# Oracle Intelligent Agent

ユーザーズ・ガイド

リリース 9.0.1

2001 年 10 月

部品番号 : J04227-01

ORACLE®

Oracle Intelligent Agent ユーザーズ・ガイド , リリース 9.0.1

部品番号 : J04227-01

原本名 ：Oracle Intelligent Agent User's Guide, Release 9.0.1

原本部品番号 ：A88771-01

# 目次

## 3 9*i* の新機能



## 4 ジョブ・スクリプトとイベント・スクリプト

## B トラブルシューティング



## C キーボード・ショートカット

## 用語集

## 索引

# はじめに

## このマニュアルの目的

このマニュアルには、Oracle Intelligent Agent の構成情報およびトラブルシューティングに関する重要な疑問への回答が記載されています。『Oracle Intelligent Agent ユーザーズ・ガイド』は、バージョン 1 およびバージョン 2 の Oracle Enterprise Manager の利用者に加えて、Intelligent Agent を介して Oracle データベースと通信する、その他のサポート対象システム管理フレームワークの利用者を対象としています。

## このマニュアルの対象読者

このマニュアルは、UNIX または Windows NT プラットフォーム上で Oracle Intelligent Agent のインストール、構成またはトラブルシューティングを行う、すべての人を対象としています。ほとんどの状況下において、Agent が必要とする構成およびメンテナンスの手段はごくわずかです。このため、『Oracle Intelligent Agent ユーザーズ・ガイド』は、最初から順番に読むのではなくリファレンスとして使用してください。

# このマニュアルの構成

| | |
|---|---|
| **第 1 章** | Intelligent Agent および検出サービスの機能の概要について説明します。 |
| **第 2 章** | Intelligent Agent のインストールおよび構成の手順について説明します。 |
| **第 3 章** | Intelligent Agent の現行のリリースについて、新機能を説明します。 |
| **第 4 章** | Tcl を使用した、ジョブおよびイベントのスクリプト記述について説明します。 |
| **付録 A** | Oracle Enterprise Manager によって使用される、必要な構成ファイルについて説明します。 |
| **付録 B** | Intelligent Agent のトラブルシューティングのガイドラインを提供し、その手順について説明します。 |
| **付録 C** | 一般的な Windows のキーボード・ショートカットのリストが記載されています。 |

# ドキュメント・セット

Oracle Enterprise Manager リリース 9.0.1 のドキュメントには、次のものがあります。

- 『Oracle Enterprise Manager 日本語リリース・ノート』には、ソフトウェアの更新その他の最新情報に関する重要な注意や、製品の動作とドキュメントでの記述との違いが記載されています。
- 『Oracle Enterprise Manager 構成ガイド』には、Oracle Enterprise Manager システムの構成に関する情報が記載されています。
- 『Oracle Enterprise Manager 概説』には、Enterprise Manager システムの概要が記載されています。
- 『Oracle Enterprise Manager 管理者ガイド』では、Oracle Enterprise Manager システムのコンポーネントおよび機能を説明しています。
- 『Oracle Intelligent Agent ユーザーズ・ガイド』では、Oracle Intelligent Agent の管理方法を説明しています。
- 『Oracle Enterprise Manager メッセージ・マニュアル』には、Oracle Enterprise Manager でのエラーについて、考えられる原因および推奨される処置が記載されています。

Oracle Enterprise Manager ドキュメント・セットに加えて、Oracle Enterprise Manager のコンポーネントには、詳細なオンライン・ヘルプが用意されています。

# 関連ドキュメント

詳細は、次の各ドキュメントを参照してください。

- 『Oracle Enterprise Manager 構成ガイド』
- 『Oracle SNMP サポート・リファレンス・ガイド』

# 1

# Intelligent Agent の概要

この章では、Intelligent Agent の概要と特長について簡単に説明します。

- Oracle Intelligent Agent:　概要
- 特長
- SNMP のサポート

# Oracle Intelligent Agent: 概要

Oracle Intelligent Agent は、ネットワーク内のリモート・ノード上で独立して動作するプロセスです。Agent は、サポートするサービスと同じターゲット上に存在し、次の機能を実行します。

- ローカル・サービスを提供する、またはオペレーティング・システム依存のサービスをコールすることにより、管理対象のターゲットとローカルに対話します。
- イベントをチェックし、結果として得られたイベント・レポートを Oracle Enterprise Manager にキューイングします。
- Oracle Enterprise Manager のジョブを実行してその結果および出力を収集し、必要に応じてその結果をキューイングします。
- データ収集を行います。
- コンソールまたはその他のアプリケーションの指示に従って、ジョブまたはイベントを取り消します。
- Intelligent Agent のプラットフォーム上で SNMP がサポートされている場合、イベントの SNMP トラップ送信の要求を処理します。

Agent の構成の詳細は、ご使用のシステムに適した、プラットフォーム固有の Oracle Server インストール・マニュアルを参照してください。

---

**注意：** リリース 9.0 では、Data Gatherer の機能が Intelligent Agent に統合されたため、独立したアプリケーションではなくなりました。

---

# 特長

Intelligent Agent は独立して動作するプログラムであり、その動作の前提としてコンソールまたは Management Server が動作している必要はありません。データベースにサービスする Agent は、データベースがダウンしているときも動作できるため、Agent によるデータベースの起動またはシャットダウンを行うことができます。Intelligent Agent は、管理者の介入なしに、管理ジョブを常に独立して実行できます。同様に、Agent は独自にイベントを検出して、そのイベントに対応できます。このため、Agent は、管理者の介入なしにシステムを監視し、修正ジョブを実行して問題を解決できます。

Agent はコンソールおよび Management Server とは無関係に稼働し、管理者がコンソールからログアウトしているときでもジョブを実行したり、イベントを監視したりできます。Agent は、管理者宛てのすべてのジョブまたはイベント・メッセージをキューイングし、それらを Management Server に通知します。管理者が再びコンソールにログインしたときに、Management Server は保留中のメッセージを現在ログインしている管理者に通知します。ジョブおよびイベントの状態に関する情報は、Agent の動作するノード上のファイルに格納されます。それらのファイルには .q の拡張子が付き、`$ORACLE_HOME/network/agent` ディレクトリに格納されます。

ジョブとイベントは Tcl スクリプトとして実装されます。あるイベントに対して Agent がジョブまたはテストを実行するとき、Agent により適切な Tcl スクリプトが実行されます。

Management Server が、コンソールにログインしている管理者の代理として Agent にメッセージを送るとき、管理者の言語およびキャラクタ・セット環境に関する情報も同時に送られます。Agent では、管理者の代理としてデータベース管理タスクを実行するとき、NLS 環境情報を使用します。これにより、管理者は自国語でデータベースを管理できます。たとえば、フランスの管理者が、ドイツ語でデータベースを管理し、メッセージをフランス語で受け取ることができます。

Intelligent Agent では、特定のカートリッジを使用して特定の型のデータを収集します。たとえば、オペレーティング・システムまたはデータベースの情報です。Agent は、そのノード上で検出されたサービスに関する統計データを収集できます。要求があると、Oracle Performance Manager および Oracle Capacity Planner とは無関係に収集が開始されます。また、オペレーティング・システムの評価、および Enterprise Manager コンソールで登録されるデータベース固有の測定イベントに使用される統計データも収集されます。

他の型のデータを収集する場合、インテグレータが独自のカートリッジを記述できます。

# SNMP のサポート

Agent は SNMP をサポートします。これにより、サード・パーティのシステム管理フレームワークは、SNMP を使用して Agent から SNMP トラップを直接受け取ることができます。Agent では、Oracle のデータベース MIB 変数にアクセスできます。SNMP をサポートしないプラットフォーム上にデータベースが存在しているときでも、Oracle MIB 変数にアクセスするジョブまたはイベントを送ることができます。SNMP の詳細は、『Oracle SNMP サポート・リファレンス・ガイド』を参照してください。

# 2

# Intelligent Agent のインストール

この章では、Intelligent Agent の基本設定および構成手順について説明します。次の項目について説明します。

- Intelligent Agent のインストール
- NT Agent の運用制御
- Windows NT および Windows 2000 での SNMP の構成
- UNIX Agent の運用制御
- UNIX での SNMP の構成
- 複数のネットワーク・カード（NIC）で使用するためのリリース 9.0.1 Agent の構成
- 複数のネットワーク・カード使用時の Agent の動作
- Oracle Intelligent Agent と Oracle Names
- Intelligent Agent に必要なロールとユーザー
- 自動検出
- サービス検出プロセス
- 8.0.6/8.1.6/8.1.7 Intelligent Agent の 9.0 へのアップグレード

# Intelligent Agent のインストール

Intelligent Agent はデータベースとともに出荷され、ORACLE_HOME 環境下で管理されるリモート・マシン上にインストールできます。Oracle Universal Installer では、Enterprise Manager のツリー・リストか、データベース・サーバーのツリー・リスト（この場合、データベースおよび Agent がインストールされます）のいずれかから Agent のインストールを選択できます。Intelligent Agent をスタンドアロンのサービスとしてインストールする場合は、Enterprise Manager のツリー・リストから Agent を選択してください。

# NT Agent の運用制御

Microsoft Windows NT システムで Intelligent Agent の運用を制御するには、次の手順に従います。

## Windows NT 上での Intelligent Agent の起動

Windows NT 上で Agent を起動する手順は、次のとおりです。

1. 「コントロール パネル」フォルダにある「サービス」アイコンをダブルクリックします。
2. 「Oracle<ORACLE_HOME_NAME>Agent」サービスを選択します。

   スタートアップの種類は「手動」に設定されています。これは、ユーザーが手動で Agent を起動するための設定です。システムを起動するたびに Agent が自動的に起動されるようにするには、スタートアップの種類を「自動」に設定します。

   a. 「スタートアップ」ボタンをクリックします。「サービス」ダイアログ・ボックスが表示されます。
   b. 「スタートアップの種類」で「自動」を選択します。
   c. 「サービス」ダイアログ・ボックスで「OK」をクリックします。
3. 「起動」ボタンをクリックして、Agent を起動します。

---

**注意：** コマンドラインで次のように入力しても、Agent を起動できます。

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

---

## Windows NT 上での Intelligent Agent の停止

Windows NT 上で Agent を停止する手順は、次のとおりです。

1. 「コントロール パネル」フォルダにある「サービス」アイコンをダブルクリックします。
2. 「OracleAgent」サービスを選択します。
3. 「停止」ボタンをクリックして、Agent を停止します。

---

**注意：** コマンドラインから次のように入力しても、Agent を停止できます。

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

---

### Agent の動作検証

Agent が動作していることを検証するには、「コントロール パネル」の「サービス」でそのステータスを調べるか、コマンド・プロンプトで `net start` と入力します。動作しているサービスのリストに「OracleAgent」が表示されます。

「Windows NT タスク マネージャ」で、`dbsnmp` プロセスの情報を確認するという方法もあります。

## ジョブ実行のための Windows NT ユーザー・アカウントの作成

管理対象のノード上で Agent がジョブを実行するためには、次の条件が満たされている必要があります。

- 高度なユーザー権限、「バッチ ジョブとしてログオン」を持つ NT ユーザー・アカウントが存在すること。権限は既存のローカル・ユーザーまたはドメイン・ユーザー、あるいは新規 NT ユーザーに割り当てることができます。
- Oracle Enterprise Manager コンソールで、ノードに対する優先接続情報リストが前述のユーザーに対して設定されていること。優先接続情報リストの設定の詳細は、『Oracle Enterprise Manager 管理者ガイド』を参照してください。
- Agent を起動するユーザーに、`TEMP` ディレクトリまたは `ORACLE_HOME` ディレクトリに対する書込み許可と、`ORACLE_HOME¥network` ディレクトリに対する読取り / 書込み許可が与えられていること。

  ---

  **注意：** 「バッチ ジョブとしてログオン」権限が設定されていないと、管理対象のターゲット上でジョブを実行しようとする際に、「ユーザーの認証に失敗しました」というメッセージが表示されます。

  ---

- ユーザーがデータベースやリスナーなどの Windows NT サービスを開始および停止するために管理者権限を持っていること。

次の手順のいずれかを実行してください。

## 新規 NT ユーザー・アカウントの作成

ローカルの Windows NT マシン上で新規 NT ユーザー・アカウントを作成し、そのユーザーに「バッチ ジョブとしてログオン」権限を付与する手順は、次のとおりです。

1. 「管理ツール」プログラム・グループから「ユーザー マネージャ」を選択します。このツールの詳細は、Windows NT のドキュメントを参照してください。
2. 「ユーザー」メニューの「新しいユーザー」を選択して、次の点を確認します。
   - 「ユーザーは次回ログオン時にパスワード変更が必要」オプション・ボックスがチェックされていないこと
   - ユーザー名に SYSTEM または system が使用されていないこと
3. ユーザー・マネージャの「原則」メニューから「ユーザーの権利」を選択します。
4. 「高度なユーザー権利の表示」ボックスをチェックします。
5. 権限のリストから「バッチ ジョブとしてログオン」を選択します。
6. 選択したユーザーにこの権限を付与します。

## 既存の NT ユーザー・アカウントへの権限の割当て

既存のローカル・ユーザー・アカウントに権限を割り当てる手順は、次のとおりです。

1. 「ユーザー マネージャ」パネル上でユーザーを選択し、次の点を確認します。
   - 「ユーザーは次回ログオン時にパスワード変更が必要」オプション・ボックスがチェックされていないこと
   - ユーザー名に SYSTEM または system が使用されていないこと
2. ユーザー・マネージャの「原則」メニューから「ユーザーの権利」を選択します。
3. 「高度なユーザー権利の表示」ボックスをチェックします。
4. 権限のリストから「バッチ ジョブとしてログオン」を選択します。
5. このユーザーに高度なユーザー権利を追加します。

## Windows 2000 の新規ユーザー・アカウントの作成

ローカルの Windows 2000 マシン上で新規ユーザー・アカウントを作成し、そのユーザーに「バッチ ジョブとしてログオン」権限を付与する手順は、次のとおりです。

1. 「コントロール パネル」→「管理ツール」→「ローカル セキュリティ ポリシー」→「ローカル ポリシー」→「ユーザー権利の割り当て」を選択し、右側の画面で「バッチ ジョブとしてログオン」をハイライトします。
2. 「バッチ ジョブとしてログオン」を右クリックし、「セキュリティ」を選択します。
3. 「追加」をクリックし、追加するユーザーを選択します。
4. ユーザーを選択して「追加」をクリックし、「OK」をクリックします。

## ドメイン・ユーザーを Agent のユーザーとして構成

**注意：** ドメイン・ユーザーは、リリース 7.3.3 以下の Agent ではサポートされていません。

リポジトリ・ユーザーを Agent のユーザーとして構成する手順は、次のとおりです。

1. ユーザー・マネージャの「原則」メニューから「ユーザーの権利」を選択します。
2. 「高度なユーザー権利の表示」ボックスをチェックします。
3. 権限のリストから「バッチ ジョブとしてログオン」を選択します。
4. 「追加」ボタンをクリックします。
   a. 「ドメインまたはコンピュータ」フィールドに値を入力します。（ドメインを選択します。）
   b. 「ユーザーの表示」ボタンをクリックします。
   c. リスト・ボックスからドメイン・ユーザーを選択します。
   d. 「追加」をクリックします。
   e. 「OK」をクリックします。
5. 「ユーザー権利の原則」ダイアログで、「OK」をクリックします。

**注意：** 同じ名前のローカル・ユーザーとドメイン・ユーザーが存在する場合、ローカル・ユーザーが優先されます。

# Windows NT および Windows 2000 での SNMP の構成

Windows NT および Windows 2000 上のサブエージェントをサポートするよう SNMP マスター・エージェントを構成する手順は、次のとおりです。

1. 次のようなサービス・ファイルがあるドライブを選択します。

   x:¥winnt¥system32¥drivers¥etc¥services

   SNMP エントリを次のように変更します。

   snmp 1161/udp

   snmp-trap 1162/udp

2. 次のパスにある Peer SNMP Master Agent 構成ファイル、MASTER.CFG を開きます。

   ```
   ORACLE_HOME¥network¥admin
   ```

   この構成ファイルを編集して、次の内容を追加します。

   a. トランスポート・エントリは次のとおりです。

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

   b. SNMP トラップを受け取る必要のあるコミュニティおよびマシン（例：dlsun1000.us.oracle.com、または IP アドレス）を指定するエントリは、次のとおりです。

   ```
   COMMUNITY   public
   ALLOW ALL OPERATIONS
   USE NO ENCRYPTION

   MANAGER dlsun1000.us.oracle.com
   SEND ALL TRAPS
   WITH COMMUNITY PUBLIC
   ```

3. 「コントロール パネル」の「サービス」パネルから、Peer SNMP Master Agent およびカプセル化機能を起動します。

   カプセル化機能が必要なのは、マシン上に複数のサブエージェントがインストールおよび構成されている場合のみであることに注意してください。Peer SNMP Master Agent 実行可能ファイルは ORACLE_HOME¥bin¥agent.exe、カプセル化機能の実行可能ファイルは ORACLE_HOME¥bin¥encaps.exe です。

4. 「コントロール パネル」の「サービス」パネルからサブエージェント（Oracle Intelligent Agent）を起動します。サブエージェントは、起動するとマスター・エージェントに登録されます。

   Oracle の SNMP をテストする場合は、Oracle SNMP マスター・エージェントとの通信、および Oracle 固有データの問合せに SNMP を使用する、サード・パーティ製アプリケーションを使用してください。

SNMP をサポートしているサード・パーティ製アプリケーションは、『Oracle SNMP サポート・リファレンス・ガイド』を参照してください。

# UNIX Agent の運用制御

UNIX システムで Intelligent Agent の運用を制御するには、次の手順に従います。

## root.sh シェル・スクリプトの実行

Agent のインストールに成功すると、`root.sh` の実行を求めるメッセージが Oracle Universal Installer により表示されます。

`root.sh` は、`oratab` ファイルの更新または作成を行うシェル・スクリプトです。`oratab` ファイルは、Agent により検出され、Oracle Enterprise Manager により制御される、すべてのデータベースへの参照をユーザーが記述するファイルです。作成されるデータベースごとに、`<SID>:<$ORACLE_HOME>:[Y/N]` という形式のエントリがあります。

Agent は通常、`root.sh` により、`setuid` プログラムとして構成されます。`root.sh` の実行に成功した場合、`setuid root` という名前で Agent がインストールされます。これにより Agent は、ホストに対する優先接続情報リストでその名前とパスワードが与えられるユーザーとして、ジョブを実行できるようになります。

---

**注意：** Agent が `setuid root` に設定されることは、`root` ユーザーが Agent を起動することと同じ効果を持つわけではありません。`root` ユーザーが Agent を起動すると、セキュリティ上の問題が生じる可能性があります。`setuid` プログラムの詳細は、ご使用のプラットフォームのマニュアルを参照してください。

---

ノード・ジョブを UNIX Agent に送るユーザーには、Agent の ORACLE_HOME ディレクトリに、読取り / 書込みのアクセス権を付与してください。`root.sh` に `setuid` の設定がない場合、Agent に送られるすべてのジョブは、Agent の実行可能ファイル（dbsnmp.exe）を所有するユーザーの権限のもとで実行されます。`root.sh` は、Agent 上のすべてのジョブに対して、Oracle Enterprise Manager コンソールで優先接続情報リストを設定するようユーザーに強制します。

### root.sh の実行成功の検証

root.sh が正しく実行されたことを検証するには、dbsnmp についてファイルの許可を確認します。

1. cd $ORACLE_HOME/bin と入力します。

   これにより、Agent の実行可能ファイルがある、$ORACLE_HOME/bin にディレクトリが変更されます。

2. ls -al dbsnmp と入力します。

   dbsnmp について、すべての関連詳細がリスト表示されます。

dbsnmp に対する ls -al コマンドの出力は、次のようになります。

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

root は所有者です。dbsnmp は Agent の実行可能ファイルです。この例では、グループの名前は g651 です。所有者が root、許可が -rwsr-s--- であれば、root.sh は正しく実行されています。

