Sun Identity Manager 8.1 システム管理者ガイド

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 821–0780–10
2009 年 2 月

091118@22749

目次

はじめに

この Identity Manager 8.1 システム管理者ガイド には、システム管理者が Sun™ Identity Manager ソフトウェア (Identity Manager) を実行しているシステムを管理する上で理解しておくべき情報が記載されています。

お読みになる前に

Identity Manager は、ネットワークまたはインターネット環境に分散したエンタープライズアプリケーションをサポートするソフトウェアインフラストラクチャーとなる、Sun Java Enterprise System のコンポーネントです。Sun Java Enterprise System で提供されているドキュメントをよく読んでください。ドキュメントは、`http://docs.sun.com/coll/entsys_04q4` からオンラインで入手できます。

Identity Manager の配備では、Identity Manager Directory Server がデータストアとして使用されるので、製品で提供されているドキュメントをよくお読みください。Directory Server のドキュメントは、`http://docs.sun.com/coll/DirectoryServer_04q2` からオンラインで入手できます。

対象読者

本書は、アプリケーションサーバーとデータベースの管理者、第一線のサポートエンジニア、および Identity Manager を実行中のシステムを管理するパートナーで、Identity Manager パフォーマンスの最適化に関心があり、配備環境で Identity Manager の維持管理を担当している方を対象としています。

Identity Manager 管理者は、次のテクノロジを理解している必要があります。

- Lightweight Directory Access Protocol (LDAP)
- Java™ テクノロジ
- JavaServer Pages™ (JSP™) テクノロジ
- ハイパーテキストトランスポートプロトコル (HTTP)
- ハイパーテキストマークアップ言語 (HTML)
- XML (Extensible Markup Language)

本書の構成

このガイドは、次の章で構成されています。

第 1 章「Identity Manager サーバーの設定」では、Identity Manager プロパティーの設定方法と Identity Manager サーバーが所定のタスクを実行できるように設定する方法について説明します。

第 2 章「ファイアウォール、ロードバランサ、またはプロキシサーバー使用時」では、ファイアウォール、ロードバランサ、またはプロキシサーバーを設置した場合、Identity Manager が URL (Uniform Resource Locators) を使用して Identity Manager の使用方法についての情報を記載することを説明します。

第 3 章「ログの使用」では、監査ログやシステムログで実行する可能性のあるタスクについて説明します。

第 4 章「チューニングの実行」では、Identity Manager のパフォーマンスを向上させ、パフォーマンス関連の問題をデバッグできるツール、方法、および関連資料について説明します。

第 5 章「トレースとトラブルシューティング」では、Identity Manager トレースを使用した問題の解決方法、パフォーマンス問題の解決方法、および本製品コンポーネントのフローを理解する方法について説明します。

第 6 章「エラーと例外」では、Identity Manager のエラーと例外メッセージについて説明します。

関連ドキュメントとヘルプ

Sun Identity Manager ライブラリには、本書以外にも次のマニュアルがあります。

主な対象読者	タイトル	説明
すべてのユーザー	『Sun Identity Manager の概要』	Identity Manager の機能の概要を説明しています。製品のアーキテクチャー情報を示し、Identity Manager を Sun のほかの製品 (Sun Open SSO Enterprise、Role Manager など) と統合する方法について説明します。
	『Sun Identity Manager 8.1 リリースノート』	既知の問題、修正された問題、および Identity Manager のドキュメントセットに記載されていない最新の情報を説明しています。

主な対象読者	タイトル	説明
システム管理者	『Sun Identity Manager 8.1 Installation 』	Identity Manager と、Sun Identity Manager Gateway や PasswordSync などのオプションコンポーネントのインストール方法について説明しています。
	『Sun Identity Manager 8.1 アップグレード』	古いバージョンの Identity Manager を新しいバージョンにアップグレードする方法について説明しています。
	『Sun Identity Manager 8.1 システム管理者ガイド』	システム管理者がインストールした Identity Manager の管理、調整、およびトラブルシューティングを行う際に役立つ情報と手順を説明しています。
ビジネス管理者	『Sun Identity Manager 8.1 ビジネス管理者ガイド』	Identity Manager のプロビジョニング機能および監査機能を使用する方法について説明しています。ユーザーインタフェース、ユーザーとアカウント管理、報告機能などが説明されています。

主な対象読者	タイトル	説明
システムインテグレータ	『Sun Identity Manager Deployment Guide 』	Identity Manager を複雑な IT 環境に配備する方法について説明しています。説明している内容は、アイデンティティー属性の操作、データの読み込みと同期、ユーザーのアクションの設定、カスタムブランディングの適用などです。
	『Sun Identity Manager Deployment Reference 』	ワークフロー、フォーム、ビュー、およびルールについて説明し、XPRESS 言語の章が含まれています。
	『Sun Identity Manager 8.1 リソースリファレンス』	リソースアダプタのインストール、設定、および使用法について説明しています。
	『Sun Identity Manager Service Provider 8.1 Deployment 』	Identity Manager Service Provider の配備方法と、ビュー、フォーム、およびリソースの標準 Identity Manager 製品との違いを説明しています。
	『Sun Identity Manager 8.1 Web Services 』	SPML サポートの設定方法、サポートされる SPML 機能とサポートの理由、およびフィールドでサポートを拡張する方法について説明しています。

ドキュメントの更新

このドキュメントやほかの Identity Manager のドキュメントに関する訂正や更新は、Identity Manager Documentation Updates の Web サイトに掲載されます。

`http://blogs.sun.com/idmdocupdates/`

RSS フィードリーダーを使用して Web サイトを定期的に確認し、更新を利用できる場合に通知を受けることができます。サイトを購読するには、フィードリーダーをダウンロードして、ページの右側の「Feeds」の下にあるリンクをクリックします。バージョン 8.0 から、メジャーリリースごとのフィードを利用できます。

関連するサードパーティー Web サイト

このドキュメントでは、サードパーティー URL を参照して、追加の関連情報を提供します。

注 – このドキュメントで取り上げる他社の Web サイトが使用可能かどうかについて、Sun は関知いたしません。Sun は、このようなサイトまたはリソースで得られるあらゆる内容、広告、製品、およびその他素材を保証するものではなく、責任または義務を負いません。Sun は、このようなサイトまたはリソースで得られるあらゆるコンテンツ、製品、またはサービスによって生じる、または生じたと主張される、または使用に関連して生じる、または信頼することによって生じる、いかなる損害または損失についても責任または義務を負いません。

ドキュメント、サポート、トレーニング

Sun の Web サイトでは、次の追加リソースに関する情報を入手できます。

- ドキュメント (http://www.sun.com/documentation/)
- サポート (http://www.sun.com/support/)
- トレーニング (http://www.sun.com/training/)

ご意見、ご要望の送付先

Sun ではマニュアルの品質向上のため、ご意見、ご要望をお受けしております。

ご意見を送信するには、http://docs.sun.com にアクセスして、「フィードバック」リンクをクリックしてください。オンラインフォームに、マニュアルのタイトルと Part No. を入力してください。Part No. は、マニュアルタイトルページまたは最上部に記載されている 7 桁または 9 桁の番号です。

たとえば、本書のタイトルは「Identity Manager 8.1 システム管理者ガイド」で、Part No. は *821-0780-10* です。

書体の表記規則

次の表では、このガイドで使用する書体の違いについて説明します。

表 P–1 書体の表記規則

字体または記号	意味	例
`AaBbCc123` (モノスペース)	API および言語要素、HTML タグ、Web サイトの URL、コマンド名、ファイル名、ディレクトリパス名、画面上のコンピュータ出力、サンプルコード。	`.login` ファイルを編集します。 すべてのファイルを一覧表示するには、`ls -a` を使用します。 `% You have mail.`
`AaBbCc123` (モノスペース太字)	ユーザーが入力する文字を、画面上のコンピュータ出力とは区別して示します。	`% su Password:`
AaBbCc123 (斜体)	書名、新規用語、強調すべき言葉。 実際の名前または値によって置き換えられるコマンドまたはパス名の可変部分。	『Identity Manager 8.1 システム管理者ガイド』の第 6 章をお読みください。 これらのオプションは、クラスオプションと呼ばれています。 ファイルを保存しないでください。 このファイルは、*install-dir*`/bin` ディレクトリにあります。

コマンドのシェルプロンプトの例

次の表は、C シェル、Bourne シェル、および Korn シェルの、デフォルトの UNIX® システムプロンプトとスーパーユーザーのプロンプトを示しています。

表 P–2 シェルプロンプト

シェル	プロンプト
C シェル	`machine_name%`
C シェル (スーパーユーザーの場合)	`machine_name#`
Bourne シェルおよび Korn シェル	`$`
Bourne シェルおよび Korn シェル (スーパーユーザーの場合)	`#`

注 – Windows のコマンドラインのプロンプトは `c:\` です。

◆ ◆ ◆ 第 1 章

Identity Manager サーバーの設定

この章では、指定したタスクを実行できるように、Identity Manager のプロパティの設定方法と Identity Manager サーバーの設定方法について説明します。

この章の情報は、次のトピックで構成されています。

Waveset.properties を使用した Identity Manager の設定

Identity Manager の設定の中には、Waveset.properties をテキストエディタで編集しないと更新できないものがあります。この Waveset.properties ファイルは、Identity Manager の標準インストール先ディレクトリの config ディレクトリ内にあります ($WSHOME/config または %WSHOME%/config)。

Waveset.properties ファイルは、大まかに次の部分から構成されています。

- Misc Options
- Authentication Options
- Object Cache Options
- Provisioning Options
- Task Scheduler Options
- Resource Adapter Options
- List Cache Properties
- Rule Driven Members Cache Properties
- Session Settings

- Server Listeners
- UI Options
- Global Options for the JNDI Context Pooling
- Task Executor Options
- XML Parsing Options

▼ Waveset.properties を編集する

1. テキストエディタで、`Waveset.properties` を開きます。これは、**Identity Manager** の標準インストール先ディレクトリの `config` ディレクトリ内にあります (`$WSHOME/config` または `%WSHOME%/config`)。
2. 変更が必要な設定があれば修正して、ファイルを保存します。
3. 他の **Identity Manager** インスタンスにも、必要に応じてこの手順を繰り返します。

Administrator Interface を使用した Identity Manager の設定

ここでは、調整サーバー、スケジューラ、JMX などのタスクに、管理者ツールを使用してサーバー固有の設定を行う方法について簡単に説明します。

▼ 管理者インタフェースを使用して Identity Manager を設定する

Identity Manager サーバーで、指定したタスクだけを実行できるようにサーバー固有の設定を編集する

1. 管理者インタフェースから、「設定」>「サーバー」の順にクリックします。
   「サーバーの設定」ページが開きます。
2. 「サーバーの設定」ページが表示されたら、リストからサーバー名をクリックします。
   Identity Manager に、「サーバー設定の編集」ページが表示されます。
3. 「サーバー設定の編集」ページに変更が必要な設定があれば修正し、「保存」をクリックします。

調整サーバーの設定

調整サーバーとは、調整を行う Identity Manager のコンポーネントです。調整については、『Sun Identity Manager 8.1 ビジネス管理者ガイド』の「アカウント調整」を参照してください。

ここでは、次のタスクを行う際に役立つ手順を説明します。

- 15 ページの「調整の設定を行う」
- 16 ページの「Reconciler Status を表示する」

注 – 調整サーバーのチューニングとトラブルシューティングについては、第 5 章「トレースとトラブルシューティング」を参照してください。

▼ 調整の設定を行う

調整を設定するには、次の項目を実行します。

1. 14 ページの「**Administrator Interface を使用した Identity Manager の設定**」に記載されている項目を実行します。
2. 「調整」タブを選択します。
   「サーバー設定の編集」ページにデフォルトで調整の設定内容が表示されます。
3. デフォルト値のままにするか、カスタム値を指定するには、「デフォルトオプションを使用」の選択を解除します。

注 – Identity Manager サーバーで使用している調整のデフォルト設定を変更するには、20 ページの「サーバーのデフォルト設定の編集」を参照してください。

---

4. 次の設定を行います。
   - 「並列リソース制限」。調整サーバーが並列処理できるリソーススレッドの最大数を指定します。リソーススレッドは作業項目をワークスレッドに割り当てます。 このため、リソーススレッドを追加する場合、ワークスレッドの最大数を増やすことが必要になることがあります。初めてインストールした場合、デフォルト値の 3 が指定されます。
   - 「最小ワークスレッド」。調整サーバーが常に状態を維持する処理スレッド数を指定します。初めてインストールした場合、デフォルト値の 2 が指定されます。

- 「最大ワークスレッド」。調整サーバーが使用できる処理スレッドの最大数を指定します。調整サーバーは、作業の負荷に応じて、スレッドを必要な数だけ開始します。ここで指定する値でその数が制限されます。ワークスレッドは、短時間アイドル状態が続くと自動的に閉じます。初めてインストールした場合、デフォルト値の 6 が指定されます。

▼ Reconciler Status を表示する

Reconciler Status を表示するには、次の項目を実行します。

1 次の **URL** をブラウザに入力して、「**Reconciler Status**」デバッグページを開きます。

`http://<AppServerHost>:<Port>/idm/debug/Show_Reconciler.jsp`

ここで `AppServerHost` は、調整サーバーを有効に設定しているホストを指します。

注 – `/idm/debug/` ページを表示するには、デバッグ機能を使用できる必要があります。この機能については、『Sun Identity Manager 8.1 ビジネス管理者ガイド』の「ユーザーへの機能の割り当て」を参照してください。

---

2 「**Reconciler Status**」ページを更新して、更新された **Reconciler Status** 情報を表示します。
このページの詳細は、「ヘルプ」をクリックします。

スケジューラの設定

スケジューラのコンポーネントは、Identity Manager 内のタスクのスケジュールを制御するものです。

特定のサーバーでスケジューラを設定するには、

1. 14 ページの「Administrator Interface を使用した Identity Manager の設定」に記載されている項目を実行します。
2. 「スケジューラ」タブを選択します。

   デフォルト値のままにするか、「デフォルトオプションを使用」を選択解除して、次のカスタム値を指定することもできます。

   - 「スケジューラの起動」。このサーバーに搭載されているスケジューラの起動モードを選択します。
     - 「自動」。サーバーの起動時に起動します。これはデフォルトの起動モードです。
     - 「手動」。サーバー起動時に起動しますが、手動で起動するまで一時停止したままになります。

- 「無効」。サーバー起動時に起動しません。
- 「トレースの有効化」。このサーバーで、標準出力に対するスケジューラのデバッグ追跡をオンにするには、このオプションを選択します。
- 「最大同時タスク数」。スケジューラが同時に実行する最大タスク数をデフォルト以外の値に指定するには、このオプションを選択します。この制限を超える追加タスクのリクエストは、延期されるか、別のサーバーで実行します。
- 「タスク指定」。このサーバーで実行できるタスク一式を指定します。タスクを指定するには、利用可能タスクのリストから 1 つ以上のタスクを選択します。選択されたタスクのリストは、選択するオプションに応じて、包含リストまたは除外リストになります。リストから選択されたタスク以外のすべてを有効にすることも (デフォルト)、逆に選択されたタスクのみを有効にすることも可能です。

3. サーバー設定に変更内容を保存するには、「保存」をクリックします。

Identity Manager サーバーに使用するスケジューラのデフォルト設定を変更するには、20 ページの「サーバーのデフォルト設定の編集」を参照してください。スケジューラのチューニングとトラブルシューティングについては、68 ページの「スケジューラのチューニング」および123 ページの「タスクスケジューラのトレース」を参照してください。

電子メールテンプレートサーバーの設定

SMTP サーバーの設定を行うには、14 ページの「Administrator Interface を使用した Identity Manager の設定」に記載されている項目を実行します。「電子メールテンプレート」タブを選択します。

デフォルト以外を指定するには、「デフォルトの選択を使用」を選択解除し、使用するメールサーバーを入力して、デフォルトの電子メールサーバーを指定します。入力するテキストは、電子メールテンプレートの *smtpHost* 変数の置換に使用されます。

SMTP (Simple Mail Transfer Protocol) は、インターネット経由での電子メール転送の標準規格です。

Identity Manager サーバーに使用する SMTP のデフォルト設定を変更するには、20 ページの「サーバーのデフォルト設定の編集」を参照してください。

JMX 監視の設定

JMX (Java Management Extensions) は、アプリケーション、システムオブジェクト、デバイス、およびサービス指向ネットワークの管理や監視を可能にする Java テクノロジです。管理/監視対象のエンティティーは、MBean (Managed Bean) と呼ばれるオブジェクトによって表されます。

ここでは、JMX クライアントからシステムを代わりに監視できるよう、Identity Manager サーバーに搭載した JMX を設定する方法について説明します。

注 – また、JMX から監査イベントを使用できるように Identity Manager を設定することもできます。これについては、『Sun Identity Manager 8.1 ビジネス管理者ガイド』の「JMX パブリッシャータイプ」を参照してください。

▼ JMX のポーリング設定を行う

個々のサーバーに JMX ポーリングの設定を行うには、次の項目を実行します。

1. 14 ページの「Administrator Interface を使用した Identity Manager の設定」に記載されている項目を実行します。「**JMX**」タブを選択します。

2. **JMX** クラスタポーリングを有効にして、ポーリングスレッドの間隔を設定します。

   次のオプションを使用します。

   - 「**JMX** の有効化」。JMX Cluster MBean に使用するポーリングスレッドを有効または無効にします。JMX を有効にするには、デフォルト設定の選択を解除します (デフォルト設定に対して `false` を選択します)。ポーリングサイクルにシステムリソースを使用するため、JMX の使用を計画している場合にのみこのオプションを有効にしてください。
   - 「ポーリング間隔 **(ms)**」。JMX が有効に設定されている場合、サーバーがリポジトリのポーリングを行うデフォルト間隔を変更します。間隔はミリ秒単位で指定します。

     デフォルトポーリング間隔は 60000 ミリ秒に設定されます。これを変更するには、このオプションのチェックボックスを選択解除し、表示される入力フィールドに新しい値を入力します。

3. サーバー設定に変更内容を保存するには、「保存」をクリックします。

注 – Identity Manager サーバーに使用する JMX ポーリングのデフォルト設定を変更するには、20 ページの「サーバーのデフォルト設定の編集」を参照してください。

JMX データの表示

JMX で収集したデータを表示するには、JMX クライアントを使用します。このクライアントは JConsole と呼ばれ、JDK 1.5 に付属しています。

ローカルでの JConsole の使用

実行中のサーバーと同じマシンで JConsole を使用するには、 `JAVA_OPTS` プロパティを次のように設定します。

```
-Dcom.sun.management.jmxremote
```

JConsole は正しい PID を使用して接続します。

リモートでの JConsole の使用

JConsole をリモートで使用するには、`JAVA_OPTS` プロパティーを次のように設定します。

- `-Dcom.sun.management.jmxremote.port=9004`
- `-Dcom.sun.management.jmxremote.authenticate=false`
- `-Dcom.sun.management.jmxremote.ssl=false`
- `jre/lib/management` ディレクトリ内の `jmxremote.access` ファイルを編集し、次の 2 つの行のコメントを解除します。
  - `monitorRole readonly`
  - `controlRole readwrite`

  Identity Manager MBeans, を表示するには、次のような URL のサーバーに接続します。

  ```
  service:jmx:rmi:///jndi/rmi://localhost:9004/jmxrmi
  ```

使用する環境によっては、その他の設定が必要になることもあります。詳細は、JConsole のマニュアルを参照してください。

注 – このほか、Identity Manager デバッグページ (http://ホスト:ポート/idm/debug) に移動して「Show MBean Info」ボタンをクリックすれば、JMX データを表示することもできます。

JMX については、次の Web サイトにアクセスしてください。

`http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/docs.jsp`

サーバーのデフォルト設定の編集

サーバーのデフォルト設定機能を使用すると、Identity Manager サーバーすべてにデフォルト設定を設定することができます。個別のサーバー設定ページで異なる項目を選択しないかぎり、サーバーはこれらの設定を継承します。

▼ サーバーのデフォルト設定を編集する

1 管理者インタフェースから、「設定」>「サーバー」の順にクリックします。

「サーバーの設定」ページが開きます。

2 「サーバーのデフォルト設定の編集」をクリックします。

「サーバーのデフォルト設定の編集」ページが開きます。

「サーバーのデフォルト設定の編集」ページには、個々のサーバー設定ページと同じオプションが表示されます。ヘルプについては、個別のサーバー設定ページのマニュアルを参照してください。

それぞれのデフォルトサーバーに行った変更内容は、対応する個々のサーバー設定に反映されます。ただし、その設定の「デフォルトオプションを使用」を選択解除している場合は反映されません。

サーバー設定に変更内容を保存するには、「保存」をクリックします。

◆ ◆ ◆ 第 2 章

2 ファイアウォール、ロードバランサ、またはプロキシサーバー使用時

この章では、Identity Manager が URL (Uniform Resource Locator) を使用する仕組みを説明するほか、ファイアウォール、ロードバランサ、またはプロキシサーバーを設置した場合に正しい URL データを取得できるように Identity Manager を設定する方法について説明します。

Servlet API

Web を使用する Identity Manager のユーザーインタフェースは、Web クライアントの検索対象となるページの位置を指定する際に URL (Uniform Resource Locator) を基にしています。

Identity Manager は、生成された HTML と HTTP 応答に有効な URL を配備できるように、アプリケーションサーバー (Glassfish, Apache Tomcat, IBM WebSphere, or BEA WebLogic など) に搭載されている Servlet API に基づいて、HTTP 要求にある URL が省略されていないかどうかを判断します。

設定によっては、Web クライアントが HTTP 要求に使用する URL をアプリケーションサーバーが特定できないものもあります。この例には次のものがあります。

- Web クライアントと Web サーバーの間、または Web サーバーとアプリケーションサーバーの間に配備されたポート転送またはネットワークアドレス変換 (NAT) ファイアウォール
- Web クライアントと Web サーバーの間、または Web サーバーとアプリケーションサーバーの間に配備されたプロキシサーバー (Tivoli Policy Director WebSEAL など)

例えば HTTP 要求からの正確な URL データを提供しない Servlet API の場合、正確なデータを `Waveset.properties` ファイル (Identity Manager のインストール先の `config` ディレクトリ内にあります) に設定することができます。

次の属性によって、Identity Manager の Web ベースドキュメントのルートと、Identity Manager が HTML BASE HREF タグを使用するかどうかが決まります。

- `ui.web.useBaseHref` (デフォルト値: `true`)— この値には、次のいずれかの値を設定します。
  - `true`— Identity Manager は HTML BASE HREF タグを使用して、関連する URL パスのルートを表します。
  - `false`— HTML に記載されるすべての URL には、スキーム、ホスト、ポートなどを含めて省略されていないパスが記載されます。
- `ui.web.baseHrefURL`— 生成済み HTML に使用される BASE HREF を明示するには、この属性に空白以外の値を設定します。この値は、servlet API で算出された値よりも優先されます。

この計算された値の上書きは、これらの API から返される値が必ずしも正確でない場合に有効です。 この状況は、次の場合に発生します。

- アプリケーションサーバーが、ポート転送または NAT を使用するファイアウォールの背後に位置している
- アプリケーションサーバーと Web サーバーの間のコネクタから正確な情報が提供されない
- アプリケーションサーバーが、プロキシサーバーによって前処理されている

ロードバランサ使用時のログの設定

`x-Forwarded-For` HTTP 要求ヘッダーに記載されているクライアント IP アドレスを、自動的にログに記録するように Identity Manager を設定することができます。これは、ロードバランサや HTTP プロキシを経由しても Web サーバーの接続先クライアントの発信 IP アドレスを特定できる標準ヘッダーです。必要に応じて、カスタム HTTP ヘッダーを使用するように Identity Manager を設定することも可能です。

カスタム HTTP 要求ヘッダーに記載されている IP アドレスを Identity Manager でログに記録するには、次の項目を実行します。

1. テキストエディタで `Waveset.properties` を開きます。
2. `client.headerIPVariable=` を検索し、この行のコメントを解除します。
3. 代わりに Identity Manager で使用する HTTP 要求ヘッダー名を入力します。

   または、プロパティに =0 を設定すれば、この機能を無効にすることができます。
4. `Waveset.properties` を保存します。
5. Identity Manager を再起動します。

注 – この設定は、監査ログとシステムログの両方に適用することができます。

Identity Manager で自動的にクライアント IP アドレスをログに記録しない場合は、`Waveset.properties` ファイル内の `client.headerIPVariable` のコメントを解除すれば、この機能を無効にできます。

◆ ◆ ◆ 第 3 章

3 ログの使用

この章では、監査ログまたはシステムログを使用する際に実行すべきタスクについて説明します。

このトピックの内容は次のとおりです。

監査ログの改ざん防止

次の形式の監査ログの改ざんを防止するように、Identity Manager を設定できます。

- 監査ログレコードの追加または挿入
- 既存の監査ログレコードの変更
- 監査ログレコードまたは監査ログ全体の削除
- 監査ログの切り捨て

すべての Identity Manager の監査ログレコードには、一意のサーバー別シーケンス番号、およびレコードとシーケンス番号の暗号化ハッシュがあります。

改ざん検出レポートを作成するときに、サーバーごとに監査ログが走査され、次の点が調べられます。

- シーケンス番号の欠如 (削除されたレコードを示す)
- ハッシュの不一致 (変更されたレコードを示す)
- 重複したシーケンス番号 (コピーされたレコードを示す)
- 予想より小さな最後のシーケンス番号 (切り捨てられたログを示す)

▼ 改ざん防止ログを設定する

1 「レポート」→「新規」→「監査ログの改ざんレポート」の順に選択して、改ざんレポートを作成します。

2 図 3–1 のような「改ざんレポートの定義」ページが表示されたら、レポートにタイトルを入力してから保存します。

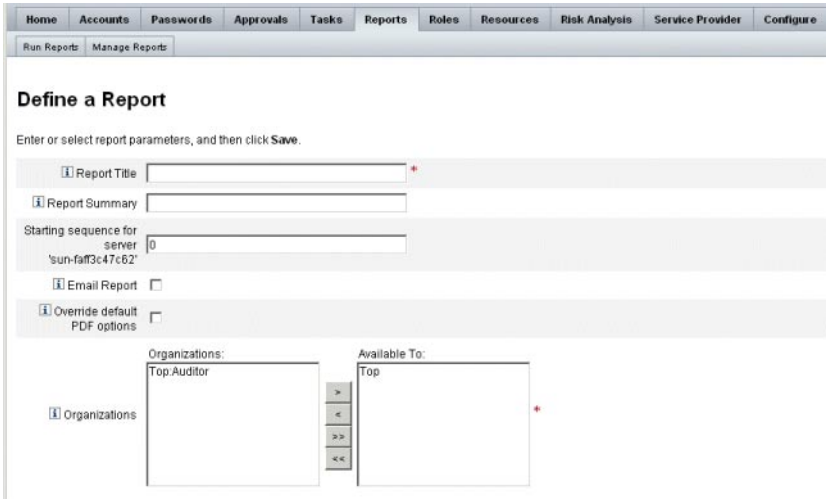


図 3–1 監査ログの改ざんレポートの設定

次のオプションパラメータも指定できます。

- 「レポート概要」。レポートのわかりやすい名称を入力します。
- <サーバー名>サーバーの開始シーケンス番号。そのサーバーの開始シーケンス番号を入力します。
- このオプションを使用すると、改ざんのフラグを付けなくても古いログのエントリを削除できるようになり、レポートのパフォーマンス上のスコープを制限します。
- 「レポート結果を送信」。レポート結果を、指定の電子メールアドレスに送信できます。
- このオプションを選択すると、ページが更新され、電子メールアドレスを指定するようにリクエストされます。ただし電子メールは、テキストコンテンツの機密情報 (アカウント ID やアカウント履歴など) が漏れる危険性があるため安全ではないことに注意してください。

- 「デフォルトの **PDF** オプションを上書き」。このレポートに、デフォルト PDF オプションの上書きを選択します。
- 「組織」。このレポートにアクセスできる組織を選択します。

3 次に、「設定」→「監査」の順に選択して、図 3–2 のように「監査設定」ページを表示します。

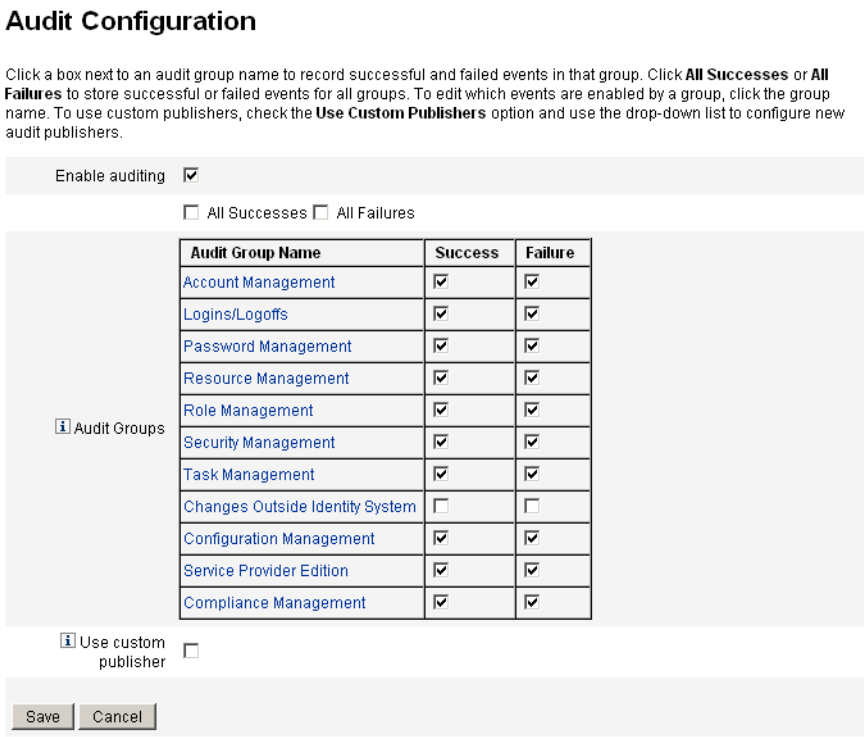


図 3–2 改ざん防止監査ログ設定

4 「カスタムパブリッシャーの使用」を選択してから、「**Repository**」のリンクをクリックします。

5 「改ざん防止ログ」を選択してから、「**OK**」をクリックします。

6 「保存」をクリックして、この設定を保存します。

このオプションをもう一度選択解除できますが、解除したエントリには、監査ログの改ざんレポートで、解除されていることを示すフラグが付けられます。 これらのエントリを無視するようにレポートを再設定する必要があります。

システムログからのレコードの削除

システムログは、Identity Manager から生成されたエラーを取り込みます。システムログは、大きくなりすぎないように、定期的に切り捨ててください。システムログから古いレコードを削除するには、「システムログメンテナンスタスク」を使用します。

▼ タスクをスケジュールしてシステムログから古いレコードを削除する

1. 管理者インタフェースで、「サーバータスク」→「スケジュールの管理」の順にクリックします。
2. 「スケジューリング可能なタスク」セクションで、「システムログメンテナンスタスク」をクリックします。
   「システムログの維持管理タスクの新規作成」ページの「タスクスケジュール」ページが表示されます。
3. フォームに値を入力し、「保存」をクリックします。

◆ ◆ ◆ 第 4 章

チューニングの実行

適切にチューニングを行えば、ほぼすべての動作に渡って Sun Identity Manager (Identity Manager) ソフトウェアのパフォーマンスを大幅に向上させることができます。このソフトウェア内の設定を変更する以外にも、アプリケーションサーバー、Java™ Virtual Machine (JVM™ マシン)、ハードウェア、オペレーティングシステム、およびネットワークトポロジをチューニングすることによって、パフォーマンスを向上させることができます。

また、パフォーマンスの診断や監視には、複数のツールを使用することもできます。トレースやメソッドタイマーなど、パフォーマンスの診断や監視を行うための複数のツールが Identity Manager 内に用意されています。また、ほかの Sun Microsystems のツールや他社製のツールを使用して、Identity Manager のパフォーマンスに関する問題をデバッグすることもできます。

この章では、パフォーマンスの向上とパフォーマンス関連問題のデバッグに使用できるツール、方法、および参考資料について説明します。

注 – チューニング処理は、多数のエンティティーに渡り、配備環境によって変わります。この章では、Identity Manager のパフォーマンスを最適化する各種チューニング方法を説明しますが、これらの方法はガイドラインとしてのみご利用ください。これらの方法は配備環境に応じて調整することが必要な場合があります。

この章では、次のトピックについて説明します。

チューニングを開始する前に

Identity Manager のチューニングを開始する前に、ここで説明することにすべて目を通してください。

重要な注意点

注 – この章で説明するチューニング方法は、ガイドラインとしてのみご利用ください。配備に合わせてこれらのチューニングの一部の変更が必要になる場合があります。さらに、本稼働環境に変更内容を適用する前に、チューニング内容を必ず検証しておいてください。

Identity Manager をチューニングする前に、次の項目が必要です。

- アプリケーションサーバーのチューニングに慣れておくこと。
- Java 5.0 (Sun Identity Manager 8.1 に必須) に使い慣れておくこと。
- 配備環境におけるパフォーマンス制限事項を理解しておくこと。
- パフォーマンスの向上が必要な領域を特定できること。
- この章に記載したチェックリストを理解しておくこと。

関連ドキュメントと Web サイト

Identity Manager のチューニングに関しては、この章の説明のほかに、この節で紹介しているドキュメントや Web サイトも参照してください。

推奨ドキュメント

パフォーマンスのチューニングについては、次のドキュメントを参照してください。

表 4–1 関連ドキュメント

ドキュメントのタイトル	説明
『IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 5.0 Diagnostics Guide』	AIX JVM を使用したパフォーマンス問題の診断方法について説明します。
『Java Tuning White Paper』	Java パフォーマンスのチューニングに関する情報、テクニック、およびポインタが記載されています。

表 4–1 関連ドキュメント (続き)

