manager Version 2, 9.x, and any other supported system management frameworks that communicate with the Oracle database through the Intelligent Agent.

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 Organized

Chapter 1, "Intelligent Agent Overview"

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

Chapter 2, "Installation, Configuration and Usage"

Describes Agent installation and configuration procedures.

Chapter 3, "Job and Event Scripts"

Describes Job and Event scripting using the Tool Command Language (Tcl).

Appendix A, "Agent Configuration Files"

Describes requisite configuration files used by Oracle Enterprise Manager.

Appendix B, "Troubleshooting"

Provides Intelligent Agent troubleshooting guidelines and procedures.

Documentation Set

The Oracle Enterprise Manager Release 9*i* documentation includes the following:

- The *Oracle Enterprise Manager Readme* Release 9*i* provides important notes on updates to the software and other late-breaking news, as well as any differences between the product's behavior and how it is documented.
- The *Oracle Enterprise Manager Configuration Guide* Release 9*i* provides information about configuring the Oracle Enterprise Manager system.
- The *Oracle Enterprise Manager Concepts Guide* Release 9*i* provides an overview of the Enterprise Manager system.
- The *Oracle Enterprise Manager Administrator's Guide* Release 9*i* describes the components and features of the Oracle Enterprise Manager system.
- The *Oracle Intelligent Agent User's Guide* describes how to administer the Oracle Intelligent Agent.
- The *Oracle Enterprise Manager Messages Manual* Release 9*i* contains probable causes and recommended actions for Oracle Enterprise Manager errors.

In addition to the Oracle Enterprise Manager documentation set, extensive on-line help is provided for components in Oracle Enterprise Manager.

To download free release notes or installation documentation, please visit the Oracle Documentation Center at http://docs.oracle.com/

Printed documentation is available for sale in the Oracle Store at http://oraclestore.oracle.com/

Related Documents

For more information, see the following documents:

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

Oracle documentation is available online from the Internet at

`http://tahiti.oracle.com`

In North America, printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/admin/account/membership.html`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/docs/index.htm`

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

1

Intelligent Agent Overview

This chapter provides a brief overview of the Intelligent Agent and its characteristics.

- Oracle Intelligent Agent: An Overview
- Characteristics
- Simple Network Management Protocol Support

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 target as the services it supports and performs the following functions:

- Provides local services or calling operating system dependent services to interact locally with the managed targets.
- Checks for events, and queueing the resulting event reports for Oracle Enterprise Manager.
- Runs Oracle Enterprise Manager jobs, collecting their results and output, and / or queuing the results as required.
- Handle data collection.
- Cancels jobs or events as directed by the Console or other applications.
- Handles requests to send SNMP traps for events if SNMP is supported on the Intelligent Agent's platform.

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

Note: With version 9.0, the functionality of the Data Gatherer has been integrated into the Intelligent Agent and no longer exists as a separate application.

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.

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.

The Intelligent Agent uses specific cartridges to collect specific types of data. For example, operating system or database information. Agents can collect statistical date for discovered services on its node. Once requested, collections occur independent of the Oracle Performance Manager and Oracle Capacity Planner. Also collected are statistical data used for evaluation of operating system and database specific metric events registered from the Enterprise Manager Console.

Integrators can write their own cartridges to collect other types of data.

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*.

2

Installation, Configuration and Usage

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

- Installing the Intelligent Agent
- Controlling Operations of the NT Agent
- Configuring SNMP on Windows NT and Windows 2000
- Controlling Operations of the UNIX Agent
- Configuring SNMP for UNIX
- Configuring the 9i Agent for Use with Multiple Network Cards (NIC)
- Agent Behavior when Using Multiple Network Cards
- Oracle Intelligent Agent and Oracle Names
- Roles and Users Required by the Intelligent Agent
- Auto-Discovery
- Service Discovery Process
- Upgrading from 8.0.6/8.1.6/8.1.7 Intelligent Agents to 9.x

Installing the Intelligent Agent

The Intelligent Agent is shipped with the database and can be installed on remote, managed machines under an ORACLE_HOME environment. Using the Oracle Universal Installer, the Agent can be selected for installation from one of two locations: the Enterprise Manager tree list or the database server tree list (this option installs the database and the Agent). To install the Intelligent Agent as a standalone service, select the Agent from the Enterprise Manager tree list.

Controlling Operations of the NT Agent

The following procedures are used to control the operation of the Intelligent Agent on Microsoft Windows NT system.

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
```

When the Agent is started, the batch file *dbsnmpwd*.bat is executed automatically. *dbsnmpwd* is a Windows watchdog process that is responsible for automatically restarting the Intelligent Agent if the Agent goes down. This assures that the Agent

is up an running at all times unless explicitly shut down. You can configure dbsnmpwd by editing the batch file directly. For more information regarding configuration, see "DBSNMPWD" on page 2-9.

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.

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 managed target.

- 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.

Creating a New Windows 2000 User Account

To create a new user account on the local Windows 2000 machine and grant the "log in as batch jobs" privilege to this user, perform the procedure below.

1. Control Panel-->Administrative Tools-->Local Security Policy-->LocalPolicies-->User Rights Assignment (On the right hand side, highlight "Log on as bacth job".
2. Right click on "Log on as batch job" and select Security.
3. Click on Add and select a user to add.
4. Click on Add after selecting user and click OK)

Configuring a Domain User as the Agent User

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

To configure a domain 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.

Configuring SNMP on Windows NT and Windows 2000

The following procedure explains how to configure the SNMP master agent to support the sub agents on Windows NT and Windows 2000:

1. Go to the drive that contains your services file say

   x:\winnt\system32\drivers\etc\services.

   Change the snmp entries to:

   ```
   snmp 1161/udp
   snmp-trap 1162/udp
   ```

2. Go to the Peer SNMP Master Agent config file MASTER.CFG under the

   `ORACLE_HOME\network\admin`
   directory and edit the config file to add the following:

   a. A transport entry as follows:

      ```
      TRANSPORT ordinary SNMP
      OVER UDP SOCKET
      AT PORT 1161
      ```

   b. An entry to indicate the community and the machine (example: dlsun1000.us.oracle.com or an IP address) that should receive the snmp traps:

      ```
      COMMUNITY   public
      ALLOW ALL OPERATIONS
      USE NO ENCRYPTION

      MANAGER trap_machine.acme.com
      SEND ALL TRAPS
      WITH COMMUNITY PUBLIC
      ```

3. Start the Peer SNMP Master Agent and the encapsulator from the Services Panel in the Control Panel.

   Please note that the encapsulator is only needed when multiple sub-agents are installed and configured on the machine. The Peer SNMP Master Agent

executable is ORACLE_HOME\bin\agent.exe and the encapsulator executable is ORACLE_HOME\bin\encaps.exe.

4. Start the sub-agent (Oracle Intelligent Agent) through the Services Panel under Control Panel. The sub-agent starts and registers itself with the master agent.

   The way to test Oracle's SNMP is by using a third party application which uses snmp to communicate with the Oracle SNMP master agent and query Oracle specific data.

Third party applications that support SNMP are listed in the Oracle SNMP Support Reference Guide

Controlling Operations of the UNIX Agent

The following procedures are used to control the operation of the Intelligent Agent on UNIX systems.

Running the root.sh Shell Script

After you have successfully installed the Agent, the Oracle Universal 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.

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.

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-s---   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-s---` are the permissions, then `root.sh` had been run successfully.

Starting and Stopping the Agent on UNIX Platforms

On UNIX, the `agentctl` utility is used to start and stop the Agent. In addition, *dbsnmpwd* is run. *dbsnmpwd* is a UNIX watchdog script that is responsible for automatically restarting the Intelligent Agent if the Agent goes down. This assures that the Agent is up an running at all times unless explicitly shut down.

The relevant `agentctl` commands are listed in the table below.

If you want to...	Enter the command...
Start the Agent and *dbsnmpwd*.	agentctl start agent
Stop the Agent and *dbsnmpwd*.	agentctl stop agent
Verify status of the Agent	agentctl status agent
Stop and then restart (bounce) the Agent. This option will only restart the Agent if it is already running.	agentctl restart agent

DBSNMPWD

Dbsnmpwd is UNIX watchdog script that ensures the *dbsnmp* (Intelligent Agent) process is always present on monitored targets. It restarts the agent when it exits abnormally i.e. exits with an unexpected return code.

The behavior of the watchdog script can be configured using the following environmental variables. These variable are contained within the script itself.

Note: Environmental variables should not be set within the snmp_rw.ora file.

DBSNMP_WDLOGFILE - Logfile where startup messages are written. It defaults to $ORACLE_HOME/network/log/dbsnmp.nohup

DBSNMP_RESTART - To disable the automatic restart mechanism set this environmental variable to 0. Default is set to 1.

DBSNMP_MAX_ABNORMALEXIT / DBSNMP_TIME_DELTA - These two variables help the watchdog script to determine when it would be safe to assume that the agent is thrashing (the process of starting and immediately exiting because initial conditions to start the agent successfully are not met). The default values are 3 and 60 respectively. It means if the agent exits more than 3 times within 60 seconds then it would be safe to assume that the agent is thrashing and the watchdog would not restart it.

Specifying a password for encrypting Agent files

By default, the agent files are encrypted using the hostname (of the host on which the agent resides) as a key or password for encryption. To specify a different password to be used for encryption, start the agent as follows:

```
agentctl start agent password=<new password>
```

Example:

```
% agentctl start agent password=mynewpassword
```

To prevent people from doing a "ps -efo" and looking at the command-line arguments to get the password, there is an option of putting the password in a file and passing the file as an argument to `agentctl`. This file can then be permission protected to prevent viewing by unauthorized users. Only the first 8 characters of the file are used as the password. The usage is as follows:

```
% agentctl start agent password_file=<location of password file>
```

Example:

```
% agentctl start agent password_file=/u1/myhome/newpassword.txt
```

Blackouts

Blackouts allow Enterprise Manager users to suspend any or all management and/or data collection activity on one or more managed targets. This capability permits maintenance or emergency operations to be performed

Specifically, blackouts can suspend:

- **Events**: All events registered on a target will not be evaluated or triggered for the duration of the blackout.
- **Jobs**: All jobs submitted to a target will not be scheduled or run for the duration of the blackout. Job skipped notifications will be sent to the Enterprise Manager Console for regular interval jobs scheduled during a blackout on a target.
- **Data Collections**: All current historical data collection activities for a target are stopped. However, loading of data collected for a target **prior** to the blackout will continue as long as the database is up. New collections can be submitted but will not proceed unless the blackout ends.

Defining Blackouts

Blackouts must be created at the target-level, i.e., they must be defined on the node where the Intelligent Agent resides. Blackouts are controlled with a command line interface. The blackout subsytem associates any command line request with a special type of Agent user called the CLI user. Only one immediate blackout can be set at a time. Multiple target blackouts can exists simultaneously.

Once in effect, blackouts cannot be modified. To change the status of a particular blackout, you must first delete the existing blackout and then re-create a new blackout with the desired changes.

Important: A blackout can only be cancelled by the user who originally set it.

Blackout Command Line Interface

The blackout command line tool exists on the node where the Agent resides and can be used by administrators to set/cancel blackouts. The Intelligent Agent must be running in order to set a blackout. User commands are as follows:

Table 2–1 Blackout Commands

Blackout Action	Command
Define a blackout.	agentctl start blackout [-d [DD] HH:MM] [<target name>] where the -d option is used to specify duration in the format DD HH:MM where: *DD* indicates number of days which is optional *HH* indicates number of hours *MM* indicates number of minutes
Remove a blackout	agentctl stop blackout [<target name>]
Display the status of a blackout (machine name, target type, and duration).	agentctl status blackout
Blackout specific subsystems (jobs, events, historical collections)	agentctl start blackout <target> -d<uration> -s<ubsystem/s> By default, all subsystems (jobs and events and collections) will be blacked out.

Command Line Examples

The following examples illustrate how to use the blackout command line utility under different situations and the output generated.

Situation 1: You want to start a blackout on the target vnukal-pc.world for 10 minutes.

```
$ agentctl start blackout -d 0:10 vnukal-pc.world
Blackout registered on vnukal-pc.world database
```

Situation 2: You want to stop any blackout set by (the CLI user) effective on target vnukal-pc.world. Note: Blackouts set by other Agent users are not cancelled.

```
$ agentctl stop blackout vnukal-pc.world
Blackout canceled on vnukal-pc.world database
```

Situation 3: You want to start blackouts on all managed targets for an indefinite length of time. This is equivalent to blacking out the entire Agent.