## UNIX プラットフォーム上での Agent の起動および停止

UNIX では、Agent の起動および停止に agentctl コマンドを使用します。また、*dbsnmpwd* も実行されます。*dbsnmpwd* とは、Agent が停止した場合に Intelligent Agent を自動的に起動する UNIX の監視スクリプトです。これによって、明示的に停止しないかぎり Agent は常に起動状態になります。

関連する agentctl コマンドの一覧を、次の表に示します。

| 操作 | 入力するコマンド |
|---|---|
| Agent および *dbsnmpwd* を起動 | agentctl start agent |
| Agent および *dbsnmpwd* を停止 | agentctl stop agent |
| Agent のステータスを検証 | agentctl status agent |

### DBSNMPWD

Dbsnmpwd は、*dbsnmp*（Intelligent Agent）のプロセスが監視対象のターゲットに常に存在するようにする UNIX の監視スクリプトです。Agent が異常終了した、すなわち予期しないリターン・コードで終了した場合、このスクリプトにより Agent が再起動されます。

監視スクリプトの動作は、次の環境変数を使用して構成できます。これらの変数は、スクリプト自体の中に記述されています。

> **注意：** 環境変数は、snmp_rw.ora ファイル内では設定しないでください。

DBSNMP_WDLOGFILE: 起動メッセージが記述されるログ・ファイル。デフォルトでは、$ORACLE_HOME/network/log/dbsnmp.nohup になります。

DBSNMP_RESTART: 自動再起動のメカニズムを無効にする場合、この環境変数を 0 に設定します。デフォルトは 1 です。

DBSNMP_MAX_ABNORMAL_EXIT および DBSNMP_TIME_DELTA: これら 2 つの変数は、Agent がスラッシングしている（Agent を正常に起動するための初期条件が満たされず、起動直後に停止する状態にある）と想定しても問題のないタイミングを、監視スクリプトが判断する際に機能します。デフォルト値は、それぞれ 3 と 60 です。この場合、Agent が 60 秒以内に 4 回以上異常終了すると、Agent がスラッシングしていると判断され、監視スクリプトが再起動を行わなくなります。

## UNIX での SNMP の構成

UNIX システムでは、Oracle Intelligent Agent（IA）のインストール時に、SNMP ファイルも自動的にインストールされます。SNMP を構成する手順は、次のとおりです。

1. Intelligent Agent をインストールします。

次のコマンドで、SNMP マスター・エージェント（snmpd プロセス）が稼働していないことを確認します。

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

snmpd（Solaris では snmpd.cfddi）プロセスが稼働している場合は、それを停止します。

2. cd $ORACLE_HOME/network/snmp/peer
3. CONFIG.master スクリプトで、同じディレクトリ内の SNMP Console を実行しているマシンの IP アドレスを編集し、start_peer スクリプトが存在することを確認します。
4. su root
5. start_peer -a（Master Peer エージェント、カプセル化機能およびシステム固有の SNMP デーモンを起動）
6. root から exit します。

7. Enterprise Manager コンソールで、イベントを登録し、「外部サービス（Agent による SNMP トラップ）への通知を使用可能にする」のチェックボックスをチェックします。トリガーされる各 EM イベントについて、トラップが SNMP Master コンソールに送られます。

---

**注意：**

- SNMP の構成はプラットフォームごとに異なります。各プラットフォームのドキュメントを確認してください（インストールおよび構成のガイドに、この情報が記載されています）。
- 一部の UNIX プラットフォームでは、SNMP がサポートされていません。そのプラットフォームのドキュメントを確認してください。
- リリース 8.0.5 より前では、Windows NT で SNMP がサポートされません。

---

サード・パーティのシステム管理アプリケーションは、SNMP Master Agent を使用して Intelligent Agent と通信します。Oracle Intelligent Agent が SNMP を使用して Master Agent と通信できるようにするには、SNMP Master Agent および Oracle Intelligent Agent を正しく構成する必要があります。

Oracle データベースおよび Management Server に対して SNMP を構成する一般的な手順については、『Oracle SNMP サポート・リファレンス・ガイド』を参照してください。

より広範な構成情報については、ご使用のプラットフォームのインストレーション・ガイドまたは構成ガイドを参照してください。SNMP の構成は、プラットフォームによって異なります。

# 複数のネットワーク・カード（NIC）で使用するためのリリース 9.0.1 Agent の構成

リリース 8.1.7 の Intelligent Agent と同様に、9*i* Intelligent Agent ユーザーは、複数のネットワーク・カードを持つ 1 台のマシン上にある Agent の構成において、3 つのオプションを選択できます。デフォルトでは、Agent はそのマシン上の主 NIC（UNIX プラットフォームでは le0、Windows NT では network0）にバインドされます。他の 2 つのオプションは、次のとおりです。

**a.** ユーザーが指定した NIC にバインドできます。

**b.** マシン上のすべての NIC にバイドできます。このオプションは、すべての NIC 上でその Agent にリスニングをさせることが好ましくない場合は、使用しないでください。

**注意：** Agent は、EM ジョブ、イベントおよびデータ収集の実行を求めて送られてくるすべての要求をリスニングするために、IP アドレスにバインドされ、そのアドレスを使用します。

Agent では、Agent で使用されている IP アドレス /NIC と異なる IP アドレス /NIC でリスニング中のサービス（リスナーなど）を検出することもできるようになります。

# 複数のネットワーク・カード使用時の Agent の動作

1. **リスニング・アドレスが指定されていない場合**

   snmp_rw.ora に明示的なリスニング・アドレス・ディレクティブがない場合、Agent は、Agent マシンの主ネットワーク・インタフェース・カードを介した接続をリスニングします。

2. **リスニング IP アドレスが指定されている場合**

   *snmp_rw.ora* で明示的なリスニング・アドレスが指定されている場合、Agent はそのアドレスでの接続のみをリスニングします。

   主ネットワーク・カード以外の特定のネットワーク・インタフェース・カードに 9*i* Intelligent Agent をバインドする手順は、次のとおりです。

   1. dbsnmp.hostname パラメータを設定します。

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

   2. コマンドラインで次のように入力し、Agent を起動します。

      ```
      >agentctl start agent
      ```

3. **ホスト名が指定されている場合**

   *snmp_rw.ora* でホスト名が指定されている場合、Agent はマシンのすべてのネットワーク・インタフェース・カードでの接続をリスニングします。

   8.1.7 より前のリリースの Agent（8.1.5 以上）では、これが Agent のデフォルトの動作です（Agent で使用するネットワーク・レイヤー・コードのデフォルトの動作です）。

   9*i* Intelligent Agent をホスト上のすべてのネットワーク・インタフェース・カードにバインドする手順は、次のとおりです。

   1. dbsnmp.hostname を設定します。

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

   2. Agent を起動します。

4. **Windows NT Fail Safe 構成が使用されている場合**

   Windows NT Fail Safe 構成では、Agent は Fail Safe Agent の NT レジストリに格納されている IP アドレスでの接続をリスニングします。

   Agent では、ターゲットの構成ファイルで使用されているマシン名または IP アドレスに関係なく、マシン上の各ターゲットを検出します。

# Oracle Intelligent Agent と Oracle Names

Oracle Intelligent Agent の管理対象となるマシン上で Oracle Names を実行している場合、データベースがすでに Names Server で登録されており、データベースの別名が `listener.ora` ファイルの GLOBAL_DBNAME パラメータによって定義されていることが想定されます。

Intelligent Agent では、管理対象のサービスを検出する際に Oracle Names を使用しません。リスナーがどのデータベースにサービスするかは、`listener.ora` ファイルの GLOBAL_DBNAME パラメータによって判断されます。この名前は、管理対象のデータベース名として Enterprise Manager コンソールのナビゲータに表示されます。

GLOBAL_DBNAME パラメータには一般に、Names Server で登録されているものと同一のデータベース名が記述されます。たとえば、データベース初期化パラメータ・ファイルで与えられている、データベースの名前およびドメインが記述されます。GLOBAL_DBNAME パラメータの値は一意である必要があります。

この環境でジョブを実行している、またはイベントを監視しているとき、Intelligent Agent でのデータベース別名の解決に Oracle Names は使用されません。Agent は Bequeath プロトコルを使用して、独自の TNS 接続文字列を生成します。

---

**注意：** 同じノード上で複数の Oracle データベースを管理しようとしている場合、`listener.ora` ファイルの GLOBAL_DBNAME パラメータがデータベースごとに異なっていることを確認してください。

---

# Intelligent Agent に必要なロールとユーザー

Agent に対するデフォルト・データベースのユーザー名 / パスワードは、dbsnmp/dbsnmp です。catsnmp.sql スクリプトが、データベースとともにインストールされます。データベースのロールおよび権限は、catsnmp.sql スクリプトを使用して Agent ユーザーに割り当てられます。Oracle データベースのインストール時には、catalog.sql によって catsnmp.sql スクリプトが自動的に実行されます。

Intelligent Agent のデータベース・ログイン用のユーザー名およびパスワードを、ユーザーが変更しなければならない場合があります。ユーザー名およびパスワードを dbsnmp/dbsnmp 以外のものに変更するには、snmp_rw.ora を編集して、次のパラメータを追加する必要があります。

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

新規の Agent ユーザーに必須のロールおよびユーザー権限を付与するには、新規 Agent ユーザーを参照する catsnmp.sql で指定されている SQL コマンドを実行します。Server Manager または SQL*Plus を使用できます。この目的で、catsnmp.sql スクリプトを直接編集しないでください。

SNMPAGENT のロールがデータベースに存在するかどうかを確認するには、次の SQL コマンドを入力します。

```
SELECT * FROM dba_roles;
```

SNMPAGENT ロールが表示されない場合、データベースに対して catsnmp.sql スクリプトを実行します。

複数のバージョンのデータベースがすでに動作している場合、Agent がコンタクトする必要のあるすべての付与およびビューに対して正しい設定を行うには、各データベース上で catsnmp.sql スクリプトを実行する必要があります。

スクリプトを実行するには、SYS または INTERNAL でログインする必要があります。

---

**注意：** catsnmp.sql の場所は、動作しているデータベースのバージョンおよびプラットフォームによって異なります。たとえば、Windows NT 上の Oracle 9.x データベースでは、スクリプトは次の場所にあります。

```
ORACLE_HOME¥rdbms¥admin
```

---

# 自動検出

Intelligent Agent は、ビルトインの自動検出機能を備えています。これは、管理対象のサービスに関する情報が含まれる必要な構成ファイルを、プロセスの起動時に毎回、自動的に生成する機能です。検出プロセスの間に、次の 3 つのファイルが作成または追加されます。

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

  snmp_ro.ora ファイルは、Agent によって作成される読取り専用ファイルです。このファイルには、Agent によって監視されるサービスに関する情報が含まれています。

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

  snmp_rw.ora ファイルには、管理対象のサービスの索引情報が含まれています。この情報は Agent によって内部的に使用されます。また、ユーザーがトレースなどの変数を指定できるようにする働きもあります。

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

  services.ora ファイルには、Agent が監視する必要のあるすべてのサービスの別名が含まれています。このファイルにリストされたサービスのみが、Agent により監視されます。このファイルの内容はその後、検出の間にコンソールに送られます。リリース 9.0 では、Agent が稼働している環境のオペレーティング・システムの種類およびバージョンに関する情報も、このファイルに含まれます。

---

**注意：** これらのファイルで使用されるパラメータの詳細は、付録 A「Agent 構成ファイル」を参照してください。

---

Agent が起動すると、自動検出プロセスは次のソースから構成パラメータを読み込みます。

- `oratab`（UNIX ノード上）
- Windows NT レジストリ（Windows NT ノード上）
- `listener.ora`
- `tnsnames.ora`（存在する場合）

検出プロセスにより、ノード上にインストールされているサービスが抽出され、前述の構成ファイル群がコンパイルされます。

Agent は各 ORACLE_HOME に対する SID 情報を、`ORATAB` ファイル（UNIX の場合）または Windows NT レジストリから集めます。Agent は次に、listener.ora ファイルを解析して、関連する SID およびリスナー情報を探します。`listener.ora` ファイルに GLOBAL_DBNAME セクションがある場合、Agent はデータベース・サービス名を GLOBAL_DBNAME 変数に設定します。変数が存在しない場合、Agent は SID に対する有効なサービス名を含む tnsnames.ora を、そのマシン上で探します。該当するファイルを Agent が見つけられない場合、<SID>_<HOSTNAME> というサービス名が各 SID に対して作成されます。

**注意：** tnsnames.ora で、同じインスタンスに対して複数の別名が存在する場合、Agent は最初にリストされている別名を使用します。それ以外の別名を使用するには、tnsnames.ora のエントリの順序を入れ替えて、Agent を再起動します。

**注意：** 1 台のマシン上に複数のデータベース・インスタンスが存在し、`listener.ora`ファイルのGLOBAL_DBNAMEパラメータを使用している場合、これらのインスタンスは listener.ora に一意の GLOBAL_DBNAME を持つ必要があります。listener.ora を手動で編集しなければならない場合もあります。

## 自動検出のための前提条件

- SQL*Net バージョン 2 または Oracle Net TCP/IP が存在し、必要なファイル群が Intelligent Agent の起動に先立って作成されている必要があります。必要な SQL*Net（または Oracle Net）ファイルは `listener.ora` のみですが、特定のサービスを検出するには、`tnsnames.ora` および `sqlnet.ora` を正しく構成してください。Agent はこれらのファイルを、`$ORACLE_HOME/network/admin` ディレクトリで検索します。
- Agent による検出における TNS_ADMIN 変数の使用方法は次のとおりです。

  （UNIX）すべてのバージョンの UNIX 検出スクリプトで、TNS_ADMIN 変数を使用して入力構成ファイル（`listener.ora` および `tnsnames.ora`）の場所を特定できます。この環境変数が設定されている場合、7.3.4 または 8.0.3 より後のリリースでのみ、出力ファイル（`snmp_ro.ora` および `snmp_rw.ora`）を TNS_ADMIN に正しく書き込むことができます。TNS_ADMIN 変数が設定されていない場合、Agent は出力ファイルを `$ORACLE_HOME/network/admin` ディレクトリに書き込みます。

  （NT）リリース 8.0.5 以上では、前述のものに加えて、検出スクリプトにより TNS_ADMIN の値が NT レジストリから読み取られます。この変数は、次の場所にあります。

  - （NT）「コントロール パネル」→「システム」→「環境」タブ内の TNS_ADMIN 変数
  - （NT）NT レジストリ内の TNS_ADMIN キー

# サービス検出プロセス

Agent が起動時に最初に実行する操作は、Agent が監視するノード上にどのようなサービスが存在するかを検出することです。次の検出アルゴリズムに関するドキュメントでは、Agent が動作する最も一般的な 2 つのプラットフォームでのサービス検出プロセスについて説明します。

## Agent の検出プロセス（Windows NT）

Intelligent Agent を起動すると、スクリプトが実行され、Windows NT レジストリ、listener.ora ファイルおよび tnsnames.ora ファイル（存在する場合）から構成パラメータが読み込まれます。

Agent は、インストールされているマシン上で新規のサービスを検出し、その構成ファイルである snmp_ro.ora、snmp_rw.ora および services.ora を作成、再書込みまたは追加します。

Agent では、次の検出アルゴリズムを使用して、そのマシン上で使用可能なサービス（Agent により管理されるサービス）を判断します。

1. Agent は、Windows NT レジストリから読み取られた各データベース・サービスの ORACLE_SID および ORACLE_HOME 情報を記録します。
2. Windows NT レジストリの値を基に、Agent は listener.ora ファイルを読み取り、どのリスナーがどのデータベースにサービスするかを判断します。構成ファイル listener.ora の場所は、SQL*Net の構成ファイルの場所に基づきます。たとえば、TNS_ADMIN 環境変数と、$ORACLE_HOME¥network¥admin ディレクトリの場所は、Windows NT レジストリから読み取られた ORACLE_HOME 情報を基にして決まります。
3. 検出されたデータベースの名前は、そのデータベースの listener.ora ファイルで定義されている GLOBAL_DBNAME パラメータに基づきます。
4. listener.ora に GLOBAL_DBNAME パラメータが存在しない場合、Agent は listener.ora ファイルを検索したときと同じ方法で tnsnames.ora ファイルを検索します。
5. tnsnames.ora ファイルが見つからない場合、データベース別名 <SID>_<hostnames> がデータベース・サービスに割り当てられます。サービスはこの別名で Agent に認識され、Oracle Enterprise Manager コンソールにその名前で表示されます。

**注意：** TNSNAMES.ORA ファイルで、同じインスタンスに対して複数の別名が存在する場合、Agent は最初にリストされている別名を使用します。それ以外の別名を使用するには、TNSNAMES.ORA ファイルのエントリの順序を入れ替えて、Agent を再起動します。

Agent が存在するノード上に、データベースまたはその他の新しいサービスがインストールされた場合、Agent を再起動して、新しいサービスを Agent の構成ファイルに追加する必要があります。この手順は、UNIX バージョンの Intelligent Agent についても適用されます。

## Agent の検出プロセス（UNIX）

起動時に Agent は、インストールされているマシン上で新しいサービスを検出し、その構成ファイル snmp_ro.ora、snmp_rw.ora および services.ora を作成します。

Agent は次の検出アルゴリズムを使用して、そのマシン上で使用可能なサービス（Agent により管理されるサービス）を判断します。

1. Agent は oratab ファイルから、すべての Oracle ホームおよび SID の値を読み取ります。oratab ファイルが次のどちらの場所に格納されるかは、プラットフォームによって異なります。
   - /etc
   - /var/opt/oracle
2. oratab ファイルから読み取られた Oracle ホームの値を基に、Agent は listener.ora ファイルを検索して、どのリスナーがどのデータベースにサービスするかを判断します。
3. 検出されたデータベースの名前は、そのデータベースの listener.ora ファイルで定義されている GLOBAL_DBNAME パラメータに基づきます。
4. listener.ora に GLOBAL_DBNAME パラメータが存在しない場合、Agent は listener.ora ファイルを検索したときと同じ方法で tnsnames.ora ファイルを検索します。
5. tnsnames.ora ファイルが見つからない場合、データベース別名 <SID>_<hostnames> がデータベース・サービスに割り当てられます。サービスはこの別名で Agent に認識され、Oracle Enterprise Manager コンソールにその名前で表示されます。

# 8.0.6/8.1.6/8.1.7 Intelligent Agent の 9.0 へのアップグレード

各リリースの Intelligent Agent により、Agent のパフォーマンス、機能性および信頼性が向上します。そのため、ご使用のプラットフォームで使用できる最新バージョンに Intelligent Agent をアップグレードすることをお薦めします。より新しい Agent に移行する際、Enterprise Manager の既存のジョブ、イベントおよびデータ収集を維持するには、$ORACLE_HOME/bin にある NMUMIGR8 ユーティリティを使用します。このユーティリティでは、既存のジョブ、イベントおよびデータ収集を、リリース 9.0 の Intelligent Agent で認識されるフォーマットに移行できます。

### 使用方法

```
nmumigr8 [-source_home <source ORACLE_HOME>] [-password <password>] [-verbose]
```

### パラメータ

#### -source_home

既存の Intelligent Agent キューおよびデータ収集ファイルがあるソース Oracle ホーム。source_home を指定しない場合は、ORACLE_HOME 環境変数の値である宛先 Oracle ホームになります。