ドキュメントのタイトル	説明
『Oracle MetaLink: Note:114671.1: Gathering Statistics for the Cost Based Optimizer』	システム統計と Oracle® の Cost-Based Optimizer (CBO) の使用方法について説明します。 注意: このドキュメントは、Oracle Metalink 購読者が入手できるものです。登録が必要です。
『Solaris Dynamic Tracing Guide』	DTrace を使用したシステム動作の監視、デバッグ、およびチューニング方法について説明します。
『Sun Java™ System Application Server Performance Tuning Guide』	お使いの Sun Java System アプリケーションサーバーから最適なパフォーマンスを得る方法について説明します。Sun Microsystems ドキュメント Web サイト (`http://docs.sun.com/app/docs`) から、必要なバージョンの本書をダウンロードしてください。
『Tuning Garbage Collection with the 5.0 Java Virtual Machine』	JVM を使用したガベージコレクションアプリケーションのチューニング方法について説明します。
『Turbo-charging Java HotSpot Virtual Machine, v1.4.x to Improve the Performance and Scalability of Application Servers』	`PrintGCStats` スクリプトのダウンロード方法と使用方法、および JVM チューニングを最適なものにするための統計を収集する方法について説明します。
『Understanding System Statistics』	Oracle の Cost-Based Optimizer がシステム統計を使用する仕組みについて説明します。
『Using JConsole to Monitor Applications』	JConsole を使用して、Java プラットフォームで実行中のアプリケーションの監視方法について説明します。

有用な Web サイト

Identity Manager パフォーマンスをチューニングする際に有用と思われる Web サイトをいくつか次の表に示します。

表 4–2 有用な Web サイト

Web サイト URL	説明
http://sunsolve.sun.com/	診断ツール、フォーラム、機能と記事、セキュリティー情報、パッチの内容を含む Sun の Web サイト。 注意: このサイトの情報は、次の 3 分野に分かれています。 ■ 社内。Sun 従業員のみ。 ■ 契約。契約アクセス権をお持ちのお客様のみ。 ■ 公開。すべての人に公開。
http://forum.java.sun.com/	フォーラムを参照したり、質問を投稿したりできる SDN (Sun Developer Network) Web サイト。
http://jrat.sourceforge.net/	Java プラットフォーム用オープンソースのパフォーマンスプロファイラである Java Runtime Analysis Toolkit の使用方法を説明する JRat Web サイト。
https://metalink.oracle.com/	Oracle データベースのチューニング情報を記載した、Oracle の社内フォーラムサイト。 注意: このサイトに記載されている情報にアクセスするには、Oracle Metalink の購読者になる必要があります。
http://performance.netbeans.org/howto/jvmswitches/index.html	パフォーマンスの向上に役立つ JVM スイッチのチューニング情報が記載された NetBeans™ Web サイト。
https://sharespace.sun.com/gm/folder-1.11.60181?	Sun の Share Space の Identity Manager リンク。 注意: このサイトに記載されている情報にアクセスするには、Share Space ID を登録する必要があります。
https://sharespace.sun.com/gm/document-1.26.2296	Sun の Share Space の Identity Manager FAQ。 注意: この FAQ にアクセスするには、Share Space ID を登録する必要があります。
http://www.slamd.com/	SLAMD Distributed Load Generation Engine の Web サイト。
http://www.opensolaris.org/os/community/dtrace/	OpenSolaris Community: DTrace の Web ページ。

表 4–2 有用な Web サイト (続き)

Web サイト URL	説明
http://www.solarisinternals.com/ http://docs.sun.com/app/docs/prod/solaris.10	Solaris OS のチューニングに関する情報が記載されている Web サイト。

チューニングのロードマップ

Identity Manager のソリューションのパフォーマンスは、次の配備固有の設定によって決まります。

- リソース構成
- 接続しているリソースの数
- 接続しているリソースの種類
- 属性がリソースにマッピングされている方法
- リソースの正確なバージョン
- ネットワークトポロジ
- ドメインコントローラの数と分散
- インストールした Identity Manager ゲートウェイの数
- 同時設定の数
- 並行プロセスの数 (実効ワークフローの数)
- 同時に接続しているユーザーの数
- 同時に接続している Identity Manager 管理者の数
- 管理対象となるユーザーの総数

パフォーマンス問題をデバッグする際には、まず問題を解析して記述してください。次の設問に答えます。

- パフォーマンスに問題がある場所はどこですか。調整、ワークフロー、カスタムワークフロー、GUI ページの読み込み、プロビジョニング、アクセスレビューうちのどれかですか。
- CPU 境界、メモリー境界、リソース境界、ネットワーク境界のうちのどれかにあてはまりますか。
- 問題に対して設定は確認しましたか (ハードウェア、ネットワーク、パラメータなど)。
- 使用中の設置環境で、最近何か変更したものはありますか。
- 本質的に煩雑なリソースのプロファイルを試みて、問題がリソース側にあって Identity Manager 側にはないかどうか確認しましたか。
- ビューのサイズはいくつですか。
- 複数のリソースにプロビジョニングしていませんか。
- 低速ネットワーク上のリソースが Identity Manager に接続されていませんか。

- Identity Manager を搭載したサーバーで、余分なアプリケーションを実行していませんか。
- 組織に、ルール駆動型メンバーのルールが用意されていますか。
- 一貫したパターンがあるかどうかを実行して、一連のスレッドダンプを確認しましたか。

注 – 単一のスレッドダンプを見ただけでは、判断できないことがあります。

- 最近、トレースを起動しましたか。
- JVM ガベージコレクションを確認しましたか。
- メモリー管理に負荷が掛かる組織を追加しましたか。

配備環境のチューニング

ここでは、次のような配備環境のチューニングについて説明します。

- 34 ページの「Java EE 環境のチューニング」
- 36 ページの「アプリケーションサーバーのチューニング」
- 38 ページの「リポジトリデータベースのチューニング」

Java EE 環境のチューニング

ここでは、Java プラットフォームである Enterprise Edition (Java EE プラットフォーム) 環境の最適化に使用できるチューニング案をいくつか説明します。

これらのチューニング案は、一連の実験結果のうち、ユースケースのテストにおいてスループットにかなりの増加が確認されたものを基にしています。この向上は、JVM のサイズ変更と、ガベージコレクタの動作に影響を及ぼすスイッチによるものです。

注 – Java、JConsole、または JVM のチューニングの詳細は、表 4–1 および表 4–2 に記載されている Web サイトをご覧ください。

次の節では、Java EE 環境で Java と JVM をチューニングする方法について説明します。

Java のチューニング

Java パフォーマンスのチューニングの詳細、ベストプラクティス、および用例については、次の『Java Tuning White Paper』を参照してください。

http://java.sun.com/performance/reference/whitepapers/tuning.html

JVM のチューニング

この節に記載したチューニング案は、次のチューニングスクリプトを基にしています。これらのスクリプトは、Sun Java System Application Server の domain.xml ファイル (ドメイン設定ディレクトリにあり、通常は domain-dir/config です。) に追加されました。

- PrintGCStats – データマイニングのシェルスクリプト。verbose:gc ログからデータを収集し、ユーザーが指定した間隔でデータのサンプルを抽出して、ガーベジコレクションの中断時間、パラメータの算出、アプリケーションの実行時に渡るタイムラインの解析などの情報を表示します。

  注 – このスクリプトとガベージコレクション統計を使用して JVM チューニングを最適なものにする方法については、次の Web サイトを参照してください。

  http://java.sun.com/developer/technicalArticles/Programming/turbo/#PrintGCStats

  ---

- PrintGCDetails – より詳細なガベージコレクション統計が得られるシェルスクリプト。
- PrintGCTimeStamps – PrintGCDetails スクリプトを使用して収集したガベージコレクション統計にタイムスタンプ情報を追加するシェルスクリプト。

最良の JVM パフォーマンスを得られるようにするには、次の点を確認してください。

- 『Sun Identity Manager 8.1 リリースノート』の「サポートされているソフトウェアと環境」節に記載されている必須の Java バージョンを使用し、最新の機能、バグの修正、およびパフォーマンス拡張機能を使用していること。
- ガベージコレクションの最新バージョンを使用していること。

  アプリケーションサーバーをインストールしたときのデフォルトのガベージコレクションスキームである旧バージョンを削除していないことがよくあります。Identity Manager で旧バージョンのガベージコレクタを実行すると、多数のオブジェクトが生成されるため、JVM が絶え間なくガベージを収集することになります。
- Identity Manager を Sun Java System Application Server に配備している場合は、配備した Identity Manager インスタンスの server.xml ファイルにガベージコレクションの要素を追加すれば、スループットを高めることができます。

最大負荷に 300 人以上のユーザーが見込まれる場合は、次の設定を修正してパフォーマンスを高めます。

- 配備した Identity Manager インスタンスに HTTP リスナーを設定している場合は、`server.xml` ファイル内のリスナー定義要素を編集し、「ホスト上のアクティブ *CPU* 数」を「ホスト上のアクティブ *HTTP* リスナー数」で割った数をアクセプタスレッド数に設定します。

  たとえば、次のようにします。

  `<HTTP リスナー ID="HTTP リスナー-1" \アドレス="0.0.0.0" ポート="80" \アクセプタスレッド="アクセプタスレッドの算出結果" ...>`

- 大部分の Identity Manager 配備の静的コンテンツは頻繁に変更されるように設計されていないため、(「ファイルキャッシュ設定」ページにある) 静的コンテンツのファイルキャッシュ設定を編集しても構いません。コンテンツが再読み込みされる前のファイルキャッシュ内のコンテンツ最大有効期間に、大きな数値を指定します (24 時間単位の秒数など)。

  「ファイルキャッシュ設定」ページにアクセスするには、HTTP サーバーノード用の Web ベース管理者コンソールから「ファイルキャッシュ」タブをクリックします。(詳細な手順については、最新の『Sun Java System Web Server Administrator's Guide』を参照してください。)

注 – Sun Application Server では、チューニング可能なものを公開しています。これによって、HTTP コンテナで確保される各種スレッドプールと接続キューのサイズが変わります。

デフォルトでは、チューニング可能なものの大半は同時接続ユーザー数 300 人以下の負荷とされています。

アプリケーションサーバーのチューニング

アプリケーションサーバーのチューニングには、次のガイドラインが役立ちます。

- 36 ページの「Sun Application Server のチューニング」
- 38 ページの「WebSphere Application Server のチューニング」

注 – ヒープサイズ以外にも、大半のアプリケーションサーバーにはデフォルトのパラメータ設定が使用できます。ご使用中のリリースに応じて、サーバーのヒープサイズを変更してください。

Sun Application Server のチューニング

最新の『Sun Java System Application Server パフォーマンスのチューニングガイド』の「アプリケーションサーバーのチューニング」の章では、Sun Java System

Application Server のチューニング方法について説明しています。このドキュメントは、次の URL から入手できます。`http://docs.sun.com/app/docs`

また、Sun Application Server 8.2 Enterprise Edition をお使いの場合は、次のように変更すれば「並列モードのエラー」が解決され、パフォーマンスが改善され予測しやすくなるはずです。

- 古い世代のコレクションを常に実行している場合は、アプリケーションのヒープ面積を見直し、このサイズを増やしてみてください。たとえば、次のようにします。
  - 500M バイトは小さめなサイズですので、この値を 3G バイトに増やせば向上する場合があります。
  - 若い世代コレクションが 2G バイトの場合、スカベンジするたびに約 70M バイトをプロモートします。スカベンジを少なくとも 1 回行うと 70M バイトをプロモートするということを念頭に置いてください。たとえば、次の `SurvivorRatio` が必要になることがあります。

    `2 G バイト/70 M バイト X 2 = 4096/70 = 55`

    ここで、次のように設定します。

    ```
    -XX:SurvivorRatio=32 -XX:MaxTenuringThreshold=1
    ```

    この比率に設定すれば、早すぎるプロモートを防ぐことができ、またスカベンジパフォーマンスの低下の原因となりうる「偏り」も防ぐことができます。
- `-XX:CMSInitiatingOccupancyFraction=60` と設定しておけば、CMS コレクションがこのしきい値に達するまで起動したままになります。たとえば、次のようにします。

  ```
  56402.647: [GC [1 CMS-initial-mark: 265707K(1048576K)] 1729560K(3129600K),
  3.4141523 secs]
  ```

  `-XX:CMSInitiatingOccupancy=60` を削除し (デフォルト値の 69% を使用)、次の行を加えます。

  ```
  -XX:UseCMSInitiatingOccupancyOnly
  ```
- 若い世代のコレクションが 2G バイトで、古い世代のコレクションが 1G バイトの場合でも、早すぎる CMS コレクションが行われてしまうことがあります。この比率を逆転してみてください。次のように、若い世代のコレクションを 1 G バイト、古い世代のコレクションを 2 G バイトに設定します。

  ```
  -Xms3G -Xmx3G -Xmn1G
  ```

  また、`-XX:NewRatio` を削除します。若い世代と全体のヒープサイズを明示的に指定していた場合は、この比率は余分です。
- Java 開発キット (JDK™ ソフトウェア) の `5uXX` バージョンを使用していて、過度に長すぎる「abortable preclean」サイクルがある場合は、一時的な回避策として `-XX:CMSMaxAbortablePrecleanLoops=5` を使用することができます。

この値は、さらに調整する必要があります。

- ガベージコレクタのパフォーマンスの詳細を表示するには、次の行を追加します。

  ```
  -XX:+PrintHeapAtGC
  ```

  注 – 冗長的なガベージコレクションデータの生成量が増えるので、このコマンドは慎重に使用してください。ガベージコレクタの出力の保存先に十分なディスク空き容量があることを確認してください。

- Sun Fire™ T2000 サーバーを使用している場合は、DTLB (Data Translation Look-aside Buffer) のヒープが大きいとリソースが不足してしまうことがあります。Java ヒープに大きなページを使用することでパフォーマンスに役立つことがよくあります。たとえば、次のようにします。

  ```
  -XX:+UseLargePages
  ```

WebSphere Application Server のチューニング

IBM WebSphere® アプリケーションサーバーに搭載した Identity Manager をチューニングする場合、ヒープメモリーはスレッドに使用するメモリーに影響するため、ヒープに割り当てるメモリー量を制限するようにしてください。

多数のスレッドが同時に作成されてヒープサイズが増えると、アプリケーションのディスク空き容量限度を直ちに超えて、次のエラーが返されます。

```
JVMCI015:OutOfMemoryError
```

リポジトリデータベースのチューニング

Identity Manager は、ID と構成データの保存と管理にリポジトリデータベースを基にしています。このため、データベースのパフォーマンスが Identity Manager のパフォーマンスに大きく影響することがあります。

注 – データベースのパフォーマンスチューニングと管理の詳細は、データセットやベンダーによって異なるので、このドキュメントでは説明しません。また、データベース管理者 (DBA) は、すでにお使いのデータベースに精通している必要があります。

ここでは、データベースの計画、チューニング、および管理に役立つよう、Identity Manager アプリケーションの特徴を説明し、Identity Manager のデータとその典型的な使用法パターンの概要を説明します。

この情報は、次の節で構成されています。

リポジトリテーブルの種類

Identity Manager のリポジトリには、3 種類のテーブルがあり、テーブルごとに使用方法の特徴が多少異なります。このテーブルについての情報は、次の節で構成されています。

属性テーブル

属性テーブルを使用すると、定義済みの単一値または複数値のオブジェクト属性を照会できます。

大半のオブジェクト型のストアド属性はハードコードされます。

注 – `User` オブジェクト型と `Role` オブジェクト型は、この規則における例外です。`User` と `Role` オブジェクトテーブルに格納されるインライン属性は構成可能なため、追加のカスタム属性をクエリー可能として設定できます。

属性状態を基にオブジェクトを検索すると、Identity Manager は対応するオブジェクトテーブルを使用して結合内の属性テーブルにアクセスします。結合の形式 (`JOIN`、`EXISTS` 述語、`SUB-SELECT` など) によっては、属性の状態ごとに発生するものもあります。

属性テーブルの行数は、対応するオブジェクトテーブルの行数に比例しています。この値は傾斜 *(スキュー)* の形に分布することがあります。複数値の属性には値ごとに行があります。オブジェクトの中にはほかと比べて属性や属性値が多いものもあります。通常、属性テーブルの行とオブジェクトテーブルの行には、多対 1 の関係があります。

属性テーブルのテーブル名は `ATTR` です。

変更テーブル

Identity Manager では、対応するオブジェクトテーブルに加えた変更内容の履歴を追跡するのに変更テーブルを使用します。これらのテーブルのサイズは、オブジェクトの変更率に比例しますが、際限なく増えることはありません。Identity Manager は、自動的に変更テーブルを切り捨てます。

変更テーブルは、行の有効期間が比較的短く新しい行が頻繁に作成されるため、変動率が高くなることがあります。

変更テーブルへのアクセスは、通常、タイムスタンプフィールドの範囲スキャンで実行されます。

変更テーブルのテーブル名は CHANGE です。

オブジェクトテーブル

Identity Manager のリポジトリは、オブジェクトテーブルを使用してラージオブジェクト (LOB) などの直列化データオブジェクトを保持します。オブジェクトテーブルでは、よく問い合わされる単一値のオブジェクト属性も保持できます。

大半のオブジェクト型のストアド属性はハードコードされます。

注 – User オブジェクト型と Role オブジェクト型は、この規則における例外です。オブジェクトテーブルに格納されるインライン属性は構成可能なため、User と Role は、追加のカスタム属性をクエリー可能として設定できます。

オブジェクトテーブル内の行数と、格納されるオブジェクト数は一致します。オブジェクトテーブルごとに格納されるオブジェクト数は、そのテーブルに格納されるオブジェクト型によって決まります。オブジェクト型によっては多数のものもあれば、少数のものもあります。

Identity Manager は通常、オブジェクト ID またはオブジェクト名を使ってオブジェクトテーブルにアクセスしますが、テーブル内に格納されている属性のうちの 1 つを使ってアクセスすることもできます。オブジェクト ID とオブジェクト名は単一のオブジェクト型をとおして一意ですが、属性値は一意ではなかったり均等に分布されていたりします。属性によっては値が多いものもあれば、比較的少ないものもあります。さらに、複数のオブジェクト型で同じ属性を使用することもあります。属性の中には、あるオブジェクト型に対する値は多く、別のオブジェクト型に対する値は少ないものもあります。値の分布が均一でないと、インデックスページの分布も均一でなくなり、スキューと呼ばれる状態になります。

オブジェクトテーブルは、テーブル名に ATTR または CHANGE 接尾辞が付かないテーブルです。

XML 列

オブジェクトテーブルにはそれぞれ 1 つの XML 列があり、LOG テーブルセットを除く各直列化オブジェクトを格納するのに使用されます。特定の LOG テーブルセットのオプションの属性は、それが存在する場合には XML 列に格納されます。たとえば、デジタル署名が有効に設定されている場合などがあります。

データクラス

Identity Manager のデータは、アクセスパターン、カーディナリティー、有効期間、データの変更頻度などの点で類似したプロパティーを持つクラスにほぼ分類できます。リポジトリのテーブルセットに対応しているクラスは、次のとおりです。

- 41 ページの「ユーザーデータ」
- 41 ページの「ロールデータ」
- 42 ページの「アカウントデータ」
- 42 ページの「コンプライアンス違反データ」
- 42 ページの「権限付与データ」
- 43 ページの「組織データ」
- 43 ページの「タスクデータ」
- 43 ページの「構成データ」
- エクスポートキューデータ44 ページの「エクスポートキューデータ」
- 44 ページの「ログデータ」

ユーザーデータ

ユーザーデータは、ユーザーオブジェクトから構成されます。

管理対象 ID ごとにオブジェクトがあるため、このデータはかなり大きくなることが予想されます。大部分の演算は既存オブジェクトを更新したものになるため、最初に生成してからは、作成されるものはあまりないはずです。

ユーザーオブジェクトは有効期間が長いものが多く、削除される確率は比較的低くなります。

ユーザーデータは、USEROBJ テーブル、USERATTR テーブル、および USERCHANGE テーブルに格納されます。

ロールデータ

ロールデータは、ビジネスロール、IT ロール、アプリケーション、アセットなどのロールのサブタイプを含む Role オブジェクトから構成されます。

ロールデータは組織データと似ており、お客様が Identity Manager を配備した後、このオブジェクトは比較的固定されます。

注 – ただし、権限ロールセットがある外部ソースと一体化した配備は例外です。たとえば、一体化型にすると Identity Manager にロール変更の入力が必要になり、Identity Manager の Role データが変動しやすくなります。

(複数のユーザーが各ロールを共有しているとすれば) ロールオブジェクト数は、ユーザーなどの ID オブジェクト数よりも少ないと言えますが、これは企業のロール定義の仕方によって異なります。

ロールデータは、ROLEOBJ テーブル、ROLEATTR テーブル、および ROLECHANGE テーブルに格納されます。

アカウントデータ

アカウントデータは、Account Index のアカウントオブジェクトからのみ構成されます。

ユーザーデータと同様、既知のリソースアカウントごとにオブジェクトがあるため、アカウントデータは大きくなりがちです。アカウントオブジェクトは、一般的に有効期間が長く、削除される確率は比較的低く、最初に生成されてからは、作成されることはあまりありません。ネイティブアカウントの追加や削除を頻繁に行わない限り、またはネイティブ変更の検出を有効に設定しない限り、アカウントオブジェクトの変化はあまり発生しません。

Identity Manager は、アカウントデータを ACCOUNT テーブル、 ACCTATTR テーブル、および ACCTCHANGE テーブルに格納します。

コンプライアンス違反データ

コンプライアンス違反データには、監査ポリシーの評価に失敗した時期を表す違反レコードが記載されます。この違反レコードは、同じユーザーに対して同じ監査ポリシーが評価され、評価に合格するまで保持されます。違反レコードは、監査ポリシーのスキャンまたはアクセスレビューの一環として作成、変更、または削除されます。

違反レコードの数は、スキャンで使用される監査ポリシー数とユーザー数に比例します。インストールユーザー数が 5000、監査ポリシー数が 10 とすると、違反レコードは 500 (5000 x 10 x 0.01) と考えられます。この乗数 0.01 は、ポリシーの厳密度とユーザーアカウントの変更の仕方によって変わります。

Identity Manager は、コンプライアンス違反レコードを OBJECT テーブル、ATTRIBUTE テーブル、および OBJCHANGE テーブルに格納します。

権限付与データ

権限付与データは、コンプライアンスのアクセスレビューを実行する場合にのみ作成される、ユーザー権限付与オブジェクトから主に構成されます。

権限付与レコードは大きなバッチに作成され、最初の作成から (何日もかけて) ゆっくりと変更され、その後は手つかずのままになります。このレコードは、アクセスレビューが削除されてから削除されます。

Identity Manager は、権限付与データを ENTITLE テーブル、ENTATTR テーブル、および ENTCHANGE テーブルに格納します。

組織データ

組織データは、オブジェクトグループまたは組織オブジェクトから構成されます。

オブジェクトグループデータは構成データと似ており、このデータは配備された後は比較的固定されます。一般的に、タスクオブジェクトと比べたり、ユーザーやアカウントなどの ID オブジェクトと比べるとオブジェクト数は (定義されている組織を単位とすると) 少ないですが、ほかの構成オブジェクトと比べるとこの数の方が多くなることがあります。

組織データは、ORG, ORGATTR テーブルと ORGCHANGE テーブルに格納されます。

タスクデータ

タスクデータは、状態と結果データを含むタスクとワークフローに関するオブジェクトから構成されます。

オブジェクトが高速で作成、変更、および削除されるため、これらのテーブルにあるこのデータの有効期間は、ほかのクラスと比べると短いです。このテーブルのデータ量は、使用システムの動作量に比例します。

タスクデータは、TASK テーブル、TASKATTR テーブル、TASKCHANGE およびテーブルに格納されます。

構成データ

構成データは、フォーム、ロール、ルールなどの Identity Manager のシステム構成に関するオブジェクトから構成されます。

構成データの特徴は、一般的に次のとおりです。

- ほかのクラスと比べると比較的小さい。
- 変更は配備中とアップグレード中にのみ、大きいバッチで行われる。
- 配備された後は、あまり変更されることがない。

Identity Manager は、構成データを ATTRIBUTE テーブル、OBJCHANGE テーブル、および OBJECT テーブルに格納します。

エクスポートキューデータ

データのエクスポートを有効に設定すると、レコードの中には、エクスポートタスクがレコードをデータウェアハウスに書き込むまで、Identity Manager のキューに入れられるものがあります。キューに入れられるレコードの数は、データエクスポートの構成と、キューの中のすべての型に対するエクスポート間隔と相関関係にあります。

デフォルトでキューに入れられるデータ型は次のとおりです。これ以外のデータ型はすべてキューに入れられません。

- `ResourceAccount`
- `WorkflowActivity`
- `TaskInstace`
- `WorkItem`

これらのテーブルのレコード数は、エクスポートタスクがキューを排出するまで増えます。最新のテーブルサイズは、JMX™ Bean から表示できます。

このテーブルに追加されたレコードは、変更されることがありません。これらのレコードは、調整、プロビジョニング、ワークフローの実行など、ほかの Identity Manager 動作中に書き込まれます。このデータエクスポータがタスク実行をエクスポートすると、タスクがテーブルを排出します。

Identity Manager は、エクスポートキューデータレコードを `QUEUE` テーブル、`QATTR` テーブル、および `QCHANGE` テーブルに格納します。

ログデータ

ログデータは、監査オブジェクトとエラーログオブジェクトから構成されます。ログデータは 1 度限り書き込めるので、新規の監査オブジェクトとエラーログオブジェクトは作成できても、これらのオブジェクトを変更することはできません。

ログデータは明示的要求によってのみ消去できるため、ログデータの有効期間は非常に大きくなりがちです。ログデータに頻繁にアクセスするかどうかは、属性テーブル内ではなくオブジェクトテーブル内に格納されている属性によって決まります。ログに対する属性値の分布もクエリーも、具体的には Identity Manager の使い方によって決まります。

たとえば、ログテーブル内の属性値の分布は、次によって決まります。

- どの種類の変更が加えられたか。
- どちらの Identity Manager インタフェースから変更が加えられたか。
- どのオブジェクト型が変更されたか。

ログテーブルに対するクエリーのパターンは、お客様がそのログテーブルに対して実行する Identity Manager レポート、カスタムレポート、または外部データマイニングのクエリーによっても異なります。

Identity Manager は、監査ログレコードを LOG テーブルと LOGATTR テーブルに、そしてエラーログレコードを SYSLOG テーブルと SLOGATTR テーブルに格納します。このデータには、対応する変更テーブルがありません。

オブジェクト ID

Identity Manager は、JDK ソフトウェアに用意されている VMID クラスでオブジェクトにグローバル一意識別子 (GUID) を生成します。

これらの GUID 値は、オブジェクトが作成された順に基づいて、格納されたプロパティーを文字列表現で表します。たとえば、Identity Manager でオブジェクトを新規作成すると、古い方のオブジェクトより新しい方のオブジェクトに大きいオブジェクト ID が付きます。このため、Identity Manager がデータベースに複数の新しいオブジェクトを挿入すると、オブジェクト ID に基づいたインデックスが同じブロックを競合することがあります。

準備済み文

Identity Manager は、準備済み文を動作 (データベース行の挿入や更新など) に使用することはあっても、クエリーに使用することは通常ありません。

Oracle® を使用している場合は、この動作のためにライブラリキャッシュに問題が生じることがあります。特に、文のバージョンが多数存在するとライブラリキャッシュのラッチに競合が起こることがあります。

この競合を解決するには、Oracle CURSOR_SHARING パラメータ値を EXACT から SIMILAR に変更します。この値を変更すると、SQL 文のリテラルが Oracle によってバインド変数に置き換えられるため、バージョンの数が減ります。

文字コードセットとエンコーディング

Identity Manager はバイトよりも文字データを通常読み書きする Java アプリケーションのため、データベースが用いるエンコーディングは制限されません。

Identity Manager は、データが正しく送信および返信されることだけを要求します。たとえば、書き込み時と再読み取り時にデータが破損することはありません。複数バイト文字をサポートし、データに合ったエンコーディングを使用してください。通常は UTF-8 エンコーディングで十分ですが、日本語やアラビア語などの真の複数バイト文字を多数用いる企業では、UTF-16 の方が良いかもしれません。

大多数のデータベース管理者が複数バイト文字をサポートしているエンコーディングの方を望む理由は、次のとおりです。

- 配備が拡張され、国際文字セットのサポートが必要になることが多い。
- ASCII のように見えるが実際には全角ダッシュ (–) のような複数バイトを含んだ Microsoft アプリケーションのテキストをカット＆ペーストすることがある。

リポジトリデータベースのチューニングのガイドライン

ここでは、リポジトリデータベースをチューニングする際のガイドラインをいくつか説明します。

- データベース管理者は、統計を頻繁に実行してリポジトリデータベースの現状を監視する必要があります。
- データソースを使用している場合は、RepositoryConfiguration オブジェクトの connectionPoolDisable 属性を true に設定して、Identity Manager リポジトリの内部接続プールの自動化を無効に設定します。

  たとえば、<RepositoryConfiguration connectionPoolDisable='true'> と設定すると (Identity Manager 用とアプリケーションサーバー用に) 接続プールが 2 つにならないようにできます。
- RepostioryConfiguration オブジェクトを編集して、固有の単一値属性に対する検索パフォーマンスを向上させることもできます。たとえば、このオブジェクトを編集して、ユーザー検索に使用したり相関キーとして使用する employeeID などの拡張属性を追加してください。

  デフォルトの RepositoryConfiguration オブジェクトは、次の例のようになります。

```
<RepositoryConfiguration ... >
   <TypeDataStore Type="User" ... attr1="MemberObjectGroups",
attr2="lastname" attr3="firstname" attr4="" attr5="">
   </TypeDataStore>
</RepositoryConfiguration>
```

注 – 省略符号は、ここには無関係の XML 属性を表しています。

---

attr1、attr2、attr3 、attr4、および attr5 の XML 属性はそれぞれ、waveset.userobj テーブルにコピーすべき単一値属性を示しています。waveset.userobj テーブルは、最大 5 つまでのインライン属性を格納できます。RepositoryConfiguration 内の attr1 という属性値は、このテーブルの "attr1" データベース列にコピーされます。

インライン属性は、(子の attribute テーブル列としてではなく) Type の基本 object テーブルに格納されます。

インライン属性を使用すると、属性に対するリポジトリのクエリーのパフォーマンスが向上します。(インライン属性はメインの「object」テーブルにあるため、インライン属性に対するクエリーの方が、子の「attribute」テーブルに格納されているインライン属性以外に対するクエリーよりも高速です。インライン属性以外に対するクエリー条件には、属性テーブルに「結合」が必要になります。)

デフォルトでは、Identity Manager が MemberObjectGroups、lastname、および firstname のインライン属性を使用します。

- クエリーに使用できる属性であれば、検索を高速化するためにもう 2 つ属性を追加しても構いません。

  たとえば、配備に `employeeID` の拡張属性がある場合、この属性インラインを追加すれば、その属性に対するリポジトリ検索のパフォーマンスが向上します。
- `lastname` や `firstname` が不要な場合は、削除したりほかの属性と代えても構いません。
- `MemberObjectGroups` は削除しないでください。Identity Manager は、承認チェックの高速化にこの属性を内部で使用します。

各テーブルセットに格納されるオブジェクト型の詳細は、41 ページの「データクラス」を参照してください。

ベンダー固有データベースのチューニングに関するガイドライン

ここでは、Oracle と SQL Server のリポジトリデータベースをチューニング際のベンダー固有のガイドラインについていくつか説明します。

注 – 現在、開発とデモ用にサポートされているのは、MySQL™ データベースだけです。

Oracle データベース

ここでは、Oracle リポジトリデータベースをチューニングする際のガイドラインについて説明します。

- Identity Manager アプリケーションは、Oracle データベースの機能やオプションを必要としません。
- Oracle リポジトリデータベースと Identity Manager サービスプロバイダ や Identity Manager を使用する場合、Identity Manager は `LOB` データ型ではなく `LONG` データ型をデフォルトで使用するため、オブジェクトテーブルの断片化の問題に遭遇するかもしれません。`LONG` データ型を使用すると、使用可能スペースに作成できないエクステントスペースの「未割り当て」量が増えることがあります。

  この問題を抑制するには、次の項目を実行します。

  - Object テーブルの `EXPORT` をダンプし、インポートし直して未割り当てのエクステントスペースを解放します。インポートしたら、データベースを終了して再起動する必要があります。
  - `LOB` データ型と、Oracle に標準の `LOB` 実装を提供する DataDirect Technologies の Merant ドライバを使用します。
  - 自動的に空きスペースを管理するローカルマネージメントテーブルスペース (LMT) を使用します。この LMT は、Oracle 8.1.5 で使用できます。

Identity Manager は、SGA サイズ変更、バッファサイズ変更、オープンカーソル、プロセスなどに Oracle の `init.ora` パラメータ設定を必要としません。

- Identity Manager リポジトリは汎用的なデータベースでありながら、まさにオブジェクトデータベースと言えます。

  Identity Manager テーブルのうち、`TASK` テーブルセットが最もトランザクション処理の特徴を備えています。`LOG` と `SYSLOG` テーブルセットは直列化オブジェクトを格納しないため、これらも例外です。

  テーブルの説明、各テーブルに格納されるオブジェクト型、および各テーブルの一般的なアクセス方式については、39 ページの「リポジトリテーブルの種類」と 41 ページの「データクラス」を参照してください。

- Oracle データベースにパフォーマンスの問題がある場合は、Identity Manager では比較的効率がよいと思われるクエリーに対して選択されているクエリー計画に問題がないか調べてください。

  たとえば、インデックスが使用可能な場合、Identity Manager はテーブルの完全スキャンが実行されるように設定されています。こういった問題は、自動ワークロードリポジトリ (AWR) レポートにある SQL の `buffer gets` テーブルによく見られます。この問題は、Enterprise Manager ツールでも表示することができます。

  パフォーマンス問題は、データベーステーブルの統計が不良か紛失した場合によく起こります。この問題を解決すれば、データベースと Identity Manager の両方のパフォーマンスを向上させることができます。