```
$ agentctl start blackout
Do you wish to blackout the entire agent (Y/N) ? [N] y
All targets on the agent are blacked out.
```

Situation 4: You want to stop blackouts on all registered targets.

```
$ agentctl stop blackout
Do you wish to cancel blackout on all targets (Y/N) ? [N] y
Blackout canceled on all targets.
```

Situation 5: You want to know the status of blackouts on a target.

```
$ agentctl status blackout
vnukal-pc.world is blacked out. Blackout will end in 2 hours.
```

Situation 6: You want to set a blackout on a target whose name matches another target of a different type. In this case, the command line interface allows you to interactively select the desired target.

```
$ agentctl start blackout payroll
Following targets matching "payroll" have been found.
1.  payRoll ( Database )
2.  payRoll ( Listener )
Choose the target to blackout [1] : 2
Blackout registered on "payroll" listener
```

Situation 7: You want to blackout events on target 'mytarget' for 30 minutes.

```
$ agentctl start blackout mytarget -d 0:30 -s events
```

Configuring SNMP for UNIX

On UNIX systems, when you install the Oracle Intelligent Agent (IA), the SNMP files are also installed. SNMP is not required if you are running an Enterprise Manager-only environment. However, if you are running a third-party SNMP monitoring system, you will need to configure the Intelligent Agent to communicate with the SNMP Master Agent or the Peer Encapsulator, depending on your platform. To configure SNMP:

1. Install the Intelligent Agent.

Make sure there is no SNMP master agent already running (snmpd) on the

```
ps -ef | grep snmp
```

If there is an snmpd (`snmpd.cfddi` on Solaris) process running, stop it.

2. cd $ORACLE_HOME/network/snmp/peer
3. Edit the `config.master` script with the IP address of the machine that is running the SNMP Console in the same directory, verify that a script start_peer exists.
4. su root
5. start_peer -a (starts the Master Peer agent, the encapsulator and the native SNMP daemon)
6. Exit from root.
7. From the Enterprise Manager Console, register an event and check the box: "Enable Notifications to External Services (SNMP traps by Agent)". Traps will be sent to the SNMP Master Console for each EM event that triggers.

Note:

- The SNMP configuration differs from platform to platform, check the platform documentation (the Installation and
- Configuration Guides usually have this information)
- Some UNIX platform do NOT offer SNMP support; check the platform documentation

---

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. SNMP configuration may differ depending on the platform.

Configuring the 9*i* Agent for Use with Multiple Network Cards (NIC)

As with version 8.1.7 of the Intelligent Agent, 9i Intelligent Agent users have three options to configure the Agent on a machine with multiple network cards. By default the Agent will bind to the primary NIC on its machine ('le0' on UNIX platforms and 'network0' on Windows NT platforms). The other two options are:

a. Ability to bind to a NIC specified by the user.

b. Ability to bind to all NICs on the machine. This option should not be used if it is not desirable to have the Agent listening on all NICs.

Note: The Agent binds to an ip address and uses that address, to listen for all incoming requests for executing EM jobs, events, and data collection requests.

The Agent will also have the capability of discovering services (listeners etc.) that are listening on an ip address / NIC that's different from the ip address / NIC being used by the Agent.

Agent Behavior when Using Multiple Network Cards

If no Listening Address is Specified

When no explicit listening address directives are in snmp_rw.ora, the Agent listens for connections via the Agent machine's primary network interface card.

If a listening IP Address is Specified

When an explicit listening address is specified in *snmp_rw.ora*, the Agent listens for connections on only that address.

To bind the 9i Intelligent Agent to a specific network interface card, other than the primary network card:

1. Set the dbsnmp.hostname parameter.

```
dbsnmp.hostname=<IPaddress of the network card>
```

2. Start the Agent by entering the following at the command line.

```
>agentctl start agent
```

If a hostname is Specified

If the hostname is specified in *snmp_rw.ora*, the Agent listens for connections on all the machine's network interface cards.

For pre-8.1.7 versions of the Agent (8.1.5 or higher) this is the default behavior of the Agent (since it is the default behavior of the network layer code used by the Agent) .

To bind the 9i Intelligent Agent to all network interface cards on a host:

1. Set the dbsnmp.hostname

```
dbsnmp.hostname=<name of host>
```

2. Start the Agent

A Windows NT FailSafe Configuration is Used

In the Windows NT FailSafe configuration, the Agent listens for connections on the IP address stored in the NT registry for the FailSafe Agent

The Agent discovers each target on a machine, regardless of which of the machine names or IP addresses is used in the target's configuration files.

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 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 Intelligent Agent

The default database username/password for the Agent is dbsnmp/dbsnmp. The `catsnmp.sql` script is installed with the database. The database roles and privileges assigned to the Agent user using the `catsnmp.sql` script. When an Oracle database is installed, the `catsnmp.sql` script is automatically run by `catalog.sql`

The customer may need to change the user/password for the Intelligent Agent's database logon. To change the user name and password to something other than dbsnmp/dbsnmp, you need to edit `snmp_rw.ora`, adding the following parameters:

```
snmp.connect.<svcname>.name = <username>
snmp.connect.<svcname>.password = <password>
```

Note: <svcname> is the database service name as it appears in the Enterprise Manager Console or in the snmp_ro.ora file.

To grant the requisite roles and users privileges to the new Agent user, run the individual SQL commands specified in `catsnmp.sql` referring to the new Agent user. You can use Server Manager or SQL Plus. Do not edit the `catsnmp.sql` script for the sole purpose of performing this task.

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.

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 9.x database, the script is located at

`ORACLE_HOME\rdbms\admin`.

Auto-Discovery

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. With version 9.0, this file also contains information about the type of operating system and the operating system version of the environment in which the Agent is running.

  Note: Please refer to Appendix A, "Agent 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.

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: 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

- Oracle Net TCP/IP must be present, and the necessary files must be created, prior to launching the Intelligent Agent. The only required SQL*Net (or Oracle Net) 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

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 Intelligent 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.

Discovering Additional Services

Along with database and listener services, the Intelligent Agent also discovers other services (concurrent managers, forms servers, application servers, etc.). These services are discovered through individual discovery scripts that are defined in the nmiconf.lst file. The nmiconf.tcl parses the nmiconf.lst to see if any non-basic services have been installed on the server and then runs all discovery scripts that it finds listed in this file.

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. The Agent searches for the listener.ora file:

- If the tns_admin environment variable is set, the Agent searches for the listener.ora file in this directory
- If the tns_admin environment variable is not set, the Agent searches for the listener.ora file in `/etc` or `/var/opt/oracle`
- If neither directory (/etc or /var/opt/oracle) exists, the agent searches for the listener.ora file in the `/network/admin` directories of the ORACLE_HOMEs listed in oratab file.

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.

Real Application Cluster Environments

In Real Application Cluster environments, multiple instances of the Intelligent Agent (one per node) must share a single ORACLE_HOME. In order to support this configuration, files created by the Agent (queue files, logs and traces, CS recovery files, configuration files) must be stored in an alternate directory outside the ORACLE_HOME.

To override the default ORACLE_HOME location, set the ORA_OEMAGENT_DIR environment variable to a new directory outside of the ORACLE_HOME.

Example (UNIX environment):

```
setenv ORA_OEMAGENT_DIR /private/agentstate
```

Upgrading from 8.0.6/8.1.6/8.1.7 Intelligent Agents to 9.x

Each release of the Intelligent Agent improves Agent performance, functionality, and reliability. We therefore recommend upgrading your Intelligent Agent to the latest version available for your platform. To make sure the transition to a newer Agent preserves your existing Enterprise Manager jobs, events, and data collections, you can use the NMUMIGR8 utility found in $ORACLE_HOME/bin. This utility

allows you to migrate existing jobs, events, and data collections to a format recognized by 9.x version of the Intelligent Agent.

Usage:
`nmumigr8 [-source_home <source ORACLE_HOME>] [-verbose]`

Parameters

-source_home
Source Oracle Home that contains an existing Intelligent Agent queue and data collection files. If source_home is not supplied then the value defaults to the destination Oracle Home that is the value of the ORACLE_HOME environment variable.

-verbose
This flag indicates that detailed migration information should be written to the trace file `nmumigr8.trc` located under the Agent's `ORACLE_HOME/network/trace` directory. If this flag is not set, only summary information is written to the trace file.

Users upgrading from earlier versions of the Intelligent Agent to the version 9.0 Intelligent Agent should follow the guidelines described in the next section.

Best Practices: Agent Installation and Configuration

Several database products can be installed on a single machine, all of different core versions, and any of which can be upgraded or patched independently. Because the Intelligent Agent can manage multiple products, adhering to Agent implementation, upgrade and mainentance best practices will optimize Agent performance and ensure trouble-free operation.

Agent Installation

Install Only One Agent Per Machine

Although it is possible to have several versions of the Intelligent Agent installed on a single machine, only one of these versions can be active at any time. It is recommended that you only install one version of the Intelligent Agent per machine. When several versions of the Agent are present on one machine, there is a danger that the wrong Agent will be started, causing all job and event registrations and notifications to be cross-sent to different versions of the Agent. This situation

will lead to synchronization problems between the Management Server and the Intelligent Agent. The only solution at this point is to drop the node from the navigator, and clean-start the Intelligent Agent to re-synchronize the information. See "Clean Starting the Intelligent Agent" on page B-39 for more information

Create a Separate Agent User

Create a user at the Operating System level that will own the Agent installation and will control (start and stop) the Agent. The username (not necessarily its password) should be the same on all managed nodes (for example, oramon). The Agent owner needs to be a member of the 'dba' group for all monitored databases on its node, although none of them has to be its primary group.

The profile of the Agent Owner should set the following environment variables. The example of Listing 1 assumes that csh is the default shell of the Agent owner and lists the contents of the .cshrc file. A similar-looking structure should be used in the .profile file if the Bourne Shell or Korn Shell are used as the default shell of the Agent Owner.

Example 2–1 Profile of the ORAMON Operating System User

```
umask 002
# provide the appropriate value for ORACLE_HOME
setenv ORACLE_HOME /oracle
setenv PATH ${ORACLE_HOME}/bin:/usr/ccs/bin:${PATH}
setenv TCL_LIBRARY ${ORACLE_HOME}/network/agent/tcl

# unset ORACLE_HOME, LD_LIBRARY_PATH and NLS_LANG
unsetenv ORACLE_SID
unsetenv LD_LIBRARY_PATH
unsetenv NLS_LANG

# if a central location for net8 config files exist,
# set it here otherwise, make sure TNS_ADMIN is not set
# setenv TNS_ADMIN /usr/local/oracle/admin
unsetenv TNS_ADMIN

# point ORATAB to an OEM-specific oratab file if such exists
if ( -e ${ORACLE_HOME}/network/agent/oratab.oem ) then
  setenv ORATAB ${ORACLE_HOME}/network/agent/oratab.oem
endif
```

The TNS_ADMIN environment variable should point to the central location of all Oracle Net configuration files, if such a location exists. If not, make sure that the TNS_ADMIN variable is not set.

Install the Agent in Its Own ORACLE_HOME

In order to avoid conflicts during maintenance upgrades of the various pieces of software, and to be able to upgrade an individual product independently, it is recommended that you install the Intelligent Agent in its own ORACLE_HOME.

It is also recommended that the standard operating system user, "oramon" be used to install the Agent on all machines in your environment. You need to change both the owner of the directory and the group to which the directory belongs. To do this:

1. Create a new *ORACLE_HOME* directory.

   ```
   > mkdir oemagent_9.2.0
   ```

2. Change the owner to 'oramon'.

   ```
   > chown oramon oemagent_9.2.0
   ```

3. Change the group to 'oramon'.

   ```
   > chgrp oramon oemagent_9.2.0
   ```

Starting from Oracle 8.1.5, the Universal Installer is used to install the Oracle software. To install the Intelligent Agent using the Universal Installer, perform the following steps:

1. In the Installer's File Location window, specify a new ORACLE_HOME name, and a new physical location.
2. Perform a 'Custom Install'
3. In the product list, select the following products:

   Oracle Enterprise Manager Products --> Oracle Intelligent Agent

4. Deselect all other products. Expand all elements of the product list to make sure only the Intelligent Agent is being installed. Deselecting only the top level entry for a product does not guarantee lower-level elements have been deselected.
5. Finish the installation, and perform the OS specific post-install tasks, such as running the root.sh script on UNIX.

Note: If you decide to install the Agent in an Oracle Home which has an Oracle Database Server, make sure that the version of the Agent matches the database version. It is NOT required to shutdown a database while the Agent is being installed. If you decide to de-install the Agent, this will NOT remove any components that the Oracle Database Server is dependent on.

Use Symbolic Links to the Agent Agent Home

A standard Agent ORACLE_HOME should be used on all machines, such that the ORACLE_HOME is a symbolic link to the location of the current Agent. Thus, when an Agent is upgraded to another version, the link will then point to the location of the latest Agent. This will help maintain a consistent ORACLE_HOME for the Agent on every machine at all times.

For example, if using version 9.2.0 of the Agent, set ORACLE_HOME to point to the directory /u01/app/oracle/product/oemagent which is a symbolic link to the directory /u01/app/oracle/product/oemagent_9.2.0.