#### -password

Intelligent Agent のキュー・ファイルを暗号化する際に使用されるパスワード。指定しない場合は、デフォルトの暗号化パスワードが使用されます。

#### -verbose

Agent の `ORACLE_HOME/network/trace` ディレクトリにあるトレース・ファイル `nmumigr8.trc` に、詳細な移行情報を書き込むかどうかを指定するフラグ。このフラグを設定しない場合は、要約情報のみがトレース・ファイルに書き込まれます。

旧リリースの Intelligent Agent からリリース 9.0 の Intelligent Agent にアップグレードするユーザーは、次の項で説明するガイドラインに従ってください。

## Intelligent Agent アップグレードのガイドライン

1. 最新の Intelligent Agent を新しい Oracle ホームの下にインストールします。
2. 保存しておくジョブまたはイベントが、それぞれジョブ・ライブラリまたはイベント・ライブラリに保存されていることを確認します。ジョブ / イベントをジョブ / イベント・ライブラリに追加するには、ジョブ / イベント・ペインから該当のジョブ / イベントを選択し、マウスの右ボタンを使用して目的のエントリをクリックして、コンテキスト依存メニューから「ライブラリへのコピー」を選択します。
3. イベントの警告をイベントの履歴に移動します。履歴ペインの内容は保存または消去できます。

   **注意：** 複数のターゲットに対して登録されているイベントがある場合、「類似作成」メニュー・オプションを使用して、各ターゲットについて個々のイベントを作成し、これらのイベントをイベント・ライブラリに保存します。

4. Enterprise Manager コンソールから、既存のイベントの登録を解除し、Agent をアップグレードするノードに対してスケジュールされているアクティブ・ジョブを削除します。
5. 古い Agent をシャットダウンします。
6. 新しい Agent を起動します。
7. OEM コンソールから、ナビゲータ上のノードをリフレッシュします。
8. 新しい Agent に保存したジョブおよびイベントを再送します。

# 3

# 9*i* の新機能

この章では、次の項目について説明します。

- ブラックアウト
- 新規のコントロール・ユーティリティ

# ブラックアウト

ブラックアウトを使用すると、Enterprise Manager のユーザーは、管理対象である 1 つ以上のターゲットに対する任意またはすべての管理、またはデータ収集アクティビティ（あるいはその両方）を一時停止できます。この機能を利用して、メンテナンスや緊急時の操作を行うことが可能です。

具体的には、ブラックアウトにより次の操作が停止します。

- **イベント：** ターゲットに登録されているすべてのイベントが、ブラックアウトの期間中は評価または起動されません。
- **ジョブ：** ターゲットに発行されたすべてのジョブが、ブラックアウトの期間中はスケジュールまたは実行されません。ブラックアウトの期間中にターゲットに対してスケジュールされる通常間隔のジョブについては、ジョブがスキップされたことを示す通知が Enterprise Manager コンソールに送られます。
- **データ収集：** ターゲットに関する現行の履歴データ収集アクティビティは、すべて停止します。ただし、ブラックアウトの**前に**ターゲットに関して収集されたデータのロードは、データベースが稼働中であるかぎり続行します。新規の収集を発行することはできますが、ブラックアウトが終了するまで処理は続行されません。

## ブラックアウトの定義

ブラックアウトは、ターゲットレベルで作成する、つまり Intelligent Agent が常駐しているノード上で定義する必要があります。ブラックアウトは、コマンドライン・インタフェースで制御できます。ブラックアウトのサブシステムによって、コマンドラインでの要求はすべて、CLI ユーザーと呼ばれる特別なタイプの Agent ユーザーに関連付けられます。即時ブラックアウトを設定できるのは、一度に 1 つのみです。ターゲット・ブラックアウトは、同時に複数存在できます。

一度有効になったブラックアウトは変更できません。特定のブラックアウトのステータスを変更する場合は、既存のブラックアウトを削除してから、変更した新規のブラックアウトを再作成してください。

---

**重要：** ブラックアウトの取消しは、それを設定したユーザーによってのみ可能です。

---

## ブラックアウトのコマンドライン・インタフェース

Agent が常駐するノード上には、ブラックアウトのコマンドライン・ツールがあり、管理者はこれを使用してブラックアウトの設定および取消しを行うことができます。ブラックアウトを設定するには、Intelligent Agent が稼働している必要があります。ユーザー・コマンドは、次のとおりです。

**表 3-1**

| ブラックアウトの操作 | コマンド |
|---|---|
| ブラックアウトを定義 | `agentctl start blackout [-d [DD] HH:MM] [<target name>]`<br>-d オプションでは、DD HH:MM の形式で期間を指定します。<br>*DD* は日数を表します（オプション）。<br>*HH* は時間を表します。<br>*MM* は分数を表します。 |
| ブラックアウトを削除 | `agentctl stop blackout [<target name>]` |
| ブラックアウトのステータスを表示（マシン名、ターゲット・タイプおよび期間） | `agentctl show blackout` |

## コマンドラインの例

次の例は、状況に応じたブラックアウト・コマンドライン・ユーティリティの使用方法と、その結果を示したものです。

**状況例 1:** ターゲット vnukal-pc.world 上で、10 分間のブラックアウトを開始する場合。

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

**状況例 2:** （CLI ユーザーにより）ターゲット vnukal-pc.world 上に設定されたブラックアウトを停止する場合。注意： 他の Agent ユーザーにより設定されたブラックアウトは、取り消せません。

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

**状況例 3:** 管理対象のすべてのターゲット上で、無期限のブラックアウトを開始する場合。これは、Agent 全体のブラックアウトと同じです。

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

**状況例 4:** 登録されているすべてのターゲット上で、ブラックアウトを停止する場合。

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

**状況例 5:** あるターゲット上でのブラックアウトのステータスを確認する場合。

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

**状況例 6:** タイプの異なる別のターゲットと名前が一致するターゲット上で、ブラックアウトを設定する場合。この場合、コマンドライン・インタフェースでは、対話形式で任意のターゲットを選択できます。

```
$ 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
```

# 新規のコントロール・ユーティリティ

9*i* の Intelligent Agent では、独自の制御ユーティリティが用意されています。

| 操作 | 入力するコマンド |
|---|---|
| Agent および *dbsnmpwd* を起動 | `agentctl start agent` |
| Agent および *dbsnmpwd* を停止 | `agentctl stop agent` |
| Agent のステータスを検証 | `agentctl status agent` |

# 4

# ジョブ・スクリプトとイベント・スクリプト

この章では、次の項目について説明します。

- スクリプト言語
- サーバー・メッセージとエラー情報
- 修正ジョブの Tcl 配列に対するイベント
- Intelligent Agent での Tcl の使用
- NLS の問題とエラー・メッセージ
- OraTcl の関数とパラメータ

# スクリプト言語

ジョブ・スクリプトおよびイベント・スクリプトの記述には、OraTcl 拡張機能付き Tcl 言語を使用します。Tcl は次の要件を満たすことから、スクリプト用の言語として使用されます。

- ファイルおよびデバイスの処理、プログラムの起動、およびオペレーティング・システム関数の実行のためのホスト・システム・アクセス
- RDBMS アクセスのための SQL および PL/SQL 関数
- RDBMS 管理用関数
- Agent 自体がサポートするデータベース MIB 変数、および、ホストのサービスまたは SNMP 上で実現されるその他のサービスなどの外部 MIB の両方への SNMP アクセス
- Oracle Intelligent Agent、および Oracle Trace などその他の Oracle ソフトウェアとの通信
- 次のことを実現するジョブ・スクリプトおよびイベント・スクリプトを記述するための構文
  - ユーザー・インタフェースの駆動
  - ジョブまたはイベントの性質、および任意の入出力に関する情報の提供
  - NLS サポートのための Oracle メッセージ・ファイル・システムへのアクセス

## Tcl 言語の説明

Tcl は、カリフォルニア州立大学バークレー校の John Ousterhout 博士によって開発されました。Tcl は Tool Command Language の略称で、現行リリースのバージョンは 7.5 です。

Tcl は言語とライブラリの両方を指します。Tcl はシンプルなテキスト言語で、テキスト・エディタ、デバッガ、イラスト作成ツール、シェルなどの対話形式プログラムにコマンドを発行することがその主な用途です。Tcl はシンプルな構文を持ち、言語の拡張が容易です。Tcl の利用者は、独自のコマンド・プロシージャを記述して、ビルトイン・セットに備わっているものよりもさらに強力なコマンドを組み込むことができます。

Tcl は、アプリケーション・プログラムに埋込み可能なライブラリ・パッケージでもあります。Tcl ライブラリは、Tcl 言語のパーサー、Tcl のビルトイン関数を実装するルーチン、および各アプリケーションに固有の追加コマンドで Tcl を拡張するためのプロシージャから構成されます。アプリケーション・プログラムは Tcl コマンドを生成し、実行のためにそれらを Tcl パーサーに渡します。コマンドは、入力ソースから文字を読み取ることによって生成できます。また、アプリケーションのメニュー・エントリ、ボタン、キーストロークなどのユーザー・インタフェースに、コマンド文字列を関連付けることによっても生成できます。Tcl ライブラリはコマンドを受け取ると、コマンドをコンポーネント・フィールドに解析し、ビルトイン・コマンドを直接実行します。アプリケーションによって独自に実装されるコマンドについては、Tcl はアプリケーションにコール・バックしてコマンドを実行します。多くの場合、コマンドに実行のための追加の文字列を渡すことによって、Tcl インタプリタが

再帰的に起動されます。プロシージャ、ループ・コマンドおよび条件コマンドは、すべてこのようなしくみで動作します。

アプリケーション・プログラムのコマンド言語に Tcl を使用することには、様々な利点があります。

- Tcl は標準的な構文を提供します。Tcl をいったん学習すれば、Tcl ベースのアプリケーションに容易にコマンドを送ることができます。
- Tcl は言語拡張性を備えています。すべての Tcl アプリケーションに必要なのは、アプリケーションに固有の低レベル・コマンドをいくつか実装することです。Tcl は多くのユーティリティ・コマンドに加えて、複雑なコマンド・プロシージャを構築するための一般的なプログラミング・インタフェースを提供します。Tcl を使用すれば、これらの機能をアプリケーションの側で繰り返し実装する必要はありません。
- Tcl の拡張機能は、Tcl コマンドの送受信によってアプリケーション間で通信を行うための機構を提供します。Tcl 言語の共通的なフレームワークは、アプリケーション間の通信を容易にします。

Tcl はもともと、大規模なソフトウェア・システムの設計には複数の言語を使用する必要がある、という思想のもとに設計されたものです。これは、複雑な内部データ構造を操作する、パフォーマンスを重要視するなどの各用途に応じて最適な言語を使い分けることを意味します。Tcl については、C プログラムの断片をつなぎ合わせたり、拡張のためのフックを提供したりするための短いスクリプトを記述する、などの用途があります。Tcl スクリプトの場合、パフォーマンスや、複雑なデータ構造およびアルゴリズムを扱うための機能性よりも、学習、プログラミングおよび統合の簡便さが重視されます。Tcl は、より低いレベルでの実現に適したタスクに遭遇した際に、それをより低レベルの言語に移植する容易さを考慮して設計されています。このように、基本的なコア機能は小さなまま保つことができ、特定の目的または要求に必要な断片を持ち寄って組み合せる作業に集中することができます。Tcl/Tk の詳細は、次の Web サイトを参照してください。

- http://www.neosoft.com/tcl
- http://www.scriptics.com/resource/doc/papers/

**注意：** World Wide Web サイトの場所は頻繁に変化するため、これらのアドレスには将来アクセスできなくなる可能性があります。

## OraTcl の説明

Agent のジョブとイベント・スクリプトはともに、ファイルとデバイスの処理、プログラムの起動、オペレーティング・システム関数の実行、および Oracle データベースへのアクセスのためのホスト・システム・アクセスを必要とします。OraTcl は、Oracle の用途および SNMP アクセスのために Tcl を拡張する目的で開発されました。OraTcl の機能のカテゴリは、次のように分かれています。

- SQL および PL/SQL 関数
- RDBMS 管理用関数
- SNMP アクセス
- Intelligent Agent およびその他の Oracle ソフトウェアとの通信
- キャラクタ・セット変換およびエラー処理用の関数
- 汎用ユーティリティ関数

OraTcl の関数および変数の説明は、4-12 ページの「OraTcl の関数とパラメータ」を参照してください。

### 例： OraTcl スクリプト

次の例は、OraTcl の基本的な使用方法を示しています。

```
#
# 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
```

# サーバー・メッセージとエラー情報

OraTcl は、Tcl グローバル配列 oramsg を作成、維持して、Oracle Server メッセージのフィードバックを提供します。oramsg は、OraTcl インタフェース・ルーチンと通信して、NULL の戻り値と LONG の制限を指定するためにも使用されます。NULLVALUE と MAXLONG を除くすべての場合、任意の OraTcl コマンドの起動時に各要素が NULL にリセットされ、コマンドの影響を受ける任意の要素が設定されます。oramsg 配列は、オープンされているすべての OraTcl ハンドルの間で共有されます。

**注意：** oramsg は、それを必要とする任意の Tcl プロシージャの中で、グローバル文を使用して定義してください。

## oramsg の要素

oramsg の要素を次に示します。

**oramsg (agent_characterset)**
Agent のキャラクタ・セットで、US7ASCII のような値をとります。convertin および convertout 関数によるキャラクタ・セットの変換に使用されます。4-14 ページの「convertin」および 4-14 ページの「convertout」を参照してください。

**oramsg (db_characterset)**
データベースのキャラクタ・セットで、US7ASCII などの値をとります。convertin および convertout 関数によるキャラクタ・セットの変換に使用されます。4-14 ページの「convertin」および 4-14 ページの「convertout」を参照してください。この変数は、Tcl スクリプト内では oralogon 関数の後でのみ使用できます。

**oramsg (collengths)**
oracols によって返される列の長さの Tcl リストです。collengths は、oracols によってのみ設定されます。

**oramsg (colprecs)**
oracols によって返される数値列の精度の Tcl リストです。colprecs は、oracols によってのみ設定されます。数値以外の列については、リスト・エントリは NULL 文字列になります。

**oramsg (colscales)**
oracols によって返される数値列のスケールの Tcl リストです。Colscales は、oracols によってのみ設定されます。数値以外の列については、リスト・エントリは NULL 文字列になります。

**oramsg (coltypes)**
oracols によって返される列の型の Tcl リストです。coltypes は、oracols によってのみ設定されます。返される型には、CHAR、VARCHAR2（バージョン 7）、NUMBER、LONG、rowid、DATE、RAW、LONG_RAW、MLSLABEL、RAW_MLSLABEL または不明の型などがあります。

**oramsg (errortxt)**
rc に関連付けられるメッセージ・テキストです。oraplexec 関数によって複数の SQL 文が実行されることがあるため、サーバーから複数のメッセージが受け取られる可能性があります。

**oramsg (handle)**
最後に実行された OraTcl 関数のハンドルを指します。ハンドルは、コマンドを追跡するために使用されるメモリー内でのマッピングのことで、無効なハンドルが使用されている場合を除き、OraTcl コマンドの実行のたびに設定されます。

**oramsg (jobid)**
現行のジョブのジョブ ID です。ジョブ・スクリプトに対してのみ定義されます。

**oramsg (language)**
コンソールの NLS 言語で、AMERICAN_AMERICA.US7ASCII のように設定します。

**oramsg (maxlong)**
プログラマが設定できる要素で、orafetch によって返される LONG または LONG RAW データの量を制限します。デフォルトは 32KB です。上限値は 64KB（バージョン 6）または 2147483647 バイト（バージョン 7）です。ゼロ以下の値はすべて無視されます。変更された maxlong の値は、次に orasql をコールしたときに有効になります。orafetch の、MAXLONG の使用に関する注記を参照してください。

**oramsg (nullvalue)**
プログラマが設定できる要素で、NULL の結果に対して返される文字列値を指定します。oramsg (nullvalue) を DEFAULT に設定すると、INTEGER、FLOAT および MONEY のような数値 NULL データ型に対しては 0 が返され、その他のすべてのデータ型に対しては NULL 文字列が返されます。NULLVALUE は、初期値では default に設定されています。

**oramsg (oraobject)**
スクリプトの動作対象オブジェクトを示します。イベント・スクリプトに対してのみ定義されます。

**oramsg (orahome)**
ORACLE_HOME ディレクトリです。

**oramsg (oraindex)**
構成ファイル snmp.ora から読み取られた SNMP 索引値の Tcl リストです。

**oramsg (orainput)**
ジョブの入力ファイルの名前を示す Tcl リストです。ほとんどのジョブは入力ファイルを必要としませんが、SQL スクリプトで SQL*Plus を起動するジョブ、または仕様ファイルで Export を起動するジョブは、入力ファイル機能を使用します。ジョブ・スクリプトに対してのみ定義されます。

**oramsg (rc)**
最後の SQL コマンドと、後続の orafetch 関数による処理の結果を示します。rc は orasql、orafetch、oraplexec によって設定され、OraTcl コマンドによって最後にコールされた OCI ライブラリ関数からの数値リターン・コードが入ります。

詳細は、『Oracle9*i* データベース・エラー・メッセージ』を参照してください。一般的な値のリストを表 4-1「エラー・メッセージ」に示します。

**表 4-1 エラー・メッセージ**

| エラー | 意味 |
|---|---|
| 0000 | 関数は正常に終了。エラーはありません。 |
| 0900 ～ 0999 | 無効な SQL 文、キーワード不足、無効な列名など。 |
| 1000 ～ 1099 | プログラム・インタフェース・エラー。SQL 文の欠如、ログイン拒否、必要な権限不足など。 |
| 1400 ～ 1499 | 実行エラーまたはフィードバック。 |
| 1403 | `orafetch` コマンドの実行時にデータの終端に到達。 |
| 1406 | `orafetch` によってフェッチされる列が切り捨てられた。LONG または LONG RAW データのフェッチ時に、maxlong の値が実際のデータ・サイズよりも小さい場合に発生する可能性があります。 |

**oramsg (rows)**
`orasql` コマンドでの挿入、更新または削除によって変化する行の数、または `orafetch` によってフェッチされる累計の行数です。

**oramsg (starttime)**
ジョブが開始するようにスケジューリングされた時刻です。ジョブに対してのみ定義されます。

# 修正ジョブの Tcl 配列に対するイベント

OraTcl は、Tcl グローバル配列 `trigevent` を作成、維持して、Tcl 配列を Enterprise Manager の修正ジョブに渡します。`trigevent` 配列は、修正ジョブ自体からも使用できます。

## trigevent の要素

trigevent の要素を次に示します。

**trigevent (name)**
修正ジョブを起動するイベントの名前です。

**trigevent (object)**
ネットワークのターゲットまたはノードです。

**trigevent (arguments)**
トリガー・イベントへの引数です。イベントにより異なります。

**trigevent (results)**
イベントの結果です。たとえば、この要素は CPU 使用率イベント・テストの使用率が 35% であることを示す数値 35 を返します。

**trigevent (severity)**
イベントの重大度です。-1（安全）、1（警戒）、2（警告）があります。

---

**警告：** trigevent は、修正ジョブでのみ機能します。この配列が修正ジョブ以外に渡されると、trigevent は未定義（NULL）になりジョブは失敗します。このため、trigevent 配列は、その要素の参照が解除される前に必ずチェックする必要があります。

---

### 例

次の例は、2 つの異なるタスクを実装する修正ジョブを示しています。

- trigevent を使用して修正ジョブに配列を渡す Tcl スクリプト

```
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"
}
```

- オペレーティング・システム・コマンドの実行

```
top -d1 -ocpu
```

この例の修正ジョブは、特定のレベルの CPU の使用率を監視する CPUUTIL イベント・テストと対応付けられています。このイベント・テストの場合、次のようなパラメータが設定されます。