  次の記事は Oracle から入手でき、Oracle のコストベースオプティマイザ (CBO) を調べるのに役立ちます。

  - 『Understanding System Statistics』
  - 『Oracle MetaLink: Note:114671.1: Gathering Statistics for the Cost Based Optimizer』
  - 『Cost Control: Inside the Oracle Optimizer』
  - 最適なクエリー計画を選択するもう一つの方法として、SQL プロファイルを用いた調査も行えます。パフォーマンスの悪い SQL を特定する際には、Enterprise Manager から SQL Advisor を使用してこれらのプロファイルを作成します。

  Oracle 再実行ログに予想外の増加が検出された場合は、手動操作で無限ループに陥っているワークフローがあるかもしれません。ループによってリポジトリが絶えず更新され、その結果 `TaskInstance` ごとのサイズが大幅に増加してしまいます。ワークフローのエラーは、`WF_ACTION_TIMEOUT` を不適切に取り扱ったり、ワークフローの途中でブラウザを閉じたりすると起こります。

  問題のあるワークフローを回避するには、本番稼動前に各手動操作のプレビューを行い、次の点を検証します。

  - タイムアウトを設定しているか。
  - 手動操作で用いる動作に、タイムアウトの処理に合った遷移ロジックを作成しているか。

- TaskInstance に大量のデータがある場合、手動操作に公開変数タグを使用しているか。

CURSOR_SHARING パラメータ値を EXACT から SIMILAR に変更すると、Identity Manager のパフォーマンスを大幅に向上できることがよくあります。

Identity Manager は、準備済み文をいくつかの動作 (データベース行の挿入や更新など) に使用することはあっても、これらの文をクエリーに使用することはほぼありません。

Oracle を使用するとき、この動作によってライブラリキャッシュに問題が生じることがあります。特に、文のバージョンが多数存在するとライブラリキャッシュのラッチに競合が生じることがあります。CURSOR_SHARING を SIMILAR に変更すると、Oracle が SQL 文内のリテラルをバインド変数に置き換えるため、バージョンの数を大幅に減少できます。

詳細は、45 ページの「準備済み文」を参照してください。

SQL Server データベース

SQL Server 2000 データベースをリポジトリとして使用している顧客より、並行処理が増加するときに SQL Server の内部で悲観的 (Pessimistic) ロックを使用すること (主にロックエスカレーション) によるデッドロックの問題が SQL Server 2000 に発生するという報告がありました。

これらのデッドロックエラーは、次の形で表示されます。

```
com.waveset.util.IOException:
  ==> com.microsoft.sqlserver.jdbc.SQLServerException: Transaction (Process ID 51)
was deadlocked on lock | communication buffer resources with another
process and has been chosen as the deadlock victim. Rerun the transaction.
```

このデッドロック問題を回避または解決するには、次の項目を実行します。

1. SQL Server 2005 データベースを使用します。
2. 次のようにコマンドを初期化して、READ_COMMITTED_SNAPSHOT パラメータを設定します。

   ```
   ALTER DATABASE waveset SET READ_COMMITTED_SNAPSHOT ON
   ```

   READ_COMMITTED_SNAPSHOT パラメータを有効にすると、次のようになります。

   - SELECT 文の実行中にブロックの原因となりうる競合がなくなります。そのため、SQL Server 内部へのデッドロックの危険性が大幅に減少します。
   - 不確実なデータが読まれなくなります。そのため、SELECT 文が確実なデータの一貫したビューを受信できるようになります。

   READ_COMMITTED_SNAPSHOT パラメータの詳細は、http://msdn2.microsoft.com/en-us/library/ms188277.aspx を参照してください。

Identity Manager パフォーマンスのチューニング

Identity Manager のパフォーマンスを最適化するための推奨は、次の領域に分かれています。

一般的なパフォーマンスのチューニング

通常は、次のとおり実行すると Identity Manager のパフォーマンスを最適化できます。

- トレースを無効にする (Java クラス、userform、ワークフローのトレースなど)。トレースによって、かなりのオーバーヘッドが追加されます。
- Identity Manager 内蔵の監査ログの管理タスクとシステムログの管理タスクを実行して、ログレコードの有効期限を設定する。ログレコードは際限なく増えてしまうことがあるため、これらのタスクを使用してリポジトリデータベースが空きスペースを使い切ってしまうことを防ぎます。詳細は、『Sun Identity Manager 8.1 ビジネス管理者ガイド』を参照してください。
- Identity Manager の更新 (以前は サービスパック または インストールパック と呼ばれていました) に含まれている `README` ファイルをチェックして、製品にパフォーマンス改善が施されていないか調べる。改善が施されていれば、アップグレードを計画します。
- Identity Manager リポジトリなど、1 台以上のリモートシステムからデータを取得する際のパフォーマンスの影響を考慮する。

- Identity Manager を実行中のアプリケーションサーバーのインスタンス数を同一サーバー上で、またはサーバーを追加して増やし、負荷分散ツールを使用してインスタンス間に要求を分散させる。
- バイナリ属性で参照されているファイルのサイズをできる限り小さく保つ。たとえば極端に大きいグラフィックファイルを読み込むと、Identity Manager のパフォーマンスが低下することがあります。
- たとえばリファクタリングを行なって重複を抑え、メモリーを効率的に活用し、システム全体のパフォーマンスに対する影響を軽減するような、堅牢で理解しやすい XML コードを記述する。
- イベントをリアルタイムに追跡するよう、Identity Manager のシステム監視を設定する。

  これらのイベントはダッシュボードグラフに表示されるため、システムリソースをすばやく評価して異常を見つけ、時刻や曜日などを基にこれまでのパフォーマンスの傾向を把握することで、監査ログを調べる前に問題を対話的に特定することができます。計器盤では監査ログほど詳細がわかりませんが、ログのどこを見れば問題が見つかるかのヒントがわかります。

  計器盤の詳細は、『Sun Identity Manager 8.1 ビジネス管理者ガイド』の第 8 章「レポート」を参照してください。

Active Sync アダプタのパフォーマンスのチューニング

同期はバックグラウンドタスクであるため、Active Sync アダプタ設定によってはサーバーのパフォーマンスが影響を受ける可能性があります。

Active Sync アダプタの管理には、「リソース」リストを使用します。Active Sync アダプタを選択し、「リソースアクション」リストの「同期」セクションから処理を制御する実行、停止、ステータス更新を利用してください。

Active Sync アダプタのパフォーマンスを向上するには、次のとおり実行します。

- 実行中の動作の種類に応じて、ポーリング間隔を評価して調節します。

  ポーリング間隔は、Active Sync アダプタが新しい情報の処理を開始する時期を決定します。たとえば、アダプタがデータベースから多数のユーザーのリストを読み込み、毎回 Identity Manager の全ユーザーを更新する場合、この処理を毎日早朝に実行することを検討してください。アダプタによっては処理する新しい項目を即座に検索するため、毎分実行するよう設定できるかもしれません。
- リソースの同期ファイルを編集し、アダプタの実行先ホストを指定します。

  多くのメモリーと CPU サイクルを必要とする Active Sync アダプタを専用サーバーで実行するように設定すると、システムの負荷を分散させることができます。

- 適切な管理者権限があれば、Active Syncリソースを変更してActive Syncアダプタを無効にしたり、手動または自動的に開始したりできます。

  アダプタを自動に設定すると、アプリケーションサーバーを起動したときにアダプタが再起動します。アダプタを開始すると、アダプタは指定のポーリング間隔で即座に実行されます。アダプタを終了すると、次回アダプタが終了フラグを調べたときに終了します。
- 同期ログから取り込まれた詳細レベルを調節します。

  同期ログは、現在処理中のリソースに関する情報を取り込みます。それぞれのリソースには、独自のログファイル、パス、およびログレベルがあります。アダプタログから取り込まれる詳細の量は、指定したログレベルに応じて変わります。この値は、該当するユーザータイプ(Identity Managerまたはサービスプロバイダ)の「同期ポリシー」の「ログ設定」セクションから指定します。

一括ロードのチューニング

一括ロード操作中のパフォーマンスを向上させるには、次のとおり実行します。

- たとえばActive Sync、一括動作、調節などの一括処理動作の処理時間を改善するため、承認サブプロセスへの呼び出しを削除して、デフォルトのワークフローを単純化します。
- 管理者に割り当てられているユーザーフォームをできる限り単純化します。たとえば、次のようにします。
  - データのロードにフォームを作成する際、データ表示のためのコードがあれば削除します。
  - 一括追加操作を使用する場合は、`firstname`や`lastname`といった基本属性がCSVファイルに定義されていることを確認します。次に、管理者ユーザーフォームからこれらの属性を削除します。

注 – Identity Managerに用意されているデフォルトのフォームは変更しないでください。代わりに、そのフォームのコピーを作り、そのコピーに一意の名前を付け、この名前を変更したコピーの方を変更します。この方法により、アップグレード中でも製品アップデート中でもカスタマイズしたフォームが上書きされなくなります。

フォームの作成方法と編集方法の詳細は、『Sun Identity Manager Deployment Reference』の第2章「Identity Manager Forms」を参照してください。

- 次の機能を、NIS(ネットワーク情報サービス)実装先の配備環境に実装します。

- `user_make_nis` という名前のアカウント属性をスキーママップに追加し、この属性を調整やその他の一括プロビジョニングワークフローで使用します。この属性を追加した場合、リソース上の各ユーザーが更新された後は、システムで NIS データベースへの接続手順がバイパスされます。
- プロビジョニングが完了してから NIS データベースに変更内容を書き込むには、ワークフローに `NIS_password_make` という ResourceAction を作成します。

設定を変更可能な XML オブジェクト設定のチューニング

設定可能な XML オブジェクトを使用すると、広範囲のユーザーインタフェース仕様が利用でき、タスクごとにユーザーへのデータ表示方法を定義したり、複雑な業務プロセスを自動化したりできるようになります。ただしこの柔軟性は、効率、パフォーマンス、および信頼性に影響を与えることがあります。

ここでは、フォーム、ルール、およびワークフローから構成される、Identity Manager の設定可能な XML オブジェクトをチューニングする際のガイドラインをいくつか説明します。これらの情報は、次のように構成されています。

- 53 ページの「フォームのチューニング」
- 56 ページの「ルールのチューニング」
- 56 ページの「ワークフローのチューニング」

フォームのチューニング

実行中タスクのビューや変数コンテキストと相互に作用するインタフェースを定義するには、Identity Manager のフォームを使用します。フォームには、データ要素のセットにビジネスロジックと変換ロジックに使用する実行コンテキストがあります。各種タスクを実行する非常に強力で動的なフォームを作成できるとはいえ、フォームの複雑さを抑えれば効率が上がります。

次に、カスタマイズしたフォームのパフォーマンスを向上させる方法をいくつか説明します。

- 54 ページの「新しいフォームの最適化」
- 54 ページの「管理者フォームの最適化」
- 55 ページの「エンドユーザーフォームの最適化」
- 55 ページの「フォームフィールド内の式の最適化」

新しいフォームの最適化

Identity Managerの新しいフォームを設計する際、システムインテグレーターは次の手順に従えばフォームのパフォーマンスを最適化できます。

- 「負荷が大きい」クエリーは、できる限り一度だけ実行する。このクエリーを最小限に抑えるには、次の項目を実行します。
  - `<Field>` `<Default>` 要素を使用して、クエリーを実行し結果を格納します。
  - これ以降のフィールドの値を参照するには、フィールド名を使用します。
  - カスタムタスクの場合
    - `ManualAction` の前にこのタスク内の値を計算してから、タスク変数にこの値を格納します。
    - このフォーム内の変数を参照するには、`variables.tmpVar` を使用します。
    - `ManualAction` の後に変数をクリアするには、`<setvar name='tempVar'/>` を使用します。
- 計算には、初期表示と更新ごとに実行される `<defvar>` を使用します。

管理者フォームの最適化

管理者フォームのパフォーマンスを向上させるには、次の項目を実行します。

- `TargetResources` を指定して、特定のリソースのみを取得して編集します。(詳細は、56 ページの「ワークフローのチューニング」を参照してください。)
- `FormUtil.getResourceObjects` または `FormUtil.listResourceObjects` を使用している場合は、変更頻度の低いオブジェクトに `cacheList` と `cacheTimeout` のキャッシュパラメータを使用します。
- 時間の掛かる計算とフェッチの結果は `<Field>` 要素に格納し、`<Default>` にある式で評価することで、演算が一度だけ行われるようにします。
- `update.constraints` を使用して、実行時に取得されるリソースを制限します(『Sun Identity Manager Deployment Reference』の「Dynamic Tabbed User Form」を参照してください)。
- バックグラウンドの承認を使用 (`ManualAction` の各所有者に 1 秒のタイムアウトを設定) して、ページ送信の高速化を図ります。
- 選択したパネルにかかわらず、ページの再読み込み時に Identity Manager が「タブパネルフォーム」のすべてのパネルに定義されているすべてのフィールドを必ず更新するようにしてください。

エンドユーザーフォームの最適化

エンドユーザーフォームのパフォーマンスを向上させるには、次の項目を実行します。

- `TargetResources` を使用して、ビューのチェックアウトを対象となるリソースアカウントのものだけに制限します。これにより、ビューのフェッチ時間が短縮され、`TaskInstance` と `WorkItems` で消費されるメモリーが少なくて済むようになります。
- Identity Manager のユーザーオブジェクトのビュープロパティーと属性だけが対象となっている場合、`WSUser` を返す際に `Session.getObject(Type, name)` の使用を検討します (複数の延期タスクのトリガーの管理に役立ちます)。
- 通常はエンドユーザーのタスクの方がプロビジョニングタスクよりも `WorkItems` が多いため、エンドユーザーのタスクは`WorkItem` サイズに特に影響を受けやすくなります。
- 「ビュー」をチェックアウトして構築し、編集したあとでビュー全体にマージし直してチェックインする際には、一時的な汎用オブジェクトを使用することを検討します。
- デフォルトの「ユーザーの作成」と「ユーザーの編集」インタフェースを使用する代わりに、スケーラブルフォームを使用するようにします。

  ユーザーを編集する際にデフォルトのユーザーフォームを使用すると、ユーザーのアカウントを編集し始めた時点で Identity Manager はそのユーザーが所有しているリソースを取得します。多数のリソースを抱えるユーザーアカウントがある配備環境では、この時間の掛かりがちな操作によってパフォーマンスが低下する危険性があります。

フォームフィールド内の式の最適化

フォームで実行される動作の中には、Identity Manager 外部のリソースを呼び出すものがあります。特にグループリストや電子メール配信リストのコンパイルなど、結果の値が長いリストになるような外部リソースへのアクセスは、Identity Manager のパフォーマンスに影響を与えることがあります。

このような呼び出し中のパフォーマンスを向上させるには、『Sun Identity Manager Deployment Reference』の「Java クラスを使用したフィールドデータの取得」のガイドラインに従ってください。

また、`<Disable>` 式のようなパフォーマンスに影響を受けやすい式で JavaScript™ を使用するのは避けてください。組み込まれたトレース機能を使用する場合は、短い XPRESS 式の方がデバッグしやすいです。ワークフローアクションの複雑なロジックには JavaScript を使用します。

フォームの表示速度が低下した場合、debug/Show_Timings.jsp ページで問題を見つけ出すことができます。Formconvert.convertField() への呼び出しを探してください。これで、フィールドごとにその値の計算にどれくらい時間が掛かっているかがわかります。

ルールのチューニング

Identity Manager ルールを使用して、本製品の中でフォームやワークフローなど設定可能なコンポーネントで再利用できる定数と XPRESS ロジックをカプセル化します。

ルールを記述する際は、最適なパフォーマンスを得られるよう、必要に応じて次のガイドラインを使用してください。

- 定数値を返すには、静的な宣言を使用します。
- 増分された値や一度だけ参照される値には、defvar メソッドを使用して一時的な値でアルゴリズムを実装します。
- 複数回返すべき値がある複雑で負荷のかかる計算には、putmap、setlist、または setvar メソッドを使用します。最終的には値を <null> に設定します。

ワークフローのチューニング

さまざまな人的および電子的タッチポイントを使用した複雑な業務処理を、使いやすく自動化できるように Identity Manager のワークフローをカスタマイズします。

カスタムのワークフローのパフォーマンスを向上させるには、次の方法に従ってください。

- 処理時間を改善するため、承認サブプロセスへの呼び出しを削除して、デフォルトのワークフローを単純化します (特に、Active Sync、一括動作、調節などの一括処理動作)。
- ワークフロー内に無限ループがないことを確認します。特に、承認サブプロセス内にあるループで、ブレークフラグが更新され適切に確認されるようにします。
- 複数回同じオブジェクトのリポジトリを参照する必要がある場合は、取得したオブジェクトを後から使用できるよう変数に置きます。

  Identity Manager はすべてのオブジェクトをキャッシュしないため、変数の使用が必要です。
- WorkflowServers の checkoutView の TargetResources オプションで、アカウント情報のクエリー対象となるリソース数を制限します。

  アカウント情報のクエリー対象となるリソース数を制限する方法については、次の例をご覧ください。

```
<Argument name='TargetResources'>
  <list>
    <string>resource name[| #]</string>
```

```
  </list>
</Argument>
```

注 – 上の例の [| #] は、特定のリソースに複数アカウントがあるときに使用できるオプションのパラメータです。たいていの場合、リソース名で十分です。

- フォームで残った不要なビュー変数を消去します。特に大きなマップとリストを消去します。たとえば、次のようにします。

  ```
  <setvar name='myLargeList'></null></setvar>
  ```

  このビューは、TaskInstance オブジェクトで複数回コピーされるため、大きなビューでは TaskInstance とそれに対応する TaskResult のそれぞれのサイズが著しく増加します。

- 完了したタスクをすぐに廃棄するよう、TaskDefinition の resultLimit (数秒内) を使用するか、タスク実行中のこのオプションを設定します。TaskInstances の数が多いと、次に影響が出ます。
  - フッター内の taskResults.jsp と管理者インタフェース内の JSP™ タスクの表示方法
  - JSP タスクの表示方法
  - タスク名を変更する際の TaskInstance ごとのクエリー
  - データベースのサイズ

  次のオプションを必要に応じて設定します。
  - (選択を推奨) delete — 新しいタスクの実行が開始される前に、同一名の古い方の TaskInstance を削除します。
  - wait — 古い方の TaskInstance が削除されるか resultLimit に達して期限が切れるまで、現在の TaskInstance を一時停止します。
  - rename — 命名が競合しないように、TaskInstance 名にタイムスタンプを挿入します。
  - terminate — 同一名の古い方の TaskInstance を削除します。現在実行中の同一名の TaskInstance がそれぞれ中止されます。

WorkItems (ManualActions) のチューニング

WorkItems (ワークフロー内では ManualActions と表示されています) の数とサイズによって、メモリーとシステムパフォーマンスが著しく影響を受けることがあります。デフォルトでは、WorkItem に Identity Manager が全体ワークフローのコンテキストをコピーしてから、送信後にこのワークフローコンテキストを書き戻します。

WorkItems と ManualActions のパフォーマンスを向上させるには、次の手順に従ってください。

- WorkItems のサイズを減らします。

  デフォルトでは、ManualAction が WorkItem を作成してから、そのタスクコンテキストのそれぞれの変数を WorkItem.variables にコピーします。タスクコンテキスト変数を制限すると、並列承認から上書きされなくなります。

  - WorkItem へコピーし直される変数を制限するには、ExposedVariables を使用します。たとえば、次のようにします。

    ```
    <ExposedVariables><List><String>user ...
    ```

  - WorkItem から実行中のタスクへ解析し直される変数を制限するには、EditableVariables を使用します。たとえば、次のようにします。

    ```
    <EditableVariables><List><String>user ...
    ```

    承認フラグ、フォームボタンの値、および実際の承認者名を必ず含めてください。

- 確認ページとバックグラウンド処理を変更し、ユーザーインタフェースの応答時間を向上させます。
  - 設定プログラムなどの別ユーザーが所有している、確認の ManualAction またはバックグラウンドの ManualAction を作成します。
  - タスクが実行されて WorkItems が削除されてからユーザーがアクションを送信する場合にエラーメッセージが表示されないようにするには、timeout='-5'(5 秒) と ignoreTimeout='true' を設定します。
- 値のマップやリストなどの大きな属性値を送信時に null に設定してメモリー使用を最適化します。または、スコープからすばやく排出されるようにそれらを Activity スコープ指定変数としてインスタンス化します。
- 完了したタスクの有効期間を短くします。
  - 各 WorkItem に Timeout を指定してそのワークフローが Timeout を WorkItem ごとに予測するようにして、デッドエンドのタスクがなくなるようにします。
  - タスク完了後のスケジューラによるタスクの処理方法がわかるよう、TaskDefinition の resultLimit と resultOption オプションを使用するようにします。
    - ❒ resultLimit を使用して、タスク完了後のタスクの有効期間を秒単位で制御します。デフォルトのサイズはゼロ (0) です。つまり、タスク完了後にタスクインスタンスが即時削除されます。
    - ❒ resultOption を使用して、タスクの繰り返しインスタンスが開始されたときに行うべき動作を指定します (wait、delete、rename、または terminate など)。デフォルトは delete です。

      正常に完了したタスクを即時削除するとはいえ、デバッグできるまでエラーを含むタスクを残しておくには、条件付きで完了したタスクを削除することもできます。

TaskDefinition の resultLimit に、問題をデバッグするのに十分な期間を設定します。WorkflowServices 呼び出し後に、実行時にエラーが何も報告されれば(WF_ACTION_ERROR が <null/> など)、resultLimit を 0 (または 小さい値) に設定できます。

- うまくスコープ指定されていない変数を評価して修正します。次のように、宣言された場所に応じて変数をスコープ指定します。
  - **Global** 変数は、多くの動作 (ケース所有者やビューなど) に使用され、サブプロセス内で承認フラグとして使用されるべき値です。

    <TaskDefinition> の要素として宣言されている変数があれば、その変数をグローバルにスコープ指定してください。external に宣言されている変数があれば、その値はスタックの呼び出し時に解決されます。
  - 負荷の掛かる値の **Activity** 変数 (リソースのフェッチを必要としたり、大きなリストやマップの値を格納する Activity 変数など) が、WorkItem で参照されることがあります。

    <Activity> 要素として宣言されている変数があれば、その変数を Activity 内の動作と遷移要素に表示されるようにします。

    注 – Identity Manager Version 2005Q3M1 SP1 以降は、WorkItem の作成に値がコピーされないよう、ワークフローではなくフォームに <Activity> 変数値を使用してください。

  - **Activity** 変数は、遷移ロジックで使用される値です。

    <Action> の要素として宣言されている変数があれば、動作の完了時にその変数がスコープを配るはずです。Action 変数は通常、アプリケーションから設定された変数名 (View → User など) を「変更」するために WorkflowApplication の呼び出しで使用されます。
- ウィザードのワークフローの最終ページには、同期実行 (syncExec='true') を指定しないでください。

  true に設定すると、ユーザーがタスクを実行したときに、そのタスクが完了するまで実行されます。そのタスクが完了するか、別の停止点 (別の ManualAction など) に達するまでインタフェースがハングしてしまいます。
- 不要な承認チェックを削除します。

  注 – Active Sync には、viewOptions.Process から指定されたシステム指定のプロビジョニングタスクの代わりに、能率的なプロビジョニングタスクを使用します。

- Identity Manager に用意されているプロビジョニングタスクは変更しないでください。

(リストされていないタスクである限り、) タスクを新規作成してから、フォームと処理マッピング構成の中でそのタスクを指定します。

データベース統計のチューニング

データベース管理者の場合、頻繁に統計を実行してリポジトリデータベースを監視しているはずです。

パフォーマンス問題は、データベーステーブルの統計が悪いか抜けている場合によく起こります。この問題を解決すれば、データベースと Identity Manager パフォーマンスの両方のパフォーマンスが向上します。

詳細は、次の Oracle 記事を参照してください。

- 『Understanding System Statistics』
- 『Oracle MetaLink: Note:114671.1: Gathering Statistics for the Cost Based Optimizer』
- 『Cost Control: Inside the Oracle Optimizer』

最適なクエリー計画を選択するもう一つの方法として、SQL プロファイルを使用するようにします。パフォーマンスの悪い SQL を特定する際には、Enterprise Manager から SQL Advisor を使用してこれらのプロファイルを作成します。

データエクスポータのチューニング

データエクスポータを使用すると、Identity Manager の新規データ、変更データまたは削除データを、レポート作成と解析作業に適した外部のリポジトリにエクスポートできるようになります。実際のエクスポートデータはバッチで行われます。ここで、エクスポート対象となる各種データが独自のエクスポート周期を指定できます。Identity Manager リポジトリに付属しているエクスポート対象のデータと、エクスポート周期の長さと変更されるデータ量にもよりますが、エクスポートされるデータ量は大きくなりがちです。

Identity Manager のデータ型の中には、後からエクスポートできるように、特殊なテーブルのキューに入れられるものがあります。具体的に言うと、`WorkflowActivity` と `ResourceAccount` データがキューに入れられます。こうしないとこのデータが持続しないためです。型に加えられたすべての変更点をウェアハウスで監視する必要があったり、`TaskInstance` や `WorkItem` データなど、エクスポート周期に対応しないライフサイクルの型があると、どの持続性のデータ型でもキューに入れられることもあります。

パフォーマンスを最大限に高めるには、ウェアハウスで必要なデータ型だけをキューに入れてエクスポートしてください。データエクスポートはデフォルトで無効になっていますが、データエクスポートを有効にすると、すべてのデータ型がエクスポートされます。不要なデータ型はすべて無効にしてください。

エクスポートタスクがデータをエクスポートする際、このタスクは複数スレッドを使用して、できる限り多くのスループットを達成するためにエクスポートを一刻も早く完了しようとします。Identity Manager リポジトリとウェアハウスの入出力速度にもよりますが、export タスクが Identity Manager サーバーのプロセッサを完全に占有し、これが対話式パフォーマンスが低下する原因になります。できればエクスポートは、エクスポートタスクに専用のマシンで実行するか、少なくともそのマシンに対話式処理がない期間に実行するようにします。

エクスポートタスクでサポートしているチューニングパラメータは、次のとおりです。

- 読み取りブロックサイズのキュー化
- 書き込みブロックサイズのキュー化
- 排出スレッドカウントのキュー化

排出スレッドカウントは、最も需要なスループットです。キューテーブルに多数のレコードがある場合、(最大 24 まで) スレッド数を増せばスループットが高まるようになります。ただし、そのキューがある種のレコードに占有されていると、実際に高速化できる排出スレッドは少なくなります。エクスポートタスクは、割り当てできるスレッドのセット数までキューテーブルの中身を分けて、それぞれのスレッドに排出対象のセットを入れようとします。これらのスレッドには、ほかのリポジトリテーブルを排出している排出スレッドも含まれます。

一般的な XML のチューニング

静的な XMLObject 宣言を可能な限り使用すれば、一般的な XML を最適化できることがよくあります。たとえば、以下を使用します。

- <list> の代わりに <List>
- <s> の代わりに <String>
- <map> の代わりに <Map><MapEntry ...></Map>

また、コンテキストにもよりますが、<o></o> 要素の代わりにラップオブジェクトを使用するようにします。

Identity Manager Service Provider のチューニング

Identity Manager の計器盤グラフを使用すると、Sun™ Identity Manager サービスプロバイダ (サービスプロバイダ) の現在のシステムを評価し、異常を見つけ、これまでの傾向を理解することができます (時間枠内の並列ユーザーやリソースの動作など)。

注 – サービスプロバイダには、管理者インタフェースがありません。Identity Managerの管理者インタフェースから、管理者タスクのほぼすべてを実行してください (計器盤グラフなど)。

サービスプロバイダのチューニングの詳細は、『Sun Identity Manager Service Provider 8.1 Deployment』を参照してください。

Identity Manager Webインタフェースのチューニング

Identity Managerの Webインタフェースを使用している場合は、Identity Managerに同梱されている OpenSPMLツールキットでパフォーマンスを最適化できます。

注 – `openspml.jar` ファイルを `http://openspml.org/` の Webサイトから使用すると、メモリーリークが起こります。

初期ロードのチューニング

大きな初期ユーザーのロード中にパフォーマンスを向上させるには、次の項目を実行します。

1. Identity Managerの管理者インタフェースから、すべての監査イベントを無効にします。

   注 – 監査ロギングを使用すると操作ごとに複数のレコードが追加され、それ以降の監査レポートの実行速度が低下することがあります。

   a. 「設定」→「監査」の順に選択します。
   b. 「監査の設定」ページで、「監査を有効にする」ボックスの選択を解除して「保存」をクリックします。
2. Webサーバをシャットダウンしてリストキャッシュを無効にするか、(`debug/Show_WSProp.jsp` デバッグページにある) `ChangeNotifier.updateSearchIntervalCount` プロパティーを **`0`**.. に変更して、リストキャッシュを無効にします。

   リストキャッシュには、メモリー内で頻繁にアクセスされる組織内のユーザーリストが保持されています。これらのリストを保持するため、リストキャッシュは新規作成されたユーザーを探して確認します。

3. debug/ Clear_List_Cache.jsp ページにある最新のリストキャッシュを消去します。
4. ユーザーの処理に使用されるワークフローに、承認が記載されていないことを確認します。
5. 次のように、代替となるロード方法を使用します。
   - ロードを分割し、そのデータをゾーンで実行する。
   - より高速な一括ロードを使用する。
   - ファイルからロードを行う。
6. WorkflowActivity 型のデータエクスポータを無効にします。

必要メモリーのチューニング

Java コマンドラインへの最大ヒープサイズと最小ヒープサイズを追加して、必要メモリーを定め、アプリケーションサーバーの JVM に値を設定します。たとえば、次のようにします。

```
java -Xmx512M -Xms512M
```

パフォーマンスを向上させるには、次の項目を実行します。

- 最大と最小のヒープサイズの値に同じ値を設定します。
- 調整を行う場合は、お客様固有の実装に応じてこの値を増やしても構いません。
- パフォーマンスチューニングの目的上、次を waveset.property ファイルに設定することもできます。

  ```
  max.post.memory.size value
  ```

注 – max.post.memory.size は、ディスクへスプールせずに (たとえば HTML FileSelect コントロールで) 送信されたファイルに含まれている最大バイト数を指定します。一時ファイルへの書き込み権がない場合は、max.post.memory.size を増やして、ディスクへスプールしないようにします。デフォルト値は 8K バイトです。

システム要件の詳細は、『Sun Identity Manager 8.1 リリースノート』を参照してください。

オペレーティングシステムのカーネルのチューニング

Solaris™ と Linux オペレーティングシステムのカーネルのチューニングの詳細は、『Sun Java System Application Server Enterprise Edition Performance Tuning Guide』の「Tuning the Operating System」の章を参照してください。

Oracleオペレーティングシステムのカーネルのチューニングについては、Oracleシステムに付属の製品マニュアルを参照してください。

プロビジョニングツールのチューニング

ビューのプロビジョニングを処理する際のパフォーマンス問題は、ネットワーク遅延に原因があることがよくあります。個々のリソースアダプタをトレースすれば、何がパフォーマンス問題の原因になっているかがわかりやすくなります。

プロビジョニングツールのパフォーマンスを向上させるには、次の項目を実行します。

- `Waveset.properties` ファイルの `provisioner.maxThreads` を設定して、リソース処理ごとにスレッドが起動しているような、スレッドをプロビジョニングしている同時アカウントの数を制御します。

  通常、この値を **`10`** に設定すると最適なパフォーマンスを達成できます。20を超える値を指定すると、プロビジョニングツールのパフォーマンスが著しく低下します。

- `Waveset.properties` ファイルの割り当て設定を行い、指定タスクにユーザーが実行できる並列処理(再プロビジョニングなど)の数を制御します。並列動作の数を増やすと、より多くの処理が高速に完了できるようになりますが、一度に多すぎる動作を処理しようとするとボトルネックの原因になりかねません。

  設定セットは、プール単位で作成できます。たとえば、設定A、設定B、設定Cを作成する場合、`TaskDefinition`(ワークフロー)作成する際に、定義済みの設定から特定のプール設定をワークフローに割り当てられます。

  次の例は、1つの再プロビジョニングタスクを一度に実行するユーザーのバイナリオブジェクト(`BOB`)を制限する、割り当て設定を示したものです。

  ```
  Quota.poolNames=ReProvision,Provision
  Quota.pool.ReProvision.defaultLimit=1
  Quota.pool.ReProvision.unlimitedItems=Configurator
  Quota.pool.ReProvision.items=bob,jan,ted
  Quota.pool.ReProvision.item.bob.limit=1
  ```

  タスクの割り当てを強制するには、`TaskDefinition` の `poolName` を参照します。このフォーマットは、次のとおりです。

  ```
  <TaskDefinition ... quotaName='{poolName}'..>
  ```

  ほとんどのユーザーが一度に実行するタスクは1つだけです。調整を行ったりActive Syncタスクを実行するプロキシ管理者の場合、このタスクの割り当てを大きく設定します。

注 – 調整や Active Sync タスクは、設定プログラムのユーザーが使用しないようにしてください。設定プログラムは、無限のタスクにアクセスでき、使用可能なリソースを独占できるため、並列プロセスに悪影響を及ぼしかねません。

調整のチューニング

調整サーバーとは、調整を行う Identity Manager のコンポーネントです。ここでは、以下を初めとする調整サーバーのパフォーマンスを向上させるための推奨方法を説明します。

- 65 ページの「調整のチューニングで用いる一般的な推奨事項」
- 66 ページの「調整サーバーの設定のチューニング」
- 67 ページの「複数リソースに用いる調整のチューニング」

調整のチューニングで用いる一般的な推奨事項

通常は、次のとおり実行すると調整サーバーのパフォーマンスを最適化できます。

- 調整タスクは、設定プログラムのユーザーが使用しないようにしてください。設定プログラムは、無限のタスクにアクセスでき、使用可能なリソースを独占できるため、並列プロセスに悪影響を及ぼしかねません。