```
ln -s  /u01/app/oracle/product/oemagent /u01/app/oracle/product/oemagent_9.2.0
```

Agent Configuration

Disable Unused Agents

To prevent accidental starting of an older Agent, rename the DBSNMP executable of all older versions of the Agent. This will prevent someone from accidentally starting the wrong Intelligent Agent.

Auto-Start the Agent Upon Machine Startup

Your Agent should be configured to start automatically upon machine startup/reboot. Include a script like the one listed below in your boot up procedures so that the Intelligent Agent is started automatically at machine bootup time.

Example 2–2 Sample Agent Boot Time Startup Script (UNIX)

```
#!/bin/sh

PATH=/usr/bin:/usr/sbin:/sbin; export PATH

# modify the next line to point to the agent's ORACLE_HOME
```

```
ORACLE_HOME="/u01/app/oracle/product/oemagent"

# find the agent owner by looking at file ownership
USERID=‘ls -ld $ORACLE_HOME/bin | awk ’{print $3}’‘

case $1 in
’start’| ’’)
        su - $USERID -c "$ORACLE_HOME/scripts/oemagent start"
        "Starting OEM agent"
        ;;
’stop’)
        su - $USERID -c "$ORACLE_HOME/scripts/oemagent stop"
        "Stopping OEM agent"
        ;;
*)
        echo "usage: $0 {start|stop}"
        ;;
esac
```

Agent Compatibility

To ensure Agent-Enterprise Manager-RDBMS compatibility, and to access all features available for a given version of the Enterprise Manager framework, we recommended using the Intelligent Agent that matches the version of the framework. The following table lists Agent version compatibility.

Table 2–2 Agent Compatibility

Database Version	Enterprise Manager Version	Agent Version
7.3.4.x	1.3.4	7.3.4.x
8.0.5.x, 7.3.4.x	1.6.0	8.0.5.x
8.0.6.x, 8.0.5.x, 7.3.4.x	1.6.5	8.0.6.x
8.1.5.x, 8.0.6.x, 8.0.5.x, 7.3.4.x	2.0.4	8.1.5.x
8.1.6.x, 8.1.5.x, 8.0.6.x, 8.0.5.x, 7.3.4.x	2.1	8.1.6.x
8.1.7.x, 8.1.6.x, 8.1.5.x, 8.0.6.x, 8.0.5.x, 7.3.4.x	2.2	8.1.7.x
9.2.0, 9.0.1, 8.1.7.x	9.2.0	9.2.0

After Agent installation is completed, the SQL*Net / Oracle Net files for the Intelligent Agent (SQLNET.ORA and TNSNAMES.ORA) have to be created / updated, to be able to contact all the databases the Agent needs to manage. Also, by default the Agent will use the ORATAB file (unless the TNS_ADMIN environment variable is set) on its machine to find the ORACLE_HOMES and

discover services. If you want the Agent to find a subset of ORACLE_HOMES and discover corresponding services, we recommend creating another file that contains this subset. The ORATAB environment variable should be set in the Agent's environment and should point to this file prior to starting the Agent.

General Agent-Enterprise Manager Upgrade Process

1. Install the latest Intelligent Agent under a new Oracle Home. See above.
2. Make sure that any jobs or events you wish to keep have been saved in the job or event library respectively. To add a job / event to a job / event library, select the job / event from the job / event pane, click on the desired entry using the right mouse button and select Copy to Library from the context-sensitive menu.
3. Move any event alerts to event history. You can save the contents of the history pane or clear them.

> **Note:** If you have events registered against multiple targets, use the Create Like menu option to create individual events for each target and save these events to the Event Library.

4. From the Enterprise Manager Console, de-register any existing events and remove any active jobs scheduled against the node on which you are upgrading the agent.
5. Shut down the old Agent.
6. Start the new Agent
7. From the OEM Console, refresh the node in the Navigator.
8. Resubmit the saved jobs and events to the new Agent.

3

Job and Event Scripts

Topics covered in this document include:

- Scripting Language
- Server Message and Error Information
- Event to Fixit Job Tcl Array
- 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://www.sun.com/960710/cover/testimonials.html
- http://www.neosoft.com/tcl
- http://www.scriptics.com/resource/doc/papers/

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 3-14.

Example: OraTcl Script

The following example illustrates the basic use of OraTcl.

```
#
# 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} {
  set total_for_job [lindex [oraplexec $cur2 $monthly_pay_pl :monthly ""
:jobclass "$job"] 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 3-16 and "convertout" on page 3-17.

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 3-16 and "convertout" on page 3-17. This variable can only be used after an `oralogon` function within a Tcl script.

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 (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 *Oracle9i Database Error Messages* for detailed information.Typical values are listed in Table 3–1, "Error Messages"

Table 3–1 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 (starttime)
The time at which the job was scheduled to be started. Defined for jobs only.

Event to Fixit Job Tcl Array

OraTcl creates and maintains a Tcl global array `trigevent` to pass a Tcl array to an Enterprise Manager Fixit job. The `trigevent` array can also be used from within the Fixit job itself.

`trigevent` Element

The following are trigevent elements:

trigevent (name)
Name of the event that caused the Fixit job to be fired.

trigevent (object)
Target or node in your network.

trigevent (arguments)
Arguments to the triggering event (varies according to the event)

trigevent (results)
Results of the event. For example, this element returns the number 35 indicating 35 percent for the CPU Utilization event test.

trigevent (severity)
Severity of the event as indicated by the following numbers: -1 (*Clear*), 1 (*Warning*), 2 (*Alert*)

Warning: `trigevent` only works with Fixit jobs. If this array is passed to a non-Fixit job, `trigevent` will be undefined (NULL) and the job will fail. For this reason, the `trigevent` array should always be checked before its elements are dereferenced.

Example

The following example shows a Fixit job that implements two separate tasks:

- Tcl script using `trigevent` to pass an array to a Fixit job.

```
global trigevent
if {[info exists trigevent]} {
  puts "Event name: $trigevent(name)"
  puts "Event object: $trigevent(object)"
  puts "Event arguments: $trigevent(arguments)"
```

```
  puts "Event results: $trigevent(results)"
  puts "Event severity: $trigevent(severity)"
} else {
  puts "Not a fixit"
}
```

- Execution of an operating system command

  ```
  top -d1 -ocpu
  ```

The Fixit job in this example is associated with the CPUUTIL event test, which monitors for specific levels of CPU activity. For this event test, the parameters are set as follows:

- **Alert Threshold** = 20
- **Warning Threshold** = 10
- **Fixit Job** = Option is selected in the property sheet.

When the event containing the CPUUTIL event test is triggered, the associated Fixit job is executed. The Fixit job generates the following output:

First task executed: Information from the trigevent array is displayed.

```
Event name: /oracle/host/perf/cpuutil
Event object: aholser-sun
Event arguments: {1} {20} {10}
Event results: 63
Event severity: 2
```

Second task executed: Operating system command `top -d1 -ocpu` *is executed.*

```
last pid: 26420;  load averages:  0.64,  0.56,  0.48    13:16:00
111 processes: 110 sleeping, 1 on cpu

Memory: 128M real, 7080K free, 89M swap in use, 912M swap free

  PID USERNAME THR PRI NICE  SIZE   RES STATE   TIME    CPU COMMAND
  727 root       1  30    0  128M   19M sleep  17:22 10.54% Xsun
25915 aholser    4  31    0   12M 5032K sleep   4:47 10.32% dbsnmp
  823 aholser    1  34    0   10M 4368K sleep   2:11  7.37% dtterm
26402 aholser    1  34    0  976K  872K sleep   0:03  3.57% find
26415 aholser    4  34    0   11M 4304K sleep   0:00  3.11% dbsnmp
26403 aholser    1  23    0  976K  872K sleep   0:02  2.60% grep
25914 aholser    4  34    0   11M 4832K sleep   0:56  2.56% dbsnmp
26419 aholser    1  -5    0 1576K 1368K cpu     0:00  1.64% top
```

```
  894 aholser    1  33    0 5904K 3456K sleep   7:07  0.86% view_server
  159 root       5  23    0 3176K 1976K sleep   6:21  0.29% automountd
26418 aholser    1  25    0  920K  760K sleep   0:00  0.28% sh
 1007 root       3  33    0 1840K 1400K sleep   0:40  0.06% cachefsd
```

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*
- *orainfo*
- *orajobstat*
- *orareportevent*

Character set conversion and error handling functions

- *convertin*
- *convertout*
- *msgtxt*
- *msgtxt1*

General purpose utility functions

- *orasleep*
- *oratime*

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.

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 jobs or events, use $oramsg(agent_characterset). See "oramsg Elements" on page 3-6.

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 jobs or events, use $oramsg(agent_characterset). See "oramsg Elements" on page 3-6.

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.

msgtxt

Purpose

This function returns message text in the client's (Console) language and characterset 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.

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 3-15.

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 3-15.

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 3-15.

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 3-15.

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 3-15.

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 3-32 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 3-15.

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.

`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 3-15.

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 3-15.

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 3-15..

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 3-15..

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.

Additional Information: 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 3-13 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.

oralogon_unreached

Purpose This function connects to an Oracle server using a connect string and an optional role. this connection cannot be shared.

Syntax oralogon connect_string [AS] [SYSDBA | SYSOPER | NORMAL]

connect_string See common parameters on "Common Parameters" on page 3-15..

Comments This verb is identical to oralogon except that the returned connection is not shared. Also, an optional role can be specified.

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 3-15..

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 3-15..

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. `oraplexec` returns the result set as its return value in a Tcl list.

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 3-15..

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 3-15..

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 3-6.

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 3-22 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 3-15.

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 3-15.

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 3-15.

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 3-15.

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.

A

Agent 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
- User-configurable Parameters
- Intelligent Agent Log Files

Configuration Files

The following files control the operation of the Intelligent Agent.

snmp_ro.ora

The `snmp_ro.ora` file is located in `$ORACLE_HOME/network/admin` or a directory specified by the TNS_ADMIN environment variable. Do **NOT** update this read-only file.

snmp_rw.ora

The `snmp_rw.ora` is located in the Agent's `$ORACLE_HOME\network\admin`. You can modify this read-write file, but this should be done carefully.

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. Beginning with Agent version 9.0, this file also contains version and platform information about the operating system of the environment in which the Agent is running. 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.

User-configurable Parameters

These parameters are used in the `snmp_rw.ora` configuration file for the Intelligent Agent release.

agentctl.trace_level = off | user | admin | nn

Turns on tracing at the specified level for the AGENTCTL executable. 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, you can see that packets are being passed. This parameter is optional.

agentctl.trace_directory = *directory*
Directory where trace file is written. The setting is only relevant in conjunction with DBSNMP.TRACE_LEVEL. If omitted, trace files are written to `$ORACLE_HOME\network\trace`. This parameter is optional.

agentctl.trace_file = *filename*
Filename of the trace file. This parameter is optional.

agentctl.trace_timestamp = *true/false*
When set to TRUE, inserts a timestamp after each line of trace in the trace files.

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.connect.<service_name>.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>.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 re-executed 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.

When the dbsnmp password is changed it is automatically encrypted. The password parameter entry in the snmp_rw.ora file will appear in a non-readable format. For example:

```
snmp.connect.<service>.encryptedpassword="f1d03f5c7912faa3"
```

After the new password is accepted and encrypted, the Agent's dbsnmp.log file will contain "NMS-00056: Agent's db-account password for database '<service>' was encrypted"

snmp.contact.<service_name> ="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.addnl_db_conns = nn
Increases the size of the Agent session pool. The default size of the Agent session pool is based on the number of ORACLE_DATABASE targets that the Agent manages. The connection limit is the default value plus the value specified for DBSNMP.ADDNL_DB_CONNS.

dbsnmp.no_job_skipped_notifications = <false/true>
When this parameter is true, skipped notifications are not sent. Skipped notifications are sent if this parameter is set to false (default value).

dbsnmp.notificationtimeout = <nn>
If the Agent fails to deliver a notification to the Oracle Management Server within this time, the Agent will be automatically re-started. The default value is 6 minutes. <nn> is number of milli-seconds.