- **警告のしきい値** = 20
- **警戒のしきい値** = 10
- **修正ジョブ** = プロパティ・シートで選択したオプション

CPUUTIL イベント・テストを含むイベントが引き起こされると、対応する修正ジョブが実行されます。修正ジョブでは、次のような出力が生成されます。

実行された最初のタスク： trigevent 配列の情報が表示されます。

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

実行された 2 番目のタスク： オペレーティング・システム・コマンド `top -d1 -ocpu` が実行されます。

```
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
```

# Intelligent Agent での Tcl の使用

Tcl スクリプトは、Intelligent Agent でジョブまたはイベントを扱うために使用されます。同じ Tcl スクリプトでも、Agent とユーザー・インタフェースではその扱いが異なります。

ジョブとは、1 回または複数回実行されるようにスケジューリングされるスクリプトのことです。ジョブは一般に、データベースを起動する、バックアップを実行する、または puts コマンドを通して画面に出力を送るなどの副次作用を伴い、実行には長い時間を要することがあります。イベント・スクリプトと異なり、ジョブには SQL スクリプトなどの入出力ファイルを与えることができます。UNIX、DOS または OS/2 上では、出力ファイルが `stdout` でリダイレクトされることに注意してください。

一方イベント・スクリプトは、例外の検出にその目的を限って使用されます。Tcl イベント・スクリプトは様々な手段を利用して、データベース、ホスト・システムまたは SQL*Net サービスを監視できます。特定の条件が満たされたと判断すると、スクリプトはイベントの重大度を示すリターン・コードを Agent に送ります。イベント・スクリプトはジョブよりも頻繁に実行される傾向があり、実行時間は比較的短いことが想定されます。また、イベント・スクリプトには副次作用が伴わないことも想定されます。

ジョブとイベントはともに Tcl を使用してタスクを遂行する一方で、実行環境が異なるなど、両者の性質は大きく異なっています。特に UNIX システム上では、イベントは通常 Agent コードとともにインラインで実行される一方で、ジョブは独立したプロセスへと分岐します。

インラインのイベント・スクリプトにかぎり、Tcl インタプリタの状態が実行の間に保存され、Tcl グローバル変数の値が保持されます。これらは仮想プロセスの存在を装うための動作です。これにより、イベント・スクリプトは履歴を維持して、イベントが何度も呼び出されるのを防ぐことができます。たとえば、ある値が 90 を超えたことをコンソールに通知した後で、次に値が 80 を下回り再び 90 を超えるまでの間、再度の通知を控えることができます。`oralogon` 関数を使用しているデータベース接続は、インラインの全イベント・スクリプトにまたがってキャッシュされます。これにより、同じ接続文字列を使用して繰り返し実行されるイベント・スクリプトが、同じ接続を利用できます。

ジョブとイベントの両方に対して、すべてのコマンドとグローバル変数が使用可能なわけではありません。スクリプトの動作対象となっているサービスの種類をイベントに知らせる `oraobject` グローバル変数は、ジョブには存在しません。また、ジョブが SQL*Plus スクリプトに対して使用する `orainput` グローバル変数はイベントにはありません。

# NLS の問題とエラー・メッセージ

ユーザーがイベントを登録する、またはジョブをスケジューリングするとき、Agent に対してユーザーの言語環境設定を使用できます。各コンソール・ユーザーの言語および現在のアドレスを報告するための、特殊なリモート・プロシージャ・コールが用意されています。oralogon 関数がコールされるたびに、Agent は指定された言語への ALTER SESSION コマンドを前もって発行します。これは、それ以降に Oracle Server から送られてくるメッセージまたは出力が、ユーザーの言語になることを意味します。加えて、キャラクタ・セットの変換は Agent 上で明示的に行われず、コンソールがユーザー側でその変換を行うことができます。

イベント・スクリプトまたはジョブ・スクリプトの実行が失敗すると、エラー・メッセージがユーザーの言語でコンソールに返されます。通常これは、関数が不適切なパラメータを与えられた場合に、Oracle Tcl 拡張機能のいずれかによって返される Oracle メッセージです。たとえば、不正な接続文字列を指定すると、oralogon では「ERROR: ORA-01017: ユーザー名 / パスワードが無効です。ログインは拒否されました。」というエラーが返されます。ただし、エラー・メッセージは「ERROR: Tcl-00456: division by zero error」という Tcl 固有のメッセージになることもあります。これはメッセージ・ファイルに格納されるため、ユーザーの作業環境の言語で返されることはありません。ユーザーの言語作業環境が指定されていないか、ユーザーの言語のエラー・メッセージ・テキストが存在しない場合、Agent によって使用されるデフォルトの言語はアメリカ英語です。

# OraTcl の関数とパラメータ

この項では、OraTcl の関数とパラメータの一覧を示します。OraTcl 構文中の関数またはその他の語は、function のようなフォントで示されます。角カッコ内のパラメータ [option] はオプションで、文字 '|' は「または」の意味です。すべてのパラメータは関数に渡され、IN モードです。

- SQL および PL/SQL 関数

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

- RDBMS 管理用関数

| | |
|---|---|
| *orastart* | *orastop* |

- SNMP アクセス用関数

| | |
|---|---|
| *oradbsnmp* | *orasnmp* |

- Intelligent Agent およびその他の Oracle ソフトウェアとの通信用関数

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

- キャラクタ・セット変換およびエラー処理用関数

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

- 汎用ユーティリティ関数

  *orasleep* *oratime*

## 共通パラメータ

次のパラメータは複数の OraTcl 関数で使用されるもので、この項に説明があります。

**column**
LONG または LONG RAW 列である列名です。

**connect_string**
有効な Oracle データベース接続文字列で、次のいずれかの形式になります。

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

**destaddress**
`destaddress` は、Agent の宛先アドレスです。

**filename**
列に書き込まれる LONG または LONG RAW データを含むファイルの名前、または LONG または LONG RAW データが書き込まれるファイルの名前です。

**logon-handle**
以前に `oraopen` でオープンされた、有効なカーソル・ハンドルです。ハンドルとは、関数の追跡に使用されるメモリー内でのマッピングのことです。

**rowid**
既存の行の Oracle データベース `rowid` で、Oracle の `rowid` データ型の形式である必要があります。

**table**
行および列を含む Oracle データベースの表の名前です。

## convertin

**用途**　この関数は、パラメータ文字列をクライアント（コンソール）のキャラクタ・セットから接続先のキャラクタ・セットに変換します。返される値は、変換後の文字列です。

**構文**　`convertin dest_characterset string`

**パラメータ**

**dest_characterset**
接続先のキャラクタ・セットです。ジョブまたはイベントに対しては、$oramsg(agent_characterset) を使用します。4-6 ページの「oramsg の要素」を参照してください。

**string**
変換される文字列です。

**コメント**　クライアントおよび Agent のノードで、異なる言語またはキャラクタ・セットが使用される場合があります。キャラクタ・セットの変換を実行することは、Tcl スクリプト作成者の責任です。一般に、それらが ASCII であることが保証されていないかぎり、ジョブまたはイベントのすべての入力パラメータを変換してください。

## convertout

**用途**　この関数は、パラメータ文字列を接続先のキャラクタ・セットからクライアント（コンソール）のキャラクタ・セットに変換します。返される値は、変換後の文字列です。

**構文**　`convertout dest_characterset string`

**パラメータ**

**dest_characterset**
接続先のキャラクタ・セットです。ジョブまたはイベントに対しては、$oramsg(agent_characterset) を使用します。4-6 ページの「oramsg の要素」を参照してください。

**string**
変換される文字列です。

**コメント**　クライアントおよび Agent のノードで、異なる言語またはキャラクタ・セットが使用される場合があります。キャラクタ・セットの変換を実行することは、Tcl スクリプト作成者の責任です。一般に、それらが ASCII であることが保証されていないかぎり、ジョブまたはイベントのすべての出力を変換してください。

## msgtxt

**用途** この関数は、特定の製品名、機能およびメッセージ番号に対応するメッセージ・テキストを、クライアント（コンソール）の言語およびキャラクタ・セットで返します。出力は FACILITY-ERROR : MESSAGE TEXT のような形式になります。

**構文** `msgtxt product facility error_no`

**パラメータ**

**product**
製品名です。出力は `rdbms` のようになります。

**facility**
機能名です。出力は `ora` のようになります。

**error_no**
エラー番号です。出力は `1101` のようになります。

**コメント** この関数は、ジョブの出力ファイルにエラー・メッセージを出力するために使用されます。メッセージはクライアント（コンソール）の言語で表示されます。

## msgtxt1

**用途** この関数は、特定の製品名、機能およびメッセージ番号に対応するメッセージをクライアント（コンソール）の言語で返します。出力は MESSAGE TEXT のような形式になります。

**構文** `msgtxt1 product facility error_no`

**パラメータ**

**product**
製品名です。出力は `rdbms` のようになります。

**facility**
機能名です。出力は `ora` のようになります。

**error_no**
エラー番号です。出力は `1101` のようになります。

**コメント** この関数は、ジョブの出力ファイルに確認メッセージを出力するために使用されます。メッセージはクライアント（コンソール）の言語で表示されます。

## oraautocom

**用途**　この関数は、logon-handle が指示する接続を通してオープンされたハンドルを使用する、SQL データ操作文の自動コミットを有効または無効にします。

**構文**　oraautocom logon-handle {on | off}

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　指定された logon-handle がオープンされていない場合、oraautocom は Tcl エラーを返します。

on または off のどちらかを指定する必要があります。自動コミット機能は、デフォルトではオフになっています。

## oracancel

**用途**　この関数は、logon-handle が指示する接続を通してオープンされたカーソルを使用する、先行の orasql 関数によって返されたすべての保留中の結果を取り消します。

**構文**　oracancel logon-handle

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　指定された logon-handle がオープンされていない場合、oracancel は Tcl エラーを返します。

## oraclose

**用途**　この関数は、logon-handle に関連付けられたカーソルをクローズします。

**構文**　oraclose logon-handle

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　指定された logon-handle がオープンされていない場合、oraclose は Tcl エラーを返します。

## oracols

**用途**　この関数は、最後に実行された orasql、orafetch または oraplexec 関数から、列の名前を Tcl リストで返します。oracols は、oraplexec の後に使用できますが、この場合はバインドされた変数名が返されます。

**構文**　oracols logon-handle

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　指定された logon-handle がオープンされていない場合、oracols は Tcl エラーを返します。

oramsg の配列索引 collengths には、列の長さに対応する Tcl リストが設定されます。索引 coltypes には、列の型に対応する Tcl リストが設定されます。索引 colprecs には、数値列の精度に対応する Tcl リストが設定され、対応するその他の非数値列は NULL 文字列になります（バージョン 7 のみ）。索引 colscales には、数値列のスケールに対応する Tcl リストが設定され、対応するその他の非数値列は NULL 文字列になります（バージョン 7 のみ）。

## oracommit

**用途** この関数は、logon-handle が指示する接続を通してオープンされたカーソルを使用する、先行の orasql 関数によって返されたすべての保留中のトランザクションをコミットします。

**構文** oracommit logon-handle

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**コメント** 指定されたログイン・ハンドルがオープンされていない場合、oracommit は Tcl エラーを返します。

## oradbsnmp

**用途** この関数は、SNMP MIB 値を取り出します。

**構文** oradbsnmp get | getnext object_Id

**パラメータ**

**object_Id**
object_Id の取り得る値は、1.3.6.1.2.1.1.1.0 のような実際の MIB オブジェクト ID か、sysDescr または sysDescr.0 のように、それに付加できる索引を伴ったオブジェクト名のどちらかです。

**コメント** oradbsnmp は、RDBMS パブリック MIB、または Oracle RDBMS プライベート MIB など、Agent によってメンテナンスされる SNMP MIB 値を取り出すための関数です。この関数は、通常の SNMP 用 UDP ポートへの書込みを行わず、SNMP MIB 値を Agent の内部データ構造から直接取得します。この関数は、ホスト上で SNMP Master Agent が動作していない場合に有用です。get および getnext の機能の詳細は、4-28 ページの「orasnmp」の項を参照してください。SQL コマンドで V$ 表から値をフェッチするかわりに、oradbsnmp の使用を薦める理由には、次のようなものがあります。

- Agent は、V$ 表からフェッチされる MIB 値のキャッシュをメンテナンスして、RDBMS への過度の負荷を回避します。oradbsnmp は SQL よりも高速な場合が多いため、システムへのオーバーヘッドが軽減されます。
- SGA アクセスが実装されているとき、SGA から直接フェッチされるこれらの MIB 変数について、SGA アクセスはこの関数に対して透過です。

- getnext の場合、次の object_id は、プライベートおよびパブリックの RDBMS MIB 内部の次の object_id であって、別の MIB のそれではありません。この関数を使用してシステム固有の情報を取り出すことはできません。かわりに orasnmp を使用してください。

## orafail

**用途** この関数は、Tcl スクリプトを強制的に失敗させます。

**構文** `orafail errormsg`

**パラメータ**

**errormsg**
errormsg の取り得る値は、引用符で囲まれたテキスト文字列、または FAC-XXXXX のような形式の文字列のどちらかです。XXXXX の部分には、VOC-99999 のように、特定の機能に対する Oracle メッセージ番号が入ります。

**コメント** エラー・メッセージは、クライアント側での表示目的に使用されます。

## orafetch

**用途** この関数は、orasql で実行された最後の SQL 文からの次の行を Tcl リストで返します。

**構文** `orafetch logon-handle [commands]`

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**commands**
オプションのコマンドを使用することにより、orafetch で行を繰り返しフェッチして、各行に対してコマンドを実行できます。

**コメント** 指定された logon-handle がオープンされていない場合、orafetch は Tcl エラーを返します。

返されるすべての列はキャラクタ文字列に変換されます。現行の結果セット内にそれ以上行がない場合、NULL 文字列が返されます。orafetch によって返される Tcl リストには、選択された列の値が、SELECT 文によって指定された順序で格納されます。

それぞれの行に対して Tcl リストを Tcl_Eval() に渡す前に、コマンド上で置換が行われます。orafetch では、コマンド中の文字列 @n を使用して結果の列を指定します。たとえば、@1、@2、@3 はそれぞれ結果中の第 1 列、第 2 列、第 3 列を指します。@0 は、結果行全体を Tcl リストで指します。置換列は任意の順番で、あるいは同じコマンド中に複数回指定できます。置換される列は、適切なリスト要素としてコマンド文字列に挿入されます。たとえば、置換の前後に 1 文字分の空白が追加され、埋め込まれた空白を伴う列値は必要に応じて中カッコ（{}）で囲まれます。

列の置換番号が結果中の列数を超える場合、Tcl エラーが返されます。コマンドがブレークを実行すると、orafetch の実行は中断され、Tcl_OK が返されます。残りの行は、それ以降の orafetch 関数でフェッチできます。コマンドがリターンまたはコンティニューを実行すると、残りのコマンドはスキップされ、orafetch の実行は次の行から継続します。コマンドがエラーを返すと、orafetch は Tcl エラーを返します。コマンドは二重引用符（""）または中カッコ（{}）で囲んでください。

OraTcl は、すべてのデータ型に対する変換を実行します。ロー・データは、先頭に 0x の付かない 16 進文字列で返されます。特定の変換を強制するには、SQL 関数を使用します。

oramsg の配列索引 rc は、フェッチのリターン・コードに設定されます。0 は行のフェッチに成功したことを示し、1403 はデータの終端に到達したことを示します。行の索引は、その時点までにフェッチされた累計の行数に設定されます。

oramsg の配列索引 maxlong は、返される各列に対して返される LONG または LONG RAW データの量を制限します。デフォルトは 32768 バイトです。oramsg の配列索引 nullvalue を設定して、列が NULL のときに返される値を指定できます。デフォルトは数値データに対しては "0"、その他のデータ型に対しては "" です。

destaddress は、orainfo 関数を使用して取得できます。提供されるアドレスは Agent の子アドレス、つまり Agent がファイル転送要求をリスニングする特別のアドレスであり、他のすべての RPC に対して使用される通常のアドレスではありません。

**追加情報：** Intelligent Agent のアドレスの詳細は、Oracle Server のインストレーション・ガイドの Agent の構成に関する章を参照してください。

## orainfo

**用途**　この関数はジョブによって使用され、構成情報を取得します。

**構文**　`orainfo destaddress`

**パラメータ**

**destaddress**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　`orainfo` は、`destaddress` の Agent から、Agent の構成情報をフェッチします。`destaddress` が存在しない場合、ローカル・マシン上の Agent から構成情報がフェッチされます。Agent の構成は、次の要素からなる Tcl リストです。

- この Agent によって監視されるデータベースのリスト。このリストには各データベースのデータベース名、`ORACLE_HOME` および SID が含まれます。
- Agent の通常の RPC アドレス。tnsnames（TNS）文字列で表されます。
- Agent のファイル転送アドレス。TNS 文字列で表されます。

## orajobstat

**用途**　この関数はジョブによって使用され、即時性の出力をコンソールに返します。

**構文**　`orajobstat destaddress string`

**パラメータ**

**destaddress**
4-13 ページの「共通パラメータ」を参照してください。

**string**
`string` の取り得る値は、引用符で囲まれたテキスト文字列、または FAC-XXXXX のような形式の文字列のどちらかです。XXXXX の部分には、VOC-99999 のような、特定の機能に対する Oracle のメッセージ番号が入ります。文字列はクライアント側での表示目的に使用されます。

**コメント**　`destaddress` は、デーモンではなく Agent のアドレスです。この関数は、Agent プロセスの内部からではなくジョブ・プロセスから発行されます。Agent のアドレスは `orainfo` を使用して取得できます。

## oralogoff

**用途** `logon-handle` に関連付けられた Oracle Server 接続からログオフします。

**構文** `oralogoff logon-handle`

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**コメント** 指定されたログイン・ハンドルがオープンされていない場合、`oralogoff` は Tcl エラーを返します。`oralogoff` は NULL 文字列を返します。

## oralogon

**用途** `connect_string` を使用して Oracle Server に接続します。

**構文** `oralogon connect_string`

**パラメータ**

**connect_string**
4-13 ページの「共通パラメータ」を参照してください。

**コメント** この関数で確立された接続を使用し、`logon-handle` を必要とする他のすべての OraTcl 関数に対して、返された `logon-handle` を使用してください。同一または異なるサーバーへの複数の接続が可能です。

**追加情報：** `oralogon` がイベント・スクリプト内で使用されるとき、接続キャッシュが役に立ちます。oralogon は通常、同じデータベースに対して他のイベント・スクリプトがオープンした接続を再使用できます。詳細は、4-12 ページの「NLS の問題とエラー・メッセージ」を参照してください。無効なログインまたはネットワークの使用不可など、なんらかの理由で接続が確立されない場合、`oralogon` は Tcl エラーを返します。connect_string でデータベースが指定されていない場合、サーバーとして環境変数 `ORACLE_SID` の値が使用されます。

## oralogon_unreached

**用途**　この関数により、接続文字列およびオプションのロールを使用して、Oracle サーバーに接続します。この接続は共有できません。

**構文**　`oralogon connect_string [AS] [SYSDBA|SYSOPER|NORMAL]`

**パラメータ**

**connect_string**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　この動詞は、返される接続が共有不可である点を除いて oralogon と同じです。また、オプションでロールを指定できます。

## oraopen

**用途**　この関数は、サーバーへの SQL カーソルをオープンします。`oraopen` は、`logon-handle` を必要とするそれ以降の OraTcl 関数で使用するためのカーソルを返します。

**構文**　`oraopen logon-handle`

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　指定された `logon-handle` がオープンされていない場合、`oraopen` は Tcl エラーを返します。合計で 25 のカーソルを上限として、同じまたは異なるログイン・ハンドルを通して複数のカーソルをオープンできます。