  代わりに、調整と Active Sync タスクには、能率化された最小限のユーザーを設定してください。タスクを実行する対象はタスクの一部として連続しているため、最小限のユーザーにすれば、タスクごとおよびリポジトリ内で更新するたびにスペースやオーバーヘッドの消費が少なくて済みます。
- 調整サーバーの状態ページ (`debug/Show_Reconciler.jsp`) の説明を基に、キューサイズ別、利用可能なシステムリソース別、およびパフォーマンスのベンチマーク別に調整すべき設定を決定してください。これらの設定は、環境に応じて変わります。
- 利用可能な総メモリー量と空きメモリー量については、システムメモリー概要ページ (`debug/Show_Memory.jsp`) をご使用ください。調整はメモリーを多く使う機能のため、この説明を使用すれば JVM に十分なメモリーが割り当てられてないかどうかが判断できます。また、ガベージコレクションの起動や、ヒープ使用量を調査するための JVM 内の未使用メモリーの消去にも、このページが役に立ちます。
- 調整を行うプロキシ管理者にユーザーフォームを割り当てるときは、ユーザーフォームをできる限り単純に保ち、必須フィールドのみを使用します。スキーママップよって変わりますが、`waveset.organization` 属性を計算するフィールドなどは十分間に合います。

注 – ユーザーとロールにIdentity Managerスキーマを表示したり編集する必要がある管理者は、IDM スキーマ設定管理者グループに設定し、IDM Schema Configuration権限を付与する必要があります。

- アカウント単位のワークフローを慎重に使用します。調整プロセスは、パフォーマンス目的でプロビジョニングタスクを開始することはデフォルトでありません。

  アカウント単位のワークフロータスクを使用しなければならない場合は、調整ポリシーを編集して、調整サーバーの自動応答を対象イベントのみに制限してください。(「調整ポリシーの編集」ページの「状況」領域を参照してください。)

調整サーバーの設定のチューニング

デフォルト設定で適切な場合が多いですが、「サーバー設定の編集」ページにある次の設定を調節すると、調整サーバーのパフォーマンスを向上できることがあります。

- 「並列リソース制限」。調整サーバーが並列処理できるリソーススレッドの最大数を指定します。

  リソーススレッドは作業項目をワークスレッドに割り当てます。 このため、リソーススレッドを追加する場合、ワークスレッドの最大数を増やすことが必要になることがあります。
- 「最小ワークスレッド」。調整サーバーが常にオープンを保つ処理スレッド数を指定します。
- 「最大ワークスレッド」。調整サーバーが使用できる処理スレッドの最大数を指定します。調整サーバーは、ワークフローが要求するスレッドの数だけ起動します。この数には制限があります。ワークスレッドは、短時間アイドル状態が続くと自動的に閉じます。

アイドル中は、スレッドに行うべき処理がなく、指定した最小限のスレッド数にまで低下した場合に限り、スレッドが停止します。ロードが増えるに従って、最大スレッド数に達するまで、調整サーバーはさらにスレッドを追加します。調整サーバーは、最小限のスレッド数を下回ったり最大限のスレッド数を上回ることはありません。

通常、スレッドが増えると並列処理も増えます。ただし、多すぎるスレッドによっていつしかマシンに大きすぎる負荷が掛かったり、単に効果が得られなくなることがあります。

注–配備はそれぞれ異なるため、汎用的で最適な設定を推奨することはできません。調整サーバーの設定は、配備環境ごとに個別に調整する必要があります。

▼ 調整サーバーの設定を変更する

調整サーバーの設定を変更するには、次の項目を実行します。

1. 管理者インタフェースにログインします。
2. 「設定」→「サーバー」→「調整サーバー」タブの順にクリックします。
3. 「サーバー設定の編集」ページが表示されたら、必要に応じて設定を調整します。
   詳細は、20 ページの「サーバーのデフォルト設定の編集」を参照してください。

複数リソースに用いる調整のチューニング

Identity Manager で複数リソースに調整を設定する場合は、いくつかオプションがあります。

- このサーバーに搭載されているすべてのリソースを一括して

  このオプションは、Identity Manager から見ると一番効果的ですが、リソースが多いと (20 台を超えるなど)、Java リソースの問題が生じやすくなります。
- このサーバーに搭載されているすべてのリソースを個別に

  このオプションは、Java リソースのローディング時に楽になりが、スケジュール設定に大きな負荷がかかります。
- 各サーバーに搭載されているリソース別に一括して

  このオプションは、経過時間を最小限に抑えますが、サーバー数をが大きくなります。

配備はそれぞれ異なるため、この設定には最適な解決策がありません。配備に合った解決策を見つけるには、これらのオプションを組み合わせて調整する必要があります。

この機能の業務目的を基に使用量アンケートを作成すると、進め方が決定しやすくなるかもしれません。

次の問題に取り組みます。

- これらのリソースを調整している理由は。
- どのリソースにも同じ目標があるか。
- どのリソースも等しく重要または重大か。
- すべてのリソースを同じスケジュールで調整しなくてはならないか。調整をかち合わないようにすることはできるか。

- 各リソースを調整すべき頻度は。

また、調整サーバーは、Webトラフィックを処理するプールに含める必要はありません。このサーバーはとトランザクション処理専用にあるため、決して直接対話しないようなサーバーを追加してください。トランザクション処理専用のサーバーを用意すると、大規模システムには一番上のオプションが最適かもしれません。

リソースクエリのチューニング

注 – ビューをプロビジョニングする際のパフォーマンス問題は、ネットワーク遅延に原因があることがよくあります。個々のリソースアダプタをトレースすれば、何がパフォーマンス問題の原因になっているかがわかりやすくなります。

クエリーを実装する際に `FormUtil.getResourceObjects` を使用すると、リソースクエリーのパフォーマンスを向上させることができます。

クエリー結果のキャッシュに、次のうちいずれかの方法を用います。

- `getResourceObjects(Session session, String objectType, String resID, Map options, String cacheList, String cacheTimeout, String cacheIfExists)`
- `getResourceObjects(String subjectString, String objectType, String resId, Map options, String cacheList, String cacheTimeout, String clearCacheIfExists)`

注 –

- `cacheTimeout` をミリ秒単位で設定します。
- 具体的な `searchContext` があれば、その検索を絞り込みます。
- `options.searchAttrsToGet` に属性の最小数を返します。

スケジューラのチューニング

スケジューラコンポーネントは、Identity Managerのタスクのスケジュール作成を管理します。

ここでは、以下を初めとするスケジューラのパフォーマンスを向上させるための推奨方法を説明します。

- 69 ページの「スケジューラをチューニングするための一般的な推奨事項」
- 70 ページの「スケジューラサーバーの設定のチューニング」

スケジューラをチューニングするための一般的な推奨事項

次の TaskDefinition オプションによって、タスク完了後のスケジューラによるタスクの処理の仕方が決まります。

- resultLimit — タスク完了後のタスクの実行期間を秒単位で制御します。デフォルト設定は、タスクごとに異なります。0 を設定すると、タスクは完了後に即時削除されます。
- resultOption — タスクの繰り返しインスタンスが開始されたときに行うべき動作を制御します。デフォルト設定は delete で、余分なタスクのインスタンスが削除されます。

これらのデフォルト設定は、完了したスケジューラのタスクの有効期間を短縮することでメモリーを最適化できるようにするためのものです。これらの設定を変更せざるをえない事情がない限り、デフォルトのままにしておいてください。

正常に完了したタスクを即時削除するとはいえ、デバッグできるまでエラーを含むタスクを残しておくには、次を実行することもできます。

- resultLimit 値を、問題をデバッグするのに十分な期間に設定して、条件付きで完了したタスクを削除します。
- WorkflowServices 呼び出し後に、実行時にエラーが何も報告さければ (WF_ACTION_ERROR is <null/> など)、resultLimit を 0 (または 小さい値) に設定します。

スケジューラサーバーの設定のチューニング

「サーバー設定の編集」ページで次の設定を調整すると、スケジューラのパフォーマンスを向上できることがあります。

- 「最大同時タスク数」。スケジューラが一度に実行できるタスクの最大数を指定します。

  並列タスクの最大数の設定の許容値よりも多くのタスクが実行できる場合は、それ以上のタスクは空きができるまで、または別のサーバーで実行されるまで待機しなければなりません。

  メモリーが足りなくなりCPU時間を共有するほど多すぎるタスクがスワップしていると、オーバーヘッドによってパフォーマンス速度が低下します。また、最大数を小さく設定しすぎても、アイドル時間が増えます。スケジューラは、待機中のタスクが1分以内に実行できるように使用可能なタスクを毎分チェックします。

  デフォルトの並列タスクの最大数設定(`100`)で通常は十分です。配備で実行中のタスクを基に、もしくは配備が完了した後の実行時間の動作を調べて、この設定の増減を調整するかどうかを決定します。

  場合によっては、このスケジューラを一時停止したり無効に設定しても構いません。たとえば、エンドユーザーインタフェースの処理専用にするサーバーがある場合、スケジューラを無効にすれば、そのサーバーでタスクの実行が行われないようになります。そのサーバーは、エンドユーザーインタフェース専用になり、ほかのサーバーに実行するよう開始したタスクが格納されます。
- 「タスク指定」。このサーバーで実行できるタスク一式を指定します。

  この「タスク指定」設定を使用すると、サーバーで実行できるタスクをきめ細かく制御できるようになります。個別にタスクを制限することも、サーバー設定で制限することも可能です。

注–配備はそれぞれ異なるため、汎用的で最適な設定を推奨することはできません。スケジューラの設定は、配備環境ごとに個別に調整する必要があります。

▼ スケジューラサーバーの設定を変更する

1. 管理者インタフェースにログインします。
2. 「設定」→「サーバー」→「スケジューラ」タブの順にクリックします。
3. 「サーバー設定の編集」ページが表示されたら、必要に応じて設定を調整します。

   詳細は、20ページの「サーバーのデフォルト設定の編集」を参照してください。

セッションのチューニング

Identity Manager は、認証済みセッションで認証されたユーザーが使用できる最長時間未使用の (LRU) キャッシュを保持します。既存の認証済みセッションを使用すると、セッションを必要とするオブジェクトとアクションに対してリポジトリアクセスの速度をアップすることができます。

認証プールサイズを最適化するには、`Waveset.properties` ファイルの `session.userPoolSize` 値を、そのサーバーで見込まれる並列ユーザーセッションの最大数に変更します。

Sun Identity Manager Gateway のチューニング

Sun Identity Manager Gateway は、接続ごとにスレッドを生成して、リソース種類、ゲートウェイホスト、およびゲートウェイポートの一意の組み合わせごとに異なるプールを使用します。ゲートウェイは、5 分ごとにアイドル中の接続がないかチェックします。60 分間アイドル中の接続が合った場合、ゲートウェイはその接続を閉じてプールから削除します。

ゲートウェイは要求を受け取ると、次の操作を行います。

- そのプールにアイドル中の接続がなければ、ゲートウェイは接続を新規作成します。
- そのプールにアイドル中のプールがあれば、ゲートウェイはその接続を検索して再利用します。

そのリソースの最大接続数を設定してください。また、その種類のすべてのリソースに対し、そのゲートウェイを使用しているのと同じ方法でこの接続を設定してください。このリソース種類では、所定のホストでゲートウェイに行った最初の接続とポートが、そのリソースの最大接続数を使用します。

注 – そのリソースの最大接続数を変更する場合は、サーバーを再起動して変更内容を反映してください。

次の例は、接続、要求、およびゲートウェイスレッドの関係を示したものです。

Active Directory リソースで最大接続数を 10 に設定し、2 台の Identity Manager サーバーを使用している場合は、その Active Directory リソースのゲートウェイに最大 20 までの同時接続を設定できます (Identity Manager サーバーごとに 10)。このゲートウェイは、各サーバーから送信された 10 の同時要求を受け付けることができ、各要求を別々のスレッドで処理します。同時要求数がゲートウェイの最大接続数を超えた場合、それ以上の要求はゲートウェイが要求を完了して接続をプールに戻すまで、キューの中に入れられます。

注 – ゲートウェイコードはマルチスレッドですが、この特性はゲートウェイで使用中の API やサービスに対応しません。Active Directory には、ゲートウェイは Microsoft から提供されている ADSI インタフェースを使用します。このインタフェースがゲートウェイの要求を並列に処理するかどうかを判断する調査は一切行われていません。

このほか、ゲートウェイのパフォーマンスを向上させる方法には、次のようなものがあります。

- ゲートウェイを、(ネットワーク接続の観点から) 管理対象ドメインのドメインコントローラの近くに配備します。
- ゲートウェイリソースのブロックサイズを大きく設定すると、調整やロード操作中のスループットを高めることができます。

  スループットが高まる結果は、カスタムワークフローがなく、実行中の属性調整が一切ない基本的な調整で説明したとおりです。最初はゲートウェイが多くのシステムメモリーを消費しますが、このメモリーは次第に解放されていきます。

  これは先細りしていくことをご了承ください。ある時点を境に、ブロックサイズを大きく設定してもパフォーマンスがそれほど向上しなくなります。たとえば次のデータは、Active Directory リソースから 10,000 ユーザーのリソースからの読み込みを監視した速度を表したものです。読み込み中のゲートウェイ処理には、ピークメモリー使用量も含まれています。

ブロックの設定	時間単位で作成されたユーザー	ゲートウェイのピークメモリー使用量
100	500	20 M バイト
200	250	25 M バイト
500	9690	60 M バイト
1000	10044	92 M バイト

- Exchange Server 2007 では、`PowerShellExecutor` が Exchange Server 2007 の動作を実行します。次のレジストリ設定を修正すれば、ゲートウェイ中の `PowerShellExecutor` の動作を変更できます。

  注 – どちらの設定も、ゲートウェイの動作とメモリー使用量に大きな影響を及ぼすことができます。これらのパラメータに加える変更内容は、慎重にテストを行った上で検討してください。

  – `powerShellTimeout`

- ❒ 「内容」。PowerShell 動作 (レジストリ型 REG_DWORD) のタイムアウト
- ❒ 「デフォルト」。60000 ミリ秒 (1 分)

  powerShellTimeout 設定がタイムアウトすると、PowerShell 環境内の異常動作を防ぐために、すべての RunSpace 動作が中断され取り消されます。これにより、ゲートウェイが反応しなくなります。

  powerShellTimeout 値を小さい値に下げると、動作が途中で取り消され、RunSpace の初期化が正しく完了しなくなることがあります。プール範囲の最初の RunSpace の監視起動時間は、2 秒から 5 秒までです。

  powerShellTimeout 値は、起動時に読み取り専用のため、ゲートウェイを再起動しないとこの値は変更できません。

– runSpacePoolSize
  - ❒ 「内容」。プール内の RunSpaces 数 (レジストリ型 REG_DWORD)
  - ❒ 「デフォルト」。5
  - ❒ 「最小」。5
  - ❒ 「最大」。25

  ゲートウェイから PowerShell 動作を並列実行できるプール内の RunSpaces 数。Exchange 2007 でのユーザーのプロビジョニング動作または更新が、実行中の複数の PowerShell 動作につながることがあります。

  起動した RunSpace が、大量のメモリーを消費することがあります。最初の RunSpace の通常サイズは、約 40 M バイトです。その後の RunSpaces は、通常 10 M バイトから 20 M バイトの間を消費します。

  注 – 以上の数値は環境ごとに異なるため、ガイドラインとしてのみ記載したものです。この値を変更する際はご注意ください。

  ---

  runSpacePoolSize 値は、起動時に読み取り専用のため、ゲートウェイを再起動しないとプールサイズ値は変更できません。

タスクバーのチューニング

管理者インタフェースのタスクバーには、すでに実行したプロビジョニングタスクのリンクが表示されます。このため、多数のタスクがあると、インタフェースの速度が低下します。

インタフェースのパフォーマンスを向上させるには、<List>...</List> 要素を UserUIConfig オブジェクトから削除して、taskResults.jsp リンクを各 JSP から削除します。

次の例に、<TaskBarPages> 内の <List>...</List> エントリを示します。

例 4–1 UserUIConfig オブジェクトの修正

```
<TaskBarPages>
  <List>
    <String>account/list.jsp</String>
    <String>account/find.jsp</String>
    <String>account/dofindexisting.jsp</String>
    <String>account/resourceReprovision.jsp></String>
    <String>task/newresults.jsp</String>
    <String>home/index.jsp</String>
  </List>
 </TaskBarPages>
```

パフォーマンス問題のデバッグ

ここでは、パフォーマンス問題のデバッグに使用できる、各種 Identity Manager と、第三者のデバッグツールについて説明します。

これらの情報は、次のように構成されています。

- 74 ページの「Identity Manager のデバッグページの使用」
- 80 ページの「それ以外のデバッグツールの使用」

Identity Manager のデバッグページの使用

注 – トレースは、システムパフォーマンスに影響を及ぼします。最適なパフォーマンスが得られるようにするには、最小限のトレースレベルを指定するか、システムのデバッグ後にトレースを無効に設定してください。

ここでは、Identity Manager のデバッグページにアクセスする手順を説明するとともに、このページを使用して Identity Manager のパフォーマンス問題を突き止めデバッグする方法について説明します。

これについては、次のセクションを参照してください。

- 75 ページの「デバッグページへのアクセス方法」
- 75 ページの「制御タイミング (callTimer.jsp) 」
- 76 ページの「トレース設定の編集 (Show_Trace.jsp)」
- 77 ページの「ホスト接続プール (Show_ConnectionPools.jsp)」
- 77 ページの「消去したキャッシュのリスト (Clear_XMLParser_Cache.jsp)」
- 77 ページの「メソッドタイミング (Show_Timings.jsp)」
- 78 ページの「オブジェクトサイズの概要 (Show_Sizes.jsp)」
- 79 ページの「管理者とコンフィギュレータ用のプロビジョニングスレッド (Show_Provisioning.jsp)」

- 79 ページの「システムキャッシュの概要 (Show_CacheSummary.jsp)」
- 79 ページの「システムメモリーの概要 (Show_Memory.jsp)」
- 79 ページの「システムプロパティー (SysInfo.jsp)」
- 79 ページの「システムスレッド (Show_Threads.jsp)」
- 80 ページの「消去されたユーザーセッションプール (Clear_User_Cache.jsp)」
- 80 ページの「Waveset プロパティー (Show_WSProp.jsp)」
- 80 ページの「フラッシュして消去された XML リソースアダプタのキャッシュ (Clear_XMLResourceAdapter_Cache.jsp)」

デバッグページへのアクセス方法

注 – Identity Manager のデバッグページにアクセスして操作を実行するには、*Debug*、*Security Administrator*、または *Waveset Administrator* 機能が必要です。管理者とコンフィギュレータには、デフォルトでこの機能が割り当てられています。

デバッグ機能がない場合は、エラーメッセージが表示されます。

▼ Identity Manager のデバッグページにアクセスする

1 ブラウザを開き、管理者インタフェースにログインします。

2 次の **URL** を入力します。

`http://` *host*`:`*port*`/idm/debug`

各表記の意味は次のとおりです。

- *host* は、Identity Manager の実行先アプリケーションサーバーです。
- *port* は、このサーバーが監視中の TCP ポート数です。

3 システム設定ページが表示されたら、開くデバッグページの `.jsp` ファイル名を入力します。

たとえば、次のようにします。

`http://` *host*`:` *port* `/idm/debug/`*pageName.jsp*

注 – デバッグユーティリティーの中には、システム設定ページからリンクされていないものもありますが、これらを使用すると、製品のパフォーマンスと使いやすさのデータを収集できるようになります。デバッグページの全体リストについては、コマンドウィンドウを開いて `idm/debug` ディレクトリの中身をリストします。

制御タイミング（callTimer.jsp)

各種メソッドの呼び出しタイマー統計を収集して表示するには、制御タイミングページを使用します。この情報を基に、特定メソッドに対するボトルネックと呼び

出された API を追跡できます。呼び出しタイマーの基準値をインポートまたはエクスポートするのにも、呼び出しタイミングページのこのオプションが使用できます。

注 – 呼び出しタイミングの統計は、トレースが有効に設定されている間にだけ収集されます。

▼ 呼び出しタイマーの統計を表示する

1. トレースとタイミングを有効にするには、「制御タイミング」ページを開き、「タイミングとトレースの開始」をクリックします。
2. タイミングを終了するには、「タイミングとトレースの終了」をクリックするか「タイミングの終了」をクリックします。

   ページが再表示され、統計が使用できるメソッドのリストとメソッドの集約呼び出しタイマー統計が「タイミングの表示」テーブルに入力されます (呼び出し側から中断されません)。

   このテーブルには、次の情報が記載されています。

   - メソッド名 (呼び出すメソッドを表示するには、そのメソッド名をクリックします)
   - 総使用時間
   - 平均時間
   - 最小時間
   - 最大時間
   - 総呼び出し数
   - 総エラー数
3. このリストを消去するには、「タイミングの消去」をクリックします。

注 – コンソールから呼び出しタイマーのデータを収集するには、`callTimer` コマンドを使用します。このコマンドは、アップグレード中や、アプリケーションサーバーで Identity Manager を実行していないなどの状況で、パフォーマンス問題をデバッグする際に有用です。

トレース設定の編集 (`Show_Trace.jsp`)

Identity Manager をインストールしたときに付属している Java クラスにトレースを有効にを設定するには、「トレース設定の編集」ページを使用します。

具体的には、このページから次のトレース設定を行えます。

- トレースするには、メソッド、クラス、またはパッケージを選択して、取り込むトレースのレベルを指定します。
- トレース情報をファイルまたは標準出力先に送信します。
- 格納対象となるトレースファイルの最大数と、各ファイルの最大サイズを指定します。
- トレース出力ファイル内の日時の書式設定方法を指定します。
- キャッシュ対象となるメソッドの最大数を指定します。
- トレースファイルへのデータの書き込み方を指定します。

  データが生成されるに従って、データがトレースファイルに書き込まれるか、データがキューに入れられてからファイルに書き込まれます。

ホスト接続プール (Show_ConnectionPools.jsp)

データソースを使用しない場合は、「ホスト接続プール」ページから接続プール統計を表示することができます。これらの統計には、プールのバージョン、作成された接続数、アクティブ数、プール内の接続数、プールから処理されている要求数、および破棄された接続数が記載されます。

この「ホスト接続プール」ページは、ゲートウェイへの接続管理に使用された接続プールの概要を表示する際にも使用できます。この情報を使用して、低位アドレスメモリー状態を調べられます。

消去したキャッシュのリスト (Clear_XMLParser_Cache.jsp)

最近使用した XML パーサーをキャッシュから消去して低位アドレスメモリー状態を調べるには、「消去したキャッシュのリスト」ページを使用します。

メソッドタイミング (Show_Timings.jsp)

メソッドレベルですばやくホットスポットを検出して評価するには、「メソッドタイミング」ページを使用します。

Identity Manager メソッドから収集され、「メソッドタイミング」ページに表示される情報は、次のとおりです。

- メソッド名
- メソッドが呼び出された回数
- メソッドがエラー状態で終了した回数
- メソッドによって消費された平均時間
- 各メソッドの呼び出しによって消費された最小時間と最大時間

「メソッドタイミング」ページには、次のリンクを記載したテーブルもあります。これらのリンクをクリックすると、詳細を表示できます。

- 「詳細」。呼び出しスタックの情報が表示されます。
- 「履歴」。最後の呼び出し時刻とともに、呼び出し期間のグラフが表示されます。
- 「履歴データ」。呼び出しが生成された時刻とその呼び出し期間とともに、最後の呼び出しのリストが表示されます。

  Identity Manager は、デフォルトでスタック履歴を保持しません。スタック履歴を保持してその深さを制御するには、`Waveset.properites` を編集して `MethodTimer` キーを調べます。

注– 「メソッドタイミング」ページにある「すべて消去」オプションは、全ての結果を消去します。このオプションは、デフォルトで有効になっています。

オブジェクトサイズの概要 (Show_Sizes.jsp)

システムに影響を及ぼしかねない大きそうなオブジェクトを検出するには、この「オブジェクトサイズの概要」ページを使用します。

この「オブジェクトサイズの概要」ページには、リポジトリに格納されるオブジェクトのサイズが (文字単位) で表示されます。このオブジェクトは型別にリストされ、型別の総オブジェクト数とオブジェクトの組み合わせ総サイズ、平均サイズ、最大サイズ、および最小サイズが記載されます。

そのオブジェクト型のサイズの詳細は、「型」列のエントリをクリックします。たとえば、リポジトリ内の最大設定オブジェクトの ID、名、サイズを表示するには、「設定」をクリックします。

このサイズ情報は、「コンソール」コマンドラインからもアクセスできます。

▼ コマンドラインからオブジェクトサイズ情報にアクセスする

1. コンソールを開きます。
2. コマンドプロンプトで、次を入力します。

   `showSizes [ type[limit ]]`

注– アップグレードの場合、更新または再表示されるまで既存オブジェクトのサイズが 0 と記録されます。

管理者とコンフィギュレータ用のプロビジョニングスレッド (Show_Provisioning.jsp)

システムで使用中のプロビジョニングスレッドの概要を表示するには、この「管理者とコンフィギュレータ用のプロビジョニングスレッド」を使用します。この概要は、Show_Threads.jsp で用意される情報のサブセットです。

注– 単一のスレッドダンプを見ただけでは、判断できないことがあります。

システムキャッシュの概要 (Show_CacheSummary.jsp)

低位アドレスメモリー状態の調査に役立つよう、次の項目について表示するには、この「システムキャッシュの概要」ページを使用します。

- 管理者に関連付けられているオブジェクトキャッシュ
- システムオブジェクトのキャッシュ
- ユーザーのログインセッション
- XML パーサーのキャッシュ

システムメモリーの概要 (Show_Memory.jsp)

利用可能な総メモリー数と空きメモリーを表示するには、この「システムメモリーの概要」ページを使用します。調整などのメモリーを消費する機能を使用している場合は、この情報を基にして JVM に十分なメモリーが割り当てられているかどうかが突き止められます。

また、ガベージコレクションの起動や、ヒープ使用量を調査するための JVM 内の未使用メモリーの消去にも、このページが役に立ちます。

システムプロパティー (SysInfo.jsp)

この「システムプロパティー」ページには、ソフトウェアバージョン、パス、環境変数など、使用中の環境についての情報が記載されます。

システムスレッド (Show_Threads.jsp)

(調整や Active Sync などの) 自動プロセスが実行中であることを確認できるように、実行中のプロセスを表示するには、この「システムスレッド」ページを使用します。

このページには、プロセス種類、プロセス名、そのプロパティー、プロセスがデーモンかどうか、およびプロセスがいまだに実行中かどうかについての情報が記載されます。

注 – 単一のスレッドダンプを見ただけでは、判断できないことがあります。

消去されたユーザーセッションプール (Clear_User_Cache.jsp)

最近ログインしたユーザーのキャッシュ済みセッションをすべて消去し、低位アドレスメモリー状態を調べるには、この「消去されたセッションプール」ページを使用します。

Waveset プロパティー (Show_WSProp.jsp)

Waveset.properties ファイルのプロパティーを表示して一時的に編集するには、この「Waveset プロパティー」ページを使用します。Waveset.properties ファイルの常駐先の特定サーバーには、サーバーを再起動して変更内容を反映していなくても、各種のプロパティー設定をテストできます。編集したプロパティー設定は、次回サーバーを再起動するまで、反映されたままになります。

フラッシュして消去された XML リソースアダプタのキャッシュ (Clear_XMLResourceAdapter_Cache.jsp)

テスト XML リソースアダプタをキャッシュから消去し、低位アドレスメモリー状態を調べるには、この「フラッシュして消去された XML リソースアダプタのキャッシュ」ページを使用します。

それ以外のデバッグツールの使用

パフォーマンスの潜在的ボトルネックを見つけるには、次の Sun Microsystems ツールと第三者ツールが使用できます。

- 80 ページの「Identity Manager Profiler」
- 81 ページの「DTrace の使用」
- 82 ページの「JMX の使用」
- 85 ページの「JConsole の使用」
- 86 ページの「JRat の使用」

これらのツールは、使用中の配備でカスタム Java クラスを使用する場合に有用です。

Identity Manager Profiler

Identity Manager には、使用中の配備のパフォーマンス問題をトラブルシューティングするのに役立つ Profiler ユーティリティーが搭載されています。

カスタマイズしたフォーム、Java、ルール、ワークフロー、およびXPRESSがパフォーマンス問題とスケール問題の原因になることがあります。このProfilerは各領域での経過時間を調べるので、フォーム、Java、ルール、ワークフロー、XPRESSオブジェクトのうちどれがパフォーマンス問題とスケール問題の一因となっているか、一因となっている場合は、そのオブジェクトのどの部分が問題の原因になっているかがわかります。

注 – Profilerの詳細は、『Sun Identity Manager 8.1 リリースノート』の「Identity Managerプロファイラの操作」を参照してください。

DTraceの使用

DTrace機能は、Solaris 10オペレーティングシステム用に動的にトレースするフレームワークで、JVM動作を監視できるようになります。

DTraceには、30,000を超える検証機能が搭載されており、統合されたユーザーレベルとカーネルレベルのトレース機能を使用して、プロダクションシステム内を把握できるようにします。また、C言語やawk言語に似たD言語で、任意のデータと式をトレースすることもできます。このDTrace機能にも、JVMを監視するための特殊なサポートが備わっているので、使用中のシステム全体とJVM外部のスパンが監視できるようになります。

DTraceは、検証機能がJVMに内蔵されているのでJava 6で一番使用しやすくなっています。この機能は、Java 1.4とJava 5でも動作しますが、次のURLからJVM PIまたはJVM TIエージェントをダウンロードする必要があります。

https://solaris10-dtrace-vm-agents.dev.java.net/

次の例は、DTraceスクリプトの記述方法を示したものです。

例4–2　DTraceスクリプトの例

```
#!/usr/sbin/dtrace -Zs
#pragma D option quiet
hotspot$1:::
{
  printf("%s\n", probename);
}
```

この例では、*$1*をスクリプトの最初の引数に置き換えてください。これは、管理対象となるJavaプロセスのPIDとしてください。たとえば、次のようにします。

```
# ./all-jvm-probes.d 1234
```

次の表は、各種DTraceの検証機能を有効にできるコマンドを説明したものです。

表 4–3 DTrace コマンド

コマンド	説明
`-XX:+DTraceMonitorProbes`	Java 6 の JVM サポートを有効にします (Java 1.4 と Java 5 用パッチ)。
`-XX:+ExtendedDTraceProbes`	次の情報を提供します。 ■ JVM の起動 (開始と終了) とシャットダウン ■ 開始スレッドと停止スレッド ■ 読み込み中クラスと読み込み解除中のクラス ■ ガベージコレクション (各種オプションが利用可能) ■ JIT コンパイルの開始と終了 ■ 読み込み中コンパイル済みメソッドと読み込み解除中のコンパイル済みメソッド ■ 監視競合、待機、通知 ■ メソッドのエントリ、メソッドの戻り値、オブジェクト割り当て
`/usr/sbin/dtrace -n 'hotspot*:::'`	そのシステム上のすべての Java プロセスに、すべての JVM 検証機能を有効にします。
`/usr/sbin/dtrace -n 'hotspot1234:::'`	PID が *1234* の Java プロセスにだけ、すべての JVM 検証機能を有効にします。
`/usr/sbin/dtrace -n 'hotspot1234:::gc-begin'`	プロセス *1234* を開始するためのガベージコレクション時に起動する検証機能だけを有効にします。

注 – DTrace はシステム処理を増やすため、この機能はシステムパフォーマンスに影響します。この影響はごくわずかですが、負荷のかかる有効化で大量の検証機能を有効にすると大きくなります。

DTrace のパフォーマンスへの影響を最小限に抑える手順については、『Solaris Dynamic Tracing Guide』の「Performance Considerations」の章に記載されています。

DTrace の詳細は、`/usr/demo/dtrace` および `man dtrace` を参照してください。

JMX の使用

Identity Manager を使用すると、Java Management Extensions (JMX™) を使用して、所定のリソースアダプタの操作における操作上の統計を取り込み表示することができます。このデータは、システム状態とレポートを監視するなど、診断と予測に使用できます。

この統計データには、次の情報が含まれています。

- 動作が実行された回数
- 操作の最小期間、最大期間、平均期間

オブジェクト	監視対象となった動作
アカウントの場合	■ Create ■ Update ■ Delete ■ Get ■ Authenticate
動作の場合	Run
その他オブジェクトの場合	■ Create ■ Update ■ Delete ■ Get ■ List

JMX は、サーバー別のリソースアダプタごとに MBeans を作成して、これらの Beans を次のパターンに一致する名前で登録します。

`serverName=`*server name*`, resourceAdapterType=`*Resource Adapter Type*`,`
`resourceAdapterName=`*Resource Adapter Name*

Identity Manager は、正常に完了したかエラーで完了したか、完了した処理のすべてに統計を記録します。ただし Identity Manager は、例外を投げる処理など未完成の処理の統計は記録しません。

`excludes` の設定方法は、次のとおりです。

1. 管理者インタフェースから、「設定」→「サーバー」の順に選択します。
2. 「サーバーの設定」ページで、次のいずれかのタスクを実行します。
   - サーバーのデフォルト設定を編集するには、「サーバーのデフォルト設定の編集」ボタンをクリックします。
   - サーバーのポリシーを編集するには、そのサーバーのリンクをクリックします。
3. リソース監視を有効にするには、「JMX」タブをクリックして「JMX リソースアダプタモニターの有効化」ボックスを有効にします。
   - 特定のリソースを除外するには、「JJMX リソースアダプタモニターの対象外」リストに正規表現を追加します。

- 特定動作の監視を除外するには、「JMX リソースアダプタモニター操作の対象外」リストに正規表現を追加します。

どの excludes も、正規表現を使用します。特定リソースを除外した場合、JMX はリソース名だけで照合します。たとえば次の名前のアダプタがある場合は、

```
resource1
resource2
resource3
resource10
resource11
```

次のパターンを指定します。

```
.*1$
```

つまり、何かが 1 (1$) で終わるまで、0 以上の任意の文字 (.*) にマッチします。JMX は、resource1 と resource11 を除外します。