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.noheuristic = {true/false}
The value of this parameter determines whether the Intelligent Agent will use a connection heuristic to ascertain the state of a monitored database (whether the database is up or down). By default, this value is set to FALSE (Agent uses the heuristic).

Note: If the monitored target is a Real Application Clusters database instance, DBSNMP.NOHEURISTIC must be set to TRUE since the heuristic does not work against Real Application Clusters database instances.

---

dbsnmp.notificationtimeout = <nn>
If the Agent fails to deliver a notification to the Oracle Management Server within the specified time (nn is number of milliseconds), the Agent is automatically re-started. The default value is 6 minutes.

dbsnmp.trace_level = off | user | admin | nn
Turns on tracing at the specified level for the Intelligent Agent process (dbsnmp). 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, you can only see that packets are being passed. This parameter is optional.

dbsnmp.trace_directory = *directory*
Directory where trace file is written. The setting is only relevant in conjunction with the DBSNMP.TRACE_LEVEL. If omitted, trace files are written to `$ORACLE_HOME\network\trace`. This parameter is optional.

dbsnmp.trace_file = *filename*
Filename of the trace file. This parameter is optional.

dbsnmp.trace_filecnt = nn
Maximum number of trace files generated by the Agent. This optional parameter should be used when full tracing is desired, but disk space on the Agent machine is limited. This parameter is available beginning with version 8.1.7 of the Intelligent Agent.

dbsnmp.trace_filesize = nn
Maximum size of the individual trace file (in Kilobytes) generated by the Agent. For example, a value of 1024=1 megabyte trace file. This parameter is used in conjunction with DBSNMP.TRACE_FILECNT when full tracing is desired, but disk

space on the Agent machine is limited. This parameter is available beginning with version 8.1.7 of the Intelligent Agent.

dbsnmp.trace_unique = true/false
When set to TRUE, generates unique log files for each log entry.

dbsnmp.log_directory = directory
Directory where log file is written. This parameter is optional.

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

dbsnmp.log_unique = true/false
When set to TRUE, generates unique log files for each log entry.

dbsnmp.trace_timestamp = true/false
When set to TRUE, inserts a timestamp after each line of trace in the trace files.

dbsnmp.hostname = <hostname or ip address>
Overrides the hostname the Agent is known by. This parameter allows the Agent to bind to a particular network interface card on its machine when an IP address is specified. This parameter also allows the Agent to bind to all network interface cards on its machine when hostname is specified.

The string that is returned is the name of the machine and is used in crypto exchanges. DBSNMP.HOSTNAME will also override the default host that is used for RPC listens (dbsnmp.address and dbsnmp.spawnaddress).

dbsnmp.threshold_job_status
This sets the number of job notifications that can be queued by the Agent for the Oracle Management Server.

dbsnmp.threshold_evocc
Sets the number of event notifications that can be queued for the OMS.

dbsnmp.avg_occ_per_event
Sets the number of notifications stored per event prior to flushing the queue when there are too many notifications.

dbsnmpj.trace_level = off | user | admin | nn
Turns on tracing at the specified level for the *dbsnmpj* process. *dbsnmpj* is spawned whenever a job is submitted to the Intelligent Agent. 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, you can only see that packets are being passed. This parameter is optional.

dbsnmpj.trace_directory = directory
Directory where trace file is written. The setting is only relevant in conjunction with the DBSNMP.TRACE_LEVEL. If omitted, trace files are written to `$ORACLE_HOME\network\trace`. This parameter is optional.

dbsnmpj.trace_file = filename
Filename of the trace file. This parameter is optional.

dbsnmpj.trace_filecnt = nn
Maximum number of trace files generated by the Agent. This optional parameter should be used when full tracing is desired, but disk space on the Agent machine is limited. This parameter is available beginning with version 8.1.7 of the Intelligent Agent.

dbsnmpj.trace_filesize = nn
Maximum size of the individual trace file (in Kilobytes) generated by the Agent. For example, a value of 1024=1 megabyte trace file. This parameter is used in conjunction with DBSNMP.TRACE_FILECNT when full tracing is desired, but disk space on the Agent machine is limited. This parameter is available beginning with version 8.1.7 of the Intelligent Agent.

dbsnmpj.trace_unique = true/false
When set to TRUE, generates unique trace files for each line of trace.

dbsnmpj.log_directory = *directory*
Directory where log file is written. This parameter is optional.

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

dbsnmpj.log_unique = true/false
When set to TRUE, generates unique log files for each log entry.

dbsnmpj.trace_timestamp = *true/false*
When set to TRUE, inserts a timestamp after each line of trace in the trace files.

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.

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.

dbsnmp.cs_base_port = <port number>
Port number used to override the default port numbers utilized by the Agent's collection service. For example, specifying a port number of 1700 tells the Agent to use 1700 and 1701 as the designated port numbers.

Intelligent Agent Log Files

When the appropriate log file parameters are set, the following log files can be generated.

Table A–1 Intelligent Agent Log Files

Log File	Description
dbsnmp.log	Logs all Intelligent Agent processes
dbsnmpj.log	Logs all jobs processed by the Intelligent Agent
dbsnmp.nohup (UNIX) <agent service name>.log (Windows NT)	Logs automatic Intelligent Agent restart (dbsnmpwd)
agntsrvc.log (Windows NT)	Logs the "agentctl" functions. File also shows any errors that occur during Agent startup.
nmiconf.log	Logs discovery information and errors.

B

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
- Additional Checks
- Intelligent Agent Error Messages and Resolutions
- Tracing the 9i Agent
- Interpreting Intelligent Agent Startup Errors on Windows NT/2000
- Understanding and Troubleshooting the Data Gatherer
- Clean Starting the Intelligent Agent
- Diagnosing Intelligent Agent Discovery Errors on UNIX
- Diagnosing Agent Discovery Errors on Windows NT

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. In general, the lowest acceptable version of the Agent should be compatible with the highest version of the software being monitored.

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\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, "Agent Configuration Files" for valid sample files.

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

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

4. 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.

   Important: When installing a 9*i* Agent on a machine running a pre-9*i* database, you must re-run a version specific copy of catsnmp.sql which is located in the ORACLE_HOME/sysman/admin directory of the 9*i* Enterprise Manager client install. For example, if you are running Oracle 8.1.7 on a machine and then install a 9*i* Agent, you must re-run the catsnmp_8*i*.sql script after installing the new Agent. This operation must be performed for each pre-9i database serviced by this Agent.

   Do not run the 9*i* version of the catsnmp.sql script against pre-9i databases.

   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: 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.

5. 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.

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`:

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

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

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

      `dbsnmp.trace_unique=true/false` (When TRUE, this parameter generates a unique trace file each time.)

   b. Restart the Agent.

   c. Check the log files located in the `oracle_home/network/log` directory.

      `DBSNMP.LOG` should show general Agent problems.

      DBSNMP.NOHUP should show any errors related to the Agent's "watchdog" dbsnmpwd process.

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

Quick Checks for UNIX Agents

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

1. Make sure requisite system requirements are met:
   - TCP / IP is configured correctly
   - Valid ORATAB file for your Oracle environment.
2. Check the Agent's status. Enter the command:

   ```
   agentctl status
   ```

   Alternatively, you can check to see if the Intelligent Agent is running by entering the following command:

   ```
   ps -eaf | grep dbsnmp
   ```

   These checks should show that a "dbsnmp" process is running and / or "dbsnmpwd" watchdog script is running.
3. Check the `ORACLE_HOME/network/log/dbsnmp*.log` file for errors on UNIX. (nmiconf.log for discovery).
4. Check that the Oracle user has write permissions to the following directories:
   - `ORACLE_HOME/network/log`
   - `ORACLE_HOME/network/trace`
   - `ORACLE_HOME/network/agent`
   - `ORACLE_HOME/network/agent/jobout`
5. 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. Alternatively, you can check the directory pointed to by the TNS_ADMIN environment variable.

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

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

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

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.

   Important: When installing a 9*i* Agent on a machine running a pre-9*i* database, you must re-run a version specific copy of catsnmp.sql which is located in the ORACLE_HOME/sysman/admin directory of the 9*i* Enterprise Manager client install. For example, if you are running Oracle 8.1.7 on a machine and then install a 9*i* Agent, you must re-run the catsnmp_8*i*.sql script after installing the new Agent. This operation must be performed for each pre-9i database serviced by this Agent.

   Do not run the 9*i* version of the catsnmp.sql script against pre-9i databases.

   ---

   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: 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.

7. If you have problems running the Intelligent Agent control utility (agentctl), set tracing for agentctl as follows:
   - agentctl.trace_level=admin
   - agentctl.trace_directory
   - agentctl.trace_file
8. If you still do not know why the Agent did not start, trace the Agent by setting the following variables in `snmp_rw.ora` and then re-start the Agent.
   - `dbsnmp.trace_level=admin` (or 16 if you want more information)
   - `dbsnmp.trace_directory=<any directory which the Oracle user can write to>`
   - `dbsnmp.trace_file=agent`

Additional Checks

If after going through the quick checks your Intelligent 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 the Agent running with the correct permissions? (UNIX) on page B-14
- Does the OS user exist and does it have the correct permissions? (Windows NT) on page B-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 Management Server 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 Management Server.
- Determine if the machine running the Management Server is available on the local network and whether the machine running the Agent can communicate with the machine running the Management Server.

1. ping <IP address of the console machine>

   Ping the machine running the Management Server from itself.

2. ping <IP address of the Management Server machine>

   Ping the Management Server 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.

   NIS / DNS setups may require modification at the NIS / DNS level to correct TCP / IP configuration problems.

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

   Example: (Windows NT)

```
HOSTS file:
        122.111.111.111   myHost

LMHOSTS file:
        122.111.111.111   myHost  #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 are binary files containing information about Agent state and current jobs and events. Do not delete these files without first removing all jobs and events registered against this Agent. Because the hostname of the machine on which the Agent is running on is used to encrypt the 'Q' files, these files cannot be copied from one system to another.

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 Oracle Net 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. Oracle Net provides the session and data communication medium between client machines and Oracle servers, or between Oracle servers. For this reason, proper Oracle Net configuration is a prerequisite for Agent communication. This section covers the most common problems that can occur when Agent communication fails.

Oracle Net configuration files are found in either the $TNS_ADMIN location or the `$ORACLE_HOME/network/admin` directory (both UNIX and NT).

Primary configuration files are:

- `listener.ora`

- sqlnet.ora
- tnsnames.ora

See Appendix A, "Agent 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 the listener.ora file has at least one TCP entry configured for the target.

   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)

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.

Is Oracle Net functioning properly?

If your Oracle Net configuration is correct and you are still unable to contact the Agent, the next step is to determine whether services in your Oracle Net 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 Oracle Net service. This indicates Oracle Net is functioning properly.

Pinging the Intelligent Agent

If you want to see if an Agent is up, do this:

```
$ tnsping (address=(protocol=tcp)(host=<<hostname>>)(port=1748))
```

Testing the Connections to the Agent

You can also use the TNSPING utility to test connections to the Agent:

```
$ tnsping "(address=(protocol=tcp)(host=<<hostname>>)(port=1748))"
```

If there is a successful connection, you will see a message similar to the following:

```
Attempting to contact (address=(protocol=tcp)(host=<<hostname>>)(port=1748))
OK (750 msec)
```

If a host which is not in the invited_nodes list is trying to contact the node, the following error will be shown in the output:

```
Attempting to contact (address=(protocol=tcp)(host=<<hostname>>)(port=1748))
TNS-12547: TNS: lost contact
```

Did the Agent startup successfully?

To check whether the Agent process is running issue the following command:

```
agentctl status
```

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

Table B–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 `$ORACLE_HOME/network/log/nmiconf.log` file for errors.
Check that the Oracle user has write permissions to the following directory: `$ORACLE_HOME/network/log`	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 directory: `$ORACLE_HOME/network/log`
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?
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.

Table B–1 Troubleshooting an Agent that Will Not Start

UNIX	Windows NT
	Check the $ORACLE_HOME/network/log/AGENTSRVC.log file. This file will show startup errors that occured when the Agent service was started.
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 9i Agent" on page B-22)	

For both UNIX and Windows NT systems check:

```
$ORACLE_HOME/network/log/dbsnmp.nohup
```

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.

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

To verify whether the Agent has the correct user permissions, see Installing the Intelligent Agent 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/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".

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.

For both UNIX and Windows NT systems check the following file for additional errors:

`$ORACLE_HOME/network/log/dbsnmp.nohup`

Intelligent Agent Error Messages and Resolutions

The following error messages and resolution are categorized by operating system. Situations that apply to all systems are listed under "Generic Agent."

Generic Agent

'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\network` directory.
- (UNIX) Shadow password files used in high security environments must be synchronized. The Intelligent Agent looks at the local "shadow" password file for authentication information. Refer to your OS-specific documentation for more information on how to synchronize passwords on high-security systems.

'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.

Important: When installing a 9*i* Agent on a machine running a pre-9*i* database, you must re-run a version specific copy of catsnmp.sql which is located in the ORACLE_HOME/sysman/admin directory of the 9*i* Enterprise Manager client install. For example, if you are running Oracle 8.1.7 on a machine and then install a 9*i* Agent, you must re-run the catsnmp_8*i*.sql script after installing the new Agent. This operation must be performed for each pre-9i database serviced by this Agent.

Do not run the 9*i* version of the catsnmp.sql script against pre-9i databases.

'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 database name by specifying one of the following:

- A unique SID.
- A unique alias in the TNSNAMES.ORA file for the database.
- A unique GLOBAL_DBNAME in the LISTENER.ORA file.

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\network\log\nmiconf.log` for errors. For more information on Agent startup problems, see "Did the Agent startup successfully?" on page B-12.

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

All the database services the Agent finds on a machine, must be defined in the relevant SQL*Net/Oracle Net 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 / Oracle Net 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 / Oracle Net 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 Oracle Net Configuration files correct?" on page B-10 and "Is Oracle Net functioning properly?" on page B-11.

'Invalid service name' while registering a job or event. (Agent-specific error)

OR

'File operation error' while registering a job or event. (OS-specific error)

These errors are usually seen when the services on the Enterprise Manager 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.

'Oralogin failed in orlon' You may receive this error while executing a TCL script using the oratcl verb oralogon. "Oralogin failed in orlon" means that the connect

string is either wrong or for some reason, the account used cannot logon to the database. To debug the this error, turn on Tcl tracing.

ORA-1017 'Invalid username/password' Invalid username/password errors may occur when starting up the Agent on UNIX systems from an X-terminal. This problem can occur because the Data Gatherer (pre-9i) cannot connect to the Capacity Planner repository to upload collected data. This message will repeat every couple of minutes.

NT Agent

For any NT Operating System Error when starting the Agent

See Oracle Intelligent Agent – Windows Event Log Messages on page B-29 for information on Windows NT-Agent error message cross-referencing.

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.
- DBSNMP.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 `$ORACLE_HOME\network\admin` directory and they are not zero length.
- Verify that user that started the Agent can read/write the queue files.
- Make sure that the Machine Name and the TCP Hostname are the same.
- See Verifying the TCP Name and the Computer Name on NT. Verify that SQL*Net/Oracle Net is running.
- Verify that TCP/IP is configured and running correctly. (See Is TCP/IP Configured and Running?)
- Remove all non-essential files from the `$ORACLE_HOME/network/agent/MIB` directory.
- Upgrade to the latest Agent

'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, thus it uses the NetBios name. 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\network\agent` directory (and TEMP directory, for some applications) and read permissions to the SYSTEM32 directory

Verify that the NT user has these permissions.

UNIX Agent

Discovery fails with no services at all

First check that all of the SQL*Net/Oracle 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.

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
   ```

   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"
   - agentctl status agent (results in "The db subagent is not started.")
   - Then do this.
   - ps -ef | grep dbsnmp
   - kill -9 _____ (fill in process numbers)
   - restart Agent with `agentctl start agent`

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 `agentctl start agent`

This will re-start the Agent and remove all of the job and event queues (.q files) it was using in the past.

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

NMS-001 while starting the Agent (SNMP environemnts only) 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 or Oracle 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-00207 Agent xxxx user account is locked for database yyyy Events registered with the Agent for monitoring a 9*i* version of the database will not work because the database account is locked.

Under these conditions, an Enterprise Manager database up-down event will always indicate that the database is down. The Agent's log file dbsnmp.log will contain a NMS-00207 error message indicating the dbsnmp user account for the database is locked.

To resolve this problem, you must log into the database and perform the following:

1. Unlock the "dbsnmp" account by running the sql statement:

   ```
   ALTER USER dbsnmp ACCOUNT UNLOCK;
   ```

2. Reset the password for dbsnmp account by running the sql statement:

   ```
   ALTER USER dbsnmp IDENTIFIED BY <password>;
   ```

3. Add the reset password to the Agent configuration file snmp_rw.ora as follows:

   ```
   SNMP.CONNECT.<service_name>.PASSWORD=<password>
   ```

   where service_name is the name of the seed database as discovered by the Agent in snmp_ro.ora / snmp_rw.ora.

4. Stop and start the Agent using agentctl.

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

NMS-205 Failure to connect to database name with username/password string The 'dbsnmp' user could not be located.

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

NMS-351 Encryption key supplied is not the one used to encrypt the file This happens if there mismatches between the ID's in the '*.q' files in the $ORACLE_HOME / network / agent directory. This condition can be caused by the following

- Hostname change on the machine.

- Encryption key was overruled with a password switch, and the Agent is started without it or with another key.

Delete all the '*.q' in the $ORACLE_HOME/network/agent directory. Rebuild your repository. Restart the Agent.

Tracing the 9*i* Agent

Tracing and logging of the Intelligent Agent allows tracking of all communication between the Intelligent Agent and Management Server(s) as well as Agent startup and discovery information. To turn on tracing for the 9i Intelligent Agent, you will need to modify the Agent's snmp_rw.ora file. This file is normally in the $ORACLE_HOME\network\admin directory. The snmp_rw.ora is created the first time the Agent process is started. If the file is not created and you need to trace the startup process, manually create a text file and add the necessary tracing parameters to the file.

The log file, $ORACLE_HOME/network/log/dbsnmp.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 upon first start up of the Agent and appended upon subsequent Agent startups. 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 TCL82) plus a large subset (not all) of the ORATCL verbs supported by the Intelligent Agent. ORATCLSH is not a general purpose utility and may only be used in combination with the Intelligent Agent as it depends on files and data structures maintained by the Agent.

There is no documentation of ORATCLSH as it has never been part of the supported feature set of the Intelligent Agent. It is provided strictly as a debugging tool to help Oracle customers and developers in developing Enterprise Manager 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.

By default the following log files are created under the Agent's ORACLE_HOME/network/log directory:

- agntsrvc.log (Windows NT)

- dbsnmp.log
- dbsnmp.nohup (for the watchdog script dbsnmpwd)
- dbsnmpj.log (job log files)
- nmiconf.log (for discovery information)

Setting various tracing and logging parameters in the snmp_rw.ora file allows you to monitor the following areas:

- Agent Process (dbsnmp) and Communication
- Agent Job Subsystem
- AGENTCTL Utility
- Data Collection Services
- Agent's Event Subsystem

The following tables organize the tracing and logging parameters according to their respective functional areas.

Parameters Used to Trace the Agent (dbsnmp) Process

Table B–2 Parameters Used to Trace the Agent (dbsnmp) Process

Parameter	Description
dbsnmp.trace_level = <OFF|USER|ADMIN|nn>	This parameter is used to turn tracing on and specify the amount of data that is collected (trace level). A trace level of 16 provides the most detailed level of information. This parameter must be in lower case. Trace levels are identical to those used by SQL*Net or Oracle Net.
dbsnmp.trace_file= <filename>	The default trace file is dbsnmp.trc. If you want the trace to be written to a different file, add the dbsnmp.trace_file parameter.
dbsnmp.trace_directory=<directory>	sThe default trace directory is $ORACLE_HOME/network/trace. If you want to change the location of the trace file, add the dbsnmp.trace_directory parameter.
dbsnmp.trace_filecnt=<integer>	This parameter defines the maximum number of trace files to be generated.
dbsnmp.trace_filesize=<integer in kilobytes>	This parameter defines the maximum size of a trace file in kilobytes.
dbsnmp.trace_unique={true/false}	This parameter generates a unique trace file each time if set to true.
dbsnmp.trace_timestamp={true/false}	This parameter determines whether to put a timestamp before each line of trace.
dbsnmp.log_directory=<directory>	The default log directory is $ORACLE_HOME\network\log. If you want to change the name, you can add the dbsnmp.log_directory parameter. However, the Agent must have write privileges to the directory that you specify.

Table B–2 Parameters Used to Trace the Agent (dbsnmp) Process

Parameter	Description
dbsnmp.log_file=<filename>	The default log file is called dbsnmp.log. If you want to change the name, you can add the dbsnmp.log_file parameter. This 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.
dbsnmp.log_unique={true/false}	This parameter will cause the Agent to create a unique log file each time the Agent is started.

Table B–3 Parameters Used to Trace the Job System

Parameter	Description
dbsnmpj.trace_level={OFF|USER|ADMIN|nn}	This parameter is used to turn tracing on for the job system and also specifies the amount of data that is collected (trace level). A trace level of 16 provides the most detailed level of information. This parameter must be in lower case.
dbsnmpj.trace_directory=<directory>	The default trace directory is $ORACLE_HOME/network/trace. If you want to change the location of the trace file, add the dbsnmpj.trace_directory parameter.
dbsnmpj.trace_file=<filename>	The default trace file is dbsnmpj.trc. If you want the trace to be written to a different file, add the dbsnmpj.trace_file parameter.
dbsnmpj.trace_filecnt=<integer>	This parameter defines the maximum number of trace files to be generated.
dbsnmpj.trace_filesize=<integer in kilobytes>	This parameter defines the maximum size of a trace file in kilobytes.
dbsnmpj.log_directory=<directory>	The default log directory is $ORACLE_HOME\network\log. If you want to change the name, you can add the dbsnmpj.log_directory parameter. However, the Agent must have write privileges to the directory that you specify.
dbsnmpj.log_file=<filename	The default log file is called dbsnmpj.log.
dbsnmpj.log_unique={true/false}	This parameter will cause the Agent to create a unique log file each time a job is ran.

Table B–4 Parameters Used to Trace Agent Startup

Parameter	Description
agentctl.trace_level={OFF|USER|ADMIN|nn}	This parameter is used to turn the tracing on for the agentctl utility (starting the agent). If a problem is occurring, Oracle Support will normally need a trace at level 16.
agentctl.trace_file=<filename>	The default trace file is agentctl.trc. If you want the trace to be written to a different file, add the agentctl.trace_file parameter.
agentctl.trace_directory=<directory>	The default trace directory is $ORACLE_HOME/network/trace. If you want to change the location of the trace file, add the agentctl.trace_directory parameter.
agentctl.trace_filecnt=<integer>	This parameter defines the maximum number of trace files to be generated.
agentctl.trace_filesize=<integer in kilobytes>	This parameter defines the maximum size of a trace file in kilobytes.
agentctl.trace_unique={true/false}	This parameter generates a unique trace file each time if set to true.
agentctl.trace_timestamp={true/false}	This parameter determines whether to put a timestamp before each line of trace.

Tracing the Data Collection Services

Because the data collection service (formerly the Data Gatherer) functionality has been integrated into the 9i Intelligent Agent, data collection-based tracing can be turned on using one of the following procedures (according to platform).

On UNIX:

1. Set the VP_DEBUG environment variable to one:

   ```
   >setenv VP_DEBUG 1
   ```

2. Start the agent:

   ```
   > agentctl start agent.
   ```

   Any collection activity will be logged in $ORACLE_HOME/network/log/dbsnmp.nohup.

On Windows NT (from a DOS window):

1. Set the VP_DEBUG environment variable to one:

   ```
   >set VP_DEBUG=1
   ```

2. Start the Agent and redirect the output to a text file:

   ```
   > dbsnmp -agent_name Oracleora920Agent > stdout.log2> odg.log
   ```

On Windows 2000 (from a DOS window):

1. Set the VP_DEBUG environment variable to one:

   ```
   >set VP_DEBUG 1
   ```

2. Start the Agent and redirect the output to a text file:

   ```
   > dbsnmp -agentname Oracleora920Agent > stdout.log2> odg.log
   ```

Multiple ORACLE_HOMEs

If there are multiple ORACLE_HOMEs on the same machine, perform the following.