## oraplexec

**用途**　この関数は、匿名 PL ブロックを実行します。オプションで、PL/SQL 変数に値をバインドします。

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

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**pl_block**
`pl_block` の取り得る値は、完全な PL/SQL プロシージャ、または匿名 PL/SQL ブロックとしてコーディングされたストアド・プロシージャへのコールのどちらかです。

**:varname value**
`:varname value` はオプションのペアです。

**コメント**　指定された `logon-handle` がオープンされていない場合、または PL/SQL ブロックがエラー状態にある場合、`oraplexec` は Tcl エラーを返します。PL/SQL ブロックの終了時に、`oraplexec` は各 :varname の内容を Tcl リストとして返します。`oraplexec` は、その戻り値として結果セットを Tcl リストで返します。

`pl_block` の後に、オプションの :varname value ペアが続くことがあります。変数名の先頭にはコロン記号を付け、プロシージャ内で使用される置換名に一致させる必要があります。値と一致しないすべての `:varname` は無視されます。`:varname` が出力に対して使用される場合、値を NULL 文字列 "" としてコーディングしてください。

`oramsg` の配列索引 rc には、ストアド・プロシージャからのリターン・コードが含まれます。

## orareadlong

**用途**　この関数は、LONG または LONG RAW 列の内容を読み取り、結果をファイルに書き込みます。

**構文**　`orareadlong logon-handle rowid table column filename`

**パラメータ**

**logon-handle rowid table column filename**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　`orareadlong` が正常に終了すると、LONG 列から読み取られたバイト数が 10 進値で返されます。

指定された `logon-handle` がオープンされていない、rowid、table または column が無効である、あるいは行が存在しないなどの場合には、`orareadlong` は Tcl エラーを返します。

`orareadlong` は、table、column および rowid の値を基に SQL の SELECT 文を構成し、実行します。SELECT rowid FROM table WHERE ... のような `orasql` を前もって実行することにより、適切な形式の ROWID を取得できます。

## orareportevent

**用途**　この関数はジョブによって使用され、不要なイベントを Agent、またはコンソール内のイベント管理システムに報告します。`oemevent` の実行可能ファイルを使用することもできます。

**構文**　`orareportevent eventname object severity message [results]`

**パラメータ**

**eventname**
`eventname` はイベントの名前です。これは、次のような形式の 4 つの部分からなるイベント名です。

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

任意のキャラクタ文字列を入力できますが、4 つの部分すべてとスラッシュが必須です。

名前の最初の 2 つの部分は特に重要で、Oracle スクリプトの作成者が使用する必要がある多くの定義済文字列が用意されています。

- レベル 1 はスクリプトの定義者で、一般には `oracle` などのインテグレータ企業名、指定のない顧客に対しては `user` のような名前です。

- レベル 2 はスクリプトを関連付ける製品の名前で、例としては rdbms、office、agent、osgeneric、sqlnet または hpux などがあります。すべての Oracle サービスには、Oracle スクリプトの記述者が使用する必要がある定義済の名前が用意されています。

eventname は 7 ビット ASCII 形式であることが想定されます。プラットフォームまたは言語によって変化しないことがその理由です。定義済イベント名のリストは、ORACLE_HOME¥net8¥admin ディレクトリにある eventdef.tcl（Windows NT プラットフォーム上の Oracle Enterprise Manager リリース 1.5.0 の場合）を参照してください。

**注意：** 実際のイベント・スクリプト名は、特定のプラットフォーム上で有効な一意のファイル名とするために、短縮されたり、大文字に変えられたり、その他の方法で操作されることがあります。操作形式はオペレーティング・システムごとに固有です。たとえば、/oracle/rdbms/security/SecurityError は、UNIX システム上では $oracle_home/network/agent/events/oracle/rdbms/security/securityerror.tcl として格納されることがあります。

**object**
object はイベントが監視しているオブジェクトの名前で、snmp.ora ファイル、または $oramsg(nodename) の snmp.visibleservices パラメータにリストされるデータベースまたはサービスの名前などが相当します。

**severity**
severity は、イベントの重大度のレベルを表します。orareportevent では、値は 1（警戒）、2（警告）、または -1（安全）のいずれかです。oemevent では、リテラル・テキスト文字列 alert、warning または clear が使用されます。

**message**
message は、" ファイルが見つかりません " のような、コンソール上に表示される引用符付きテキスト文字列です。

**[results]**
results は、イベントから発生する可能性のあるすべての結果を表します。これは、エラーのある表領域やセキュリティ違反を犯したユーザーなど、イベントに対する特定の結果からなる Tcl リストです。

**コメント** この関数は、必要でないイベントを任意のジョブが Agent に報告し、コンソールに返すための手段です。イベント管理システムの詳細は、『Oracle Enterprise Manager 管理者ガイド』を参照してください。

## oraroll

**用途**　この関数は、logon-handle が指示する接続を通してオープンされたカーソルを使用する、先行の orasql 関数によって返されるすべての保留中トランザクションをロール・バックします。

**構文**　oraroll logon-handle

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　指定されたログイン・ハンドルがオープンされていない場合、oraroll は Tcl エラーを返します。

## orasleep

**用途**　この関数は、要求された秒数のみ Tcl スクリプトを一時停止します。

**構文**　orasleep seconds

**パラメータ**

**seconds**

**コメント**　orasleep は、要求された秒数を引数として slcsleep() をコールします。デフォルト、下限または上限の値はありません。

## orasnmp

**用途** この関数は、object_id によって指定されたオブジェクト上で、SNMP の get または getnext のどちらかの操作を実行します。

**構文** orasnmp get | getnext object_Id

**パラメータ**

**object_Id**
object_Id の取り得る値は、1.3.6.1.2.1.1.1.0 のような実際の MIB オブジェクト ID か、sysDescr または sysDescr.0 のように、それに付加される索引を伴ったオブジェクト名のどちらかです。

**コメント** オブジェクト名は MIB テキスト・ファイルに由来します。OpenView のような完全なネットワーク・マネージャは、MIB ファイルを受理し、ASN.1 を解析する MIB コンパイラを備えており、すべての MIB 内のすべてのオブジェクトのデータベースを作成します。Agent はよりシンプルである必要があります。次のような形式の、1 つまたは複数の 2 列 ASCII ファイルを含む、標準の構成ディレクトリが存在します。

```
"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",
....
```

Tcl インタプリタはこれらのファイルを読み取り、実行時にこれらのファイル上でバイナリ検索を実行して、オブジェクトを object_Id に解決します。

Oracle サービスに使用する索引値は、snmp.ora ファイルを介して構成されます。これらの索引は、グローバル変数 oraindex からも取得できます。4-5 ページの「サーバー・メッセージとエラー情報」を参照してください。

orasnmp の結果は、次のような形式の Tcl リストです。

```
{object_id   value}
```

object_Id には、value に関連付けられたオブジェクト ID が入ります。object_Id は、orasnmp get の場合はオブジェクトと同じですが、getnext の場合は次の論理 object_Id になります。orasnmp 操作は、ローカル・ホストのみに適用されると想定されています。この関数は、ローカル・ホスト上の標準の SNMP ポートに SNMP 問合せを実際に送出します。したがって、SNMP をサポートするホストまたはその他のアプリケーションの MIB 変数など、Oracle 以外の MIB 変数を問い合せることができます。この関数が機能するためには、ローカル・ホスト上で SNMP Master Agent が動作している必要があります。Oracle データベースの MIB オブジェクトを取り出すための最適化された方法については、4-18 ページの「oradbsnmp」を参照してください。Master Agent が動作していない場合、この関数は失敗します。

## orasql

**用途**　この関数は、Oracle SQL 文をサーバーに送ります。

**構文**　`orasql logon-handle sql_stmt`

**パラメータ**

**logon-handle**
4-13 ページの「共通パラメータ」を参照してください。

**sql_stmt**
`sql_stmt` は、単一の有効な SQL 文です。

**コメント**　`logon-handle` は、あらかじめ `oraopen` でオープンされた有効なハンドルである必要があります。指定された `logon-handle` がオープンされていないか、SQL 文の構文が不正な場合、`orasql` は Tcl エラーを返します。

SQL 文の実行が成功すると、`orasql` は数値リターン・コード 0 を返します。`oramsg` の配列索引 rc は、リターン・コードに設定されます。rows 索引は、挿入、更新または削除の場合に SQL 文で決定される行数に設定されます。`sql_stmt` では単一の SQL 文のみを指定できます。`orafetch` を使用すると、生成される戻り行を検索できます。最後に実行された `orasql` によって返された保留中の結果がある場合、`orasql` は暗黙のうちに `oracancel` を実行します。

`orasql` で行う表への挿入は、『Oracle SQL リファレンス』に記載の変換ルールに従ってください。

## orastart

**用途** この関数は、Oracle データベースのインスタンスを起動します。

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

**パラメータ**

**connect_string**
4-13 ページの「共通パラメータ」を参照してください。

**init_file**
`init_file` は、使用する `init.ora` ファイルへのパスです。

**コメント** `init_file` のデフォルト値は、次のとおりです。

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

[SYSDBA|SYSOPER] は、データベースを起動するユーザーのロール・フラグです。[RESTRICT] [PARALLEL] [SHARED] はデータベース・オプションです。[RESTRICT] が指定されている場合、データベースは制限付きモードで起動されます。

## orastop

**用途** この関数は、Oracle データベースのインスタンスを停止します。

**構文** `orastop connect_string [SYSDBA|SYSOPER] [IMMEDIATE|ABORT]`

**パラメータ**

**connect_string**
4-13 ページの「共通パラメータ」を参照してください。

**コメント** [SYSDBA|SYSOPER] は、データベースをシャットダウンするユーザーのロールを示すフラグです。[IMMEDIATE|ABORT] は、シャットダウン・モードを示すフラグです。

**注意：** 通常のシャットダウン操作が確実に失敗することが予測されている状況も存在します。これは、Agent がデータベースへの独自の接続を維持している一方で、これを行うときにユーザーが Agent に特殊な RPC を送り、それにより Agent がデータベースから切断されることが原因です。

## oratime

**用途**　この関数は、現在の日付と時刻を返します。

**構文**　`oratime`

**パラメータ**　なし

**コメント**　なし

## orawritelong

**用途**　この関数は、ファイルの内容を LONG または LONG RAW 列に書き込みます。

**構文**　`orawritelong logon-handle rowid table column filename`

**パラメータ**

**logon-handle rowid table column filename**
4-13 ページの「共通パラメータ」を参照してください。

**コメント**　`orawritelong` は、table、column および rowid を基に SQL の UPDATE 文を構成し、実行します。`orawritelong` が正常に終了すると、LONG 列に書き込まれたバイト数が 10 進の数値で返されます。SELECT rowid FROM table WHERE ... のような `orasql` 関数を前もって実行することにより、適切な形式の ROWID を取得できます。

指定された `logon-handle` がオープンされていない、rowid、table または column が無効である、あるいは行が存在しないなどの場合には、`orawritelong` は Tcl エラーを返します。

# A

# Agent 構成ファイル

この付録では Intelligent Agent によって生成される構成ファイル、および異なるシステム設定に対して Agent の運用を最適化するために設定できるパラメータについて説明します。次の項目について説明します。

- 構成ファイル
- ユーザーが構成可能なパラメータ
- Intelligent Agent のログ・ファイル

# 構成ファイル

Intelligent Agent の操作は、次のファイルによって制御されます。

## snmp_ro.ora

snmp_ro.ora ファイルは、$ORACLE_HOME/network/admin ディレクトリ、または TNS_ADMIN 環境変数で指定されるディレクトリにあります。この読取り専用ファイルは更新**しないで**ください。

## snmp_rw.ora

snmp_rw.ora ファイルは、Agent の $ORACLE_HOME¥network¥admin ディレクトリにあります。この読み書き両用ファイルは修正が可能ですが、修正は慎重に行ってください。

## services.ora

services.ora ファイルは Agent の起動時に作成され、Windows NT プラットフォームでは $ORACLE_HOME¥network¥agent ディレクトリに、UNIX では $ORACLE_HOME/network/agent ディレクトリに置かれます。このファイルには、Oracle データベースやリスナーなど、Agent が動作するノード上のサービスの一覧が含まれます。Agent のリリース 9.0 以上では、Agent が稼働している環境のオペレーティング・システムに関するバージョンおよびプラットフォーム情報も、このファイル含まれます。このファイルは Oracle Enterprise Manager ナビゲータの「検出」メニュー・オプションを通して Agent から検索されます。

---

**注意：** services.ora ファイルを手動で編集しないでください。このファイルは、起動時に Agent によって再度書き込まれます。

---

# ユーザーが構成可能なパラメータ

これらのパラメータは、Intelligent Agent リリースの構成ファイル snmp_rw.ora で使用されます。

**AGENTCTL.TRACE_LEVEL = OFF | USER | ADMIN | nn**
指定されたレベルで、AGENTCTL 実行可能ファイルに対するトレースを有効にします。オラクル社では、トレース・レベルを 13 に設定することをお薦めしています。レベル 16 は最大の情報を提供しますが、これはバグの調査時にのみ有用です。レベル 16 では、実際の TCP/IP パケットの内容を確認できます。レベル 15 では、パケットが渡されている様子を確認できます。このパラメータはオプションです。

**AGENTCTL.TRACE_DIRECTORY = *directory***
トレース・ファイルが書き込まれるディレクトリです。この設定は、DBSNMP.TRACE_LEVEL との組合せでのみ意味を持ちます。このパラメータを省略すると、トレース・ファイルは $ORACLE_HOME¥network¥trace に書き込まれます。このパラメータはオプションです。

**AGENTCTL.TRACE_FILE = *filename***
トレース・ファイルのファイル名です。このパラメータはオプションです。

**AGENTCTL.TRACE_TIMESTAMP = *true/false***
true に設定すると、トレース・ファイルのトレース各行の後にタイムスタンプが挿入されます。

**SNMP.INDEX.service_name.world = index_number**
Agent が監視しているサービスの、一意の索引番号です。索引番号には任意の番号を割り当てることができます。唯一の制限は、複数の索引行がある場合に索引番号が一意でなければならない点です。次に例を示します。

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

**SNMP.CONNECT.<service_name>.USER = user_name**
サブエージェントがデータベースへの接続に使用するユーザー名です。デフォルトは dbsnmp です。このパラメータはオプションです。このパラメータがデフォルト設定でない場合、catsnmp.sql スクリプトを編集して再実行してください。

サブエージェントは Intelligent Agent のことを指します。サーバー上で SNMP を構成するとき、SNMP Master Agent との対比で Intelligent Agent のことをサブエージェントと呼ぶ場合があります。ただし、Intelligent Agent を動作させる前にサーバー上で SNMP を構成する必要はありません（Netware プラットフォームを除く）。セキュリティ上の理由から、Intelligent Agent データベースのデフォルトのアカウント / パスワードである dbsnmp/dbsnmp を顧客が使用しない場合があります。ここに示した例は、Intelligent Agent のデータベース・ログイン・アカウントの変更を顧客が希望する場合にかぎり使用してください。

**SNMP.CONNECT.<service_name>.PASSWORD = password**
サブエージェントがデータベースへの接続に使用するユーザー名に対するパスワードです。デフォルトは dbsnmp です。このパラメータはオプションです。このパラメータがデフォルト設定でない場合、catsnmp.sql スクリプトを編集して再実行してください。

サブエージェントは Intelligent Agent のことを指します。サーバー上で SNMP を構成するとき、SNMP Master Agent との対比で Intelligent Agent のことをサブエージェントと呼ぶ場合があります。ただし、Intelligent Agent を動作させる前にサーバー上で SNMP を構成する必要はありません（Netware プラットフォームを除く）。セキュリティ上の理由から、Intelligent Agent データベースのデフォルトのアカウント / パスワードである dbsnmp/dbsnmp を顧客が使用しない場合があります。ここに示した例は、Intelligent Agent のデータベース・ログイン・アカウントの変更を顧客が希望する場合にかぎり使用してください。

**SNMP.CONTACT.<service_name> ="contact_info"**
サービスに対する責任を持つ管理者の連絡先情報を含む文字列です。連絡先情報には名前、電話番号、電子メール・アドレスなどがあります。このパラメータはオプションです。

**DBSNMP.POLLTIME = nn**
Agent がデータベースをポーリングして、ダウンしていないかどうかを確認する時間間隔（秒単位）です。データベースがダウンしている場合、あるいは接続されなかった場合には、この値は再試行間の間隔になります。デフォルトは 30 秒です。

---

**注意：** Intelligent Agent が 3 つ以上のインスタンスを監視する必要がある場合、監視されるインスタンスの数に比例して DBSNMP.POLLTIME の値を増やします。

次に例を示します。

Agent が 10 個のインスタンスを監視する必要があります。その場合、DBSNMP.POLLTIME は 150（10/2 × 30 = 150）に設定します。

---

**DBSNMP.NOHEURISTIC={TRUE/FALSE}**
このパラメータの値は、Intelligent Agent で、監視対象データベースの状態（データベースが起動しているか停止しているか）を確認するために監視および接続を行うかどうかを指定します。デフォルトでは、この値は FALSE（Agent は監視および接続を行う）に設定されます。

**注意：** 監視対象ターゲットが Oracle Real Application Clusters データベース・インスタンスの場合、DBSNMP.NOHEURISTIC は TRUE に設定する必要があります。Oracle Real Application Clusters データベース・インスタンスには、この監視および接続が機能しないためです。

**DBSNMP.TRACE_LEVEL = OFF | USER | ADMIN | nn**
指定されたレベルで、Intelligent Agent プロセス（dbsnmp）に対するトレースを有効にします。オラクル社では、トレース・レベルを 13 に設定することをお薦めしています。レベル 16 は最大の情報を提供しますが、これはバグの調査時にのみ有用です。レベル 16 では、実際の TCP/IP パケットの内容を確認できます。レベル 15 では、パケットが渡されている様子のみを確認できます。このパラメータはオプションです。

**DBSNMP.TRACE_DIRECTORY = *directory***
トレース・ファイルが書き込まれるディレクトリです。この設定は、DBSNMP.TRACE_LEVEL との組合せでのみ意味を持ちます。このパラメータを省略すると、トレース・ファイルは `$ORACLE_HOME¥network¥trace` に書き込まれます。このパラメータはオプションです。

**DBSNMP.TRACE_FILE = *filename***
トレース・ファイルのファイル名です。このパラメータはオプションです。

**DBSNMP.TRACE_FILECNT = *nn***
Agent により生成されるトレース・ファイルの最大数。完全トレースを実行することが好ましいですが、Agent マシン上のディスク領域が限られている場合、このオプション・パラメータを使用します。このパラメータは、リリース 8.1.7 の Intelligent Agent から使用可能になりました。

**DBSNMP.TRACE_FILESIZE = *nn***
Agent により生成される個々のトレース・ファイルの最大サイズ（KB）。たとえば、値が 1024 の場合は 1MB のトレース・ファイルです。完全トレースを実行することが好ましいですが、Agent マシン上のディスク領域が限られている場合、このパラメータを DBSNMP.TRACE_FILECNT とともに使用します。このパラメータは、リリース 8.1.7 の Intelligent Agent から使用可能になりました。

**DBSNMP.TRACE_UNIQUE = *true/false***
true に設定すると、各ログ・エントリについて一意のログ・ファイルが生成されます。

**DBSNMP.LOG_DIRECTORY = *directory***
ログ・ファイルが書き込まれるディレクトリです。このパラメータはオプションです。

**DBSNMP.LOG_FILE = *filename***
ログ・ファイルのファイル名です。このパラメータはオプションです。Windows NT では、デフォルトのファイル名は `dbsnmp` です。

**DBSNMP.LOG_UNIQUE = *true/false***
true に設定すると、各ログ・エントリについて一意のログ・ファイルが生成されます。

**DBSNMP.TRACE_TIMESTAMP = *true/false***
true に設定すると、トレース・ファイルのトレース各行の後に、タイムスタンプが挿入されます。

**DBSNMP.HOSTNAME = <hostname or ip address>**
IP アドレスを指定した場合、Agent はそのマシン上の特定のネットワーク・インタフェース・カードにバインドされます。ホスト名を指定した場合は、Agent はそのマシン上のすべてのネットワーク・インタフェース・カードにバインドされます。