処理の場合も、手順は同様です。処理に次の名前が付いている場合は、パターンがこの名前に一致する必要があります。

```
ACCOUNT_CREATE
ACCOUNT_UPDATE
ACCOUNT_DELETE
ACCOUNT_GET
ACCOUNT_AUTHENTICATE
OBJECT_CREATE
OBJECT_UPDATE
OBJECT_DELETE
OBJECT_GET
OBJECT_LIST
ACTION_RUN
```

たとえば、^ACCOUNT.* パターンは ACCOUNT で始まるすべての処理を除外します。または、このパターンを使用すると updates と deletes が除外されます。

```
.*UPDATE$
.*DELETE$
```

注 – JMX の設定方法と使用法の詳細は、『『Sun Identity Manager 8.1 ビジネス管理者ガイド』』の18 ページの「JMX 監視の設定」 および 『Sun Identity Manager 8.1 ビジネス管理者ガイド』の「JMX パブリッシャータイプ」を参照してください。

JConsole の使用

Java Monitoring and Management Console (*JConsole*) は、Java Management Extension (JMX) テクノロジ対応のグラフィカル管理ツールで、JDK 5 以降に同梱されています。JConsole は実行中の JVM に接続し、接続している JMX エージェントの JVM MBeans から情報を収集します。

具体的に JConsole で実行できるタスクは、次のとおりです。

- 低位アドレスメモリーとデッドロックの検出

  JConsole は、メモリーシステム、メモリープール、および MBeans ガベージコレクタにアクセスして、メモリー消費量、メモリープール、ガベージコレクション統計などのメモリー使用量に関する情報を表示します。
- ガベージコレクションの有効化または無効化
- 冗長トレースの有効化または無効化
- ローカルおよびリモートアプリケーションの監視
- 現在のヒープメモリー使用量、ヒープ以外のメモリー使用量、ファイナライズに保留されているオブジェクト数など、MBeans の監視と管理を行います。
- パフォーマンス、リソース消費量、およびサーバー統計に関する情報を表示
- JVM と監視した値、アプリケーションで実行中のスレッド、および読み込まれたクラスに関する概要を表示
- オペレーティングシステムリソース (Sun のプラットフォーム拡張) に関する情報を表示。次のようなものがあります。
  - CPU プロセス時間
  - 利用可能な物理メモリーの総容量と空き容量
  - 確定した仮想メモリー量 (プロセスの実行に確実に利用できる仮想メモリー量)
  - 利用可能なスワップ領域の総容量と空き容量
  - オープンファイルの記述数 (UNIX® のみ)

注 – JConsole を使用した Java プラットフォーム上のアプリケーション監視の詳細は、「JConsole を使用したアプリケーション監視」という Sun Developer Network (SDN) 記事を参照してください。次の URL から入手できます。

`http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`

Identity Manager には、次の点に関する情報を提供する JMX MBeans がいくつか搭載されています。

- Identity Manager Server Cluster
- データエクスポータ
- スケジューラ

JRat の使用

潜在的なパフォーマンスボトルネックを見つけるには、Java プラットフォーム用オープンソースのパフォーマンスプロファイラである Java Runtime Analysis Toolkit (JRat) を使用します。使用中の配備でカスタム Java クラスを使用する場合は特に使用します。JRat は、使用中のアプリケーションの実行を監視し、アプリケーションのパフォーマンスの測定を維持します。

たとえば、プロビジョニングにカスタムワークフローがある場合にこの JRat を使用すると、呼び出されているクラスを表示したり、デフォルトの Identity Manager プロビジョニングワークフローと比較したワークフローの実行に掛かる時間を表示できます。

JRat の詳細は、`http://jrat.sourceforge.net` を参照してください。

◆ ◆ ◆ 第 5 章

5 トレースとトラブルシューティング

この章では、トレーシングを使用してエラーの解決に役立てる方法、パフォーマンス問題の解決方法、Identity Manager 内のフローを理解する方法について説明します。この章では、各種 Identity Manager コンポーネントで起こりうる問題のトラブルシューティング方法についても説明します。

この章では、次のトピックについて説明します。

トレースやトラブルシューティングを始める前に

Identity Manager のトレースやトラブルシューティングを始める前に、次のセクションの内容を確認しておいてください。

対象読者

この章は、アプリケーションサーバーとデータベースの管理者、第一線のサポートエンジニア、および配備環境で Identity Manager の維持管理を担当しているパートナーの方を対象としています。

Identity Manager の問題をトラブルシューティングする前に、次が必要になります。

- Java 5.0 (Sun Identity Manager 8.1 に必須) に使い慣れておくこと。
- トラブルシューティングしようとしているコンポーネントを理解していること。

Identity Manager のトレースとトラブルシューティングの重要な注意点

Identity Manager のトレースまたはトラブルシューティングを開始する前に、次の点にご注意ください。

- トレースは、システムパフォーマンスに影響を及ぼします。最適なパフォーマンスが得られるようにするには、システムのデバッグ終了後に最小限のトレースレベルに抑えるか、トレースを無効にします。
- `com.waveset` クラスにはトレースを有効に設定しないでください。`com.waveset` クラスは冗長で多数のクラスがあるため、このクラスをトレースするとサーバーがハングすることがあります。
- Sun サポートから特定の指示がない限りは、Identity Manager が高度なレベルでトレースするように設定しないでください。
- 特定の問題をデバッグするのに `Waveset.properties` ファイルの `exception.trace` を使用する場合は、長期間使用せず、デバッグが終了したら必ず無効に設定してください。`Waveset.properties` ファイルの `exception.trace` を設定すると、システムパフォーマンスに著しい影響を及ぼします。

サポートにお問い合せになる前に

Sun テクニカルサポートにサポートをお問い合わせになる前に、問題の領域を絞り込むことができる次の推奨事項を試してみてください。

- Web ブラウザの検索ツールで、問題を調べてみる。
- リソースに固有のサポートオプションがあれば、それを使用する。リソースに関連する問題では、リソースのパッチを使用すれば解決することがあります。
- 適切なクライアントツールが使用できればそれを使用して、その状況から Identity Manager をアンインストールします。たとえば、Java LDAP ブラウザを使用します。

また、Sun サポートにサポートをお問い合わせになる前に、次の情報を集めて用意しておく必要があります。

収集する情報	情報の確認方法
製品のバージョンと次の情報 ■ インストールされているパッチ、ホットフィックス、e-fix ■ カスタマイズのリスト	Identity Manager コンソールで次のコマンドを使用します。 ■ `installed` ■ `inventory`
Identity Manager のトポロジと次の情報 ■ Identity Manager クラスタの構成 ■ 問題の環境のローカリゼーションと環境に関するその他の情報 ■ サーバーのリスト	情報は次のように取得します。 ■ 手動で確認する必要があります。 ■ 手動で確認する必要があります。 ■ 実行中の Identity Manager サーバーで、Identity Manager 管理者インタフェースから「設定」>「サーバー」の順に選択します。
環境で行った最近の変更	手動で確認する必要があります。
Java のバージョンとタイプ	次の Java コマンドを使用します。 `java -version`
アプリケーションサーバーのバージョンとタイプ	使用している Application Server によって異なりますが、手動で確認する必要があります。
オペレーティングシステムのレベルと情報	手動で確認する必要があります。
デフォルト XML 出力 (すべてのリポジトリオブジェクトのリスト)	コンソールで `export default.xml default` コマンドを使用します。
タスクインスタンスのデータと次の情報 ■ 実行中のタスクのリスト ■ 現在のすべてのタスクインスタンスのサイズ	コンソールで次のコマンドを使用します。 ■ `listTasks` ■ `showSizes TaskInstance`
システムログ	注 – 報告する問題の種類に応じて、追加のシステムログの提出を要求される場合があります。

関連ドキュメントと Web サイト

Identity Manager のトレースとトラブルシューティングについては、この章で説明する内容のほかにもこのセクションに記載されているドキュメントと Web サイトをご覧ください。

推奨ドキュメント

Identity Manager のトレースとトラブルシューティングについては、次のドキュメントを参照してください。

- XPRESS 関数の推奨トレース方法については、『Sun Identity Manager Deployment Reference』の「Testing Your Customized Form」を参照してください。
- JConsole を使用して Java プラットフォームで動作するアプリケーションを監視する方法については、「JConsole を使用したアプリケーションの監視」という記事。この記事は、次の URL から入手できます。

  `http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`

有用な Web サイト

次の表に、Identity Manager のトラブルシューティングに関する Web サイトを説明します。

表 5–1 有用な Web サイト

Web サイト URL	説明
`http://sunsolve.sun.com/`	診断ツール、フォーラム、機能と記事、セキュリティー情報、パッチの内容を含む Sun の Web サイト。 注意: このサイトの情報は、次の 3 分野に分かれています。 ■ 社内。Sun 従業員のみ。 ■ 契約。契約アクセス権をお持ちのお客様のみ。 ■ 公開。全員。
`http://www.sun.com/service/online/us`	Sun テクニカルサポートの Web サイト。
`https://sharespace.sun.com/gm/folder-1.11.60181?`	Sun Microsystems の Share Space の Identity Manager フォルダ。Identity Manager の FAQ、フォーラムへのリンク、特集記事や記事などが記載されています。 注意: このサイトに記載されている情報にアクセスするには、Share Space ID を登録する必要があります。
`https://identitymanageride.dev.java.net`	オープンソースの Sun Identity Manager 統合開発環境 (Identity Manager IDE) プロジェクト。Identity Manager IDE のインストール方法と設定方法の手順が記載されています。

手順の概要

次の図に示す手順に従えば、通常は使用中の配備の問題を見つけて解決することができます。

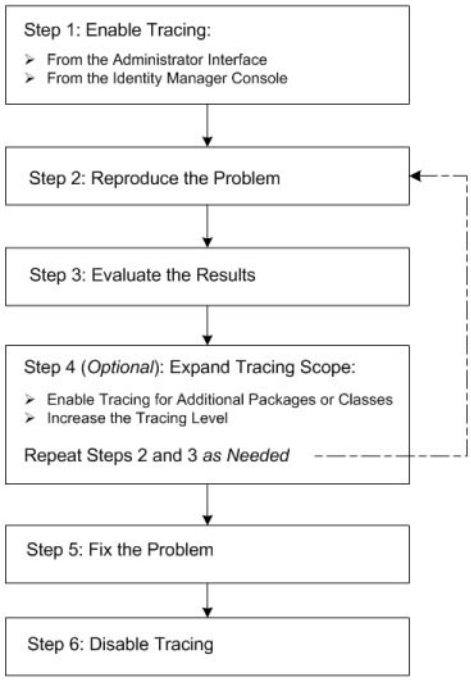


図 5–1 問題のトレースとトラブルシューティング

Identity Manager の各種オブジェクトとアクティビティーのトレースを有効にする方法については、91 ページの「Identity Manager のオブジェクトとアクティビティーのトレース」を参照してください。

オブジェクトとアクティビティーによくある問題をトラブルシューティングする方法については、134 ページの「よくある問題のトラブルシューティングと解決策」を参照してください。

Identity Managerのオブジェクトとアクティビティーのトレース

トレース出力は、問題を見つけて解決する際にも、カスタムのリソースアダプタを開発する際にも非常に有用です。

ここでは、Identity Managerの各種オブジェクトとアクティビティーのトレースを有効に設定する方法について説明します。説明する内容は次のとおりです。

注 – トレースは、システムパフォーマンスに影響を及ぼします。最適なパフォーマンスが得られるようにするには、システムのデバッグ終了後に最小限のトレースレベルに抑えるか、トレースを無効にします。

トレースの設定方法

Identity Managerの中には、トレースを有効に設定できる場所が複数用意されています。次に、この手順について説明していきます。

- 「System Settings」ページからトレースを有効にする
- 個々のデバッグページからトレースを設定する
- Identity Managerコンソールからトレースを有効にする

▼ 「System Settings」ページからトレースを有効にする

「System Settings」ページがProduct_IDMgr;のデバッグページのメイン画面になります。

このページから行える操作は、次のとおりです。

- リポジトリ内のオブジェクトの表示と編集
- キャッシュの消去
- 特殊なトレースの設定
- `Waveset.properties`ファイルの再読み込み

Identity Manager の「System Settings」ページからトレースを有効にする手順は、次のとおりです。

1 ブラウザを開いて、**Identity Manager** 管理者インタフェースにログインします。

2 次の **URL** を入力します。

`http://`*host port*`/idm/debug`

各表記の意味は次のとおりです。

- *host* は、Identity Manager の実行先ローカルサーバーです。
- *port* は、このサーバーが監視中の TCP ポート数です。

3 「**System Settings**」ページが表示されたら「**Show Trace**」をクリックして、単一のトレース設定を操作します。ここから、最大 **10** 個までのトレース設定を作成、変更、削除できます。

注 – これ以降の説明は、トレース設定が 1 つしかないものとして説明します。

複数のトレース設定を操作するには、代わりに「Show Trace List」ボタンをクリックしてください。「Trace Configuration」ページが表示されたら、これまでの設定を編集する設定名をクリックします。

Identity Manager には、デフォルトでグローバル設定が用意されています。しかし、Identity Manager インスタンスに複数のサーバーがある場合は、これ以外の設定を定義した方が便利かもしれません。トレース設定名と現ホスト名が重複する場合は、ホスト設定の方がグローバル設定よりも優先されます。

次の手順 トレースを設定した後は、次のセクションで説明するように、デフォルトのグローバルトレース設定オブジェクトを表示して編集したり、設定オブジェクトを新規作成できます。

- 93 ページの「デフォルトの設定オブジェクトを編集する」
- 96 ページの「トレース設定オブジェクトを新規作成する」

▼ デフォルトの設定オブジェクトを編集する

1 「**Edit Trace Configuration**」ページから、「**Trace Enabled**」ボックスをクリックしてトレースを有効にします。

注 – このボックスの選択を解除するとトレースは終了しますが、設定はそのまま残ります。トレースしていたクラスを覚えておいて入力し直さなくても、トレースの有効と無効を切り替えられます。

---

2 トレースするクラス、パッケージ、またはメソッド名をテーブルに入力して指定します。

たとえば、次のようにします。

- waveset.repository パッケージ内にあるすべてのクラスをトレースするには、com.waveset.repository と入力します。
- waveset.repository パッケージ内にある AbstractDataStore クラスをトレースするには、**com.waveset.repository.AbstractDataStore** と入力します。
- waveset パッケージ内にある AbstractDataStore クラスのリストメソッドをトレースするには、 **com.waveset.repository.AbstractDataStore#list** と入力します。

com.waveset クラスにはトレースを有効に設定しないでください。com.waveset クラスは冗長で多数のクラスがあるため、このクラスをトレースするとサーバーがハングすることがあります。

3 このテーブルの「**Level**」メニューから、「**Method**」または「**Class**」のトレースレベルを選択します。

次の表で説明するように、それぞれのレベルで異なる種類の情報が取り込まれます。

トレースレベル	説明
0	最小デバッグ出力、トレース例外、およびエラー情報のみ
1	トレースレベル 0 イベント + Public メソッドのエントリと終了
2	トレースレベル 1 イベント + Public 以外のメソッドのエントリと終了
3	トレースレベル 2 + 決定点と重要変数
4	最大デバッグ出力

注 – メソッド/クラスのトレースは予測できますが、場合によってはトレース出力が大量になることがあります。トレースするメソッドとクラスを指定するときは、できる限り明確に指定するようにしてください。

4 (オプショナル) 下位呼び出しのトレースを有効にするには、「**Subcall Tracing**」メニューからレベルを選択します。このメニューで使用されるトレース番号のレベルは、先の表で説明したものと同じです。

注 –

- デフォルトの下位呼び出しトレースレベルは、なしです。これにより、メソッド単位またはクラス単位での下位呼び出しトレースが無効になります。
- 下位呼び出しトレースレベルは、先の手順で指定したメソッド/クラスのトレースレベルとは関係ありません。

下位呼び出しトレースをサポートしている特定のメソッドに「下位呼び出しトレース」を有効にすると、このメソッドから呼び出されるメソッドにはトレースレベルが自動的に設定されるようになります。下位呼び出しトレースを使用すると、指定メソッドへのエントリとメソッドの終了により終了したエントリを契機に、短くても詳細なトレース出力のバーストが生成できます。

たとえば、`com.waveset.adapter.NewRes#init` メソッドのトレース設定を作成した場合は、メソッド/クラスのトレースをレベル 1 に設定し、下位呼び出しトレースをレベル 3 に設定します。

さらに、`init` メソッドが次のもう 2 つのメソッドを呼び出すとします。

- `NewRes#subcallA`
- `NewRes#subcallB`

  `init` メソッドが実行されると、`com.waveset.adapter.NewRes#init` メソッドは、`subcallA` に達するまで、レベル 1 でトレース出力を生成します。`subcallA` の実行が開始されると、トレースレベルが 3 に変わり、`subcallA` が終了するまでこのレベルが継続されます。`com.waveset.adapter.NewRes#init` メソッドが `init` メソッドに戻り、トレースレベルを 1 に戻します。その後、`init` が `subcallB` を呼び出すと、`subcallB` が存在する限り、別のバーストのトレースレベル 3 の詳細が続きます。最後に、`init` が存在すれば、トレースレベル 1 が終了します。

5. トレース結果を指定ファイルの場所に送信するか、`stdout` に送信します。

   ファイルへの出力を選択すると、「Trace File」フィールドが表示されます。このフィールドから、別の場所を指定したり、トレース出力ファイルのファイル名を指定します。デフォルトの出力場所とファイル名は、次のとおりです。

   path_to_idm_install`\export\pipeline\config\WSTrace.log`

6. 格納する最大トレースファイル数を指定します (デフォルトは **2**)。

7. 各ファイルの最大サイズを指定します (デフォルトは **512K**)。

8. トレース出力ファイルを生成時に書き込むか (同期的)、そのデータをキューに入れておいてからトレースファイルに書き込むか (非同期的) を指定します。

9. 変更を保存します。

▼ トレース設定オブジェクトを新規作成する

トレース設定オブジェクトを新規作成するには、次の項目を実行します。

1 トレース対象とするパッケージまたはメソッドを決定します。
通常は、リソースアダプタ名を指定するか、エラーメッセージに表示された情報を使用します。

2 **Identity Manager** 管理者インタフェースにログインし、「**System Settings**」ページからトレースを有効にするに説明したとおりに「**System Settings**」ページを開きます。

3 「**System Settings**」ページで、「**Show Trace List**」をクリックします。

4 **Identity Manager** の「**Trace Configuration**」ページが表示されたら、「**New**」をクリックします。

5 「**Edit Trace Configuration**」ページから、トレースを有効にします。
「Trace Configuration」メニューから、次のオプションのいずれかを選択します。

- 「**Global**」。すべてのサーバーのトレースを有効にするように選択します。
- サーバー名。サーバー名を選択して、特定のサーバーのトレースを有効にします。

6 「**Trace Enabled**」ボックスを選択して、このオブジェクトのトレースを有効にし、デフォルトの設定オブジェクトを編集するに説明したとおりにこのページの残りのパラメータを設定します。

7 変更を保存します。

▼ 個々のデバッグページからトレースを設定する

ここでは、Identity Manager のデバッグページからトレースを有効にする方法について説明します。

1 ブラウザを開いて、**Identity Manager** 管理者インタフェースにログインします。

2 次の **URL** を入力します。
`http://` *host*`:`*port*` /idm/debug/` *pageName.jsp*
各表記の意味は次のとおりです。

- *host* は、Identity Manager の実行先ローカルサーバーです。
- *port* は、このサーバーが監視中の TCP ポート数です。
- *pageName.jsp* は、開こうとしている個々のデバッグページです。

たとえば、カスタムアダプタにアダプタクラスとメソッドをトレースするには、次のURLを入力して「Edit Trace Configuration」ページを開きます。

```
http://host : port/idm/debug/Show_Trace.jsp
```

▼ Identity Managerコンソールからトレースを有効にする

ここでは、Identity Managerコンソールからトレースを有効にする方法について説明します。

1. `$WSHOME`を設定します。

   たとえば、この変数をデフォルトのインストールディレクトリに設定するには、次のように指定します。

   ```
   set WSHOME=C:\Program Files\tomcat\webapps\idm
   ```

2. **Identity Manager**コンソールを`bin`ディレクトリから開くには、`lh console`と入力します。

3. コンソールから、**trace**と入力して、`enable`や`disable`など使用可能なトレースオプションの詳細なまとめを表示します。

   このコマンドの構文は次のとおりです。

   ```
   trace [ -s server ] $subcommand
   ```

トレースファイルの表示方法

デフォルトでは、Identity Managerはトレース情報を`WSTrace#.log`というファイルに送信します。これは*path_to_idm_install*`\export\pipeline\config\`ディレクトリにあります。必要ならば、オブジェクトのトレースを設定する際に別のファイル名と場所を指定することもできます。

それぞれのログファイルは、「Edit Trace Configuration」ページで指定した最大ファイル数まで連続して番号が付けられます。たとえば、最大ファイル数を3に指定した場合は、ファイルには`WSTrace1.log`、`WSTrace2.log`、および`WSTrace3.log`が付けられます。

これらのログファイルを表示するには、次のいずれかの方法を使用します。

- トレース情報をファイルに送信する場合は、指定した場所からそのトレースファイルを開きます。例を示します。

  path_to_idm_install `\export\pipeline\config\WSTrace2.log`

- トレース情報を`stdout`に送信する場合は、使用しているアプリケーションサーバーの`stdout`ファイルをテキストエディタで開いて、トレース出力ログを表示します。

Identity Manager サーバーのトレース

Identity Manager は、Java ベースの製品です。この実行可能ファイルは、パッケージとしてグループ化されている Java クラスから構成されています。コードを実装すると、多くのクラスでトレースを出力できるようになります。

Identity Manager サーバーをトレースすると、サーバーの障害箇所、問題のある箇所、実行していない箇所などの有用な情報が入手できます。Identity Manager を実行中のサーバーでパッケージレベルとメソッドレベルのトレースを有効にするには、Identity Manager のデバッグページを使用します。

注 – Sun サポートから指示がない限りは、Identity Manager がこの高度なレベルでトレースするように設定しないでください。

アダプタのトレース

トレース情報を使用すると、リソースアダプタが起動していることを確認し、すべての設定の変更内容が保存されていることを確認し、アダプタの問題を診断することができます。

ここでは、アダプタのトレース設定に役立つ内容について説明します。このセクションは、次のトピックから構成されています。

- 98 ページの「アダプタのトレースに関する一般的な注意点」
- 99 ページの「トレースポイントがあるコードの計測」
- 106 ページの「その他のトレースガイドライン」

アダプタのトレースに関する一般的な注意点

アダプタのトレースを有効に設定するには、次を入力してトレース対象とするメソッドを特定する必要があります。

com.waveset.adapter.*sample*.*MyResourceAdapter*

また、アダプタがリソース設定をログファイルに書き込めるように、使用中のカスタムリソースアダプタに新メソッドがあれば、呼び出しを指定してログエントリを作成する必要があります。

場合によっては、リソースインスタンスにさらに次のようなログパラメータを指定することもできます。

- ログアーカイブの最大数
- アクティブログの最大有効期間
- ログファイルパス
- ログファイルの最大サイズ

- ログレベル

注 –

- さらに同期プロセスをデバッグするには、ActiveSync アダプタに同期ログを設定してください。この手順は、Identity Manager のオンラインヘルプで説明しています。
- カスタム Java コードのトレースは、お客様独自のリソースアダプタを書き込む際に有用です。詳細は、107 ページの「カスタムコードのトレース」を参照してください。
- 問題をより効率的に評価して解決するには、トレースポイントがあるコードを測定します。詳細は、99 ページの「トレースポイントがあるコードの計測」を参照してください。
- 特定のアダプタメソッドをトレースする方法の詳細は、『Sun Identity Manager 8.1 リソースリファレンス』を参照してください。

トレースポイントがあるコードの計測

Identity Manager は、トレース機能を使用すると、アダプタコンポーネント (リソースアダプタ) でアダプタのトレースポイントがあるコードを計測できるようになります。ここでは、トレースポイントを使用し続けてアダプタ問題を評価し、解決に役立つガイドラインをいくつか説明します。

トレースポイントがあるコードを測定するためのガイドラインの方針は、次のとおりです。

- 関連情報を表示して、本稼働環境と開発環境の両方で、トレースポイントをトラブルシューティングにできる限り役立つようにする。
- 重要なメソッドごとに、1 つのエントリポイントと 1 つの終了ポイント、または例外トレースポイントを作成する。
- 通常の処理中、および次の操作によってトレースが有効になっているときに、トレースポイントをできる限り効果的に活用する。
  - トレースレベルのチェック数を最小限におさえる。
  - トレースが有効に設定されているときにオブジェクトの作成を最小限に抑える。

表示する情報量を制御するには、トレースポイントのトレースレベルを指定します。次の表は、`com.sun.idm.logging.TraceManager` インタフェースで定義される各トレースレベルを説明したものです。

表5–2 定義されているトレースレベル

トレースレベル	トレース変数	使用状況
1	Trace.LEVEL1	リソースアダプタインタフェースの Public メソッドのエントリポイントと終了ポイント
2	Trace.LEVEL2	Public メソッド以外または Identity Manager コンポーネント外部インタフェースに含まれているメソッドのメソッドのエントリポイントと終了ポイント
3	Trace.LEVEL3	決定点または重要変数
4	Trace.LEVEL4	ループ内の重要変数など、極めて詳細な情報
n/a	Trace.ALWAYS	トレースレベルのチェックを行わない 注意: このオプションは、すでにトレースレベルを条件に指定している場合に使用してください。

注 – 循環的に依存しないようにするには、com.sun.idm.logging.TraceManager インタフェースを実装する com.sun.idm.logging パッケージに、com.sun.idm.logging.Trace クラスを使用します。

トレースポイントをコードに追加するときは、次の点に注意してください。

- メソッドへのトレース引数、およびメソッドからの戻り値。

  引数が「print large」の場合、テキスト表現が短くなければ、そのオブジェクトを null かどか指定してください。引数または戻り値が配列かリストで、その値の印刷が大きい場合は、配列サイズかリストサイズを指定すれば十分です。null 値を避けるには、com.waveset.util.Util.length() ユーティリティーメソッドが使用できます。

- トレース情報をフォーマットするか有意義にトレース情報を与えるオブジェクトを作成するには、指定トレースレベルが有効にならない限り Identity Manager がトレースポイントデータを整列化しないように、Trace.getLevel メソッドと isLogging(level) メソッドを使用してトレース文の周りに if 文を配備します。

  この構文を使用するときは、条件内のトレースポイントに Trace.ALWAYS トレースレベルを指定してください。条件内でチェックがすでに行われていれば、このトレースレベルがトレースレベルに不要なチェックを防いでくれます。

- obj.getName() などの印刷可能表現を取得するのにメソッドの呼び出しを必要とするオブジェクト値をトレースする場合は、トレースが有効になった場合に NullPointerException が発生しないように null オブジェクトを避けてください。

注 – 通常は、単純な getter や setter メソッドなどの簡易メソッドにトレースポイントを使用しないでください。

- WSUser オブジェクトをトレースするときは、 toString() メソッドを getName() メソッドの代わりに使用して、オブジェクトの値を表示します。toString()) メソッドの方が、getName() よりも堅牢性があり、有用な情報を印刷します。user.getName() のトレースの代わりに、toString() を暗黙に呼び出す user をトレースしてください。
- メソッドまたはコンストラクタが単なるラッパーの場合、つまりメソッドまたはコンストラクタが別のメソッドやコンストラクタを、同名だが異なる署名で呼び出すだけでまったく重要でない場合は、トレースポイントを行き先メソッドやコンスタラクタに直接配備します。
- パフォーマンスが不可欠で、そのメソッドが何度も呼び出される場合は、エラーを示すように情報トレースポイントを使用します。そのメソッドには、エントリや終了トレースポイントを付けないでください。このオプションは慎重に使用してください。
- トレースポイントを付ける前に、フィールドの問題を見つけるのにそのトレースポイントから有用な情報が得られるかどうか確認してください。

次のセクションでは、各トレースレベルをより詳細に説明していき、コードでのトレースポイントの使用例を紹介していきます。

- 101 ページの「エントリと終了トレースポイントの使用方法」
- 104 ページの「情報トレースポイントの使用方法」
- 105 ページの「例外トレースポイントの使用方法」
- 106 ページの「トレースポイントのテンプレートの使用方法」

エントリと終了トレースポイントの使用方法

使用するコードにエントリおよび終了トレースポイントを追加する前に、これらのガイドラインをお読みください。

- すべての Identity Manager コンポーネントの外部インタフェースメソッドには、リソースアダプタインタフェースで宣言されている public メソッドの Trace.Level1 エントリまたは終了トレースポイントを使用します。
- コンストラクタや静的メソッドや Identity Manager コンポーネントの外部インタフェースに含まれていないメソッドなど、重要な public メソッド以外には、Trace.Level2 エントリまたは終了トレースポイントを使用します。
- トレースポイントの中に引数を指定します。

その引数が基本型または java.lang.String でなく、各種オブジェクトの作成やメソッドの呼び出しを必要とする引数の場合は、エントリトレースポイントを条件付きにして、トレースが有効な場合のみフォーマット処理が行われるようにします。

- 終了トレースポイントを指定する際には、そのメソッドが基本型か java.lang.String の場合、戻り値を表示してください。戻り値に何らかのフォーマットが必要な場合は、このガイドラインに説明したとおりにオブジェクトのフォーマットを条件付きにしてください。以下を使用します。

```
int getLevel()
int getLevel (Method)
boolean isLogging(level,method)
boolean level1 (method)
boolean level2 (method)
boolean level31 (method)
boolean level4 (method)
```

- 複数の java.lang.String を一緒に付与する必要がある情報をトレースする場合は、java.lang.String の代わりに java.lang.StringBuffer を使用します。java.lang.StringBuffer を使用するよりも、付与した方が早く済みます。また、前述の箇条書き項目のとおりに条件付きにしてください。
- 重要なコンストラクタにエントリまたは終了トレースポイントを追加する際は、コンストラクタ内の this() や super() メソッド呼び出しの前にトレースポイントを配備することはできません。this() や super() メソッド呼び出しの直後に、エントリトレースポイントを配備してください。
- 例外条件をトレースするには、終了トレースポイントを使用しないでください。メソッドが例外をスローする場合は、例外がスローされる直前に、エントリまたは終了トレースポイントと同じトレースレベルで、例外トレースポイントを使用します。詳細は、105 ページの「例外トレースポイントの使用方法」を参照してください。

  メソッドにエントリトレースポイントが含まれている場合は、そのメソッドに throwing メソッド、caught メソッド、または終了のいずれかを実行するコードパスも含めてください。

次に、簡易エントリと終了トレース文の例を示します。この例では、次の CLASS 変数が各文に宣言されているものとします。

```
private static final String CLASS =
"com.waveset.adapter.MyResourceAdapter";
protected static Trace _trace = Trace.getTrace();
```

例 5–1 エントリトレースポイントの一例

```
final String METHOD = methodName;
_trace.entry(_trace.LEVEL1, CLASS, METHOD);
```

例 5–1 エントリトレースポイントの一例 *(続き)*

```
_trace.entry(_trace.LEVEL1, CLASS, METHOD, user);
if (_trace.level1(CLASS, METHOD)) {
_trace.entry(_trace.ALWAYS, CLASS, METHOD,
user= + user);
}

// Show the size of an array argument
// ASSUME: password is an argument to the method
// Note the use of the Util.length() method. It is
// a convenience method that guards against null.
if (_trace.level1(CLASS, METHOD)) {
    StringBuffer sb = new StringBuffer(32);
    sb.append(password length=);
    ab.append(Util.length(password));
    _trace.entry(_trace.ALWAYS, CLASS, METHOD, sb.toString());
}
```

例 5–2 エントリトレースポイントの一例

```
_trace.exit(_trace.LEVEL1, CLASS, METHOD);

_trace.exit(_trace.LEVEL1, CLASS, METHOD, returnValue);
if (_trace.level1(CLASS, METHOD)) {
    _trace.exit(_trace.ALWAYS, CLASS, METHOD,
      returnValue != null ? returnValue.getName() : returnValue);
}

// Show the size of an array
String[] accounts = ...
if (_trace.level1(CLASS, METHOD)) {
    StringBuffer sb = new StringBuffer(32)
    sb.append(accounts length=);
    ab.append(accounts.length);
    _trace.exit(_trace.ALWAYS, CLASS, METHOD, sb.toString());
}
```

情報トレースポイントの使用方法

使用するコードに情報トレースポイントを追加する前に、これらのガイドラインをお読みください。

- 情報データトレースポイント、変数データトレースポイント、およびデータトレースポイントには通常、Trace.Level3 または Trace.Level4 を使用します。トレースレベルは、表示された情報の重要度、工数および詳細度によって決まります。
- 重要な条件、分岐、コードパスで発生する情報などをトレースするには、Trace.Level3 と Trace.Level4 の情報と変数トレースポイントを使用します。
- 変数トレースポイントは情報トレースポイントと同様に便利なメソッドですが、変数トレースポイントが引数値をプリントするだけでなく variable=<variable> をプリントアウトします。さらにこのメソッドは、引数に指定したトレースレベルが一致しない限りは情報をフォーマットしません。
- メソッドに例外条件があってもその例外を再スローしない場合は、情報トレースポイントを使用しないでください。caught メソッドを使用してください。たとえば、105 ページの「例外トレースポイントの使用方法」を参照してください。
- バイト配列で情報を表示するには、Trace.Level3 と Trace.Level4 データトレースポイントを使用します。その情報がそのトレースレベルに合っており短い場合は、下位のトレースレベルを使用してもかまいません。
- 重要だが短い情報をトレースするには、Trace.Level1 と Trace.Level2 情報トレースポイント、変数トレースポイント、およびデータトレースポイントが使用できます。ただし、このレベルのトレースポイントはほとんど使用しないでください。