1. Navigate to the specific ORACLE_HOME/bin directory of the Agent.
2. Set the VP_DEBUG environment variable to one: >setenv VP_DEBUG 1

   ```
   e:/920/bin/ > set VP_DEBUG=1
   ```

3. Start the Agent

   ```
   > agentctl start agent.
   ```

Tracing the Event System (tracing Tcl)

You can also turn on Event tracing by setting the dbsnmp.trace_level parameter in the snmp_rw.ora file to a level greater than or equal to one (dbsnmp.trace_level >= 1). You must shut down and re-start the Agent for these parameters to take effect. Tcl tracing creates a file, oratcl.trc, in the ORACLE_HOME/network/trace directory. Every time an event is triggered, an entry is added to the oratcl.trc file.

Interpreting Intelligent Agent Startup Errors on Windows NT/2000

When the Oracle Intelligent Agent service fails or fails to start with a pre-determined error code it calls the Windows ReportEvent function to write an entry to the event log, Windows NT then passes the parameters to the event-logging

service. This in turn uses the information to write a log record to the event log. Other errors are reported to the nmi.log and nmiconf.log.

When the Windows NT event viewer application starts it uses the OpenEventLog function to open the event log for an event source. The event viewer can then use the ReadEventLog function to read event records from the log. ReadEventLog returns a buffer containing an EVENTLOGRECORD structure and additional information that describes a logged event.

Oracle Intelligent Agent – Windows Event Log Messages

When the Intelligent Agent service fails to start, the Windows Service Manager will return the underlying error code, but because it is not able to interpret the Oracle Event Message it incorrectly returns the Windows NT Win32 message text. The correct message, however, will appear in the NT event log.

The following table defines the events that the Intelligent Agent displays in the Windows NT Event viewer and the associated Win32 error text :

Table B–5 MS Windows-Intelligent Agent Error Message Translation

ID	Windows NT/2000 Message	Agent Description
1	Incorrect Function	OracleAgent failed to register its Service Control Handler
2	The system cannot find the file specified	OracleAgent failed to report its status to the Service Control Manager
3	The system cannot find the path specified	OracleAgent failed to create a thread synchronization object.
4	The system cannot open the file	OracleAgent failed to create a thread.
5	Access is denied	OracleAgent failed to allocate memory.
6	The handle is invalid	OracleAgent failed to get the encryption key.

Table B–5 MS Windows-Intelligent Agent Error Message Translation

ID	Windows NT/2000 Message	Agent Description
7	The storage control blocks were destroyed	Oracle Agent failed to fun auto-discovery script nmiconf.tcl. Look in the nmiconf.log for more information. The Intelligent Agent was unable to run the Auto Discovery process. This can be either due to invalid configuration files, a problem with the TCP/IP layer, or an error in the TCL libraries. Verify the SQL*Net or Oracle Net configuration files, and make sure that you can make normal loopback connections via SQL*Net or Oracle Net from the server to the databases. If the problem persists, reinstall the Intelligent Agent software.
8	Not enough storage is available to process this command	OracleAgent failed to initialize Oracle CORE library
9	The storage control block address is invalid	OracleAgent failed to initialize Oracle NLS library. The NLS environment and/or registry variables are not correct. Check the system for variables beginning with 'NLS'. Try to make a connection to a database using a standard tool like SQL*Plus or SQL*Worksheet, to verify the NLS client settings. If the problem persists, reinstall the Required Support Files software to correct this error.
10	The environment is incorrect	OracleAgent failed to initialize Oracle SQL*Net or Oracle Net library:%1. The SQL*Net or Oracle Net environment and/or registry variables are not correct. Try to make SQL*Net or Oracle Net connections using a standard tool, like SQL*Plus or SQL*Plus Worksheet. If the problem persists, reinstall the SQL*Net or Oracle Net client (and server if a listener is running as well) software to correct this error.
11	An attempt was made to load a program with an incorrect format	OracleAgent failed to initialize DES encryption

Table B–5 MS Windows-Intelligent Agent Error Message Translation

ID	Windows NT/2000 Message	Agent Description
12	The access code is invalid	OracleAgent failed to initialize Oracle Remote Operations Library
14	Not enough storage is available to complete this operation	OracleAgent failed to create file dbsnmp.ver
15	The system cannot find the drive specified	OracleAgent failed to create/read its queue files. The hostname of the machine is encrypted in the agent '.q' files. If this hostname found in the files by the agent does not match the name of the machine the agent is running on, this error is generated. Remove the agent '.q' files and restart the agent.
16	The directory cannot be removed	OracleAgent failed to create job scheduling symbol table
17	The system cannot move the file to a different disk drive	OracleAgent failed to initialize its connection cache. There is an incompatibility between the TCP hostname and the NetBios hostname of the machine. With the Network applet from the Control Panel, synchronize the names of the machine in the protocol properties dialogs.
18	There are no more files	OracleAgent failed to sign on to SNMP.
19	The media is write protected	OracleAgent failed to read SNMP indexes from parameter file.
20	The system cannot find the drive specified	OracleAgent failed to connect to the database. Using the information from the SQL*Net or Oracle Net configuration files, the agent was not capable of making a connection to the desired databases. Most likely a protocol error, like TCP/IP conflicts or bad TCP port being referenced in the configuration files. Check the agent log files for the specific connection errors.
21	The drive is not ready	OracleAgent failed to build SNMP cache.

Table B–5 MS Windows-Intelligent Agent Error Message Translation

ID	Windows NT/2000 Message	Agent Description
22	The device does not recognize the command.	OracleAgent failed to build MIB. There are files in the $ORACLE_HOME\network\agent\mib directory which do not comply to the MIB specifications. Verify all files in that directory. Remove all non-MIB files, or corrupt MIB files.
23	Data error <cyclic redundancy check>	OracleAgent failed to register MIB row.
24	The program issued a command but the command length is incorrect	OracleAgent failed to restart its communication thread.
25	The drive issued a command but the command length is incorrect	OracleAgent failed to listen on designated port. Another OracleAgent may already be started. There was an internal error using the encryption key used in the '.Q' files. This can either be a TCP/IP conflict on the machine, or a lack of system resources on the machine preventing a proper initialization of the Intelligent Agent routines. Check the system resources, and verify if the TCP/IP setup is correct on the machine.
The following errors are generated when the Windows NT program managing the services can no longer control, or looses control over the service.		
1067	The process terminated unexpectedly	Most likely a corruption in the DLL's, a problem with system settings, etc. Check for Dr. Watson logs, errors in the event viewer and messages in the log files to get more details on the exact condition.
2140	An internal Windows NT error occurred	The services program received conflicting information from the service. Most likely, an abnormal termination of the Intelligent Agent, due to conflicting system information discovered. More information about the abort of the Intelligent Agent can be found in the discovery log files.

Table B–5 MS Windows-Intelligent Agent Error Message Translation

ID	Windows NT/2000 Message	Agent Description
2186	The service is not responding to the control function	The Intelligent Agent failed to report it's status to the SERVICES program monitoring the services. If you can not stop the Agent service, use the 'KILL' command from the command line to stop the agent process. The 'KILL' command is part of the optional MS Resource Kit, and not a standard NT tool.

Typical Intelligent Agent Configuration Issues on UNIX

The Intelligent Agent software is delivered with the RDBMS server software. However, this does not mean that the Intelligent Agent software must be installed together with the database. It is quite possible to install the Agent alone, in a dedicated "$ORACLE_HOME", separate from the rest of the Oracle software.

The only thing the Intelligent Agent needs is SQL*Net or Oracle Net, to make connections to the databases it needs to monitor, and to be able to communicate with the rest of the Enterprise Manager framework. The SQL*Net or Oracle Net product, and the underlying 'Common Libraries' are products that are installed automatically whenever the Agent is installed on UNIX.

It is not necessary to have a SQL*Net listener running in the same "$ORACLE_HOME" as the Agent, and it also does not have to be of the same base version as the Agent.

Linking the Intelligent Agent

Important: Be sure to shut down the Agent BEFORE relinking the software.

As soon as the software is installed from the installation media, the Intelligent Agent is relinked using the current system libraries present on the system.

If a change is made to the UNIX kernel, or the system libraries, it is advised to relink the Intelligent Agent, using the following commands:

```
$ cd $ORACLE_HOME/network/lib
$ make -f ins_oemagent.mk install
```

This will create two new executables: dbsnmp and oratclsh, both created in the "$ORACLE_HOME/bin" directory. If a version of these executables already exists, the old ones will be renamed to "dbsnmp0" and "oratclsh0". As soon as the new software has been tested, these safety copies can be removed.

When relinking database software, or after making changes to the installed SQL*net protocols drivers, this does require a shutdown of all databases of that version!

As soon as the agent is relinked however, either the "root.sh" file needs to be executed again, or a manual intervention is needed to adjust the dbsnmp executable.

If the "root.sh" file is meanwhile adjusted or overwritten by a newer file, the following commands have to be executed manually, as the 'root' user:

```
$ cd $ORACLE_HOME/bin
$ chmod 6751 dbsnmp
$ chown root dbsnmp
```

These steps are essential for the proper working of the agent. If the agent has not the 'setuid' permissions (given by the chmod '6751' command), or is not owned by root ('chown root' command), the discovery can fail, and jobs will not get executed properly.

Also, when the agent has already been started on the machine, some of the files do have the 'root' ownership, making the agent fail to start, and update the wrong files, after a relink and an ownership back to the Oracle owner.

Running the Intelligent Agent

Whenever a program runs on UNIX under the 'root' permissions, the environment variable "LD_LIBRARY_PATH" is not read for security reasons. This means that all shared libraries, linked dynamically to the .EXE will have to be referenced using their absolute location on the disk.

You can check the linked shared libraries using the 'LDD' command. For example:

```
$ ldd dbsnmp
```

To avoid problems with the shared libraries, there are three options:

- Link the software statically.

If the libraries are statically linked in, the 'ld' will not consider them 'shared', solving the problem. Downside here is that the executable will be a lot larger in size.

- Copy or link the shared libraries in the "/usr/lib" directory.

  An easy and straightforward trick is to create a symbolic link to the needed shared library in "/usr/lib". Since this directory is hardcoded into the library loader, all shared objects placed in this directory will always be found.

  Potential problems can arise in this case if several versions of software are installed on the same machine. Since the "/usr/lib" directory is the same for the entire machine, any file copied or linked in this directory will be used throughout the machine, meaning that other versions of software might be picking up the wrong shared object.

- Reference the shared objects by their absolute path.

  The best solution, but also the hardest, as it includes modifying the make file and adding the full hardcoded path to the used libraries. After the modification, relink the software, to make sure the libraries are found correctly.

  Specifically for the Agent, the following errors can be encountered if there is a problem with the shared libraries:

  ```
  ld: Can not map libclntsh.so.1.0
  ld: Error opening libclntsh.so.1.0
  ld: Can not find libnet.so
  ```

If an error is generated when the agent starts, perform the following:

1. Relink the agent:

   ```
   $ make -f ins_oemagent.mk install
   ```

2. Change the owner and permissions of the executable:

   ```
   $ chmod 6751 dbsnmp
   $ chown root dbsnmp
   ```

Understanding and Troubleshooting the Data Gatherer

Important: This section **only** applies **pre-9i Intelligent Agents**. The data collection services provided by the Data Gatherer are now integrated with the 9.x Intelligent Agent.

The Oracle Data Gatherer is a daemon process which manages the collection of performance statistics from the Oracle database and from the host operating system for use by Enterprise Manager tools, such as the Oracle Performance Manager and the Oracle Capacity Planner. As mentioned above, this functionality is now an integral part of 9.x Agent. This section only applies to older versions of the Agent / Data Gatherer.

Data Gatherer Configuration

The Data Gatherer collects the following types of data:

- Operating System
- Database
- Oracle Application's Concurrent Managers
- Oracle HTTP Server
- Microsoft SQL*Server

You may not be able to collect operating system data if the Data Gatherer is not available for that particular operating system. To collect operating system data the Data Gatherer must be installed on the same host as the OS. It may not be possible to install and configure the Oracle Data Gatherer on a particular host if both of the following requirements are not met:

- The host operating system must support the installation and configuration of the Oracle Data Gatherer. Base development is done for the Data Gatherer on NT and Solaris
- An Oracle Home for Oracle 8.0.4 or later must exist on the host.