**DBSNMP.CS_BASE_PORT = <port number>**
Intelligent Agent の収集システムで使用されるデフォルトのポート 1808 および 1809 をオーバーライドします。

例： dbsnmp.cs_base_port = 1700 と指定した場合、ポート 1700 および 1701 を使用します。

**DBSNMP.THRESHOLD_JOB_STATUS**
Agent によって Oracle Management Server にキューされるジョブ通知の数を設定します。

**DBSNMP.THRESHOLD_EVOCC**
OMS にキューされるイベント通知の数を設定します。

**DBSNMP.AVG_OCC_PER_EVENT**
イベントあたりの平均通知数を設定します。

**DBSNMP.NOTIFICATIONTIMEOUT = <nn>**
タイムアウト周期を設定します（nn はミリ秒）。ここで指定する時間以内に Intelligent Agent が Oracle Management Server に通知を送信できなかった場合、Intelligent Agent は自動的に再起動します。デフォルト値は 6 秒（360,000 ミリ秒）です。

**DBSNMPJ.TRACE_LEVEL = OFF | USER | ADMIN | nn**
指定されたレベルで、*dbsnmpj* プロセスに対するトレースを有効にします。*dbsnmpj* は、Intelligent Agent にジョブが発行される際に必ず開始されます。オラクル社では、トレース・レベルを 13 に設定することをお薦めしています。レベル 16 は最大の情報を提供しますが、これはバグの調査時にのみ有用です。レベル 16 では、実際の TCP/IP パケットの内容を確認できます。レベル 15 では、パケットが渡されている様子のみを確認できます。このパラメータはオプションです。

**DBSNMPJ.TRACE_DIRECTORY = *directory***
トレース・ファイルが書き込まれるディレクトリです。この設定は、DBSNMP.TRACE_LEVEL との組合せでのみ意味を持ちます。このパラメータを省略すると、トレース・ファイルは `$ORACLE_HOME¥network¥trace` に書き込まれます。このパラメータはオプションです。

**DBSNMPJ.TRACE_FILE = *filename***
トレース・ファイルのファイル名です。このパラメータはオプションです。

**DBSNMPJ.TRACE_FILECNT = *nn***
Agent により生成されるトレース・ファイルの最大数。完全トレースを実行することが好ましいですが、Agent マシン上のディスク領域が限られている場合、このオプション・パラメータを使用します。このパラメータは、リリース 8.1.7 の Intelligent Agent から使用可能になりました。

**DBSNMPJ.TRACE_FILESIZE = *nn***
Agent により生成される個々のトレース・ファイルの最大サイズ（KB）。たとえば、値が 1024 の場合は 1MB のトレース・ファイルです。完全トレースを実行することが好ましいですが、Agent マシン上のディスク領域が限られている場合、このパラメータを DBSNMP.TRACE_FILECNT とともに使用します。このパラメータは、リリース 8.1.7 の Intelligent Agent から使用可能になりました。

**DBSNMPJ.TRACE_UNIQUE = *true/false***
true に設定すると、トレース各行について一意のトレース・ファイルが生成されます。

**DBSNMPJ.LOG_DIRECTORY = *directory***
ログ・ファイルが書き込まれるディレクトリです。このパラメータはオプションです。

**DBSNMPJ.LOG_FILE = *filename***
ログ・ファイルのファイル名です。このパラメータはオプションです。Windows NT では、デフォルトのファイル名は `dbsnmpj` です。

**DBSNMPJ.LOG_UNIQUE = *true/false***
true に設定すると、各ログ・エントリについて一意のログ・ファイルが生成されます。

**DBSNMPJ.TRACE_TIMESTAMP = *true/false***
true に設定すると、トレース・ファイルのトレース各行の後に、タイムスタンプが挿入されます。

---

**注意：** 次のアドレスは、Agent によって自動的に設定されます。アドレスを変更すると、Agent は Enterprise Manager コンソールによって検出できなくなり、手動での構成設定が必要になります。

---

**DBSNMP.ADDRESS =(ADDRESS=(PROTOCOL=<protocol>)(HOST=<host_name>)(PORT=<port_no>))**
Agent が、送られてくる要求のリスニングに使用する TNS アドレスです。アドレスには空白または改行文字を含めないようにしてください。このパラメータは、Agent がネットワーク接続に対してリスニングを行うアドレスです。

Agent がサービスを自動検出するためには TCP/IP が必須であるため、サーバー上に TCP/IP がインストールされている必要があります。

Agent は、PORT=1748 を要求します。ポート・アドレス 1748 は、Internet Assigned Number Authority（IANA）によって Oracle に付与された登録済の TCP ポートです。ポート・アドレスは自動的に設定されます。このポートを変更すると、Agent は Enterprise Manager コンソールによって検出できなくなり、手動での構成設定が必要になります。

**DBSNMP.HOSTNAME =[hostname/IP address]**
9*i* の Intelligent Agent を、特定のホストまたは IP アドレスにバインドします。

**DBSNMP.SPAWNADDRESS =(ADDRESS= (PROTOCOL=<protocol>)(HOST=<host_name>)(PORT=<spnport_no>))**
Agent が RPC を受け入れるために使用できる TNS アドレスです。このアドレスは、ファイル転送に使用されます。このパラメータで使用される spnport_no は、DBSNMP.ADDRESS パラメータで使用される port_no とは異なります。

Agent は、PORT=1754 を要求します。ポート・アドレス 1754 は、Internet Assigned Number Authority（IANA）によって Oracle に付与された登録済の TCP ポートです。このポートを変更すると、Agent は Enterprise Manager コンソールによって検出できなくなり、手動での構成設定が必要になります。

**DBSNMP.CS_BASE_PORT = <port number>**
Agent の収集サービスによって使用されるデフォルトのポート番号をオーバーライドする際に使用するポート番号。たとえば、ポート番号 1700 を指定すると、Agent はポート番号 1700 および 1701 を使用します。

# Intelligent Agent のログ・ファイル

適切なログ・ファイル・パラメータが設定されている場合、次のログ・ファイルが生成されます。

**表 A-1 Intelligent Agent のログ・ファイル**

| ログ・ファイル | 説明 |
|---|---|
| dbsnmp.log | Intelligent Agent の全プロセスのログ |
| dbsnmpj.log | Intelligent Agent により処理された全ジョブのログ |
| dbsnmp.nohup | Intelligent Agent の自動再起動（dbsnmpwd）のログ |

# B

# トラブルシューティング

この章では、Intelligent Agent が正しく機能しないときの一般的なトラブルシューティングについて説明します。次の項目について説明します。

- Intelligent Agent のトラブルシューティング
- 簡単なチェック
- 追加のチェック
- Intelligent Agent のエラー・メッセージと解決策
- 9i Agent のトレース
- TCL のトレース

# Intelligent Agent のトラブルシューティング

ほとんどの状況下において、Intelligent Agent 自体が必要とする構成の手段はごくわずかです。ただし、正しく機能するためには、Agent が管理側のホストおよび管理対象のサービスと通信できる必要があります。Oracle および使用しているオペレーティング・システムの知識が豊富なユーザーであれば、次の簡単なチェックリストを利用して、Agent の運用を妨げる可能性のある問題を高い確率で解決できます。

---

**重要：** Agent はリリースを重ねるごとに継続的に改善されているため、ご使用の特定のサーバー・リリースで使用可能な最新の Agent にアップグレードすることを**強くお薦めします**。多くの場合、アップグレードにより旧バージョンの Agent で発生した問題が解決します。

---

# 簡単なチェック

次のチェックリストでは、Agent の運用に最も影響する領域をカバーしています。Agent のトラブルシューティング・チェックリストは 2 種類あり、それぞれ、Agent が動作する最も一般的なプラットフォームである Windows NT と UNIX を対象としています。チェックリストは簡略化されており、利用にあたっては、Oracle とオペレーティング・システムの両方、そして関連する通信プロトコルについての知識が前提となります。それぞれのトラブルシューティング手順の詳細は、この章の後の部分で詳細に説明します。

## Windows NT Agent の簡単なチェック

Windows NT システム上で Agent を実行している場合、次のチェックリストを使用します。

1. 「コントロール パネル」で OracleAgent サービスの状態を調べ、Agent サービスが開始されていることを確認します。Agent が開始されていない場合、次のヒントのいずれかを参考にして、問題を解決します。
2. NT の「イベント ビューア」（「管理ツール」の下）に書き込まれたメッセージを確認します。「イベント ビューア」には、起動に関連するすべての問題が NT Agent によって書き込まれます。
3. `snmp_ro.ora`、`snmp_rw.ora` および `services.ora` の各ファイルが、起動時に Agent によって作成されていることを確認します。`snmp_ro.ora` および `snmp_rw.ora` は、`$ORACLE_HOME¥network¥admin` ディレクトリにあります。`services.ora` は、`$ORACLE_HOME¥network¥agent` ディレクトリにあります。

   リストされたサービスを、マシン上で使用可能なサービスと比較します。有効なサンプル・ファイルの例は、付録 A「Agent 構成ファイル」を参照してください。

欠落しているサービスがある場合、次のファイル間の不一致、またはファイルの破損がないかどうかを調べます。

- listener.ora
- tnsnames.ora

4. 外部ドライブに設定されているシステム・パスがないことを確認します。

   Agent は 1 つのサービスであり、デフォルトでは SYSTEM 権限で動作します。また、ORACLE_HOME¥BIN ディレクトリにある DLL も必要とします。ドライブをパス内にマップする必要がある場合、それらを SYSTEM パスの内部に設定しないでください。

   独自のパスを設定する手順は、次のとおりです。

   a. マップされるドライブのパスを、SYSTEM パス変数から独自のパスに移動します。

   b. 再起動して、システム・パスの設定を解除します。

5. TCP/IP がインストールされていることを確認します。TCP/IP は必要条件です。

6. Agent が起動しない理由がこの時点でわからない場合、Agent をトレースします。

   a. snmp_rw.ora で、次のように変数を設定します。

      dbsnmp.trace_level=admin（16 に設定すると最も多くの情報を得られます）

      dbsnmp.trace_directory=<Oracle ユーザーが書込み権限を持つ任意のディレクトリ >

      dbsnmp.trace_file=< トレース出力ファイルの名前 >

   b. Agent を再起動します。

   c. oracle_home¥network¥log ディレクトリにあるログ・ファイルを確認します。

      DBSNMP.LOG は、Agent の一般的な問題を示します。

      DBSNMP.NOHUP は、Agent の監視プロセスである dbsnmpwd に関連するエラーを示します。

      DBSNMPCONFIG.LOG は、自動検出に伴う問題を示します。

7. DNS ホストのエントリが、listener.ora および tnsnames.ora の各ファイルで指定されたノード名に設定されていることを確認します。

   a. 「スタート」→「設定」→「コントロール パネル」→「ネットワーク」→「プロトコル」→「TCP/IP プロトコル」を選択し、「プロパティ」をクリックします。

   b. DNS ホストのエントリを確認します。

## UNIX Agent の簡単なチェック

UNIX システム上で Agent を実行している場合、次のチェックリストを利用します。

1. Agent のステータスを確認します。次のコマンドを入力します。

   ```
   agentctl dbsnmp_status
   ```

   または、次のコマンドを入力して、Intelligent Agent が動作していることを確認することもできます。

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

   Agent が動作している場合、次のように表示されます。

   ```
   DBSNMP for Solaris: Version 9.0.0.0.0 - Production on 04-NOV-01 18:44:15

   (c) Copyright 2001 Oracle Corporation.  All rights reserved.

   データベース・サブエージェントはすでに実行されています。
   ```

   このチェックでは、dbsnmp プロセスが実行中であること、または dbsnmpwd 監視スクリプトが実行中であること（あるいはその両方）が表示されるはずです。

2. ORACLE_HOME/network/log/dbsnmp*.log ファイルを参照して、UNIX システムのエラーを確認します。(エラー検索の場合は、nmiconf.log ファイル)。

3. Oracle ユーザーが、ORACLE_HOME/NETWORK/AGENT および ORACLE_HOME/AGENT/LOG への書込み許可を持っていることを確認します。

4. Agent により作成されたエントリの snmp_ro.ora、snmp_rw.ora および services.ora を確認します。snmp_ro.ora および snmp_rw.ora は、ORACLE_HOME/network/admin ディレクトリにあります。services.ora は、ORACLE_HOME/network/agent ディレクトリにあります。または、環境変数 TNS_ADMIN で指定されているディレクトリをチェックすることもできます。

   リストされたサービスを、マシン上で使用可能なサービスと比較します。有効なサンプル・ファイルの例は、付録 A「Agent 構成ファイル」を参照してください。

   欠落しているサービスがある場合、次のファイル間の不一致、またはファイルの破損がないかどうかを調べます。

   - listener.ora
   - tnsnames.ora
   - oratab

5. Agent が起動しない理由がこの時点でわからない場合、snmp_rw.ora で次のように変数を設定して Agent をトレースし、Agent を再起動します。
   - dbsnmp.trace_level=admin（16 に設定するとより多くの情報を得られます）
   - dbsnmp.trace_directory=<Oracle ユーザーが書込み許可を持つ任意のディレクトリ>
   - dbsnmp.trace_file=agent
6. Intelligent Agent 制御ユーティリティ（agentctl）の実行に問題がある場合は、agentctl のトレースを次のように設定します。
   - agentctl:trace_level=admin
   - agentctl.trace_directory
   - agentctl.trace_file
7. データベース・ソフトウェアがアップグレードされており、使用中のマシンのいずれかについて、生成された snmp_ro.ora、snmp_rw.ora または services.ora の各ファイルに問題がある場合、次の指示に従います。
   a. （dbsnmp アカウントではなく）INTERNAL または SYS アカウントで catsnmp.sql を実行します。通常、catsnmp.sql スクリプトはデータベースの作成時に catalog.sql の内部で実行されますが、データベースはアップグレードされているため、このスクリプトがまだ実行されていない可能性があります。必要なスクリプトを実行しないと、dbsnmp アカウントは作成されません。
   b. oratab ファイルで複数の SID、または古い SID が参照されている場合、それぞれのデータベースに対して catsnmp.sql を実行します。
   c. snmp_ro.ra は読取り専用ファイルであるため、このファイルへのすべての変更は、Agent を起動するたびに上書きされます。必要であれば、snmp_rw.ora ファイルに変更を加えることができます。

   バックアップを実行する場合は、dbsnmp/dbsnmp アカウントで backupts.sql を実行する必要があります。

---

**警告：** Agent に付属の Tcl スクリプト（Tcl で記述されるジョブ・スクリプトおよびイベント・スクリプト）は変更しないでください。Agent に定義済のもの以外のジョブを送る場合は、TCL ジョブを使用します。このジョブでは、任意のスクリプトを渡して Agent に実行させることができます。

---

# 追加のチェック

簡単なチェックで Intelligent Agent の問題を解決できなかった場合、次の項以降で各領域について記載された事項を参考にして、Agent の運用に関する問題の原因を突き止めてください。また、チェックリストに記された手順の多くについて、Oracle または Agent が動作しているオペレーティング・システム、あるいはその両方にそれほど詳しくないユーザーのために、より詳細に説明しています。この項では、次のような問題について説明します。



---

**注意：** Agent のデバッグを行う際、`$ORACLE_HOME/network/agent` ディレクトリ内のすべての .q ファイルを削除する必要はありません。以前はこの方法をお薦めしていましたが、最近のバージョンの Intelligent Agent のトラブルシューティングでは、この操作は必要ありません。このルールには例外もありますが、詳細はこの章の後の部分で説明します。

---

## TCP/IP の構成と動作の確認

Agent の起動に際して障害となる最も一般的な問題の 1 つは、TCP/IP の構成です。TCP/IP 設定が正しく構成されていることを確認するには、コマンドラインで次のように入力します。

- ホスト・マシン（Agent が動作するマシン）および指定されたネットワーク IP アドレスが、同じマシンを指していることを確認します。コマンドラインで、次のように入力します。

  1. ping < ホスト名 >
  2. ping <IP アドレス >
  3. 2 つのコマンドによって、同じ情報が返されることを確認します（Windows NT）。UNIX システムでは、2 つのコマンドに対してそれぞれ「<hostname> is alive」、「<IP address> is alive」と表示されることを確認します。

- 次のコマンドを実行して、ホスト・マシンに到達可能なことを確認します。

  telnet < ホスト名 >

- ホスト・マシンがローカル・ネットワーク上で有効なホストであること、また、コンソール（バージョン 2 の場合は、Management Server）を実行しているマシンからホスト・マシンへのアクセスが可能であることを確認します。

  1. Agent が動作しているマシン上で、そのマシン自体の IP アドレスへの ping を実行します。
  2. コンソールを実行しているマシン上で、Agent が動作しているマシンの IP アドレスへの ping を実行します。

- コンソール・マシンがローカル・ネットワーク上で有効なホストであること、また、Agent を実行しているマシンがコンソール・マシンと通信できることを確認します。

  1. ping < コンソール・マシンの IP アドレス >

     コンソールが動作しているマシン上で、そのマシン自体への ping を実行します。
  2. ping < コンソール・マシンの IP アドレス >

     Agent が動作しているマシン上で、コンソール・マシンへの ping を実行します。
  3. これらの 2 つのコマンドにより、同じ情報が返されることを確認します。

**注意：** Windows NT システムのホスト名は、コマンド・プロンプトで hostname と入力することで確認できます。

## TCP/IP 構成上の問題の修正

1. （Windows NT）WINNT¥system32¥drivers¥etc ディレクトリ内の、hosts ファイルおよび lmhosts ファイルを編集します。

   これらのファイルが使用されていない場合、ディレクトリ内にはサンプル・ファイルのみが存在します。.sam ファイルの名前を変更するか、ファイルをコピーするかして、拡張子なしの hosts ファイルおよび lmhosts ファイルを作成します。

   （UNIX）root でログインして、/etc/hosts ファイルを編集します。

2. 各システムの IP アドレスとホスト情報が正しいことを確認します。

   例：（Windows NT）

   （カッコ内の情報は、システムの実際のホスト情報に置き換えてください。）

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

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

   **注意：** この情報は、Windows NT の「コントロール パネル」→「ネットワーク」プロパティで修正することもできます。

3. $ORACLE_HOME¥network¥agent ディレクトリ内の、*.q ファイルおよび services.ora ファイルを削除します。

   **注意：** *.q ファイルには、現行のジョブとイベントに関する情報が含まれています。これらのファイルを削除する前に、Agent に対して登録されているすべてのジョブとイベントを削除するようにしてください。

4. $ORACLE_HOME¥network¥admin¥snmp_ro.ora および $ORACLE_HOME¥network¥admin¥snmp_rw.ora ファイルを削除します。

5. Agent を再起動します。

## DNS 名およびコンピュータ名の一致の確認（Windows NT）

リリース 8.0.4 より前の Agent では、DNS ホスト名とコンピュータ名が一致している必要がありました。これらのパラメータは、Windows NT の「コントロール パネル」の、次のプロパティ・シートで確認または変更できます。

コンピュータ名を確認するには次のようにします。

- 「コントロール パネル」→「ネットワーク」→「識別」→「コンピュータ名」

DNS 名を確認するには次のようにします。

- 「コントロール パネル」→「ネットワーク」→「プロトコル」→「TCP/IP プロトコル」→「プロパティ」→「DNS」→「ホスト名」

## Oracle Net 構成ファイルの確認

ネットワークが適切に構成され、ネットワーク内のノードが互いに通信できることに加えて、Oracle 環境のコンポーネントが互いに通信できる必要があります。Oracle Net は、クライアント・マシンと Oracle サーバーの間、あるいは複数の Oracle サーバー間に、セッションおよびデータ通信の媒体を提供します。したがって、Agent が通信を行うためには、Oracle Net の適切な構成が前提条件となります。この項では、Agent の通信が失敗するときに起こりうる最も一般的な問題について説明します。