次に、簡易情報トレース文の例を示します。この例では、次の CLASS 変数が宣言されているものとします。

```
private static final String CLASS =
"com.waveset.adapter.MyResourceAdapter";
protected static Trace _trace = Trace.getTrace();
```

例 5–3　情報トレースポイントの一例

```
_trace.info(_trace.LEVEL3, CLASS, METHOD, Some Message);
WavesetResult result = new WavesetResult();
try {
   someMethod();
}  catch(Exception e) {
   String msg = Some Error Message;
   WavesetException we = new Waveset(msg, e);
   _trace.caught(_trace.LEVEL3, CLASS, METHOD, e);
   result.addException(we);
}
```

例外トレースポイントの使用方法

使用するコードに例外トレースポイントを追加する前に、これらのガイドラインをお読みください。

- エントリまたは終了トレースポイントが指定されているすべてのメソッドにトレースポイントを使用します。また、例外を取得して再スローするメソッドにトレースポイントを使用します。Identity Managerは、新しい例外が元の例外をラップしている例外をスローすることがあります。エントリと終了トレースポイントで使用しているものと同じトレースレベルを使用してください。
- 現メソッドから例外が作成されてスローされる場合は、例外トレースポイントを使用します。
- 例外が取得されて処理される場合は、例外トレースポイント`caught`メソッドを使用します。たとえば、現メソッドから例外がスローされない場合などです。この状況は、Identity Managerが例外を取得して処理する際、後から調査しようと`WavesetResult`などのコンテナにそれを配備したときによく起こります。
- 現メソッドからスローされた例外が、`java.lang.RuntimeException`または`java.lang.Error`を拡張する例外の*checked*例外でなければ、ほかの措置が講じられるまでこのメソッドの終了ポイントはトレースされません。この状況で例外トレースポイントを設定するには、`try/catch`ブロックをそのメソッドの重大域の周りに使用して、発生する例外を再スローする方法があります。つまり、メソッドが成功したか失敗したかを調べることが欠かせないときは、通常`Trace.Level3`以上の使用が必要になります。

次に、簡易例外トレース文の例を示します。この例では、次の`CLASS`変数が宣言されているものとします。

```
private static final String CLASS =
"com.waveset.adapter.MyResourceAdapter";
protected static Trace _trace = Trace.getTrace();
```

例5–4 例外トレースポイントの一例

```
try {
   someMethod();
}  catch(Exception e) {
   _trace.throwing(_trace.ALWAYS, CLASS, METHOD, e);
   throw e;
}


try {
   someMethod();
}  catch(Exception e) {
   _trace.throwing(_trace.ALWAYS, CLASS, METHOD, e);
   WavesetException we = new WavesetException(Some Message, e);
```

例 5–4 例外トレースポイントの一例 (続き)

```
   throw we;
}


if (error) {
  WavesetException e = new WavesetException(Some Error Message.;
  _trace.throwing(_trace.LEVEL3, CLASS, METHOD, e);
  throw e;
}

try {
   someMethod();
}  catch(Exception e) {
   _trace.caught(_trace.LEVEL1, CLASS, METHOD, e);
}
// execution continues.
someOtherMethod();
```

トレースポイントのテンプレートの使用方法

トレースポイントのテンプレートを使用すると、Identity Manager リソースアダプタに一貫した有用なトレースポイントを作成できるようになります。Identity Managerには、次の場所に Eclipse テンプレートが用意されています。

```
src/wps/doc/eclipse-trace-templates.xml
```

これらのテンプレートを Eclipse にインポートするには、「ウィンドウ」>「環境設定」>「Java」>「エディタ」>「テンプレート」の順に選択します。

注 – Emacs または IDEA を使用している場合は、同様のテンプレートが作成できます。

その他のトレースガイドライン

ここでは、次を含む補足的なトレースガイドラインを説明します。

- 106 ページの「内部クラスのトレース」
- 107 ページの「静的初期化子のトレース」

内部クラスのトレース

重要メソッドは内部クラスでトレースしてください。内部クラスに何かトレースメソッドがある場合は、最終的に静的な CLASS 変数を必ず宣言してください。

例 5–5　内部クラスでCLASS変数を使用した例

```
private final static String CLASS =
com.waveset.adapter.SomeResourceAdapter$AdapterInnerClass;
```

静的初期化子のトレース

通常は、静的初期化子にトレース機能を使用しないでください。Identity Managerは、リポジトリが初期化されたとき起こるクラスのロード時にこの初期化子を実行します。静的初期化子で重要なものをトレースするには、Debugクラスを使用します。

トレース監査機能

Identity Auditorを使用すると、次のメソッドをトレースして問題をトラブルシューティングできます。

- com.sun.idm.auditor.policy - Audit Scanの問題をトレースします。
- com.sun.idm.auditor.accessreview - Access Reviewsの問題をトレースします。
- com.sun.idm.auditor.report - Audit Reportsの問題をトレースします。
- com.sun.idm.auditor.view - Auditor Viewsの問題をトレースします。

▼ トレースを有効にする

1. ブラウザを開き、管理者インタフェースにログインします。
2. 「設定」>「サーバー」の順に選択します。
3. 「サーバー設定」ページが表示されたら、「サーバー」列からサーバー名をクリックしてその設定を編集します。
4. 「サーバー設定の編集」ページで、「スケジューラ」タブをクリックします。
5. スケジューラのデバッグトレースを起動してその結果をstdoutに書き込むには、「トレースの有効化」ボックスを選択します。
6. 変更を保存します。

カスタムコードのトレース

使用中の配備をトラブルシューティングする際にシームレス統合を実現するには、カスタム記述コードにIdentity Managerのトレース機能を使用します。

このトレース機能を使用するには、次のクラスを使用します。

- トレースの下位システムとを接続するには、`com.sun.idm.logging.trace.Trace` オブジェクトを使用します。
- これらのオブジェクトのファクトリには、`com.sun.idm.logging.trace.TraceManager` を使用します。

この `com.sun.idm.logging.trace` 機能は、トレース出力の記録にも使用できます。この機能とクラスの詳細は、Javadoc™ を参照してください。

注 – この `com.sun.idm.logging.trace` 機能は、Identity Manager Version 6.0 SP1 リリース以前には搭載されておりません。これ以前の Identity Manager には、代わりに `com.waveset.util.Trace` クラスを使用してください。

トレースの例外

例外ログは、Identity Manager の「デバッグ」ページまたは `config/Waveset.properties` ファイルから表示できる静的トレースです。トレースデータには、デフォルトですべての例外が含まれていませんが、例外ログは重要で有益なトラブルシューティングツールになります。

例外ログを有効にするには、次のうちいずれかのメソッドを使用します。

- Identity Manager の管理者インタフェースから「Waveset プロパティー」ページ (`debug/Show_WSProp.jsp`) を開きます。`exception.trace` キーを見つけて、値を **`true`** に変更します。
- `config/Waveset.properties` ファイルをテキストエディタで開いて、`exception.trace` キー値を **`true`** に変更します。次の図は、例外トレースの一例を示したものです。

Identity Managerは、アプリケーションサーバーのコンソールになることが多いWebアプリケーションインスタンスにある例外ログを`stdout`に送信します。

注–完了したら、アプリケーションサーバーログに不要に出力されないよう、例外ログを無効にします。例外ログを無効にするには、`exception.trace`キー値を**`false`**に戻します。

フォームのトレース

トレースを有効にすると、フォームフィールド内の編集済みフォームをトラブルシューティングし式文エラーをチェックできます。

トレースを有効にするには、次のうちいずれかの方法を使用します。

- Identity Managerの管理者インタフェースから「Waveset Properties」ページ(`debug/Show_WSProp.jsp`)を開きます。`form.trace`キーを見つけて、値を**`true`**に変更します。
- `config/Waveset.properties`ファイルをテキストエディタで開いて、`form.trace`キー値を**`true`**に変更します。

Identity Managerは、フォームの式構文に問題があれば標準の出力にレポートします。

注 – `form.trace` キーはデフォルトで無効になっています。これは、「アカウントリスト」などの各ページの各フィールドにトレース情報を生成するのでシステムパフォーマンスに影響を及ぼすためです。メソッドをトレーシングするフォームとフィールドには、より対象に近いものを使用するようにしてください。

フォームのトラブルシューティングが完了したら、 `form.trace` キー値を **`false`** に変更し直してトレースを無効にしてください。

フォームとフォームのプロセスを開発したり更新している最中には、グローバル XPRESS のトレースも有用です。グローバル XPRESS のトレースは、システムパフォーマンスに影響するほど大量の出力が生成されますが、このトレース方法は XPRESS 出力を表示して、フォームのどこに問題が発生しているかがわかることがあります。

詳細は、次を参照してください。

- 110 ページの「「グローバル XPRESS のトレース」」
- 『Sun Identity Manager Deployment Reference』の「Testing Your Customized Form」

「グローバル **XPRESS** のトレース」

通常はお勧めしていませんが、コードがどこにあっても XPRESS コードをトレースするには、グローバル XPRESS トレースが使用できます。たとえば、フォーム、ビュー、およびワークフロー内で XPRESS コードがトレースできます。結果のトレースには、潜在的な問題がわかる XPRESS 出力が表示されます。

注 – XPRESS のトレースはシステムパフォーマンスに影響するほど大量の出力が生成されるため、デフォルトで無効に設定されています。

XPRESS 関数のトレースの詳細は、『Sun Identity Manager Deployment Reference』の「Testing Your Customized Form」を参照してください。

▼ グローバル **XPRESS** のトレースを有効にする

1. コマンドウィンドウを開きます。
2. **Identity Manager** のデフォルトのインストール先ディレクトリにある `config/Waveset.properties` にディレクトリ移動します。
3. `config/Waveset.properties` ファイルを開き、`xpress.trace` 行を次のように編集します。

   ```
   xpress.trace=true
   ```

4 Waveset.properties ファイルを保存します。

5 アプリケーションサーバーを再起動するか、Waveset.properties ファイルを **Identity Manager** のデバッグページから再読み込みします。

6 次の行を Waveset.properties ファイルに追加して、**XPRESS** トレース出力をファイルにコピーします。

```
xpress.traceFile=FileName.txt

xpress.traceFileOnly=true
```

Waveset.properties に xpress.traceFileOnly=true を設定すると、xpress.traceFile に指定したファイルの中に、すべての XPRESS 文の評価によってトレースメッセージが生成されます。それ以外の場合は、 xpress.traceFile に値があれば、トレースメッセージがコンソールとファイルの両方に出力先が変更されます。

PasswordSyncのトレース

ここでは、PasswordSync のトレースを有効にする方法と、Direct アクセスまたは JMS モードでトレースを設定する方法について説明します。

PasswordSyncのトレースを有効にする

Identity Manager の PasswordSync 機能のトレースを設定するには、次の方法があります。

- 111 ページの「PasswordSync 設定ツールの使用方法」
- 112 ページの「レジストリキーの編集方法」

PasswordSync 設定ツールの使用方法

ここでは、PasswordSync 設定ツールの「Trace」タブからトレースを設定する方法について説明します。

注 – PasswordSync のインストール方法と設定方法の詳細は、『Sun Identity Manager 8.1 ビジネス管理者ガイド』の第 11 章「PasswordSync」を参照してください。

この設定ツールを初めて実行したときは、ウィザードでトレースを設定できません。それ以降は、設定ツールを実行するとウィザードに「Trace」タブが設定され、ここからトレースを設定できるようになります。

次の図は、PasswordSync 設定ツールの「Trace」タブを表したものです。

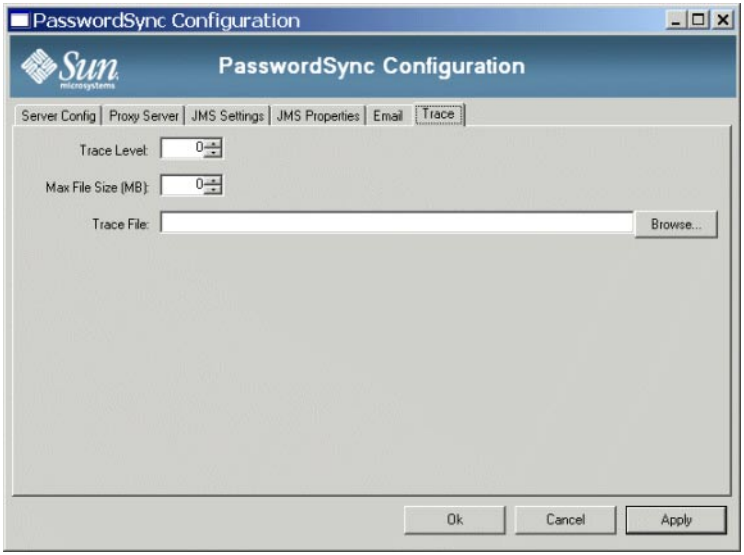


図 5–2 PasswordSync設定ツールの「Trace」タブ

このタブから指定できる操作は、次のとおりです。

- PasswordSyncがトレースログの書き込み時に表示する詳細レベルを指定するには、「Trace Level」フィールドを使用します。 `0` の値はトレースを無効にし、`4` の値は全詳細を表示します。
- ログファイルの最大サイズを指定するには、「Max File Size」フィールドを使用します。

  トレースファイルが「Max File Size (MB)」フィールドに指定したサイズを超えると、PasswordSyncが新しいトレースファイルを開始して、古い方のトレースファイル名に `.bk` を付与します。たとえばトレースレベルが100 Mバイトに設定されている場合、トレースファイルは `C:\logs\pwicsvc.log` に書き込まれますが、このトレースファイルのサイズが100 Mバイトを超えるとPasswordSyncがファイル名を `C:\logs\pwicsvc.log.bk` に変更します。するとPasswordSyncが新しい `C:\logs\pwicsvc.log` ファイル を作成して、ここに新しいトレースファイルのメッセージが書き込まれるようになります。
- PasswordSyncトレースファイルの場所を指定するには、「Trace File」フィールドを使用します。

レジストリキーの編集方法

別のPasswordSync設定を有効にするには、PasswordSync設定ツールを使用して次のPasswordSyncレジストリキーを編集します。

注 – PasswordSyncレジストリキーを編集するには、PasswordSync設定ツールを使用するのが一番安全な方法です。これらのキーを直接Windowsレジストリで編集することはお勧めしません。

表5–3 レジストリキー

キー名	種類	説明

dumpFilebase	REG_SZ	PasswordSync DLLが例外を表示する場合は、Windowsがダンプファイルを生成できるようにこのレジストリキーを設定します。
		メモリーダンプの書き込み先となる、省略されていないディレクトリパスに、このレジストリキーを設定する必要があります。例: `c:\temp`
		パスワード処理中にIdentity Managerが例外を取得するたびにメモリーダンプを書き込むには、このレジストリキーを設定します。
		注意: Windows 2000サーバー(全サービスパック)では、Microsoftから入手できる設定ディレクトリ `DbgHelp.dll` もインストールする必要があります。`DbgHelp.dll` ファイルの最低リリースバージョンは、Version 5.1です。`DbgHelp.dll` ファイルは次からダウンロードしてください。
		`http://www.microsoft.com/whdc/DevTools/Debugging/default.mspx`
		`DbgHelp.dll` がインストールされていなければ、Windows 2000にはダンプファイルが生成されません。
		ダンプファイル名の形式は、次のとおりです。
		`lhpwic-YYYYMMDD-HHmm-xxxxx.dmp`
		この名前の `YYYYMMDD` はダンプ日、`HHmm` はダンプ時刻(24時間制)、`xxxxx` は、アプリケーションのスレッド番号になります。
		ダンプファイルは手動で削除する必要があります。ダンプファイルのサイズは、Windowsの Local Security Authority Subsystem (LSASS) プロセスのサイズに応じて変わりますが、20 Mバイトから100 Mバイト以上です。このダンプファイルを削除しないと、時間が経つにつれて、ディスク空き容量の限られたシステムがいっぱいになります。

installdir	REG_SZ	PasswordSyncアプリケーションのインストール先ディレクトリ

PasswordSyncレジストリキーは、次の場所にあります。

`HKEY_LOCAL_MACHINE\SOFTWARE\Waveset\Lighthouse\PasswordSync`

ほかのキーは、この場所にあります。

各種モードのログを収集する

直接アクセスモードを使用していてもJMSモードの設定を使用していても、PasswordSyncのトレースログは変わりません。ただし、これらのトレースログには、部分情報しか表示されません。各設定にサーバー側にログを収集するには、次のセクションで説明するとおりにそれぞれのクラスを設定する必要があります。

直接モードのトレース

直接アクセスモードでPasswordSyncを使用すると、トレースログにはエラーが表示されますが、ログの中のすべてのエラーが実際のエラーとは限りません。たとえば、場合によってはビューのチェックインに時間が掛かりすぎてログにエラーが表示されることもあります。この情報を表示するには、サーバー側でトレースする必要があります。

直接モードでは、チェック対象のビューをリポジトリに生成するサーブレットとPasswordSyncが通信します。パスワードの変更内容の受信から、サーブレットに生成され返されたものの応答まで、すべてのパスワード同期段階を表示するには、`com.waveset.rpc.GenericMessageHandler`をクラスレベル4でトレースします。レベル4は、トラブルシューティングに十分な詳細情報を提供できる唯一のレベルです。

JMSモードのトレース

JMSモード設定でPasswordSyncを使用すると、ログには送信が成功したか失敗したかの情報しかJMSサーバーに表示されません。この点で、サーバー側のログの方を頼る必要があります。JMSトレースの方が、多少複雑です。

PasswordSync `dll`をJMSメッセージに生成したメッセージを変換し、そのメッセージをJMSキューに追加するには、`com.waveset.rpc.PasswordSyncHandler`クラスをレベル4でトレースします。このクラスではトレースの制限が可能で、トラブルシューティングに役立つ十分な情報を提供できるのはレベル4だけです。

PasswordSyncが正常にJMSメッセージをJMSキューに送信した場合は、トレースを見ても問題の原因が見つからないでしょう。次に、最終手段としてJMSアダプタをトレースします。この手順については、『Sun Identity Manager 8.1 リソースリファレンス』を参照してください。

ルール駆動型メンバーキャッシュのトレース

ルール駆動型メンバーのキャッシュをトレースしてその結果を基に、`Waveset.properties`のキャッシュプロパティーをチューニングできます。

com.waveset.server.RuleDrivenMembersCache をレベル 1 でトレースする場合は、結果の情報に追加数、削除数、キャッシュヒット数などが記載されます。この情報を基にキャッシュサイズを評価して、Waveset.properties のキャッシュプロパティーをチューニングする必要があるかどうかを判断します。

ルール駆動型メンバーのキャッシュを制御するには、Waveset.properties の次のプロパティーを使用します。

- サブジェクト単位でキャッシュされる最大オブジェクトリスト数を指定するには、ruledrivenmemberslistcache.size = 値を使用します。(デフォルトは 20 です。)
- サブジェクト単位でキャッシュされる最大オブジェクト総数を指定するには、ruledrivenmemberslistcache.rowlimit = 値を使用します。(デフォルトは 100000 です。)

デフォルトでは Identity Manager が、指定の組織に関連付けられているユーザーメンバールールを評価し、ユーザーの動的リストがあるユーザーメンバーリストのキャッシュを作成します。ただし、所定サブジェクトの指定ユーザー組に動的メンバー組織があるユーザーメンバーリストのキャッシュも作成できます。このキャッシュのキーは、オブジェクト ID で連結されたオブジェクト型です。たとえば、User#ID#Configurator オブジェクト ID と連結された User オブジェクト型です。それぞれキーとなる値は、このオブジェクトが動的メンバーとなるオブジェクトグループのリストです。

評価対象のオブジェクトが動的メンバーかどうかを判定するには、リストキャッシュで使用されるものと同じ組織単位のユーザーメンバールールをキャッシュが評価します。そのオブジェクトが動的メンバーであれば、Identity Manager がそのオブジェクトをリストに追加してからリストをキャッシュします。Identity Manager は、キャッシュのヒット率が最高になるよう、空と非空の両方のリストをキャッシュします。

このキャッシュには、Waveset.properties の次のプロパティーを使用して、パフォーマンスに影響する必要メモリーを制御します。

- サブジェクト単位でキャッシュされるメンバーオブジェクトグループリスト数を指定するには、ruledrivenmembersobjectcache.size = 値を使用します。デフォルト値は 100 です。
- サブジェクト単位でキャッシュされる最大メンバーオブジェクトグループ総数を指定するには、ruledrivenmembersobjectcache.rowlimit = 値 を使用します。デフォルト値は 100000 です。

Identity Manager Service Provider 委任管理のトレース

Identity Manager がメソッド/クラス レベル 2 でトレースできるように有効に設定すると、Identity Manager Service Provider (サービスプロバイダ) ユーザーをリストするかアクセスしたとき、およびサービスプロバイダ ユーザーがログインしたときに AdminRoles が動的に割り当てられていると認証フローがトレースできるようになります。

- サービスプロバイダ ユーザーを検索するときは `com.sun.idm.idmx.view.IDMXBrowseViewer` をトレースします。
- サービスプロバイダ ユーザーを作成、編集、または削除するときは `com.sun.idm.idmx.view.IDMXUserViewer` をトレースします。
- 上記クラスの両方を使用するときは `com.sun.idm.idmx.api.IDMXServerUtils` をトレースします。
- サービスプロバイダ ユーザーとしてログインしているときは、`com.waveset.security.authn.WSSPELoginModule` をトレースします。

注 – Identity Manager のデバッグページから サービスプロバイダ がトレースするように設定します。手順については、98 ページの「Identity Manager サーバーのトレース」を必要に応じて参照してください。

調整のトレース

調整タスクに問題がある場合は、`com.waveset.task.Reconciler` に標準のトレース機能を使用して、調整サーバーをトレースします。

トレースを有効にするには、次のうちいずれかの方法を使用します。

- Identity Manager の管理者インタフェースから「Waveset Properties」ページ (`debug/Show_WSProp.jsp`) を開きます。`exception.trace` キーを見つけて、値を **`true`** に変更します。
- `config/Waveset.properties` ファイルをテキストエディタで開いて、`exception.trace` キー値を **`true`** に変更します。

有用なデバッグ情報を表示するには、記載されているメソッド/クラスのトレースレベルで「System Settings 」ページからトレースして、次の調整メソッドをトレースすることもできます。

表5–4 トレース対象となる調整メソッド/クラス

トレース対象となるcom.waveset.recon.メソッド/クラス	実行すべきトレースレベル	表示対象
ReconTask$WorkerThread#reconcileAccount	2	調整されている個々のアカウント
ReconTask$WorkerThread#performResponse	2	応答中の個々のアカウントとユーザー
ReconUtil#deleteAccountIndex	2	アカウントインデックスから削除対象となるユーザー情報。
UserContext#aquireRepoLock	2	更新にロックされているユーザー。
UserContext#releaseRepoLock	2	ユーザー。リポジトリでロック解除されているユーザー。
ReconUtil#deleteAccountIndex	2	アカウントインデックスから削除対象となるユーザー情報。
UserContext#aquireRepoLock	2	更新にロックされているユーザー。
ReconTask$WorkerThread#failUserExamination	2	エラーで失敗した、すべてのユーザー検査要求
ReconTask$WorkerThread#failUserResponses	2	エラーで失敗した、すべてのユーザー応答要求
UserContext#releaseRepoLock	2	ユーザー。リポジトリでロック解除されているユーザー。
ReconTask$ResourceThread#examineResource	3	リソースから読み込まれたアカウント数。
ReconTask$ResourceThread#queueAccountReconciles	3	accountId 、accountGUID、accountDisabled など、リソースから読み込まれた各アカウントの情報
ReconTask$ResourceThread#examineLighthouse	3	キューに入れられた調整済みリソースにアカウントを所有するために使用する Identity Manager ユーザークレーム数。
ReconTask$WorkerThread#findClaimants	3	リソースのアカウントを要求するすべての Identity Manager ユーザー。
ReconTask$WorkerThread#confirmPossibleOwners	3	リソースアカウントのすべての確定済み所有者のリスト。
ReconTask$WorkerThread#applyResponse	3	適用されている応答リスト。
AccountContext#processAttributeWorkflow	3	属性変更ワークフロー開始中の、属性変更内容と、フォーマット済み変更内容。
ReconUtil#getRuleState	3	ルール処理中の、ユーザー属性を示したフルユーザービュー。
ReconUtil#evaluateCorrelationRule	3	相互規則の状態の値と、検査の結果。
ReconUtil#confirmPotentialOwners	3	確定ルールを使用して確定されたユーザーのリスト。

表 5–4 トレース対象となる調整メソッド/クラス (続き)

トレース対象となる com.waveset.recon. メソッド/クラス	実行すべきトレースレベル	表示対象
ReconUtil#getIfExistsAccountIndexEntry	3	指定したエントリに対するアカウントインデックスの検査に関する情報の出力。
ReconUtil#launchWorkflow	3	開始時のタスクインスタンスとタスク定義情報。
ReconUtil#indexFoundAccount	3	存在が判明しているアカウントのインデックスの作成中または更新中に記録されたアカウントと状況。
ReconUtil#indexMissingAccount	3	存在が判明していないアカウントのインデックスの作成中または更新中に記録されたアカウントと状況。
ReconUtil#listAccountsIndexSaysExist	3	インデックスに存在が表示されているアカウント情報。
ReconTask$ResourceThreadwaitForLighthouseWorkItems#	4	Identity Managerユーザー検査プロセス中に、キューに入れられ処理されたユーザー数。
ReconTask$ResourceThread#waitForReconcileWorkItems	4	指定リソースのキューに入れられ処理された調整と応答数。
ReconTask$WorkerThread#correlateUsers	4	相互に関係するユーザーまたは確定ユーザーのリスト。
ReconTask$WorkerThread#respondOrRequeue	4	アカウントに対し、前の応答メソッドの処理に問題があれば示されるエラーメッセージ。
Response#perform	4	修正が行われる前と行われた後のユーザー、応答、およびユーザーのリソース情報XML。
Response#perform	4	ユーザーの応答アクションの概要。
Response#createNewUserFromAccount	4	ユーザーの応答アクションの全詳細。
ReconUtil#evaluateConfirmationRule	4	確定規則の状態の値と、検査の結果。
ReconUtil#getCorrelatedUsers	4	指定ルールの結果に一致する相互関係ユーザーのリスト。

注– トレースレベルが高いほど、トレースファイルが大きくなります。また、これらのすべてのメソッドを同時にトレースすると、非常に大きなトレースファイルが作成されます。

トラブルシューティングが完了したら、exception.trace キー値を **false** に設定し直してトレースを無効にしてください。

setRepoコマンドのトレース

setRepoコマンドを使用したIdentity Managerリポジトリの設定中にエラーが表示されたら、次のフラグを使用して問題を切り離し、デバッグします。

```
-Dtrace.enabled=true
-Dtrace.level.com.waveset.repository.AbstractDataStore=2
-Dtrace.level.com.waveset.repository.DefaultTypeHandler=4
// Use one of the following based on your repository type
-Dtrace.level.com.waveset.repository.OracleDataStore=4
-Dtrace.level.com.waveset.repository.SqlServerDataStore=4
-Dtrace.level.com.waveset.repository.MysqlDataStore=4
-Dtrace.level.com.waveset.repository.DB2DataStore=4
```

Identity Managerが`setRepo`コマンドからデフォルトの`$WSHOME/config/WSTrace.log`ファイルに出力を送信します。

SPMLのトレース

ここでは、SPML Version 1.0とSPML Version 2.0のトレースを有効にするためのメソッドについて説明します。

SPML 1.0のトレースを有効にする

SPML 1.0には、Identity ManagerのSPMLトラフィックと診断問題をログできるよう、トレース出力のチューニングに次のオプションが用意されています。

メソッド1: setTraceメソッドの有効化

SPML 1.0のトレースを有効に設定するには、`SpmlClient`と`LighthouseClient`クラスから提供される`setTrace`メソッドを使用します。

この`setTrace`メソッドを有効にすると、クライアントから送信された要求のXMLと、サーバーから受信した応答のXMLが、随時送受信時にクライアントコンソールに印字されます。

`setTrace`メソッドは、Boolean引数を使用します。例を示します。

```
SpmlClient client = new SpmlClient();
 client.setURL("http://localhost:8080/idm/spml");
 client.setTrace(true);
```

メソッド 2: org.openspml.server.SOAPRouter サーブレットの初期化

サードパーティである OpenSPML 団体のオープンソースクラスである org.openspml.server.SOAPRouter サーブレットの初期化時に、トレースを有効にできます。このサーブレットは、サーブレットが SPML 要求を処理できるよう、RPC トラフィック情報の出力を制御します。

このメソッドを有効にするには、次を WEB-INF/web.xml ファイルに追加します。

```
<servlet>
    <servlet-name>rpcrouter2</servlet-name>
    <display-name>OpenSPML SOAP Router</display-name>
    <description>no description</description>
    <servlet-class>
        org.openspml.server.SOAPRouter
    </servlet-class>
    <init-param>
        <param-name>trace</param-name>
        <param-value>true</param-value>
    </init-param>
    ...
  </servlet>
```

次に、SPML 1.0 トレースの出力例を示します。

```
SpmlClient: sending to http://example.com:8080/idm/servlet/rpcrouter2
<spml:addRequest xmlns:spml='urn:oasis:names:tc:SPML:1:0'
xmlns:dsml='urn:oasis:names:tc:DSML:2:0:core'>
  <spml:operationalAttributes>
    <dsml:attr name='session'>


<dsml:value>session token</dsml:value>

    </dsml:attr>
  </spml:operationalAttributes>
  <spml:identifier type='urn:oasis:names:tc:SPML:1:0#GUID'>
    <spml:id>suetonius</spml:id>
  </spml:identifier>
  <spml:attributes>
    <dsml:attr name='objectclass'>
      <dsml:value>person</dsml:value>
    </dsml:attr
    <dsml:attr name='password'
      <dsml:value>password</dsml:value>
    </dsml:attr>
    <dsml:attr name='gn'>
      <dsml:value>Suetonius</dsml:value>
```

```
    </dsml:attr>
    <dsml:attr name='sn'>
      <dsml:value>Tranquillus</dsml:value>
    </dsml:attr>
    <dsml:attr name='email'>
      <dsml:value>twelve@example.com</dsml:value>
    </dsml:attr>
  </spml:attributes>
</spml:addRequest>


SpmlClient: received
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'>
<SOAP-ENV:Body>
<spml:addResponse xmlns:spml='urn:oasis:names:tc:SPML:1:0'
xmlns:dsml='urn:oasis:names:tc:DSML:2:0:core' result='urn:oasis:names:tc:
SPML:1:0#success'>
  <spml:operationalAttributes>
    <dsml:attr name='session'>

<dsml:value>session token</dsml:value>
    </dsml:attr>
  </spml:operationalAttributes>
  <spml:identifier type='urn:oasis:names:tc:SPML:1:0#GUID'>
    <spml:id>suetonius</spml:id>
  </spml:identifier>
</spml:addResponse>
/SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

注 – SOAP `rpcrouter` サーブレットの詳細は、OpenSPML Toolkit マニュアルを参照してください。

メソッド 3: `trace` オペレーショナル属性の渡し

個々の SPML RPC 要求のトレースを有効に設定するには、`trace` オペレーショナル属性をサーバー側の RPC 要求に渡します。

サーブレット初期化中にトレースが実行され、サーブレットの RPC トラフィックが SPML Version 1.0 の要求を処理できるように、情報を出力する方法を制御します。たとえばこのトレースは raw XML を出力します。これは `System.out` が (アプリケーションコンテナの関数である) そのサーブレット用のものかどうかにかかわらず、行ったりきたり送信されます。たとえば、次のようになります。

```
AddRequest ar = new AddRequest();
 ar.setOperationalAttribute("trace", "true");
```

trace 属性を使用したとき、属性がサーバー処理に影響を与える度合いはベンダーによって異なります。現在のところ、Identity Manager は raw 要求データと応答データをサーバーコンソールに出力しています。これは、クライアントアプリケーションがコンソールウィンドウに関連付けられていない場合に有用です。

詳細は、OpenSPML Toolkit 製品のマニュアルを参照してください。

SPML 2.0 のトレースを有効にする

SPML 2.0 には、Identity Manager の SPML トラフィックと診断問題をログできるよう、トレース出力のチューニングに次のオプションが用意されています。

メソッド 1: org.openspml.v2.transport.RPCRouterServlet サーブレットの使用

SPML 1.0 と同様、サーブレットが SPML 2.0 要求を処理できるように、RPC トラフィック情報の出力を制御する org.openspml.v2.transport.RPCRouterServlet クラスの初期化時に、SPML 2.0 用のトレースが有効に設定できます。

このメソッドを有効にするには、次を WEB-INF/web.xml ファイルに追加します。

```
<servlet>
    <servlet-name>openspmlRouter</servlet-name>
    <display-name>OpenSPML SOAP Router</display-name>
    <description>A router of RPC traffic - nominally SPML 2.0 over SOAP</description>
    <servlet-class>
        org.openspml.v2.transport.RPCRouterServlet
    </servlet-class>
    <init-param>
      <param-name>trace</param-name>
      <param-value>true</param-value>
    </init-param>
    ...
  </servlet>
```

次の例は、rpcrouter サーブレットトレースからの出力を表したものです。

```
RPCRouterServlet:
<?xml version='1.0' encoding='UTF-8'?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header/>
<SOAP-ENV:Body><lookupRequest
xmlns='urn:oasis:names:tc:SPML:2:0' requestID='random name' executionMode='synchronous'
returnData='everything'>
<openspml:operationalNameValuePair xmlns:openspml='urn:org:openspml:v2:util:xml' name='
```

```
session'value=session token'/>
<psoID ID='random name' targetID='spml2-DSML-Target'/>
</lookupRequest>
</SOAP-ENV:Body></SOAP-ENV:Envelope>

RPCRouterServlet:  response:
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope

  xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'>
<SOAP-ENV:Body>
<lookupResponse xmlns='urn:oasis:names:tc:SPML:2:0' status='failure' requestID='random
name'error='noSuchIdentifier'>
<openspml:operationalNameValuePair xmlns:openspml='urn:org:openspml:v2:util:xml'
name='session'

value=session token/>
<errorMessage>Item User:random name was not found in the repository, it may have been
deleted in another session.</errorMessage>
</lookupResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