Using the Data Gatherer when Installed on Server Host

This is the assumed configuration for using the Oracle Data Gatherer. The Data Gatherer is installed separately as part of the database server installation. In 8.0.5 or higher, the Data Gatherer is installed along with the Intelligent Agent (but not an integral part of the Agent, as is the case for 9.x versions of the Agent). The Data

Gatherer can be used to monitor database statistics for any database on that host, and also can be used to monitor OS statistics for the host itself.

Using the Data Gatherer when Installed on Client Host

It is possible to install the Data Gatherer on the same host where the client will be run from (if the client is Windows NT 4.0, this is not supported on Windows 95 or Windows 98). In this configuration, you will be able to monitor the database statistics for remote databases, but will not be able to monitor the operating system statistics on the remote host.

Using the Data Gatherer on an Alternative Port

The Data Gatherer and clients are installed and run assuming the use of a well-known (IANA registered) port (1808), which is used for communication between clients and the Data Gatherer server. If you wish to use an alternate port, you may do so, however this configuration is not supported.

Using Multiple Oracle Homes

It is possible to install the Oracle Data Gatherer in an environment with multiple Oracle homes, however there are two issues to keep in mind if you attempt to do this:

- Because the Data Gatherer uses a well known port, if you intend to run the Data Gatherer in two homes simultaneously you will need to run one on an alternate port (which is not supported)
- Database connect strings delivered to the Data Gatherer must be resolvable from the SQL*Net environment in that Oracle home.

Data Gatherer Recovery Capabilities

Data Gatherer Restart at Host Restart

The Data Gatherer is configured to save the state of all current historical collections, such that when it restarts it will create recovery threads to restart these collections from the state files. If the Data Gatherer is configured to start automatically when the system reboots, then collections should be able to continue.

Collecting Historical Data from a Cycling Database

If the database from which data is being collected is cycled (e.g. shutdown each evening for backups) then the Data Gatherer is designed to continue collecting data

from the database when it restarts. The Data Gatherer attempts to reconnect to the database at the specified collection interval until it becomes available.

Miscellaneous Data Gatherer Issues

- The Data Gatherer alert log ($ORACLE_HOME\odg\log\alert_dg.log) contains many messages of interest that the Data Gatherer wishes to save. This currently includes errors reported by the Data Gatherer and informational messages. All messages included in the alert log include a timestamp.
- Starting the Data Gatherer:

  UNIX: vppcntl -start (UNIX command).

  NT: From the Control Panel Services applet start the OracleDataGatherer service.
- Stopping the Data Gatherer:

  UNIX: vppcntl -stop(UNIX command).

  NT: From the Control Panel Services applet stop the OracleDataGatherer service.
- Checking the Data Gatherer status:

  vppcntl - status

  This status check will result in one of two messages:

  The Oracle Data Gatherer is running!

  -OR-

  The Oracle Data Gatherer is not running

Clean Starting the Intelligent Agent

Clean starting the Agent involves clearing all existing job and event definitions. This should only be necessary when the Enterprise Manager environment needs to be reinitialized, or upon specific request from Customer Support. Actions will need to be performed from both the Console and the Agent node.

To clean start the Agent:

1. From the Enterprise Manager Console.
   - Remove all jobs and events registered against the machine the agent is running on.
   - As soon as all events and jobs have been removed, shut down the Console.
2. Stop the Intelligent Agent currently running on the desired target.

   On UNIX, issue the following command:

   ```
   $ agentctl stop agent
   ```

   After the stop command has been issued, use the 'ps' command to verify that the 'dbsnmp' processes have been stopped. If the Intelligent Agent cannot be stopped, use the 'ps' command to obtain the process ID's of the dbsnmp processes, and use the 'kill -9' command to terminate these processes.

   On Windows NT, use the Control Panel / Services applet to turn the Agent service off, or issue the command line option:

   ```
   C:\> agentctl stop agent
   ```

   When the agent is stopped, use the TaskManager to verify that the 'dbsnmp' process has been stopped.
3. Go to the "$ORACLE_HOME/network/agent" directory and delete the following files:
   - SERVICES.ORA: File containing all the services the agent has discovered. This file will be recreated the next time the agent starts.
   - All files with the '.q' extension: These files are the binary representation of all registered and running jobs and events. The agent creates new 'q' files on startup when they do not exist.

Information about the host on which the agent is running is also stored in these files. If the name or the IP address of the agent machine changes, these files need to be recreated.

- All files with the '.inp' extension: These are the parameter files for registered jobs.
- All files with the '.jou': These are journal files with information the agent will use to execute jobs.
- All files with the 'out' prefix: These are temporary output files that are generated when jobs are being executed. These files should never remain in this directory after a job finishes.
- All files with the 'tcl' prefix: These are Tcl template scripts, including the specific commands, specified in the job sent to the agent
- DBSNMP.VER : This is a text file with the agent version information. This file will also be recreated when the agent starts again.

4. Go to the "$ORACLE_HOME/network/admin" directory. In this directory, remove the following files:
   - SNMP_RO.ORA : The read-only information the agent has discovered of the services on the machine.

     This file should never be edited by a user. During startup, this file is read if it exists, and then recreated again with the new discovery information.
   - SNMP_RW.ORA : The read-write information the agent has discovered of the services on the machine.

     Some of the information in this file can be edited by an administrator to provide more info about the discovered services.

     Upon Agent startup, this file is read if it exists, and after the discovery written again. Information is only added to this file.
5. Rename or remove the existing log files the agent has already created. This allows you to track any log messages since clean starting the Agent.

   The following files, located in the "$ORACLE_HOME/network/log" directory. They are used during startup of the agent:
   - NMICONF.LOG : Discovery log files
   - NMI.LOG : SQL*Net connection errors (Windows NT only)

- DBSNMP.LOG : Communication and working thread logging, if logging is enabled

You should also remove all files from $ORACLE_HOME/network/agent/library directory.

6. Create an empty SNMP_RO.ORA file and add the following line.

```
snmp.visibleservices = ()
```

7. If tracing is needed to debug a certain situation, a new "SNMP_RW.ORA" can be created, with the debug parameters:

```
dbsnmp.trace_level=16
dbsnmp.trace_unique=true
```

8. Restart the Agent.

After the agent has started, verify the "SERVICES.ORA" file first. If this file contains all the services on the machine, then check the "SNMP_RO.ORA" and "SNMP_RW.ORA" files. Discovery problems can be found in the file "NMICONF.LOG".

Important! Deleting the '.q' files erases all state information from the Agent, i.e., the Intelligent Agent has no information about the registered jobs and events.

Removing the '.q' files while jobs and events are submitted against the agent will result in synchronization errors between the framework and the agent. Therefore, use this method only on specific request by Customer Support when there is an explicit need for re-initializing the Intelligent Agent.

Events / jobs are internally identified by a sequence number. When a 'q' file is created, the sequence is reset. The Management Server on submission of an event realizes that the internal ID that it gets back is less than any other that it has previously registered for the node and triggers the message indicating there is a skew between the systems. The only option is to remove all the events and jobs and reregister them. As soon as the Management Server detects a mismatch of ID's, the Agent will be marked as 'corrupted', and all job or events you wish to send to this agent will fail with the error:

```
VNI-4040: The agent on node is not in sync with the Management
Server
```

9. Delete the contents of the $ORACLE_HOME / network / agent / reco directory.

Diagnosing Intelligent Agent Discovery Errors on UNIX

The discovery process on UNIX involves the following actions:

- Retrieve the primary host name and its aliases
- Read the ORATAB file. From this file, the list of databases, and the corresponding ORACLE_HOMEs are retrieved.
- Read the SQL*Net LISTENER.ORA and SQLNET.ORA files for each ORACLE_HOME read from the ORATAB file. The names of the database services are determined form the LISTENER.ORA file. These names are the names visible in the Enterprise Manager Console, and these names are used as TNS aliases to connect to the database.
- If a GLOBAL_DBNAME parameter for a database is not specified in the LISTENER.ORA file, the TNSNAMES.ORA file is scanned in search for an applicable TNS alias.

- If a TNS alias is not found, or if there is a duplicate TNS alias string, the Intelligent Agent generates a TNS alias for the database, and uses this generated name as the name for the service.

Files Used During Discovery on UNIX

listener.ora : Definitions of incoming SQL*Net connections
1 file per $ORACLE_HOME

Located in one of the following locations (using this order searching for it):

- $TNS_ADMIN
- /etc or /var/opt/oracle
- $ORACLE_HOME/network/admin

nmiconf.log : Intelligent Agent discovery warnings/errors
1 per Intelligent Agent

Located in $ORACLE_HOME/network/log

nmiconf.lst : List of third party additional discovery script to run
1 per Intelligent Agent

Located in $ORACLE_HOME/network/agent/config

nmiconf.tcl : Intelligent Agent discovery script
1 per Intelligent Agent

Located in $ORACLE_HOME/network/agent/config

oratab : File with all databases present on the machine
Only 1 per machine

Located in either /etc or /var/opt/oracle

services.ora : File with all service definitions the agent found
1 per Intelligent Agent

Located in $ORACLE_HOME/network/agent

snmp_ro.ora : File with all read-only service information
1 per Intelligent Agent

Located in $TNS_ADMIN or $ORACLE_HOME/network/admin

snmp_rw.ora : File with all updateable service information
1 per Intelligent Agent

Located in $TNS_ADMIN or $ORACLE_HOME/network/admin

sqlnet.ora : File with SQL*Net specific parameters
1 file per $ORACLE_HOME

Located in either (using this order searching for it):

- $TNS_ADMIN
- /etc or /var/opt/oracle
- $ORACLE_HOME/network/admin

tnsnames.ora : File with the TNS aliases to connect to databases
1 file per $ORACLE_HOME

Located in either (using this order searching for it):

- $TNS_ADMIN
- /etc or /var/opt/oracle
- $ORACLE_HOME/network/admin

Diagnosing the Discovery Problem on UNIX

Phase 1: Checking the ORATAB file

The ORATAB file is located in either the /etc, or the /var/opt/oracle directory.

You should consult your OS-specific documentation to see which directory is used to store the configuration files.

Things to look at:

- Dummy databases can be specified with a SID specified as '*'.
- The specification of the SID is case sensitive. This means that in order to have a valid service, the SID needs to specified in the exact same case in ALL configuration files.
- If you do not want certain database(s) to be visible in Enterprise Manager, you can prevent databases from being discovered by using the ORATAB file:

1. Copy the ORATAB file to another location. The physical location of the file is not important, as long as the Intelligent Agent is capable of reading the file on startup.
2. Edit the copied file, and remove the database entries that you do not want discovered.
3. Set an environment variable. Example (using Csh):

   ```
   ORATAB=$ORACLE_HOME/network/agent/oratab; export ORATAB
   ```

4. Restart the Intelligent Agent.

The result of the ORATAB parsing is two lists:

- A list with potential database services.
- A list of ORACLE_HOMEs present on the machine.

Phase 2: Checking the SQLNET.ORA file

For every ORACLE_HOME on the system, the Agent looks for the SQL*Net or Oracle Net files. It needs the SQLNET.ORA and LISTENER.ORA files first, to obtain the database service definitions. Sometimes, in the case of missing information, the TNSNAMES.ORA file is also required.

The Agent searches for the SQL*Net files in the follow order:

1. First, it checks the environment in search for a TNS_ADMIN variable. If one is found, this directory is used for the retrieval of the SQLNET information.

Note: The Intelligent Agent uses its own environment. It does not perform a login (running the profile/login scripts), nor does it run the 'oraenv' script to get the information for a specific ORACLE_HOME. If certain TNS_ADMIN values are enforced in login scripts, or in the 'oraenv' script which are different from the one in the Agent environment, they are not used by the Intelligent Agent!

If you have more than one ORACLE_HOME on the system, and a TNS_ADMIN variable is specified, 'duplicate definition' warnings will be logged in the NMICONF.LOG file starting from the second ORACLE_HOME. This occurs because the Agent finds the same SQL*Net files for that home, which will naturally contains identical information. These warnings can be ignored.

---

2. If a TNS_ADMIN environment variable is not specified, the default OS specific configuration location is searched for SQL*Net files. This can either be the '/etc' directory or '/var/opt/oracle', depending on the UNIX flavor you are working on.

   Note: If the Agent finds files in the OS specific location, and there are several ORACLE_HOMEs on the system, 'duplicate definition' warnings will be logged in the NMICONF.LOG file.