Oracle Net の構成ファイルは、`$ORACLE_HOME¥network¥admin` または $TNS_ADMIN（Windows NT）か、`$ORACLE_HOME/network/admin`（UNIX）の各ディレクトリにあります。

主要な構成ファイルには、次のようなものがあります。

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

これらのファイルの詳細および設定例は、付録 A「Agent 構成ファイル」を参照してください。

### Agent による検出の間の TNS_ADMIN 変数の使用方法

**（UNIX）**すべてのバージョンの UNIX 検出スクリプトで、TNS_ADMIN 変数を使用して入力ファイル（`listener.ora` および `tnsnames.ora`）の場所を特定できます。TNS_ADMIN が設定される場合、リリース 7.3.4 以上の Agent でのみ、出力ファイル（`snmp_ro.ora` および `snmp_rw.ora`）を TNS_ADMIN に正しく書き込むことができます。

**（Windows NT）**リリース 8.0.5 以上では、検出スクリプトにより TNS_ADMIN の値が NT レジストリから併せて読み取られます。

Agent では、listener.ora ファイル中の TNS 別名情報も使用されます。Agent のこの動作は、Oracle Names 環境の内部でも同じです。この動作は意図的なものです。これは、Oracle Names サーバーは一時的に使用不可になる可能性がある一方で、Agent は常に名前を解決できなければならないという理由によります。次の手順で、TNS 別名がローカルでどのように変換されるかを確認してください。

1. listener.ora ファイルに、それぞれのインスタンスに対する次の要素が含まれていることを確認します。

   - 2 つの IPC エントリ
   - 1 つの TCP エントリ

   Agent はポート 1748 上でリスニングするため、このポート上でリスナーをアクティブ化しないでください。（Agent に対して TNSPING を使用できるのはこのためです。TNSPING はリスナーと Agent を区別できません。）

   Agent による別名の変換には、IPC エントリと、TNS のサーバー上およびコンソール上での別名定義が必要になります。この正しい IPC エントリと TNS 別名定義は、Agent とコンソールの間（バージョン 1）、または Agent と Management Server の間（バージョン 2）で正確な通信を行うために必須です。

2. DNS ホストのエントリが、listener.ora ファイルおよび tnsnames.ora ファイル中のノード名に設定されていることを確認します。

   1. Windows NT のタスクバーから、「スタート」→「設定」→「コントロール パネル」をクリックします。
   2. 「ネットワーク」アイコンをダブルクリックします。
   3. 「プロトコル」タブをクリックします。
   4. 「TCP/IP プロトコル」を選択して「プロパティ」をクリックします。
   5. DNS ホストのエントリを確認します。

## Oracle Net の動作確認

Oracle Net の構成が正しいにもかかわらず、依然として Agent と通信できない場合、次の手順として、Oracle Net 内のどのサービスが到達可能であるかを調べます。コマンド・プロンプトで次のように入力することにより、アクセス先の各データベース上で TNSPING ユーティリティを使用できます。

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

TNSPING を使用して、クライアントからサーバーに（またはサーバー間で）正しく接続できる場合、前述のコマンドにより、Oracle Net サービスへの到達にかかる推定の往復時間（ミリ秒単位）が返されます。これは、Oracle Net が正常に動作していることを表しています。

その後、コンソールの tnsnames.ora ファイルに次の別名（Agent のデバッグ・エントリ）を追加します。

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

その後、OEM コンソールから次のように入力して、Agent への ping を実行します。

```
tnsping agent_<sid>
```

または

```
tnsping80 agent_<sid>
```

TNSPING コマンドが機能しない場合、前述の別名を Agent マシンの tnsnames.ora に追加し、Agent が動作しているマシンから TNSPING の実行を試みます。すべての Agent は、この別名を使用して TNSPING を実行できる必要があります。

## Agent の正常起動の確認

次のコマンドを発行して、Agent のプロセスが動作していることを確認します。

```
agentctl status agent
```

Agent が開始されていない場合、次の表のヒントを参考にして、問題を解決してください。

**表 B-1 Agent が起動しない場合のトラブルシューティング**

| UNIX | Windows NT |
|---|---|
| ファイル<br>`$ORACLE_HOME/network/log/dbsnmp*.log`<br>を参照して、エラーを確認します。 | NT の「イベント ビューア」(「管理ツール」の下）に書き込まれたメッセージを確認します。「イベント ビューア」には、起動に関連するすべての問題が NT Agent によって書き込まれます。 |
| ファイル<br>`$ORACLE_HOME/network/log/nmiconf.log`<br>を参照して、エラーを確認します。 | ファイル<br>`$ORACLE_HOME/network/log/nmiconf.log`<br>を参照して、エラーを確認します。 |
| Oracle ユーザーが、次のディレクトリへの書込み許可を持っていることを確認します。<br>`$ORACLE_HOME/agent/log`<br>`$ORACLE_HOME/network/agent` | Agent サービスのプロパティで、Agent によって使用される OS アカウント（デフォルトでは 'System'）を確認します。Agent ユーザーが、次のディレクトリへの書込み許可を持っていることを確認します。<br>`$ORACLE_HOME/agent/log`<br>`$ORACLE_HOME/network/agent` |
| snmp_ro.ora、snmp_rw.ora および services.ora の各ファイルを参照して、Agent によって作成されたエントリを確認します。snmp_ro.ora および snmp_rw.ora は、$ORACLE_HOME/network/admin ディレクトリにあります。services.ora は、$ORACLE_HOME/network/agent ディレクトリにあります。 | snmp_ro.ora、snmp_rw.ora および services.ora の各ファイルが、起動時に Agent によって作成されていることを確認します。snmp_ro.ora および snmp_rw.ora は、$ORACLE_HOME¥network¥admin ディレクトリにあります。services.ora は、$ORACLE_HOME¥network¥agent ディレクトリにあります。 |

**表 B-1 Agent が起動しない場合のトラブルシューティング（続き）**

| UNIX | Windows NT |
|---|---|
| リストされたサービスを、マシン上で使用可能なサービスと比較します。有効なサンプル・ファイルの例は、付録 A を参照してください。欠落しているサービスがある場合、次のファイル間の不一致、またはファイルの破損がないかどうかを調べます。<br>■ listener.ora<br>■ tnsnames.ora<br>■ oratab | リストされたサービスを、マシン上で使用可能なサービスと比較します。有効なサンプル・ファイルの例は、付録 A を参照してください。欠落しているサービスがある場合、次のファイル間の不一致、またはファイルの破損がないかどうかを調べます。<br>■ listener.ora<br>■ tnsnames.ora |
| TCP/IP がインストールされていることを確認します。TCP/IP は必要条件です。TCP/IP が正しく構成され、動作しているかどうかを確認します。 | TCP/IP がインストールされていることを確認します。TCP/IP は必要条件です。TCP/IP が正しく構成され、動作しているかどうかを確認します。 |
| Agent が起動しない理由がこの時点でわからない場合、トレースを有効にします。（「Agent のトレース」を参照してください。） | システム・パス変数に外部ドライブが含まれていないことを確認します。Agent は 1 つのサービスであり、デフォルトでは SYSTEM 権限で動作します。また、$ORACLE_HOME¥bin ディレクトリにある DLL も必要とします。パス内に外部ドライブをマップする必要がある場合、それらのドライブを SYSTEM パスの内部に設定しないでください。独自のパスを設定する手順は、次のとおりです。<br>1. マップする外部ドライブのパスを、システム・パス変数から独自のパスに移動します。<br>2. 再起動して、システム・パスの設定を解除します。 |
| Agent が起動しない理由がこの時点でわからない場合、トレースを有効にします。Agent のトレース設定の詳細は、B-24 ページの「9i Agent のトレース」を参照してください。 | |

UNIX および Windows NT システムのいずれでも、次を確認します。

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

## Agent がノード上の全インスタンスに接続していることの確認

Agent が、あるノード上で監視するデータベースに接続できることをテストするには、次の接続文字列を使用して、各データベースへの接続を試みます。

```
dbsnmp/dbsnmp@address_list
```

このテストは、Agent が動作するノード上で実行する必要があります。

> **注意：** リリース 7.3.3 より前の Agent は、ローカル・データベースへの 2 つの永続的接続を維持します。リリース 7.3.3 よりも新しい Agent では、1 つの永続的接続のみが維持されます。

## Agent が正しい許可で動作していることの確認（UNIX）

Agent が正しいユーザー許可を持っていることを検証する方法については、2-2 ページの「Intelligent Agent のインストール」を参照してください。

## OS ユーザーが存在し、正しい許可を持っていることの確認（Windows NT）

ノードに対して OS ユーザーを指定し、次の許可を付与する必要があります。

- `$ORACLE_HOME¥network` ディレクトリと、そのすべてのサブディレクトリに対する読取りおよび書込み許可。
- $TEMP ディレクトリに対する読取りおよび書込み許可（Windows NT の $TEMP ディレクトリは、「コントロール パネル」→「システム」を選択して確認できます）。$TEMP が定義されていない場合、Oracle ホーム・ディレクトリに対する読取りおよび書込み許可を OS ユーザーが持っている必要があります。OS ユーザーは、Oracle ホーム・ディレクトリの下に work という名前で一時ディレクトリを作成します。
- 前述のユーザーが、`$ORACLE_HOME¥network¥agent` ディレクトリへの書込み許可を持っている必要があります。

## エラーの有無の確認

（Windows NT）「イベント ビューア」→「ログ」→「アプリケーション」で、DBSNMP プロセスの起動に関するエラーを確認します。

（Windows NT および UNIX）`$ORACLE_HOME/network/log/nmiconf.log` ファイルを参照して、検出エラーの有無を確認します。

UNIX および Windows NT システムのいずれでも、次のファイルで追加のエラーを確認します。

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

## ジョブが実行されたにもかかわらず、Agent がステータス通知を Enterprise Manager コンソールに返信しない理由

ほとんどの場合、ジョブは実際に実行されていますが、Agent がコンソールに通信できないために通知を返すことができません。ホスト名が解決できることを確認します。コンソールが動作している Windows NT マシンの IP アドレスおよびホスト名が、UNIX マシンの /etc/hosts ファイルに記述されており、DNS/NIS を経由してホスト名が解決できることを確認します。ジョブを再試行します。

TCP/IP の解決をテストするには、コマンド・プロンプトから次のテストを実行します。

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

サーバー上で telnet または ftp のサービスが動作していることを確認します（UNIX）。

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

PING では TCP ではなく IP が使用されるため、これは問題がパケットのルーティングにあるかどうかを判断するのに良い方法です。

問題が実際に TCP に関係しているかどうかを判断するには、telnet ユーティリティまたは ftp ユーティリティを使用します。

Enterprise Manager コンソール・マシンの名前および IP アドレスが、Sun サーバーの /etc/hosts ファイルに記述されていることを確認します。記述がない場合、Agent はコンソール・マシンの名前を IPADDRESS に解決できないため、コンソールにメッセージを返すことができません。

デフォルトのリスニング・アドレス（TNS 形式）は、次のとおりです。

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

ジョブがスケジューリングされた状態のままになっている場合、[Del] キーを使用してそのジョブの削除を繰り返します。ジョブを再起動します。起動するまでに数回の送信が必要な場合もあります。通常は、ジョブが起動するまでに最大 1 分程度の遅延があります。古い Agent（リリース 7.3.2）において、Agent が最初に OEM コンソールとの同期を試みるときに、そのような傾向があります。

# Intelligent Agent のエラー・メッセージと解決策

次に示すエラー・メッセージと解決策は、オペレーティング・システム別に分類されています。すべてのシステムに共通の解決策は、「Agent 全般」の項にリストしてあります。

## Agent 全般

**ORA-12163: 'TNS: 接続記述子が長すぎます '**

$ORACLE_HOME/network/admin/snmp_ro.ora ファイルから、snmp.address.<host_name> パラメータをコピーします。このアドレスおよびパラメータを、$ORACLE_HOME/network/admin/snmp_rw.ora ファイルにペーストします。snmp_rw.ora では、IPC に対するアドレス・エントリを削除することによって、この接続文字列の長さを切りつめます。(NMP および SPX も削除できます。)

Agent をシャットダウンし、再起動します。次の例を参照してください。

---

**注意 :** リリース 7.3.4/8.0.3 以上の Agent では、snmp_ro.ora にパラメータ snmp.address は存在しません。したがって、この例を参考にして、snmp_rw.ora に新しい変数を追加する必要があります。

---

例 :

snmp_ro.ora から、次のエントリをコピーします。

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

snmp_rw.ora で修正されたエントリは次のとおりです。

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

**TNS-12542:'TNS: アドレスがすでに使用中です '**

これは、実際には Oracle Net リスナーのエラーです。

次に示すのは、リリース 8.0.3.0.0 Intel NT 版 Oracle Net リスナーのリリース・ノートから抜粋した説明です。クライアントが専用のサーバー・モードで Oracle8 Server に接続するとき、WINSOCK2 の共有ソケット機能を使用して、クライアントの接続がリスナーからデータベース・サーバーにルーティングされます。クライアントはリスナーとのソケット接続を

クローズする必要なしに、データベース・サーバーと新しい接続を確立するため、この機能により接続時間が短縮されます。

共有ソケットを使用する場合、スレッドもリスナーと同じポートを使用します。リスナーをシャットダウンし、同じポートでそのリスナーを再び起動しようとするとき、データベースとの間のなんらかのオープンな接続によりポートが使用中であれば、リスナーは起動しません。リスナーを起動する前に、データベースに接続しているクライアントがないことを確認してください。別のポート番号でリスナーを使用する場合には、リスナーを起動できます。

> **警告：** データベースに接続しているクライアントが 1 つでもあるときには、リスナーを停止しないでください。新しいデータベースをリスニングする必要がある場合、構成ファイル listener.ora を修正して、リスナー制御ユーティリティ LSNRCTL80 から reload コマンドを発行します。

listener.ora ファイルと LSNRCTL80 ユーティリティの詳細は、Oracle ネットワーク製品の Windows プラットフォーム用のスタート・ガイドを参照してください。オラクル社では、WINSOCK2 のオプションを使用してポートの再使用を可能にすることにより制限の克服を試みましたが、オプション機能の信頼性は十分ではありません。オラクル社では現在、Microsoft 社と協力して、この問題の解決に取り組んでいます。

reload コマンドの詳細は、『Oracle Net Services 管理者ガイド』を参照してください。

**VOC-04816 ' 宛先が無効です '** ジョブの送信中、「Agent_node のアドレスの検索に失敗しました」というメッセージを表示して認証が失敗します。次に、VOC-04816 ' 宛先が無効です ' エラーが発生します。これは、コンソール・マシンの tnsnames.ora 中に無効なアドレスが記述されていることが原因である可能性もあります。

リリース 7.3.3 以上の Agent にアップグレードしてください。

また、SQL*Net の構成ファイル群が正しいことを確認してください。

**ジョブ実行時の ' ユーザー認証に失敗しました ' エラー** 管理対象のノード上で Agent がジョブを実行するために必要な条件は、次のとおりです。

- 高度なユーザー権利「バッチ ジョブとしてログオン」を持つ NT ユーザー・アカウントが存在すること。(Windows NT)。権限は既存のローカル・ユーザーまたはドメイン・ユーザー（リリース 7.3.3 以上)、あるいは新規 NT ユーザーに割り当てることができます。第 2 章「インストールと構成」の、Windows NT に固有の説明箇所を参照してください。
- Oracle Enterprise Manager コンソールで、ノードに対する優先接続情報が前述のユーザーに対して設定されていること。『Oracle Enterprise Manager 構成ガイド』の「作業環境の設定」を参照してください。
- ユーザーが $ORACLE_HOME/network ディレクトリ、または $ORACLE_HOME¥network ディレクトリへの書込み許可を持っていること。

**トレース・ファイル中の ' ログインは拒否されました '、' ユーザー名 / パスワードが無効です ' の各メッセージ** これらのメッセージは、マシン上にリリース 7.3.3 より前のデータベースが存在する場合に書き込まれます。リリース 7.3.3 以上では、CATSNMP.SQL という名前のスクリプトが、ディクショナリ・スクリプト CATALOG.SQL に組み込まれています。このスクリプトには、Agent が接続のために必要とする DBSNMP ユーザーを作成する役割があります。リリース 7.3.3 より前のデータベースには、このスクリプトはありません。

ユーザー 'DBSNMP' が存在することを確認してください。存在しない場合は、catsnmp.sql スクリプトを実行してください。

**Agent 起動時の 'ORACLE_HOME は存在しません '** このメッセージは、検出スクリプト nmiconf.tcl によるものです。環境変数 $ORACLE_HOME が Agent の ORACLE_HOME に設定されていることを確認し、Agent を再起動してください。

**Agent が 1 ノード上で 1 つのデータベースしか検出しない** 1 つのノード上に複数のデータベースがある場合、listener.ora 中に、各インスタンスに対する一意の GLOBAL_DBNAME の記述があることを確認する必要があります。listener.ora 中に、このパラメータを手動で定義しなければならない場合もあります。

**snmp_ro.ora および snmp_rw.ora が生成されていない** このエラーは、Agent が $ORACLE_HOME/network/admin ディレクトリへの書込み許可を持たない場合に発生することがあります。$ORACLE_HOME/netowrklog/nmiconf.log を参照して、エラーを調べます。Agent の起動に関する問題の詳細は、B-12 ページの「Agent の正常起動の確認」を参照してください。

**検出されないサービスがある** services.ora ファイルを参照して、どのサービスが検出済であるかを調べます。

Agent がマシン上で検出するすべてのサービスは、関連する SQL*Net/Oracle Net 構成ファイルで定義されている必要があります。定義されていないサービスがあると検出は失敗し、最悪の場合、Agent は停止するかエラーを返します。

> **Windows NT**: リリース 8.0.4 以上の Agent では、'OracleService' または 'OracleServive<SID>' で始まるサービス名が検索されます。'OracleService' で始まるすべてのエントリは、このマシン上で動作しているデータベースとみなされます。Agent が取り扱うすべての SID は、関連する SQL*Net/Oracle Net ファイルで定義されている必要があります。
>
> **UNIX**: oratab ファイルは、どの SID が存在するかを特定するために使用されます。リリース 7.3.3 以下の Agent では、不正確な（Developer/2000 環境のものなど）SID が見つかると、検出は失敗します。この問題に対処するために、$ORATAB 環境変数を使用して、Agent に検出させるデータベースのみを含む別の oratab ファイルにアクセスできます。

残りのデータベースに対しては、oratab ファイル、および SQL*Net/Oracle Net のファイル群を調べることにより、これらのファイルが存在し、すべての定義が記述されていることを確認します。listener.ora ファイルに、すべてのデータベースが記述されていること

を確認します。詳細は、B-9 ページの「Oracle Net 構成ファイルの確認」、および B-11 ページの「Oracle Net の動作確認」を参照してください。

**ジョブまたはイベント登録時の ' サービス名が無効です ' または ' ファイル操作エラー ' の表示** これらのエラーは通常、コンソール上のサービスと、Agent によって検出されるサービスが同期していないときに発生します。たとえば、あるイベントを TESTDB に対して登録し、その後でデータベースの名前を PRODDB に変更したような場合、Agent とコンソールは同期しなくなります。

この問題を修正するには、このサービスからジョブおよびイベントの登録をすべて削除し、次に、サービスが存在するノードをコンソールから削除します。自動検出ウィザードを使用し、コンソール上でノードを再検出します。