注 – 詳細は、119 ページの「SPML 1.0 のトレースを有効にする」セクションの 120 ページの「メソッド 2: `org.openspml.server.SOAPRouter` サーブレットの初期化」を参照してください。

メソッド 2: SPML アクセスログの使用

SPML 2.0 は、有用なトラブルシューティングツールとして使用できる、簡易テキストのアクセスログです。このログは常に使用可能で、受信された要求の種類、その要求の処理に掛かった所要時間、要求が完了したかどうかなどの情報を、トレースを有効に設定しなくても表示できます。

この SPML テキストのアクセスログの設定手順は、『Sun Identity Manager 8.1 Web Services』の「Configuring SPML Tracing」に記載されています。

タスクスケジューラのトレース

スケジューラタスクに問題がある場合は、`com.waveset.task.Scheduler` にある標準トレース機能を使用してタスクスケジューラをトレースできます。この出力には、タスクスケジュールについての詳細情報が表示されます。

▼ トレースを有効にする

1. ブラウザを開き、管理者インタフェースにログインします。
2. 「設定」>「サーバー」の順に選択します。
3. 「サーバー設定」ページが表示されたら、「サーバー」列からサーバー名をクリックしてその設定を編集します。
4. 「サーバー設定の編集」ページで、「スケジューラ」タブをクリックします。
5. スケジューラのデバッグトレースを起動してその結果を `stdout` に書き込むには、「トレースの有効化」ボックスを選択します。
6. 変更を保存します。

注 –

- クラスター化された環境で、各インスタンスでトレースが実行されます。
- トレース設定オブジェクトの定義と編集方法、およびトレースファイルの表示方法については、98 ページの「Identity Manager サーバーのトレース」を参照してください。

ワークフローのトレース

ワークフローのトレースを有効にすると、ワークフローアクティビティの問題を解決して、ワークフロープロセスが把握できるようになります。

注 –

- クラスター化された環境で、各インスタンスでトレースが実行されます。
- 複数台のサーバー配備でワークフローをデバッグするには、1 台のサーバーを残してすべてシャットダウンするようにしてください。すべてのサーバーが実行中の場合は、環境内でどのサーバーがワークフローを実行中かを判断できず、トラブルシューティング問題の原因が特定できません。

ワークフローのトレースを有効にする

ワークフローのトレースを有効にするには、次の項目を実行します。

▼ ワークフロートレースを有効にする

1 ブラウザを開いて、**Identity Manager** 管理者インタフェースにログインします。

2 「**System Settings**」ページの「**List Objects Type**」メニューから「**Configuration**」を選択します。

3 「**List Objects**」ボタンをクリックします。

4 「**List Objects of type: Configuration**」ページが表示されたら、オブジェクトリストをスクロールダウンして **System Configuration** オブジェクトを見つけ、その編集リンクをクリックします。

5 「**Checkout Object: Configuration,** `#ID#CONFIGURATION:SYSTEMCONFIGURATION`」ページが表示されたら、`SystemConfiguration` オブジェクトの次のいずれの `workflow` オプションでも編集できます。

注 – 通常、有効にするオプションは 1 つだけですが、同時に複数のオプションを有効にすることもできます。

これらの属性は、互いに依存していません。一種類のトレースを有効にしながら、他の種類を無効に設定することが可能です。

- アプリケーションサーバーのコンソールにワークフローのトレースメッセージをリダイレクトするには、`workflow.consoleTrace=`**`true`** を指定します。この属性の方が `workflow.fileTrace` よりも多くのトレース出力を出力するため、致命的な例外でワークフローが終了したときに役立ちます。(デフォルト値は `false` です。)
- ワークフローのトレースメッセージを読みやすいファイルにリダイレクトするには、`workflow.fileTrace=`*PathtoFile* を指定します。この属性には、デフォルトで値がありません。

  ```
  <Attribute name='fileTrace'/>
  ```

  `workflow.fileTrace` 属性に値タグを付け、Unix 式の普通のスラッシでログファイルにパスを入力します。Identity Manager が、アプリケーションサーバーのインストールディレクトリに関する相対パスを格納します。例を示します。

  - **Windows** の場合。 `<Attribute name='fileTrace' value='C:\mydir\workflow.txt'/>`
  - **Solaris/UNIX** の場合。 `<Attribute name='fileTrace' value='/mydir/workflow.txt'/>`
- 表示するワークフローのトレースレベルをを指定するには、`workflow.traceLevel=`*tracingLevel* を指定します。

ワークフロープロセスをトレースするには、workflow.Trace=**true** を指定します。このオプションでトレースを開始するには、アプリケーションサーバーを再起動する必要があります。Identity Managerが、タスクのWavesetResultオブジェクトにトレース結果を格納します。ファイルシステムにアクセスできない場合は、このトレースオプションを使用してください。(デフォルト値はfalseです。)

- タスクのWavesetResultにあるメッセージをトレースします。(デフォルト値は1です。)

workflow.Trace=true を指定すると、読みにくくて長い未フォーマットの1つの文字列にトレースメッセージが付与されます。このオプションは、ファイルシステムにアクセスできない場合のみ使用してください。

注 – 最初の2つのオプションを使用すると、致命的な例外が発生した場合に、ワークフローのトレースの一部を紛失してしまうことがあります。

---

6 SystemConfigurationオブジェクトを保存します。

7 アプリケーションサーバーを再起動するか、SystemConfiguration オブジェクトを**Identity Manager**のデバッグページから再読み込みします。

指定のワークフローのトレースを有効にする

指定のワークフローのトレースを有効にするには、次のいずれかの方法を用います。

- **WFProcess**定義の編集。特定のプロセスに条件付きトレースを有効にするには、trace='consoleの属性を<WFProcess>要素に追加して、TaskDefinitionオブジェクトのXMLを編集します。
- ワークフロー変数の編集。ワークフロー変数を使用すると、トレースのタイミングをより細かく制御できるようになります。ワークフローエンジンは、traceという最上位のワークフロー変数を検索します。

次の例は、ワークフロー変数のトレース方法の一例を示したものです。

例**5–6** ワークフロー変数のトレース

```
<Variable name='trace'>
   <cond><eq><ref>accountId</ref><s>jfaux</s></eq>
     <s>workflowTrace.txt</s>
   </cond>
 </Variable>
```

trace 変数は、そのワークフローが jfaux というユーザー名で処理されている場合だけ、トレースを有効にします。さらに、トレースを対話式に制御するには、フォームフィールドの trace を指定します。

たとえば、トレース出力は workflowTrace.txt ファイルに書き込まれます。

バージョン情報の調べ方

現在使用中の Identity Manager バージョンの情報を表示するには、次のいずれかの方法を用います。

- Identity Manager アプリケーションウィンドウの右上にある「ヘルプ」ボタンの上にカーソルを置きます。バージョン番号が記載されたポップアップが表示されます。
- Waveset.properties ファイルを開き、ファイルの最上部にある情報をチェックします。

現在使用中の JVM バージョン情報とその他システムソフトウェアのバージョン情報を表示するには、「Identity Manager System Properties」ページ (/debug/SysInfo.jsp) を使用します。

Identity Manager ゲートウェイのオブジェクトとアクティビティーのトレース

ここでは、Sun Identity Manager Gateway のオブジェクトとアクティビティーのトレース方法について説明します。この内容は、次のように構成されています。

注 –

- ゲートウェイのトレースファイルを表示したり編集するときは、メモ帳を使用してファイル制限を回避します。
- ゲートウェイを起動すると、エントリを削除せずに、プログラムがトレースファイルに新しいトレースエントリを追加します。ゲートウェイのトレースエントリ開始点を探すには、`Gateway version` 文字列を探します。
- ゲートウェイバージョンは、ゲートウェイ起動時にトレースに自動的に出力されます。このバージョンを表示するには、コマンドラインから `gateway -v` と入力することもできます。

「ゲートウェイデバッグ」ページからトレースを設定する方法

Identity Manager で Windows アカウントの問題をデバッグするには、「ゲートウェイデバッグ」ページ(`Gateway.jsp`) またはコマンドラインから有効にします。

次に、この手順について説明していきます。

- 128 ページの「「ゲートウェイデバッグ」ページから」
- 129 ページの「コマンドラインから」

「ゲートウェイデバッグ」ページから

ゲートウェイにアクセスできない場合は、「ゲートウェイデバッグ」ページ(`Gateway.jsp`) から、トレースを有効にします。このデバッグページから、ゲートウェイトレースファイルを指定して検索することができます。

▼ トレースを有効にする

1. **Identity Manager** 管理者インタフェースにログインします。
2. この「ゲートウェイデバッグページ」を開くには、ブラウザに次の **URL** を入力します。

   `http://`*host*`:`*port*`/idm/debug/Gateway.jsp`
3. 「ゲートウェイリソースリスト」から、トレースするリソースを選択します。
4. 必要に応じて、今までの設定を修正します。

設定を修正するには、次のボタンをクリックします。

- 「バージョンの取得」。ゲートウェイを実行中のマシンのゲートウェイバージョンとオペレーティングシステムのバージョンを返します。
- 「トレースファイルの取得」。トレースファイルのコンテンツを返します。
- 「トレースパラメータの取得」。トレースファイルのパス、トレースレベル、およびトレースファイルの最大サイズを返します。
- 「トーレスパラメータの設定」。以上のオプションについては、96 ページの「トレース設定オブジェクトを新規作成する」を参照してください。
- 「ロードされたモジュールの取得」。ゲートウェイで使用中のモジュール (DLL) のロードアドレスを返します。

「ロードしたモジュールの取得」リストはロードアドレスから構成され、その後にモジュール名が続き、ロードされたモジュールだけが記載されます。このリストには、呼び出されずに遅延してロードされたモジュールは記載されません。

「ロードしたモジュールの取得」オプションは、Active Directory と Domino にのみ対応しています。

コマンドラインから

幅広いオプションを使用するには、コマンドラインからのトレースを有効にすると便利です。

▼ トレースを有効にする

1. コマンドウィンドウを開きます。
2. 必要なトレースのコマンド引数を指定して、ゲートウェイを起動します。
   ゲートウェイをトレースするコマンドライン引数は、次の表のとおりです。

引数	説明
-f	トレースファイルへのパスを指定します。

引数	説明
-l	次のトレースレベルを指定します。 ■ レベル **0**。トレースを無効にします。(デフォルト) ■ レベル **1**。コンポーネント間のコントロールのフローをトレースし、高度な関数的メソッドからのエントリと終了を含む「精度の低い」トレースポイントを通常は定義します。 ■ レベル **2**。すべてのメソッドからのエントリと終了を含む「精度の中程度な」トレースポイント、および高度な機能メソッドの情報トレースポイントとデータトレースポイントを通常は定義します。レベル 2 は、各コンポーネント内、主要な決定ポイント内、および情報項目内のフローを追加します。 ■ レベル **3**。すべてのメソッドからのエントリと終了を含む「精度の高い」トレースポイント、高度な機能メソッドの情報トレースポイントとデータトレースポイント、および重要なサブルーチンを通常は定義します。レベル 3 は、低度の決定ポイントと情報項目を追加します。 ■ レベル **4**。ほかのトレースレベルでトレースされたものすべてを含み、「極めて精度の高い」トレースポイントを通常は定義します。レベル 4 は非常に低度でトレースし、ほとんど必要とされなくても一部のコンポーネントの複雑な動作を特徴付けるには役立つ精度のレベルを表示します。注意: すべてのコンポーネントがレベル 4 をサポートしているわけではありません。 オーバーヘッドが追加されるため、取得メソッドと設定メソッドなどの簡易メソッドには通常、エントリまたは終了トレースポイントがありません。

-m	トレースファイルの最大サイズを KB 単位で指定します。
	トレースファイルが `-m` KB に到達すると、Identity Manager が現トレースファイルを閉じ、既存のバックアップファイルがあれば削除し、現トレースファイルを `-f` 引数で指定した名前に変更して `.bk` を付与し、`-f` 引数名が付いた新しいトレースファイルを開きます。
	たとえば、コマンドラインに `-f beeble.trc` を指定すると、`-m` KB が記録されてから以降は次の 2 ファイルが生成されます。
	`beeble.trc.bk`
	`beeble.trc`
	ここで `beeble.trc` には、最新のトレースが格納されます。

使用状況: `gateway -f name -l -m`

例を示します。

```
cd %WSHOME%\bin\winnt
gateway -d -p 11319 -f %CD%\gateway.trc -l 2 -m 500
```

上記の呼び出しにより、次の特性を備えたゲートウェイが起動します。

- `-d` – 通常アプリケーションを使用 (サービス以外)
- `-p 11319` – ポート 11319 を使用

  ゲートウェイリソースには、Identity Manager のリソース設定からこのポートを設定しておく必要があります。たとえば、Active Directory リソースの場合
- `-f %CD%\gateway.trc` – トレース出力の書き込み先となるディレクトリ。Identity Manager は、このディレクトリのトレース出力をテキストファイルに書き込みます。
- `-l 2` – ゲートウェイトレースの出力レベル 2。
- `-m` – トレースのログファイルの最大サイズ (KB 単位)。

注 – 指定されていれば、次回ゲートウェイをコマンドラインからまたはサービスとして実行したときに同じ値が使用されるように、Identity Manager が `-f` 値、`-l` 値、および `-m` 値をレジストリに保存します。

Identity Manager はゲートウェイのトレース出力をコンソールおよびトレースファイルに送信します。

PowerShellExecutor.dll アドオンのトレースを設定する方法

`PowerShellExecutor.dll` は、ゲートウェイと Microsoft PowerShell 間の通信を実装するアドオンです。この PowerShell は、Exchange Server 2007 アカウントを管理するのに使用します。このアドオンは、残りのゲートウェイとトレース機能を共有せず、残りのゲートウェイと同様のスタンドアロンのトレース機能を備えています。

`PowerShellExecutor` のトレース設定は、ほかのゲートウェイのレジストリキーとして同じレジストリキーに格納されます。

```
HKEY_LOCAL_MACHINE\Software\Waveset\Lighthouse\Gateway
```

このベースキーは、Identity Manager のデバッグページからトレースを設定したり、トレースコマンド引数でゲートウェイを開始すると作成されます。

シャットダウン時に、レジストリにトレースできるようにゲートウェイから現PowerShellExecutor設定が書き込まれます。次の設定があります。

- traceFileName

  コンテンツ。トレース出力用のファイル名(レジストリ型REG_SZ)

  「デフォルト」。" "

  PowerShellExecutorトレースに生成するトレースファイル名。このファイル名には、

  – ファイル名などが省略されていないパスを設定できます。

  – スラッシュ(\)で終わることはできません。

    ファイルを除き、traceFileName内の省略されていないパスが必要です。

    これが設定されている場合、そのファイルがもはやアクティブでなければ、ローテーション後にログのローテーションから設定済みファイル名にタイムスタンプが付与されます。このタイムスタンプは、次の形式で表示されます。

    yyyyMMddHHmmss

- traceLevel

  コンテンツ。トレースレベル(レジストリ型REG_DWORD)

  「デフォルト」。0(トレースなし)

  許容。0–4

  このキーは、残りのゲートウェイと共有されます。ゲートウェイ全体は、常に同じレベルでトレースを行います。

- traceMaxSize

  「内容」。バイト単位の最大ファイルサイズ(レジストリ型REG_DWORDまたはREG_QWORD)

  「デフォルト」。100000バイト

  最小。100000バイト

  ほかのシステムに移動できるよう、トレーステキストは、バイトオーダーマークが付いたUTF–8エンコード化テキストで書き込まれます。

- traceMaxFiles

  「内容」。トレースファイル数(レジストリ型REG_DWORD)

  「デフォルト」。2

  最小。1

  この設定は、システムで保持するトレースファイル数を制御します。最大ファイル数の設定を**1**にし続けると、最大サイズに達したときにファイルが上書きされてしまいます。最後の書き込み時刻を基にして一番古いファイルは、最大ファイル数に達したときに削除されます。

- `traceConfigInterval`

  「内容」。ミリ秒単位のタイムアウト (レジストリ型 `REG_DWORD`)

  「デフォルト」。300000 ms (5 分)

  最小。60000 ミリ秒 (1 分)

すべてのトレース設定は、このタイムアウト値を基にしてレジストリから再読み取りされます。本稼動環境では、オーバーヘッドを最小限に抑えるために、この値を 24 時間などの大きな値に設定するようにしてください。

ワトソン博士のログを取り込む用法

ゲートウェイに深刻な問題が生じて異常終了した場合は、表示されたワトソン博士のログを Sun サポートに送信して解析を依頼することができます。

注 – これらのログを表示するには、そのシステムの管理者権限が必要です。

▼ ワトソン博士のログを取り込む

1. **Windows** の「イベントビューア」を開きます。
2. アプリケーションログを開きます。
3. `DrWatson` ソースのイベントを探します。
4. 詳細を表示するには、そのイベントを開きます。
5. 「データ」に「バイト」オプションが選択されていることを確認します。
6. ディスプレイダイアログを右クリックして、メニューから「すべて選択」を選択します。
7. 情報をコピーするには、**`Ctrl-C`** キーを押します。
8. この情報をメモ帳に貼り付け、ファイルを保存します。
9. 問題の詳細な説明を記述して、そのファイルを **Sun** サポートまで電子メールで送信します。実行中の **Identity Manager** ゲートウェイのバージョンを必ず記載してください。

よくある問題のトラブルシューティングと解決策

Identity Manager を使用中に発生した問題を診断して解決できるようにするには、次のセクションに説明している情報をご利用ください。

注 – Identity Manager の FAQ など、これ以外のトラブルシューティングについては次の URL にアクセスしてください。

```
https://sharespace.sun.com/gm/document-1.26.2296/IdM_FAQ.html?
```

注意: このサイトに記載している説明にアクセスするには、Share Space ID を登録している必要があります。

デバッグツールの使用

使用中の Identity Manager 配備で問題を特定して解決できるようにするには、数種類のデバッグツールが使用できます。次のツールがあります。

Identity Manager のデバッグページの使用

使用中の配備で問題を特定して解決できるようにするには、Identity Manager のデバッグページを使用します。たとえば、各種アクティビティーとオブジェクトのトレースの有効化と無効化、統計情報の収集、実行中のプロセスの検証、またはボトルネックとメモリー問題の調査が行えます。

一番よく使用されるデバッグページとその実際の `.jsp` ファイル名は、次の表のとおりです。

注 – すべての Identity Manager デバッグページの総覧については、コマンドウィンドウを開いき、`idm/debug` ディレクトリのコンテンツを一覧表示します。

表 5–5 Identity Manager のデバッグページ

ページ名	使用目的
制御タイミング (`callTimer.jsp`)	各種メソッドの呼び出しタイマー統計を収集して表示します。この情報を基に、特定メソッドに対するボトルネックと呼び出された API を追跡できます。次のような呼び出しタイマーのメトリックスをインポートしたりエクスポートするには、読み出しタイミングページのオプションも使用できます。 ■ スキャン中の初期ユーザービューの取得に掛かる所要時間 (リソースなし) ■ スキャン中の初期ユーザービューの更新に掛かる所要時間 (リソース含む) ■ ユーザービューのポリシー評価に掛かる所要時間 ■ 各ユーザースキャンに掛かる所要時間 (ユーザービューの取得やポリシー評価など) ■ アクセスをスキャンするためにユーザーリストの取得に掛かる所要時間 ■ アクセスレビューでアクセスアテステーションルールの評価に掛かる所要時間 注 – 呼び出しタイミングの統計は、トレースが有効に設定されている間にだけ収集されます。

表 5–5 Identity Manager のデバッグページ (続き)

ページ名	使用目的
トレース設定の編集 (Show_Trace.jsp)	Identity Manager のインストールに用意されている Java クラスのトレース設定を有効にします。設定できる項目は次のとおりです。 ■ トレース対象となるメソッド、クラス、またはパッケージ、およびトレースレベル。 ■ トレース情報の送信先をファイルにするか標準の出力先にするか、およびトレース出力ファイルの日時表示形式。 ■ 格納対象となる最大トレースファイル数と、各ファイルの最大サイズ。 ■ キャッシュ対象となるメソッドの最大数を指定します。 ■ トレースファイルへのデータの書き込み方法。データ生成時にデータをトレースファイルへ書き込むか、キューに入れてからデータをファイルへ書き込むか。
ホスト接続プール (Show_ConnectionPools.jsp)	(データソースを使用していない場合に) 接続プール統計を表示します。プールのバージョン、接続の作成回数、アクティブ数、プール内にある接続数、プールから処理された要求数、切断された接続数などがあります。 このホスト接続プールページは、ゲートウェイへの接続管理に使用された接続プールの概要を表示する際にも使用できます。この情報を使用して、低位アドレスメモリー状態を調べられます。
消去されたキャッシュの一覧表示 (Clear_XMLParser_Cache.jsp)	キャッシュから最近使用した XML パーサーを消去して、低位アドレスメモリー状態を調査します。
メソッドタイミング (Show_Timings.jsp)	ホットスポットをメソッドレベルで検出して評価します。次のような Identity Manager メソッドから情報を収集するには、このページを使用します。 ■ メソッド名 ■ メソッドが呼び出された回数 ■ メソッドがエラー状態で終了した回数 ■ メソッドによって消費された平均時間 ■ 各メソッドの呼び出しによって消費された最小時間と最大時間

表 5–5 Identity Manager のデバッグページ (続き)

ページ名	使用目的
オブジェクトサイズの概要 (Show_Sizes.jsp)	システムに影響しそうな、疑わしい大きなオブジェクトを検出します。このページは、リポジトリに格納されたオブジェクトのサイズを (文字単位で) 表示します。オブジェクトの総結合サイズ、平均サイズ、最小サイズなどがあります。リポジトリ内の最大設定オブジェクトの ID、名称、およびサイズを表示するには、「Type」列のエントリをクリックします。
管理者設定ツールのプロビジョニングスレッド (Show_Provisioning.jsp)	使用中のプロビジョニングスレッドの概要をシステム別に表示します (この情報のサブセットは、 Show_Threads.jsp から使用できます)。
システムキャッシュの概要 (Show_CacheSummary.jsp)	低位アドレスメモリーの状態を調査できるよう、次の情報を表示します。 ■ 管理者に関連付けられているオブジェクトキャッシュ ■ システムオブジェクトのキャッシュ ■ ユーザーのログインセッション ■ XML パーサーのキャッシュ
システムメモリーの概要 (Show_Memory.jsp)	JVM に十分なメモリーが割り当てられているか判断できるよう、調整などメモリーをたくさん使用する機能の使用時に使用できる合計メモリーと空きメモリーがどれくらいあるかを表示します。また、ガベージコレクションの起動や、ヒープ使用量を調査するための JVM 内の未使用メモリーの消去にも、このページが役に立ちます。
システムプロパティー (SysInfo.jsp)	使用環境について表示します。
システムスレッド (Show_Threads.jsp)	自動プロセスが実行中であることを確認できるよう、実行中のプロセスを表示します。プロセスの種類、プロセス名、優先順位、そのプロセスがデーモンかどうか、およびそのプロセスがアライブ中 (実行中) かについて記載します。
ユーザーセッションプールの消去 (Clear_User_Cache.jsp)	低位アドレスメモリーの状態を調査するには、セッションプールの消去ページを使用します。
Waveset プロパティー (Show_WSProp.jsp)	Waveset.properties ファイル内のプロパティーを表示して一時的に編集します。編集したプロパティー設定は、次回のサーバー起動時に初めて反映されます。
フラッシュされて消去された XML リソースアダプタキャッシュ (Clear_XMLResourceAdapter_Cache.jsp)	キャッシュから XML リソースアダプタを消去して、低位アドレスメモリー状態の調査に使用します。

注 – 以上のデバッグページの詳細は、74 ページの「Identity Manager のデバッグページの使用」を参照してください。

▼ 個々の Identity Manager デバッグページにアクセスする

始める前に Identity Manager のデバッグページにアクセスして操作を実行するには、デバッグ機能が必要です。デバッグ機能がない場合は、エラーメッセージが表示されます。管理者とコンフィギュレータには、デフォルトでこのデバッグ機能が割り当てられています。

1 ブラウザを開き、管理者インタフェースにログインします。

2 次の **URL** を入力して、「**System Settings**」ページを開きます。

http://*host***:***port***/idm/debug**

各表記の意味は次のとおりです。

- *host* は、Identity Manager の実行先ローカルサーバーです。
- *port* は、このサーバーが監視中の TCP ポート数です。

このページから、Identity Manager の各種アクティビティーやオブジェクトのトレースを有効にしたり無効にできるほか、これらのページに表示される情報を基に、使用中の配備の問題をトラブルシューティングできます。

デバッグページの中には、「System Settings」ページにリンクされていないものもありますので、そのページの `.jsp` ファイル名を入力してページを開く必要があります。例を示します。

http:// *host***:***port***/idm/debug/** *pageName.jsp*

ここで、*pageName.jsp* は、開こうとしている個々のデバッグページです。

Identity Manager IDE の使用

Sun™ Sun Identity Manager 統合開発環境 (Identity Manager IDE) は、使用中の配備の Sun Identity Manager (Identity Manager) オブジェクトを表示、カスタマイズ、デバッグできる Java アプリケーションです。

具体的に言うと、Identity Manager IDE にはグラフィカルなデバッガが用意されており、Identity Manager のフォーム、ルール、およびワークフローをデバッグに使用することができます。このデバッガを使用すると、ブレークポイントとウォッチの設定、コードのステップ実行、変数の検査と修正、クラスと呼び出しスタックの検査、スレッドの抽出、およびマルチセッションの実行が行えます。

Sun Identity Manager 統合開発環境 (Identity Manager IDE) のインストール手順と設定手順は、次の URL から入手できるようになりました。`https://identitymanageride.dev.java.net`

Identity Manager のシステム監視の使用

Identity Manager のシステム監視を設定すると、システムイベントを追跡できます。システム監視は、各種レベルで統計を収集して集約し、仕様に従ってシステムイベントをリアルタイムに表示します。

この情報を計器盤グラフで表示すると、監査ログを見る前に、システムリソースの即時評価、異常の表示、履歴パフォーマンスの傾向の把握、および対話式で問題を切り離すことができます。計器盤は監査ログほど詳細に表示しませんが、この計器盤を見ると、ログのどこを見れば問題が探せるかがわかります。

計器盤とシステム監視の詳細は、『Sun Identity Manager 8.1 ビジネス管理者ガイド』の第 8 章「レポート」を参照してください。

アダプタログの仕様

アダプタログは、現在処理中のアダプタ情報を取り込みます。この情報を基に、アダプタの進捗を監視し、アダプタの問題を診断してデバッグすることができます。

注 – トレースを有効に設定し、ログが実行される前にトレースが必要なメソッドを特定しておく必要があります。また、カスタマイズしたアダプタには、この新しいメソッドのログエントリを作成する呼び出しを含めておく必要があります。

ほぼすべてのアダプタに、独自のログファイル、パス、およびログレベルがあります。適切な Identity Manager または サービスプロバイダ ユーザータイプの同期ポリシーにある「ログ」セクションのほかの値をとともに、このアダプタログで取り込む詳細レベルを指定できます。

デバッグツールとしてアダプタログファイルを使用する詳細は、141 ページの「アダプタのトラブルシューティング」を参照してください。

DTrace を使用したデバッグ

DTrace とは、Solaris オペレーティング環境に使用する、総合的な動的トレースのフレームワークです。DTrace には 30,000 を超えるプローブを使用中の本稼動システムに提供し、ユーザーレベルのトレースとカーネルレベルのトレースを統合します。JVM アクティビティーを監視するには、DTrace を使用します。この機能を使用すると、D 言語 (C 言語や `awk` のようなもの) も使用して任意のデータと式をトレースできるようになります。

JConsole を使用したデバッグ

Java Monitoring and Management Console (*JConsole*) とは、Java Management Extension (*JMX*) 技術に対応したグラフィカル管理ツールで、JDK 5 (以降) に付属しています。JConsole は実行中の JVM に接続し、接続している JMX エージェントの JVM MBeans から情報を収集します。

たとえば、JConsole を使用してできる操作には次のようなものがあります。

- 低位アドレスメモリーの検出
- ガベージコレクションの有効化または無効化
- 冗長トレースの有効化または無効化
- デッドロックの検出
- Identity Manager のログレベルの制御
- システムリソースの操作に関するアクセス情報 (Sun のプラットフォーム拡張)
- MBeans の監視と管理
- JVM、監視した値、アプリケーションで実行中のスレッド、およびクラスのローディングについての情報表示

注 – JConsole の詳細は、「JConsole を使用したアプリケーションの監視」というタイトルの記事を参照してください。この記事は、次の URL から表示できます。

`http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`

ブラウザに表示されたエラーのデバッグ

アクションを実行した後に Identity Manager インタフェースに赤いエラーメッセージが表示された場合は、ページのソースを表示して保存し、情報全体を表示してエラーを解析してください。

ページのソースの表示する

- Internet Explorer を利用している場合は、メニューバーから「表示」>「ソース」の順に選択します。
- Netscape™ を利用している場合は、メニューバーから「表示」>「ページのソース」の順に選択します。

それでもまだ問題の解決にサポートが必要な場合は、

1. ページソースを表示してから、「ファイル」>「保存」の順に選択してシステムにこのファイルを保存します。
2. 保存したファイルから、エラーを見つけます。
3. エラー情報、問題が発生したページの URL、問題の説明を電子メールで Sun サポートまで送信し、解決サポートを依頼してください。

アダプタのトラブルシューティング

アダプタをトラブルシューティングするには、アダプタのログファイルをレビューします。ほとんどのアダプタはそのリソース設定をログファイルに書き込んでいるため、この情報を基に、そのアダプタが起動していること、すべての設定の変更内容が保存されていることを確認できます。

注– トレースを有効に設定し、ログが実行される前にトレースが必要なメソッドを特定しておく必要があります。また、カスタムアダプタには、この新しいメソッドのログエントリを作成する呼び出しを含めておく必要があります。

トレースを有効にする手順については、必要に応じて98 ページの「Identity Manager サーバーのトレース」を参照してください。

ほとんどのアダプタログファイルは、`$WSHOME/config` ディレクトリにあり、`WSTrace1.log` という名前が付けられています。

`ActiveSyncUtil` インスタンスへのログ呼び出しを作成し Active Sync が有効に設定されているアダプタは、`Log File Path` リソース属性で指定されたディレクトリにログファイルまたはログファイルセットを作成します。このほかにも Active Sync 関連のログエントリがないか、このログファイルを必ずチェックしてください。

この節の情報は、次のように構成されています。

アダプタをデバッグする

カスタムアダプタをデバッグするには、この一般的な手順に従ってください。

1. アダプタにテストプログラムを作成し、この Java ファイルが次の基本機能を実行することを確認します。
   a. リソースの新規作成
   b. ユーザーの作成
   c. ユーザーの取得
   d. ユーザーの更新
   e. ユーザーの削除
   f. 複数ユーザーに、作成、取得、更新、および削除の操作を実行します。

注 – サンプルテストファイル (SkeletonResourceTests.java) が、インストール CD の /REF ディレクトリに用意されています。

2. デバッグに適切なログレベルを設定します。

   たとえば、最初のデバッグパスには、ログレベル (最大デバッグ出力) を 4 まで上げ、ログファイルのパスを設定し、最大ファイルサイズを指定します。

   アダプタを起動すると、すべてのリソース設定がこのログファイルに書き込まれます。この情報を基に、アダプタが起動したこと、全ての設定の変更内容が保存されたことを確認できます。

3. アダプタをコンパイルしてテストします。
   - テストプログラムをコンパイルするには、コマンドウィンドウを開いて `javac -dtest/filename.java` コマンドを入力します。このコマンドは、適切な `com/waveset/adapter/test` ディレクトリにクラスファイルを作成します。
   - このクラスファイルを使用して新アダプタをテストするには、コンパイルしたアダプタが `com/waveset/adapter` ディレクトリにあることを確認して、次のコマンドでこのアダプタを実行してください。

     `java– D waveset.home=`*path* `com.waveset.adapter.test.`*MyResourceAdapter*

4. リソースの HTML ヘルプファイルを作成します。

   注 –
   - サンプルのヘルプファイルが、`com/waveset/msgcat/help/resources` ディレクトリの `idm.jar` ファイルに用意されています。
   - アプリケーションのオンラインヘルプを記載する方法については、『Sun Identity Manager Deployment Reference』を参照してください。

5. (*Active Sync* が有効なアダプタの場合のみ) 最後のリソースで同期をリセットするには、`XmlData SYNC_`*resourceName* オブジェクトを削除します。
6. エラーログを読み、アダプタを修正します。
7. ログレベルをリセットします。

   たとえばレベル 2 のデバッグを指定すると、アダプタ設定とエラーに関する情報が表示されますが、ログの詳細量が管理できるレベルまでに制限されます。

8. Identity Manager を起動する前に、`resource.adapters` エントリの下にあるアダプタ名を設定して `$WSHOME/config/Waveset.properties` ファイルにある新アダプタを特定しておく必要があります。そうしないと、Identity Manager がそのアダプタを認識できません。

9. アダプタとその関連ヘルプファイルを Identity Manager にインストールします。

注 – Identity Manager がディスプレイ内の新アダプタのインスタンスを認識できるようにする前に、そのタイプの新リソースを「リソースのリスト」ページから作成しておく必要があります。

このページから「新規」>「新アダプタ」の順に選択し、「リソースウィザード」から新アダプタを作成します。

---

10. Identity Manager で、そのリソースにリソースとユーザーを作成します。

ヒント – Active Sync 有効アダプタをトラブルシューティングする際、`XmlData SYNC_`*resourceName* オブジェクトを編集して Active Sync 同期プロセスの `MapEntry` を「デバッグ」ページから削除していれば、アダプタが最初に検出された変更点からやり直します。