3. Finally, if nothing is found, the default $ORACLE_HOME/network/admin directory is searched for the necessary files.

   Note: If SQL*Net files are not found in a particular $ORACLE_HOME, the Agent skips to the next home and searches for the info there. If after completion the scan of all homes, information about a specific SID found in the ORATAB file is not found, a warning will be logged in the NMICONF.LOG file saying the SID will be skipped. This is a common warning when the Intelligent Agent is installed in a separate ORACLE_HOME: During the installation of the Agent, the install routines will adjust the ORATAB file, but there is no actual database with the SID name given during the installation.

   Once the SQL*Net configuration directory is established, the actual reading of the information can begin.

   Only one parameter is read from the SQLNET.ORA file: The names.default_domain parameter.

Phase 3: Checking the LISTENER.ORA file.

Using the same SQL*Net configuration directory, the information from the listener.ora file is read.

This contains two parts:

- List of how a service can be contacted using this listener. The protocol definitions, as specified in the listener definition itself. TCP protocol definition is used by the Agent to define the way it can connect to this database. It is also used in cases where the Agent needs to scan the TNSNAMES.ORA files to verify which TNS entry matches which listener definition for a database.

Note 1: If the protocol descriptions contains a lot of IPC entries and other non-TCP definitions, the length of the protocol definition can grow quite large.

If the length of the protocol definitions, added with the SID description part for a database surpasses the SQL*Net limit of 256 characters, the error ORA-12163 is generated when a connection is made using the TNS string the Agent constructed.

ORA-12163 TNS: Connect descriptor is too long.

The workaround here is to remove some of the non-TCP entries from the listener definition and restart the Agent.

Note 2: If the hostname defined in variable HOST of TCP definition does not match the hostname discovered by the Intelligent Agent, a message will be logged in NMICONF.LOG:

Could not find corresponding listener ...

Then SID listened by this listener will be skipped and if no other listener listens to them, the following messages are logged in listener.ora :

No listener found for SID ...

- The list of all databases this listener is working for. Every SID encountered in the SID description part of the LISTENER.ORA file is verified:
  - The 'extproc' SID description is skipped automatically.
  - If the SID was not present in the ORATAB file it is skipped.
  - If the SID is already encountered in another LISTENER.ORA file on the system it is skipped.

    A message is logged in the NMICONF.LOG.

    Example:

    ```
    Warning : Multiple Listeners found for SID ORCL.
    ```

- For all remaining SIDs found, a service name is generated. This service name is unique on the machine. In case no service name is enforced in the LISTENER.ORA file with the GLOBAL_DBNAME parameter, the TNSNAMES.ORA file is searched for a valid service name.

  If there are duplicates service names encountered, the Agent constructs a new unique service name for this database. A message appears in the NMICONF.LOG warning about the newly constructed name.

  Note: When different machines are using the same service names, the discovery from the Enterprise Manager navigator fails, and the duplicate service names is added to the list of managed services in Enterprise Manager.

The end result is a list of listeners. And for each listener the list of SIDs the listener works for. Every SID in these lists on its turn has, a list of the details needed for that database service.

Phase 4: Verifying the Information

As soon as all the files are parsed and treated, and all services are found, the Agent verifies if all the information is present and valid.

- If there is a listener does not have any SID, a warning is placed in the NMICONF.LOG saying the listener is skipped.

  Example:

  Warning : Listener LISTENER defined in

  /oracle/815/network/admin/listener.ora will be ignored.

- If there is a database found in the ORATAB, which does not have a listener working for it, the SID will not be 'discovered'.

  A message appears in the NMICONF.LOG saying the SID will be skipped.

  Example:

  Warning : No Listener found for SID ORCL. ORCL will be skipped

  All remaining information is considered 'discovered' and is placed in the discovery files SNMP_RO.ORA, SNMP_RW.ORA and SERVICES.ORA.

Important: The SERVICES.ORA and SNMP_RO.ORA file are ALWAYS completely rewritten. The SNMP_RW.ORA file is, however, only updated with the information about the new services. If is therefore quite possible to have gaps in the indexes, or to find information about services in this file which are no longer present on the machine.

Diagnosing Agent Discovery Errors on Windows NT

The discovery process on Windows NT involves the following actions:

- Get the primary host name and its aliases
- Get a list of the Oracle services from the registry. Databases with their the corresponding ORACLE_HOMEs are retrieved from this list.
- For each ORACLE_HOME encountered, the SQL*Net files LISTENER.ORA and SQLNET.ORA are read. From the LISTENER.ORA file, the names of the database services are determined. These names are the names visible from the Enterprise Manager Console. These names are also used as TNS aliases to connect to the database.
- If there is no GLOBAL_DBNAME parameter specified in the LISTENER.ORA file for a database, the TNSNAMES.ORA file is scanned in search for an applicable TNS alias.
- If no TNS alias is found, or in case of duplicate TNS alias string, the Intelligent Agent generates a TNS alias for the database, and will use this generated name as the name for the service.

Files Used During Discovery

listener.ora : File with definitions of incoming SQL*Net connections

1 file per $ORACLE_HOME

Located in either (using this order searching for it):

- $TNS_ADMIN in the environment
- $TNS_ADMIN in the registry
- $ORACLE_HOME\network\admin

nmiconf.log : File with Intelligent Agent discovery warnings/errors
1 per Intelligent Agent

Located in:

- $ORACLE_HOME\network\log

nmiconf.lst : List of 3rd party additional discovery script to run
1 per Intelligent Agent

Located in:

- $ORACLE_HOME\net80\agent\config (Version 8.0.X only)
- $ORACLE_HOME\network\agent\config

nmiconf.tcl : Intelligent Agent discovery script
1 per Intelligent Agent

Located in:

- $ORACLE_HOME\net80\agent\config (Version 8.0.X only)
- $ORACLE_HOME\network\agent\config

services.ora : File with all service definitions the agent found
1 per Intelligent Agent

Located in:

- $ORACLE_HOME\net80\agent (Version 8.0.X only)
- $ORACLE_HOME\network\agent

snmp_ro.ora : File with all read-only service information
1 per Intelligent Agent

Located in either (in order according to search priority):

- $TNS_ADMIN in the environment
- $TNS_ADMIN in the registry
- $ORACLE_HOME\net80\admin (Version 8.0.X only)
- $ORACLE_HOME\network\admin

snmp_rw.ora : File with all updateable service information
1 per Intelligent Agent

Located in either (in order according to search priority):

- $TNS_ADMIN in the environment
- $TNS_ADMIN in the registry
- $ORACLE_HOME\net80\admin (Version 8.0.X only)
- $ORACLE_HOME\network\admin

sqlnet.ora : File with SQL*Net specific parameters
1 file per $ORACLE_HOME

Located in either:

- $TNS_ADMIN in the environment
- $TNS_ADMIN in the registry
- $ORACLE_HOME\net80\admin (Version 8.0.X only)
- $ORACLE_HOME\network\admin

tnsnames.ora : File with the TNS aliases to connect to databases
1 file per $ORACLE_HOME

Located in either (ordered according to search priority):

- $TNS_ADMIN in the environment
- $TNS_ADMIN in the registry
- $ORACLE_HOME\net80\admin (Version 8.0.X only)
- $ORACLE_HOME\network\admin

Diagnosing the Discovery Problem on NT

Phase 1: Scanning the registry

The registry is scanned for database services. For each 'OracleService' NT service found, a potential database service entry is created, and the corresponding ORACLE_HOME is determined.

Things to point here:

- The specification of the SID is case insensitive.
- You cannot prevent discovery of the databases here, like with the ORATAB on UNIX.

Scanning the registry generates two lists:

- A list with potential database services.
- A list of ORACLE_HOME's present on the machine.

Phase 2: Scanning the SQLNET.ORA file

For every ORACLE_HOME on the system, the Agent looks for the SQL*Net / Oracle Net files. It needs the SQLNET.ORA and LISTENER.ORA files first, to get the database service definitions. Sometimes, in case of missing information, the TNSNAMES.ORA file is also required.

The Agent looks for the SQL*Net files in this order:

1. It checks the environment in search for a TNS_ADMIN variable. If one is found, this directory is used for the retrieval of the SQL*Net / Oracle Net information.
2. Next, the registry is checked in search for a TNS_ADMIN variable in the 'HOMEx' of the ORACLE_HOME the Agent is verifying. If one is found there, it is used.
3. Finally, if nothing is found, the default $ORACLE_HOME\net80\admin (Versions 8.0.X only) or $ORACLE_HOME\network\admin directory is searched for the necessary files.

> **Note:** If no SQL*Net / Oracle Net files are found in a particular $ORACLE_HOME, the Agent skips to the next home and searches for the info there. If after all Oracle_homes are scanned and information about a specific SID found in the registry is not found, a warning is logged in the NMICONF.LOG file saying the SID will be skipped.

Once the SQL*Net / Oracle Net configuration directory is established, the actual reading of the information can begin.

Only one parameter is read from the SQLNET.ORA file: The names.default_domain parameter.

Phase 3: Looking at LISTENER.ORA file

Using the same SQL*Net / Oracle Net configuration directory, the information from the listener.ora file is read.

This contains two parts:

- List of how a service can be contacted using this listener. This list consists of protocol definitions as specified in the listener definition itself. TCP protocol definition will be used by the Agent to define the way it can contact this database. It will also be used in case the Agent needs to scan the TNSNAMES.ORA files to verify which TNS entry matches which listener definition for a database.

  Note: If the protocol descriptions contains a lot of IPC entries and other non-TCP definitions, the length of the protocol definition can grow quite large. If the length of the protocol definitions, added with the SID description part for a database surpasses the SQL*Net / Oracle Net limit of 256 characters, the error ORA-12163 will be generated when a connection is made using the TNS string the Agent constructed.

  ORA-12163 TNS:Connect descriptor is too long

  The workaround here is to remove some of the non-TCP entries from the listener definition and restart the Agent.

  Note: If the hostname defined in variable HOST of TCP definition does not match the hostname discovered by the Intelligent Agent, a message will be logged in NMICONF.LOG :

  Could not find corresponding listener ...

  Then SID listened by this listener will be skipped and if no other listener listens to them following messages will be logged in listener.ora :

  No listener found for SID ...

- The list of all databases this listener is working for. Every SID encountered in the SID description part of the LISTENER.ORA file will be verified:
  - The 'extproc' SID description is skipped automatically
  - If the SID was not present in the registry file it is skipped
  - If the SID is already encountered in another LISTENER.ORA file on the system it is skipped. A message will be logged in the NMICONF.LOG.

  Example:

```
Warning : Multiple Listeners found for SID ORCL.
```

- For all remaining SID's found a service name will be generated. This service name will be unique on the machine. In case no service name is enforced in the LISTENER.ORA file with the GLOBAL_DBNAME parameter, the TNSNAMES.ORA file will be searched for a valid service name.

  If there are duplicates service names encountered, the Agent will construct a new unique service name for this database. A message will appear in the NMICONF.LOG warning about the newly constructed name.

The end result here is a list of listeners, with for each listener the list of SID's this listener is working for. Every SID in those lists on its turn has a list of the details needed for that database service.

Phase 4: Verifying the information

As soon as all the files are parsed and processed, and all services are found, the Agent verifies that all the information is present and valid.

- If there is a listener with no SIDs, a warning is placed in the NMICONF.LOG saying the listener will be skipped.

  Example:

  ```
  Warning : Listener LISTENER defined in C:\ORA920\network\admin\listener.ora
  will be ignored
  ```

- If a database has no listener servicing it, the SID will not be 'discovered'. A message will appear in the NMICONF.LOG saying the SID will be skipped.

  Example:

  ```
  Warning : No Listener found for SID ORCL. ORCL will be skipped
  ```

- At startup the Agent tries to connect to each discovered database. Therefore each open database should have two dbsnmp sessions once the Agent is started :

  Select username from v$session where username = 'DBSNMP'

  If no data retrieved and the Agent is running, check that the user DBSNMP exists and connects ok.

- All remaining information is considered 'discovered' and is placed in the discovery files SNMP_RO.ORA, SNMP_RW.ORA and SERVICES.ORA.

Important: The SERVICES.ORA and SNMP_RO.ORA file are ALWAYS rewritten completely.

The SNMP_RW.ORA file is however only updated with the information about the new services. If is therefore quite possible to have gaps in the indexes, or to find information about services in this file which are no longer present on the machine.

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

B

C

D

E

G

I

J

L

M

N

O

P

Q

R

S

T

U

W