**注意：** リリース 7.3.2 では、別名の大文字と小文字が区別されます。

NT Agent を使用している場合は、NT Agent に関する説明のジョブまたはイベント登録時の ' サービス名が無効です ' を参照してください。

**'Transport read error' または 'Transport write error' メッセージ** これらのメッセージは、TCP/IP レイヤーでの問題を示すものです。考えられる最大の原因は、IP アドレスとホスト名が同じ物理マシンを参照していないことです。

TCP/IP が正しく構成され、動作していることを確認します。（「TCP/IP の構成と動作の確認」を参照してください。）

**'Oralogin failed in orlon'** Software Developer's Kit を通して、oratcl 動詞 oralogon を使用した TCL スクリプトを実行している間に、このエラーが発生することがあります。メッセージ「Oralogin failed in orlon」は、接続文字列が間違っているか、使用されているアカウントがなんらかの理由でデータベースにログインできないことを表しています。

## NT Agent

### Agent 起動時のすべての NT オペレーティング・システム・エラー

**Agent の起動時に OS エラーが発生する場合、それが実際に snmimsg.mc に記述されている Agent エラーであるかどうかを確認してください。** 一部の Windows API が定義どおりに機能しないことが原因で、Agent がエラーの実際の原因を出力できないことがあります。

**Windows NT の「管理ツール」グループにある「イベント ビューア」を使用してください。** 記録された問題の本当の原因を見つける必要があります。Agent エラーのソースは、サービス名 dbsnmp の下にあります。リスト中の最も新しい dbsnmp エントリを選択します。イベントをダブルクリックすると、実際の原因が表示されます。

**OS エラーの発生後に Agent の問題を解決する手順は、次のとおりです。**

- 「イベント ビューア」を起動します。（このツールは「管理ツール」プログラム・グループにあります。）メイン・メニューの「ログ」をクリックし、次に「アプリケーション」を選択します。Agent エラーのソースは、サービス名 dbsnmp の下にあります。リスト中の最も新しい dbsnmp エントリを選択します。イベントをダブルクリックすると、実際の原因が表示されます。
- DBSNMP.LOG および NMICONF.LOG の各ファイルには、発生した特定のエラーについてのより詳細な情報が記されています。
- `snmp_ro.ora` および `snmp_ro.ora` の各ファイルが `$ORACLE_HOME¥network¥admin` ディレクトリに存在し、サイズが 0 でないことを確認します。
- Agent を起動したユーザーが、キュー・ファイルに対する読取りおよび書込み許可を持っていることを確認します。
- マシン名と DNS ホスト名が一致していることを確認します。
- 「DNS 名およびコンピュータ名の一致の確認（Windows NT）」を参照してください。SQL*Net が実行中であることを確認します。
- TCP/IP が正しく構成され、動作していることを確認します。（「TCP/IP の構成と動作の確認」を参照してください。）
- `$ORACLE_HOME¥network¥agent¥MIB` ディレクトリから、必須でないファイルをすべて削除します。
- NT マシン上で、8.0.3 データベースと同じ ORACLE_HOME ディレクトリに Oracle Enterprise Manager 1.5 がインストールされていないことを確認します。
- Oracle 8.0.3 より前のデータベースを使用している場合、実行している Agent のインスタンスが 1 つのみであることを確認します。Oracle リリース 8.0.4 以上では、新しい Agent サービス（OracleAgent80）を作成することにより、マシン上で 2 つの Agent サービスを動作させることができます。
- リリース 8.0.4 の Agent にアップグレードしてください。

**'Agent への接続に失敗しました ' エラー（送られた状態のままのジョブが存在する）**

Windows NT 上では、実際には 2 種類のホスト名定義が存在します。1 つは NetBIOS の定義で、これは常にインストールされている、Windows NT 内部の Named Pipes プロトコル用に使用されます。もう 1 つは TCP/IP ホスト名で、Windows NT 上に TCP/IP をインストールしたときにのみ構成可能です。

Windows NT の NetBIOS ホスト名を調べる手順は、次のとおりです。

- 「コントロール パネル」で「ネットワーク」をダブルクリックします。
- 表示されたダイアログ・ボックスの「コンピュータ名」が、NetBIOS のホスト名を表しています。

TCP/IP のホスト名を調べる手順は、次のとおりです。

- 「コントロール パネル」から「ネットワーク」→「プロトコル」→「TCP/IP プロトコル」→「プロパティ」→「DNS」の順に開きます。
- 表示された「ホスト名」が、TCP/IP のホスト名に相当します。

Windows NT サーバー上では、前述の 2 種類の名前が異なるような構成がなされている場合でも、その両方に ping を実行できます。一方、クライアント側では、TCP/IP の実ホスト名のみに ping を実行できます。Agent がローカル IPC 接続を使用している場合、Agent は Named Pipes を使用します。これは NetBIOS の名前ですが、その一方で、すべての外部接続には TCP/IP の名前が使用されます。

この 2 種類の名前が一致しない場合、'Agent に接続できません ' エラーが発生するか、ジョブがコンソール内に保留され続けます。したがって、NetBIOS のホスト名と TCP/IP のホスト名が一致していることを確認する必要があります。

**ジョブ実行中の、失敗→ 'output from job lost' エラーの発生** Agent 用に作成した Windows NT ユーザー（第 2 章「Intelligent Agent のインストール」および『Oracle Enterprise Manager 構成ガイド』を参照）には、`$ORACLE_HOME¥network¥agent` ディレクトリ（さらに、一部のアプリケーションでは TEMP ディレクトリ）に対する読取り / 書込み許可と、SYSTEM32 ディレクトリの読取り許可を与える必要があります。

NT ユーザーがこれらの許可を持っていることを確認してください。

**検出の後で Agent がサービスを見つけられない** この問題は、リリース 7.3.4 以上の Agent で修正されています。リリース 7.3.3 以下の Agent では、次の代替手段を取ることができます。

`listener.ora` ファイルの `SID_LIST` セクションで、`$ORACLE_HOME` パラメータが指定されていないことを確認します。`SID_LIST` セクションで `$ORACLE_HOME` が指定されていると、Agent はサービス検出のために必要なファイルを見つけられなくなります。

**ジョブまたはイベント登録時の ' サービス名が無効です '** この問題は、リリース 8.0.4 の Agent を使用している場合に起きる可能性があります。.world 以外のデフォルト・ドメインがある場合、Agent は検出時に .world をデータベース名に追加しようとします。たとえば、デフォルト・ドメインが nl.oracle.com で、`GLOBAL_DBNAME = database.nl.oracle.com` と定義されている場合、Agent はデータベース名を `database.nl.oracle.com.world` と定義します。この問題は、Agent とコンソールが同じマシン上に常駐する（同じ構成ファイルを共有する）場合にかぎり発生します。

代替の手段は、指定されたドメインをその時点で持っていないすべてのサービスに `.world` を付加することです。

# UNIX Agent

## サービスがまったく見つからずに検出が失敗する

まず、SQL*Net のファイル群がすべて存在し、正しく定義されていることを確認してください。次に、oratab ファイルを編集し、動作しているリスナーの有効な SID のみを記述することで、検出の問題を解決できます。この作業が終わったら、oratab ファイルに残りのエントリを追加して、問題の原因となるエントリを特定できます。

$ORACLE_HOME/network/log/nmiconf.log ファイルを参照して、エラーを確認してください。

### NMS-0308 : ' アドレスでのリスニングに失敗しました : 別の Agent が実行されている可能性があります '

このエラーの原因は 2 種類考えられます。

1. 1 台のマシン上の異なる ORACLE_HOME ディレクトリに 2 つの Agent がインストールされている場合、Agent の 2 つ目のインスタンスを起動しようとすると、このメッセージが表示されます。これは、両方の Agent が同じ 1748 番のデフォルト・ポートをリスニングしようとすることが原因です。

1 台のマシン上には 1 つの Agent のみを配置してください。

2. Agent がリスニングするポート 1748 が他のプロセスによって使用されているか、以前にそのポートを使用していて現在は停止しているプロセスがポートを解放していません (SUN のマシン上で一般的な問題です)。

ポートが他のプロセスによって使用されていることを確認する手順は、次のとおりです。

1. 次の UNIX コマンドを入力します。

```
netstat -a | grep 1748
                    ^---- これはポート番号です
```

「LISTENING」で終わる文字列が画面に表示される場合、ポートは使用中です。

2. 次の条件に当てはまる場合、

- netstat -a | grep 1748（結果が「LISTENING」と表示されます。）
- agentctl status agent（「データベース・サブエージェントは起動されていません。」と表示されます。）

次の操作を行います。

- ps -ef | grep dbsnmp
- kill -9 _____（プロセス番号を入力）
- agentctl start agent で Agent を再起動します。

この時点でなお Agent を起動できない場合、前述の手順を再度実行します。ただし、Agent を再起動する前に次の操作を行います。

- cd $ORACLE_HOME/network/agent
- rm *.q, services.ora, snmp_ro.ora, and snmp_rw.ora
- `agentctl start agent` で Agent を再起動します。

この操作により Agent は再起動しますが、Agent が過去に使用していたジョブおよびイベントのキューはすべて削除されます。

他の手段がすべて失敗する場合、マシンを再起動することでポートが解放されます。

**Agent 起動時の NMS-205 エラー** このメッセージは、SNMP Master Agent（SNMP プロトコルを制御する UNIX 上のプロセス）と通信できないことを表します。デフォルトでは、Agent は SQL*Net を介してリスニングし動作しますが、UNIX システム上の SNMP を経由して動作することもできます。

Master Agent を使用した通信を試みる場合以外は、このメッセージを無視しても差し支えありません。

**Agent 起動時の NMS-205 エラー** 'dbsnmp' ユーザーを見つけられませんでした。

SYS または INTERNAL どちらかのアカウントで、データベースに対して catsnmp.sql スクリプトを実行します。

**Agent 起動時の NMS-351 エラー** この問題は、$ORACLE_HOME/network/agent ディレクトリにある '*.q' ファイルの ID 間に不一致がある場合に発生します。
$ORACLE_HOME/network/agent ディレクトリ中のすべての '*.q' ファイルを削除します。リポジトリを再構築します。Agent を再起動します。

# 9*i* Agent のトレース

リリース 7.3.3 以上では、Agent は $ORACLE_HOME/network/admin ディレクトリにある snmp_ro.ora および snmp_rw.ora の各ファイルから情報を読み取ります。

---

**注意**： これらのファイルは、Agent を最初に起動した後でのみ存在します。Agent の起動以後、最初に Agent をトレースする場合、snmp_rw.ora という名前の新規ファイルを手動で作成して、このファイルにトレース・パラメータを追加できます。そうでない場合は、Agent を起動し、snmp_rw.ora ファイルにトレース情報を追加して、Agent を再起動します。

---

snmp_rw.ora ファイルに加える変更の例を次に示します。

```
DBSNMP.TRACE_LEVEL = (OFF | USER | ADMIN | 16 )
```

DBSNMP.TRACLE_LEVEL の設定は、SQL*Net 用に使用されるものを反映しています。

オプションとして、次のパラメータを設定します。

```
DBSNMP.TRACE_FILE = agent         Default=dbsnmp.trc
DBSNMP.TRACE_DIRECTORY = /private/temp  Default=$ORACLE_HOME/network/trace
```

（Agent が書込み許可を持つ任意の既存ディレクトリ）

---

**注意：** Data Gatherer の機能は 9*i* の Agent に統合されたため、データ収集ベースのトレースを有効にするには、次のようにします。

1. setenv VP_DEBUG 1
2. 次に、agentctl start agent で Agent を起動します。

任意の収集アクティビティにログインします。

$ORACLE_HOME/network/log/dbsnmp.nohup.

---

トレースが有効でない場合でも、ログ・ファイル $ORACLE_HOME/network/log/dbsnmp.log は、起動のたびに Agent によって書き込まれます。このファイルには、Agent の名前とバージョン、および Agent 構成ファイルの名前とその場所が記されています。トレースが有効にされている場合、データベースおよびリスナーの接続に伴う問題も記述されます。

ログ・ファイル $ORACLE_HOME/network/log/nmiconf.log は、Agent の初回起動時に作成され、その後起動のたびに内容が追加されます。自動検出は、Tcl スクリプト nmiconf.tcl（ログ・ファイル名の由来）によって行われます。このファイルは起動の間にのみ書き込まれます。$ORACLE_HOME/agentbin/ORATCLSH は、特別な目的に使用される TCL シェルで、OEM Agent によってサポートされる ORATCL 動詞の（全部ではなく）広範な部分集合に加えて、標準の TCL 動詞（TCL75.dll でサポートされる）のすべてをサポートします。

ORATCLSH は汎用のユーティリティではなく、OEM Agent によってメンテナンスされるファイルおよびデータ構造に依存するため、OEM Agent との組合せでのみ使用できます。

ORATCLSH のドキュメントは存在せず、OEM Agent のサポート対象機能セットの一部でもありません。ORATCLSH は、OEM のジョブ・スクリプトおよびイベント・スクリプトを開発する Oracle ユーザーおよび開発者を支援する目的で限定的に提供されるデバッグ・ツールです。ORATCLSH の実行可能プログラムは、TCL スクリプトのデバック用に提供されます。ORATCLSH を実行する前に、TCL_LIBRARY 環境変数が $ORACLE_HOME/network/agent/tcl を指すように設定します。これは init.tcl ファイルのある場所です。

# TCL のトレース

ORATCL_DEBUG 環境変数を設定し、`snmp_rw.ora` ファイル中でトレースを有効にすることにより、Tcl のトレースを有効にできます。ORATCL_DEBUG は、$ORACLE_HOME/network/trace ディレクトリに設定する必要があります。これらのパラメータを有効にするには、Agent をシャットダウンし、再起動する必要があります。TCL のトレースにより、前述のディレクトリにファイル oratcl.trc が作成されます。イベントが実行されるたびに、oratcl.trc ファイルにエントリが追加されます。

# C

# キーボード・ショートカット

この付録には、一般的な Windows のキーボード・ショートカットのリストが記載されています。Windows のキーボード・ショートカットおよびナビゲーションの完全なリストは、ご使用のオペレーティング・システムのドキュメントを参照してください。

**表 C-1 一般的な Windows キーボード・ショートカット**

| キー | 動作 |
|---|---|
| [F1] | アクティブ・オブジェクトまたはウィンドウ全体のヘルプ情報を表示します。 |
| Windows キーまたは [Ctrl]+[Esc] | タスクバー上にある「スタート」メニューを開きます。 |
| [Ctrl]+[Alt]+[Del] | Microsoft Windows では、「プログラムの強制終了」ダイアログ・ボックスを開きます。ここには、終了する対象のアプリケーションのリストと、「終了」、「シャットダウン」および「キャンセル」のコマンド・ボタンが表示されます。<br>Microsoft Windows NT では、「Windows NT のセキュリティ」ダイアログ・ボックスを開きます。ここには「ワークステーションのロック」、「ログオフ」、「シャットダウン」、「パスワードの変更」、「タスク マネージャ」および「キャンセル」のオプションがあります。ログインしていない場合は、ログイン・ダイアログ・ボックスが開きます。 |
| [Del] | 選択した項目を削除します。項目がファイルの場合、それらを「ごみ箱」に移します。 |
| [Shift]+[Del] | 選択した項目を削除します。項目がファイルの場合、それらを「ごみ箱」に移さず、ただちに削除します。 |
| [Ctrl]+[N] | 「新規作成」ダイアログ・ボックスを開きます。(「ファイル」メニューの「新規作成」コマンドを選択することもできます。) |
| [Ctrl]+[O] | 「ファイルを開く」ダイアログ・ボックスを開きます。(「ファイル」メニューの「開く」コマンドを選択することもできます。) |
| [Ctrl]+[P] | 「印刷」ダイアログ・ボックスを開きます。(「ファイル」メニューの「印刷」コマンドを選択することもできます。) |
| [Ctrl]+[S] | 「ファイル名を付けて保存」ダイアログ・ボックスを開きます。(「ファイル」メニューの「上書き保存」コマンドを選択することもできます。) |
| [Ctrl]+[X] | 選択した項目を切り取り、クリップボードに移します。(「編集」メニューの「切り取り」コマンドを選択することもできます。) |
| [Ctrl]+[Insert] または [Ctrl]+[C] | 選択した項目をクリップボードにコピーします。(「編集」メニューの「コピー」コマンドを選択することもできます。) |
| [Shift]+[Insert] または [Ctrl]+[V] | クリップボードから、コピーした項目を貼り付けます。(「編集」メニューの「貼り付け」コマンドを選択することもできます。) |
| [Alt]+[Back Space] または [Ctrl]+[Z] | 直前の操作を元に戻します。操作の中には元に戻せないもの（シャットダウンなど）もあることに注意してください。(「編集」メニューの「元に戻す」コマンドを選択することもできます。) |
| [Alt]+[Shift]+[Back Space] | 直前の元に戻す操作を繰り返します。(「編集」メニューの「やり直し」コマンドを選択することもできます。) |

**表 C-1 一般的な Windows キーボード・ショートカット（続き）**

| キー | 動作 |
|---|---|
| Windows キー +[M] | すべてのウィンドウを最小化します。キーボードのフォーカスは、デスクトップ上で直前に選択されていたアイコンに移ります。前に開いていたウィンドウを拡大し、最後に使用していたアプリケーションにフォーカスを戻すには、[Shift] を一緒に押します。 |

# 用語集

### Oracle プライベート MIB（Oracle Private MIB）

Oracle 製品のみに固有の MIB。

### RDBMS パブリック MIB（RDBMS Public MIB）

Internet Engineering Task Force（IETF）によって合意された、リレーショナル・データベース用の標準 MIB。この MIB は、データベース名など、リレーショナル・データベースに対して一般に関連している様々な OID（たとえば、`rdbmsDbName`, `1.3.6.1.3.55.1.2.1.4` など）をサポートする。

### Simple Network Management Protocol（SNMP）

OID を操作するネットワーク・プロトコル。Oracle の場合、2 種類の基本的な SNMP 操作がサポートされる。`get oid` は、`oid` の値をフェッチする。`getnext oid` は、`oid` 後の次の OID 値を取得する。

### SNMP オブジェクト ID（SNMP Object ID: OID）

`a.b.c...x.y.z` という形式をとるピリオドで区切られた数列。MIB の一部である情報項目を一意に識別する。OID には通常、それに関連付けられた名前を付けることができる。OID は本質的に階層型である。したがって、`1.2.3` は `1.3` の前であるが、`1.2` の後になる。たとえば、Oracle7 データベースが実行した物理読込みの数を含む OID は、次のようになる。

```
oraDbSysPhysReads, 1.3.6.1.4.1.111.4.1.1.1.8
```

### イベント（Event）

イベントとは、Intelligent Agent によって監視されるデータベースまたはノードの上で起こりうる特定の条件。たとえば、データベースがダウンすると、DBDOWN イベントが発生する。イベントは次のどちらかの方法で検出できる。(1) 特定の条件を監視する Tcl スクリプトを定期的に実行する。(2) サード・パーティ製品を利用して、イベントの発生を Agent に直接報告させる。

**管理情報ベース（Management Information Base: MIB）**

通常関連する SNMP オブジェクトの ID（OID）の集合。

**修正ジョブ（Fixit Job）**

イベントの発生によって引き起こされる特別な種類のジョブ。たとえば、使用済の表領域が 90% を超えたことが表領域満杯イベントによって検出されると、表領域にデータファイルを追加するための修正ジョブが自動的に実行される。

**ジョブ（Job）**

一度かぎり、または繰り返しのスケジュールで実行できる Tcl スクリプトのこと。イベントが特定の条件を監視するのに対して、ジョブはなんらかのタスクを遂行することが求められる。ジョブの例には、バックアップやデータベースの起動がある。

# 索引