IAPI イベントを使用した場合に、最後に処理した変更値などのリソースの同期状態を格納するには、`Property()` メソッドを設定する必要があります。このメソッドを設定すると、アダプタのトラブルシューティングに非常に役立ちます。過去の変更内容を実行して無視するようにアダプタを設定することができます。それ以降は、アダプタを修正して、変更結果をアダプタのログファイルに表示できるようになります。

使用中のリソースが Active Sync リソースの場合は、そのリソースの編集ページでログを有効にしていれば、さらに情報が表示されることもあります。ログレベル (0-4) を設定し、ログファイルの書き込み先となるファイルパスを設定します (*resource_name*`.log` など)。

11. (*Active Sync* 有効アダプタの場合のみ) 最後のリソースの同期を再起動します。

`LoginConfig` 変更内容をデバッグする

`LoginConfig` 関連の変更点をデバッグするには、次を行う必要があります。

1. 選択したファイルのトレースを有効にし、メソッド/クラスレベル 1 トレースで次のクラスをトレースします。
   - `com.waveset.security.authn.WSResourceLoginModule`
   - `com.waveset.session.LocalSession`
   - `com.waveset.session.SessionFactory`
   - `com.waveset.ui.LoginHelper`
   - `com.waveset.ui.web.common.ContinueLoginForm`
   - `com.waveset.ui.web.common.LoginForm`
2. Telnet で、次のようにシングルサインオン (SSO) のパススルー認証ログインをテストします。

a. SSO ログインモジュールを正しく設定したら、HTTP ポートに直接 telnet し、HTTP 要求を login.jsp に送信します。

b. 次の要求を telnet セッションに貼り付けます。これには、sm_user HTTP ヘッダーを検索する SSO ログインモジュールが記載されています。

```
HEAD /idm/login.jsp HTTP/1.0
Accept: text/plain,text/html,*/*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: LOCALHOST
sm_user: Configurator
```

トレースに、ユーザーが正常にログインしたと表示されます。例を示します。

```
2003.07.08 14:14:16.837 Thread-7
WSResourceLoginModule#checkForAuthenticatedResourceInfo()
Found authenticated resource accountId, 'Configurator@Netegrity
SiteMinder'on Identity Manager user 'Configurator'.
null null 2003.07.08 14:14:16.837
Thread-7 WSResourceLoginModule#checkForAuthenticatedResourceInfo()
Exit null null 2003.07.08 14:14:16.837 Thread-7 WSResourceLoginModule#login()
Exit, return code = true null null 2003.07.08 14:14:16.847
Thread-7 LocalSession#login() Login succeeded via Netegrity SiteMinder
null null 2003.07.08 14:14:16.847 Thread-7 LocalSession#login()
Overall authentication succeeded null null 2003.07.08 14:14:16.897 Thread-7
LocalSession#checkIfUserDisabled()
Entry null null 2003.07.08 14:14:16.897 Thread-7
LocalSession#checkIfUserDisabled() Exit
null null 2003.07.08 14:14:16.927 Thread-7 LocalSession#login()
Exit null null
```

アダプタ接続問題をデバッグする

ここでは、よくあるアダプタ接続問題のデバッグ方法について説明します。

このセクションのトピックは、次のように構成されています。

注 – 通常、アダプタクラスの com.waveset.adapter.*adapter_classname* をトレースすれば、アダプタの接続問題を特定できます。例を示します。

```
com.waveset.adapter.ADSIResourceAdapter
```

トレースを有効にする手順については、98 ページの「Identity Manager サーバーのトレース」を必要に応じて参照してください。

アダプタの認証問題

よくある認証問題には、次のようなものがあります。

- 認証プロパティーが欠落している。

  指定した DataSource タイプのプロパティー名をトレースの名前セット出力に含める必要があります。

- Identity Manager ユーザーに、一致するリソースアカウントがない。

  リソースアダプタの認証に成功したが、リソースアカウント ID と一致する Identity Manager ユーザーが見つからなかったという例外が発生した場合は、そのユーザーに関連付けられているリソースの accountId が、リソースアダプタの認証メソッドで返された accountId と同じであることを確認してください。

  Identity Manager ユーザーのリソース accountId を検証するには、Identity Manager の「トレース設定の編集」ページ (debug/Show_Trace.jsp) をレビューします。不一致が合った場合は、authenticate メソッドから返される名称を変更するか、使用リソースの ID テンプレートを変更してください。テンプレートは、authenticate メソッドで返される accountId と一致するリソース accountId を生成する必要があります。

Active Sync アダプタの問題

カスタム Active Sync アダプタで最もよくある問題は、フォーム関連のものです。この種のエラーは通常、必須フィールドにパスワードや電子メール情報など必要な情報を入力しなかったために起こります。

Identity Manager は、ビューの最終 XML の後に、フォーム検証エラーをアダプタログに出力します。例を示します。

```
20030414 17:23:57.469: result from submit (blank means no errors):
 20030414 17:23:57.509: Validation error: missing required field password
```

Identity Manager は、アダプタログにすべてのメッセージも出力します。このメッセージには、アカウント作成時と更新時、アダプタエラー、スキーママップデータなどが記載されます。

Active Sync リソースアダプタは、最後に処理した変更内容についての情報を SYNC.resourceName XMLData オブジェクトに格納します。

Domino ゲートウェイアダプタの問題

次に、よくある Domino ゲートウェイとアダプタの設定エラーと、この問題の解決手順を説明します。

- エラーメッセージに「"新しい Domino ゲートウェイ" にアクセスできません、接続が拒否されました」というエラーメッセージが表示された場合は、Identity Manager のゲートウェイを終了して再起動してみてください。
- 「ID ファイル名が指定されていません、userID ファイルへのパスが正しく設定されていません」というエラーメッセージが表示された場合は、userID の目標位置を指定し、リソースアダプタを編集してこの属性を正しいパスに設定します。通常、userID ファイルの目標位置は、ゲートウェイのインストール先ディレクトリにあります。
- 「このサーバーを使用する権限がありません。」というエラーメッセージが表示された場合は、ID ファイルに正しいアクセス権限が設定されていません。このファイルに正しい権限を指定してから、やり直してください。

メインフレームホストアダプタの問題

RACF、ACF2、または TopSecret のホストアダプタが接続を再利用できなかったりキャッシュできないと、ユーザーが頻繁にログインしなくてはならなくなります。これにより、パフォーマンスに悪影響を与えます。この問題は、キャッシュのタイムアウト設定に原因があることがよくあります。

キャッシュのタイムアウト設定を確認するには、Identity Manager のアダプタ接続プールを次のようにトレースします。

1. Identity Manager の「Edit Configuration Object」ページから、`com.waveset.adapter.HostConnPool#reapConnections` メソッドをレベル 4 でトレースします。

   トレースを有効にする手順については、98 ページの「Identity Manager サーバーのトレース」を必要に応じて参照してください。
2. アダプタが操作を実行中に、十分な長さの期間のトレースを取り込みます (最低 30-60 分)。
3. アプリケーションサーバーの `stdout` のトレース出力またはトレースファイルをレビューして、`Info reaping connection` エントリを見つけます。

   もしもこのエントリが 30 分おきに 1 回以上発生している場合は、接続が不必要にタイムアウトになっているということです。

   この問題を解決するには、`Idle Timeout` リソース属性値を上げて、接続が頻繁にリープされすぎないようにします。`Idle Timeout` 属性値は、接続がログアウトされるまでのアイドル期間を制御します。デフォルト値は 90 秒ですが、これですと新たなログインが頻繁に発生します。

理想的には、配備環境に合わせて、平均的なアイドル時間より長い値を指定してください。たとえば、Idle Timeout 属性値を 30 分 (1800000 ミリ秒) 以上に調節します。

PeopleSoft アダプタの問題

ここでは、次のような PeopleSoft のアダプタ問題をトラブルシューティングする方法について説明します。

- PeopleSoft リソースアダプタから「テスト接続」を実行すると、多様な例外状態が発生する原因となります。
  1. Waveset.properties ファイルを開き、 exception.trace=true を設定します。
  2. テスト接続を再試行し、次のような結果が表示される場合があります。

```
FormState: expansion complete
java.lang.NullPointerException: PSProperties not loaded from file
WavesetException:
WavesetException:
==> com.waveset.util.WavesetException:
FormState: derivation

Log in to the PeopleSoft web interface to verify that you
are using the correct UID and password.
```

- この PeopleSoft アプリケーションサーバーのログには、ログインしようとしている試行がありません。

  この問題は、接続先の PeopleSoft インストールに付属している psjoa.jar ファイルを使用しなかったためによく起こります。(PeopleSoft リソースアダプタの詳細は、『Sun Identity Manager 8.1 リソースリファレンス』 を参照してください。)

SAP アダプタの問題

SAP or SAP HR Active Sync アダプタから SAP システムへの接続をテストしようとしてエラーが発生した場合は、コマンドウィンドウを開き、インストール先ディレクトリ WEB-INF/lib からこのコマンドを実行します。

```
java -jar sapjco.jar
```

sapjco.jar コマンドは、インストールされている SAP Java Connector (JCO) のバージョンと、このアダプタが正しくインストールされているかを表示します。このコマンドは、Java™ Native Interface (JNI™) の参照先プラットフォーム、および SAP システムと通信している　RFC ライブラリも返します。

この参照先プラットフォームのライブラリが見つからない場合は、SAP マニュアルを参照して、SAP Java Connector の正しいインストール方法を調べてください。

UNIX アダプタの問題

ここでは、UNIX アダプタによくある問題をデバッグする方法について説明します。

- UNIX リソースアダプタへのプロビジョニング中にタイムアウトエラーが表示された場合は、最大ログ出力が得られるメソッド/クラスレベル 4 で `com.waveset.adapter.ScriptedConnection` をトレースすれば、プロビジョニングプロセスが失敗している場所を見つけることができます。
- 管理上のコマンドを実行する `root` になるときに、リソースアダプタが `su root` コマンドを `su - root` コマンドの代わりに実行すると、カスタムプロンプトなど、root に定義されているカスタム環境変数が環境に何も継承されなくなります (PS1 の環境変数)。

  UNIX アダプタを設定する際、次のようにすれば Root Shell Prompt フィールドに入力すべきプロンプトを判断できます。

  1. 「ログインユーザー」フィールドに指定したユーザーでシステムに Telnet または `ssh` します。
  2. パスワードを入力してログインしたら、ダッシュなしで **`su root`** を入力して改行キーを押します。
  3. root パスワードを入力します。
  4. 次に表示されたプロンプトが、Root Shell Prompt フィールドに入力すべきプロンプトです。

Auditor のトラブルシューティング

Identity Auditor を使用すると、次のメソッドをトレースして問題をトラブルシューティングできます。

- `com.sun.idm.auditor.policy` - Audit Scan の問題をトレースします。
- `com.sun.idm.auditor.accessreview` - Access Reviews の問題をトレースします。
- `com.sun.idm.auditor.report` - Audit Reports の問題をトレースします。
- `com.sun.idm.auditor.view` - Auditor Views の問題をトレースします。

また、Form または `TaskDefinitions` を修正すれば、次の非表示フラグを設定できます。

- **Audit Policy Scan Task - maxThreads (int)**。使用した並行スレッド数。(デフォルトは 5)。
- **Audit Policy Scan Task - userLock (int)**。ユーザーオブジェクトでロックを取得するまでの時間 (ミリ秒単位)。(デフォルトは 5000)。
- **Audit Policy Scan Task - scanDelay (int)**。新しいスキャンスレッドを開始する間の遅延時間 (ミリ秒単位)。(デフォルトは 0 で、遅延なし)。
- **Audit Policy Scan Task - clearuserLocks (boolean)**。`true` の場合は、スキャンする前にスキャナがユーザーロックの消去を試みます。

また、「Show Timings」ページ (/debug/callTimer.jsp) に次の情報が表示されます。

- スキャン中の初期ユーザービューの取得に掛かる所要時間 (リソースなし)
- スキャン中の初期ユーザービューの取得に掛かる所要時間 (リソース含む)
- ユーザービューのポリシー評価に掛かる所要時間
- ユーザービューの取得やポリシー評価など、各ユーザースキャンに掛かる所要時間
- アクセスをレビューするためにユーザーリストの取得に掛かる所要時間
- アクセスレビューでアクセスアテステーションルールの評価に掛かる所要時間

ClassNotFound 例外のトラブルシューティング

ClassNotFound の例外エラーのイベント発生時には、サーバーのクラスパスに欠けているクラスがないか確認します。このクラスパスが適切に設定されている場合は、親クラスのローダーの前にアプリケーションクラスローダーがロードしたとおりにアプリケーションサーバーを設定してみてください。場合によっては、アプリケーションのクラスパスをサーバーのクラスパスの前にロードすると、この問題が解決できることがあります。この手順については、アプリケーションサーバーのマニュアルを参照してください。

フォーム問題のトラブルシューティング

ここでは、よくあるフォーム問題と、この問題の解決方法について説明します。

- <script> タグをユーザーフォームに追加したが、ユーザーフォームにアクセスしようとすると次の例外が表示される場合。

  ```
  java.lang.NoClassDefFoundError: org/mozilla/javascript/NativeScript
  ```

  WEB-INF/lib/javascript.jar ファイルをアプリケーションサーバーのクラスパスに配備する必要があります。例を示します。

  - Tomcat をサービスとしてインストールした場合は (Windows のみ)、次のレジストリキーを編集します。

    ```
    HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tomcat\Parameters\JVM Option Number 0
    ```

    JVM Option Number 0 キー文字列値があればどこにあってもその終わりに、jar ファイルへのパスを付けます。例を示します。

    ```
    ;C:\tomcat\lib\javascript.jar
    ```

  - コマンドラインから Tomcat を起動したばかりの場合は、jar ファイルを tomcat/lib ディレクトリに移動します。

- タブ付きのユーザーフォームをカスタマイズしたが、User Creation でフォームにアクセスしようとすると、次の例外が表示される場合。

```
com.waveset.util.WavesetException: Maximum form stack depth exceeded
```

スタックに転送できるハードコード要素は 1000 までと制限されており、この上限値を超えました。

`Field` コンテナ名に一致する `FieldRef` または `Field` を含む `Field` コンテナを検出するチェックを追加した場合、気づかないうちに循環参照をフォームに作成してしまった可能性があります。この問題を解決するには、`FieldRef` または `Field` 名を変更します。

- LDAP ユーザーのフォームに `MultiSelect` フィールドを使用し、このフィールドのユーザーにグループを割り当て、ユーザーを編集し直してから、両側の `MultiSelect` に同じグループが表示された場合。例を示します。
  - 左側の値。 `cn=Group1,dc=test,dc=Com`
  - 右側の値。 `cn=Group1,dc=test,dc=com`

    この例では、LDAP `baseContext` リソースフィールドが `dc=test,dc=com` に設定されており、LDAP グループが `dc=test,dc=Com` としてリストされています。LDAP は大文字と小文字を区別しませんが `MultiSelect` ウィジェットは大文字と小文字を区別するため、この問題が発生しています。

    この問題を多少とも解決するには、LDAP リソースの `baseContext` を LDAP リソース `dc=test,dc=Com` の大文字と小文字に合わせて変更するか、`<upcase>` XPRESS 関数を使用して左側と右側の両方の表示を大文字にします。

ゲートウェイのトラブルシューティング

Sun Identity Manager Gateway, をトラブルシューティングする際に、ゲートウェイをコマンドラインから実行すると役立つことがよくあります。コマンドラインのオプションを使用すれば、サービスではなく通常のアプリケーションとしてゲートウェイを起動したり、別のポートでゲートウェイを実行したりと、より広範囲の起動オプションが入力できるようになります。

注 – コマンドラインから実行する前に、サービスとしての Identity Manager ゲートウェイを `kill` しておく必要があります。たとえば、次のように入力します。

```
gateway.exe -k
```

ゲートウェイのコマンドライン引数は、次の表のとおりです。

引数	説明
-i	指定した起動で、このプログラムを NT サービスとしてインストールします。

引数	説明
-r	Service Manager からこのプログラムを削除します。
-s	サービスを開始します。
-k	サービスを強制終了します。
-t	既存サービスに起動を設定します。
-d	通常のアプリケーションとしてデバッグして実行します。
-p	TCP/IP ポート番号 (デフォルトは 9278) を指定します。
-f	トレースファイルへのパスを指定します。
-l	トレースレベルを指定します (デフォルトは 0 で、情報なし)。
-m	トレースファイルの最大サイズを KB 単位で指定します。
-v	バージョンを表示します。

用途: `gateway -i n -r -s -k -t n -d -p n -f name -l n -m n -v`。

注 –

- `-d` と `-s` オプションは、相互に排除的です。
- ゲートウェイのトレースレベルの詳細は、129 ページの「コマンドラインから」を参照してください。

ゲートウェイのトラブルシューティングは、Identity Manager ゲートウェイのデバッグページからも行えます。 (`debug/Gateway.jsp`)。 詳細は、128 ページの「「ゲートウェイデバッグ」ページからトレースを設定する方法」を参照してください。

Java コード問題のトラブルシューティング

Identity Manager を使用するのに必要な基本的な Java プログラミングのスキルがあれば、ほとんどの Java コード問題を診断して解決できるはずです。

ただし、データベースへの接続を開いてその接続を適切に閉じなかった方がいると、問題が発生することがかなりよくあります。接続を適切に閉じなかった場合、パフォーマンス問題につながります。

低位アドレスメモリー状態のトラブルシューティング

ここでは、低位アドレスメモリー状態の調査に使用できる次のツールについて説明します。

Identity Manager のデバッグページから

注 – Identity Manager のデバッグページにアクセスして操作を実行するには、デバッグ機能が必要です。管理者とコンフィギュレータには、デフォルトでこの機能が割り当てられています。

デバッグ機能がない場合は、エラーメッセージが表示されます。

次の Identity Manager デバッグページを管理者インタフェースから開いて、システムで使用中のメモリー量を監視することができます。

- ホスト接続プールページ (`debug/Show_ConnectionPools.jsp`)。 (データソースを使用していない場合に) 接続プール統計概要を表示します。プールのバージョン、接続の作成回数、アクティブ数、プール内にある接続数、プールから処理された要求数、切断された接続数などがあります。

  このホスト接続プールページは、ゲートウェイへの接続管理に使用された接続プールの概要を表示する際にも使用できます。この情報を使用して、低位アドレスメモリー状態を調べられます。
- 消去されたキャッシュの一覧表示 (`debug/Clear_XMLParser_Cache.jsp`)。 最近使用した XML パーサーのキャッシュを消去します。
- 非公開収集プール (`debug/Show_JDBC.jsp`)。 リポジトリと一部のリソースアダプタで使用されている Java DataBase Connectivity (JDBC™) 接続のキャッシュ概要を表示します。
- 「**System Memory Summary**」 ページ (`debug/Show_Memory.jsp`)。 システム内の使用中メモリーと総メモリーを表示します。直前に使用されたメモリー値を取得するには、「Garbage Collect」ボタンをクリックする必要があります。
- 「**System Memory Summary**」 ページ (`debug/Show_Memory2.jsp`)。 更新された `Show_Memory.jsp` ページを表示します。ここから、ヒープ使用量を調査できるように JVM にある未使用メモリーをすべて消去することができます。
- ユーザーセッションプールの消去 (`Clear_User_Cache.jsp`)。 最近ログインしたユーザーのキャッシュ済みセッションを消去します。

- フラッシュされて消去された **XML** リソースアダプタキャッシュ (`Clear_XMLResourceAdapter_Cache.jsp`)。 テスト XML リソースアダプタのキャッシュを消去します。

JConsole から

低位アドレスメモリーとデッドロックを検出するには、Java Monitoring and Management Console (*JConsole*) を使用します。JConsole とは、Java Management Extension (JMX™) 技術に対応したグラフィカル管理ツールで、JDK 5 (以降) に同梱されています。

JConsole は、メモリーシステム、メモリープール、MBeans ガベージコレクタにアクセスして、メモリー消費量などのメモリー使用量、メモリープール、およびガベージコレクション統計についての情報を表示します。JConsole を使用すると、現在のヒープメモリー使用とヒープメモリー使用以外を調べるために MBeans を監視することもできます。

注 – Java プラットフォームで実行するアプリケーションを JConsole を使用して監視する方法については、「JConsole を使用したアプリケーションの監視」を参照してください。 このドキュメントは、次の URL から入手できます。

`http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`

PasswordSync 問題のトラブルシューティング

PasswordSync の問題をトラブルシューティングしようとする際は、次のログを参考までにお読みください。

- **PasswordSync Error Logs**。 PasswordSync はすべての障害情報を Windows イベントビューアに書き込みます。(イベントビューアの詳細は、Windows のヘルプを参照してください。)エラーログエントリのソース名は「*PasswordSync*」です。
- **PasswordSync Trace Logs**。 PasswordSync はすべてのログを、トレースを設定したときに指定したファイルの場所に書き込みます。111 ページの「PasswordSync 設定ツールの使用方法」を参照してください。

よくある PasswordSync 問題とソリューションは、次のとおりです。

- PasswordSync は、Windows サーバーから Identity Manager へパスワード変更を伝えません。

  PasswordSync は、Active Directory から Identity Manager Server への接続を作成したときのレジストリ設定を基にしています。PasswordSync は、レジストリを読み取ってその設定を処理しますが、接続を作成できるかどうかを調べるチェックは一切行いません。

  次に、PasswordSync サーバーのレジストリエントリ例を示します。この例では、デフォルトのレジストリ設定値が含まれていますが、PasswordSync が使用したすべての設定が表示されているわけではありません。

  ```
  [HKEY_LOCAL_MACHINE\SOFTWARE\Waveset\Lighthouse\PasswordSync]
  "reinitIntervalMilli"=dword:0001d4c0
  "securityIgnoreCertRevoke"=dword:00000000
  "securityAllowInvalidCert"=dword:00000000
  "directMode"=dword:00000001
  "lhuser"="config"
  "lhcred"="rsVtQZpa5Ys="
  "endpointURL"="http://10.10.10.10:8080/idm/servlet/PasswordSync"
  "installdir"="C:\\Program Files\\Sun Microsystems\\Sun Identity Manager
  PasswordSync"
  "tracelevel"=dword:00000000
  "tracemaxKB"=dword:00002710
  "tracefile"="C:\\Program Files\\Sun Microsystems\\Sun Identity Manager
  PasswordSync\\trace.log"
  ```

  適切なレベルのトレースを有効していないと、PasswordSync はあまり詳細な接続障害情報のログを記録しません。詳細なトレース情報の詳細を表示するには、112 ページの「レジストリキーの編集方法」で説明したとおりに PasswordSync レジストリ設定を編集してください。最大トレース情報を出力するように `tracelevel` 4 を指定し、`tracefile` 値を書き込み可能ファイルを指すように変更します。例を示します。

  ```
  "tracelevel"=dword:00000004
  "tracefile"="C:\\Program Files\\Sun\\IdentityManager\\PasswordSync\\pwicsvc.log"
  ```

  このレジストリ設定は、レジストリの `<i>reinitIntervalMilli</i>` 設定を基に再読み取りされます。このレジストリ設定の再読み取り後、PasswordSync はレジストリに設定されたトレースパラメータに応じて自動的にトレースを開始したり終了したりします。阻止されたそれぞれのパスワード変更については、PasswordSync がそのパスワードを Identity Manager へ転送するために講じた措置のログを記録します。

- 作成中に接続障害がある場合は、次の状況が考えられます。

注 – このような状況には、それぞれ独自のエラーコードとログエントリ一式が用意されています。Identity Manager は、省略するためにこれらのエントリから日付、タイムスタンプ、およびプロセス番号を削除します。

- 不正または到達不能な URL エラー。サーバーが到達できない場合、サーバーが実行中でない場合、またはサーバーが正しい応答で応答しない場合などに起こります。

  PasswordSync からサーバーとページにアクセスできることを確認します。

  - ❒ サーバーが実行中であること、ファイアウォールとルーターを正しく設定していることを確認します。
  - ❒ アプリケーションサーバーが実行中であること、アプリケーションパスがなくても PasswordSync がエンドポイント URL に接続できることを確認します。PasswordSync がページもエラーも返さない場合は、アプリケーションサーバーが実行されていません。
  - ❒ エンドポイント URL を標準ブラウザで開いて、サーブレットの応答を確認します。`com.waveset.util.WavesetException` で始まるエラーが表示されない場合は、サーブレットがコンパイルで使用できるかどうか確認します。
  - ❒ PasswordSync のJMS 使用は、クラスパスで使用可能になっている `jms.jar` を基にしています。正しいファイルを配備せずにエンドポイント URL にアクセスすると、次の例外メッセージが表示されます。

    ```
    com.waveset.util.WavesetException: A JMS request arrived, but
    JMS PasswordSync is unavailable. Is JMS jar file available?
    ```

- `lhuser` エントリに格納されている userID に誤りがあると、不正なユーザー名エラーが発生することがよくあります。ユーザーを置き換えるか `lhuser` レジストリのキー値を有効な userID と置き換えるには、`Configure.exe` ユーティリティーを使用します。
- 不正なパスワードエラーは、`lhcred` エントリに格納されているパスワードが、userID stored in `lhuser` に格納されている userID と組み合わせて使用されたときに誤っているとよく起こります。パスワードを置き換えるには、`Configure.exe` ユーティリティーを使用します。ただし、`lhcred` レジストリキーは手動で編集しないでください。
- パスワードエントリエラーのガベージは、レジストリキーが壊れている場合やレジストリキーが手動で編集されている場合によく起こり、これがパスワードエントリのガベージの原因となります。
- この状況によって `RAEncryptor::Decrypt3DES` 内のプロセスがハングし、PasswordSync がそのエントリを複合化できなくなります。 パスワードを置き換えるには、`Configure.exe` ユーティリティーを使用してください。

調整問題のトラブルシューティング

調整タスクの問題をトラブルシューティングしようとする際は、調整ステータスのデバッグページ (`debug/Show_Reconciler.jsp`) を見て、どのリソーススレッドが稼動中か調べてください。

よくある調整問題には、次のようなものがあります。

- 調整が開始されない。
  1. 「System Threads」デバッグページ (`debug/Show_Threads.jsp`) を開き、Reconciler*server_name* タスクが実行中かどうか調べます。
  2. このタスクが実行中でない場合は、「System Settings」ページを開いて「Trace Scheduler」をクリックします。

     スケジューラを実行すると、Reconciler *server_name* タスクが再起動されます。
  3. 「System Threads」ページをもう一度確認します。Reconciler *server_name* タスクが再起動されていなければ、スケジューラがハングします。
  4. アプリケーションサーバーを再起動します。
- あるユーザーオブジェクトで調整が落ちる。
- そのオブジェクトが Identity Manager に存在する場合は、そのオブジェクトを直接修正してみます。その Identity Manager アカウントが存在していない場合は、そのリソースからアカウントのどれか 1 つをロードします。
- 調整ユーザーに適切なアクセス権があることを確認してください。
- 調整と同じコードパスで、そのオブジェクトをファイルへ展開します。
- 「System Memory Summary」ページ (`debug/Show_Memory.jsp` or `debug/Show_Memory2.jsp`) を開き、十分な空きメモリーがあることを確認します。

リポジトリ接続問題のトラブルシューティング

接続問題をトラブルシューティングするには、Identity Manager の `lh` コマンドが非常に有用です。 このコマンドは Identity Manager の Web アプリケーションインストールを使用しますが、等式からそのアプリケーションサーバーを削除します。

ここでは、次の項目について説明します。

- 156 ページの「lh コマンドを使用した問題のデバッグ方法」
- 158 ページの「DataSource 接続のテスト」

lh コマンドを使用した問題のデバッグ方法

ここでは、`lh` コマンドの使用方法について説明します。基本的なコマンドから始めて、Identity Manager の大部分を実行できるようなコマンドの使用方法まで進んでいきます。

これらのデバッグツールに慣れると、これらの lh コマンドを使用するのに独自のバリエーションが展開できるようになります。

lh setRepo -c -n の使用

一番基本的な接続テストを実行するには、lh setRepo -c -n コマンドを使用します。これで、現在のリポジトリの場所を接続しなくても調べられるようになります。Uこのコマンドを使用すると、RL や JDBC ドライバなどのパラメータが正しいかどうか検証できます。

- 正常に接続している場合は、ServerRepository.xml ブートストラップファイルを読みます。
- 接続障害がある場合は、この障害を先に解決してみます。この障害の原因は、複合化エラーによくあります。たとえば、J2EE 不一致やクラスパスの競合があることが考えられます。

lh setRepo -c -v の使用

現リポジトリの場所に 接続して調べるには、lh setRepo -c -v コマンドを使用します。(-v は、冗長出力を表示します。) このコマンドを使用すると、Identity Manager サーバーを必要とせずに、ほぼすべてのリポジトリコードが実行できます。

- 正常に接続している場合は、現リポジトリの場所に正常に接続します。
- 接続障害がある場合は、この問題を先に解決してみます。この接続問題を解決すると、DNS、ファイアウォール、リモート接続権限問題を解決するのに役立ちます。

注 – 詳細は、158 ページの「DataSource 接続のテスト」を参照してください。

---

setRepoの使用

新しいリポジトリの場所を指定したり、リポジトリを同じ場所に設定するには、setRepo コマンドをデバッグプロセス全体にわたって使用します。

このコマンドを使用すると、JAR ファイルなど必要なコンポーネントがすべて所定の位置にあるか確認できます。setRepo コマンドはまた、テーブルの所有者や権限問題をデバッグするのに userid や password などの接続情報を変えられます。

lh console の使用

WEB-INF/lib 内の JAR ファイルと、WSHOME の下にある WEB-INF/classes 内のクラスを使用している Identity Manager サーバーを実際に起動するには、このコマンドを使用します。lh console コマンドは、Identity Manager インストール環境を使用して、実際に Identity Manager アプリケーションを起動しますが、等式からアプリケーションサーバーを削除します。

- 正常に接続した場合、この問題はアプリケーションサーバー環境か設定に固有のものです。
- 接続障害が発生した場合は、障害メッセージをレビューします。
- 接続障害がアプリケーションサーバー障害と同じ場合は、これまでよりずっと変数を減らしてみてもこの問題が再現するはずです。
- アプリケーションサーバー障害と違う障害が発生する場合は、こちらの方が変数が少なく、Identity Manager コントロール下により多くの変数があるため、Identity Manager 接続問題を先に解決してみます。

DataSource 接続のテスト

DataSource 接続をテストしていると、lh setRepo -c コマンドが失敗することがあります。

この障害は、アプリケーションサーバーのデータベース接続サービスやアプリケーションサーバーのディレクトリサービスを使用できるように Identity Manager を設定していると特に起こりやすくなります。このサービスは、実行中のアプリケーションサーバーが Web アプリケーションに提供している環境でのみ機能します。

まず、目的の DataSource 設定から手順を追いながら取り掛かります。この手順に慣れたら、自分の目的に合うようにこの方法を改変することができます。

1. アプリケーションサーバーのデータベース接続サービスを省略する DataSource 以外の接続など、直接 JDBC DriverManager 接続を使用してみてください。
2. DataSource を使用しますが、アプリケーションサーバーのディレクトリサービス以外の ディレクトリサービスにある DataSource オブジェクトを格納します。

注 – ほかに使用可能なディレクトリサービスがない場合は、ローカルファイルシステムのみを使用する JNDI の参照実装など、フリーのディレクトリサービスをダウンロードすることができます。

これがうまくいったら、アプリケーションサーバーへの問題を突き止められます。

次に、有用であれば、アプリケーションサーバーのデータベース接続サービスまたはアプリケーションサーバのディレクトリサービスを追加することもできます。アプリケーションサーバーがWebアプリケーションに提供する環境外で機能しているサービスであればどちらでもかまいません。

サーバー関連問題のトラブルシューティング

アプリケーションサーバーのログを解析して、致命的なエラーとその他サーバー関連の問題がないか調べることができます。

サーバー問題をトラブルシューティングするには、アプリケーションサーバーの管理者コンソールを使用して、各モジュールのログレベルを高めます。詳細は、サーバーに付属している製品マニュアルを参照してください。

ほとんどのアプリケーションサーバーには、アプリケーションサーバーを実行するJVMの標準出力ファイル(`stdout`)と標準エラーファイル(`stderr`)に標準の場所があります。アプリケーションサーバーログを解析するには、Identity Managerアプリケーションサーバーに指定したこのログディレクトリまたはログファイルを見つけます。

- 軽度のメッセージとその他トレースを表示するには、`stdout`ファイルを開きます。
- 致命的な例外と重大な例外を表示するには、`stderr`ファイルを開きます。

注 – このトレース出力には、Identity Managerの起動とシャットダウンメッセージが表示されます。

Identity Manager Version 7.1, 以降、Oracle 10g on Sun Application Server Enterprise Edition 8.2を搭載したIdentity Managerを使用したときに、シーリング違反の例外がアプリケーションサーバーログに発生するようになりました。

この例外は、Oracle JDBCドライバを備えた複数のJava Archive file (JARファイル)を使用した場合によく起こります。

この問題を防ぐには、`CLASSPATH`にJDBCドライバJARファイルが1つしかないこと、Oracle 10gに用意されている`ojdbc14_g.jar`を使用していることを確認します。さらに、確実に正しく処理できるよう、Oracle 10gインストールに用意されている`ojdbc14_g.jar`だけを使用してください。

Service Provider問題のトラブルシューティング

WebSphereのIdentity Manager Service Provider「エンドユーザーログイン」ページを使用し、`javax.servlet.UnavailableException`が404エラーとともにブラウザに表示された場合は、IBM 1.5 JDKのプロパティーをいくつかリセットする必要があります。

次の手順を使用します。

1. was-install/java/jre/lib ディレクトリで、jaxb.properties.sample を jax.properties に変え、この 2 行のコメントを解除します。

```
javax.xml.parsers.SAXParserFactory=
        org.apache.xerces.jaxp.SAXParserFactoryImpl
javax.xml.parsers.DocumentBuilderFactory=
        org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
```

2. ファイルを保存してアプリケーションサーバーを再起動します。

SPML 設定のトラブルシューティング

SPML 設定をテストする

1. 「接続」ページを開き、「テスト」をクリックします。

   正常に接続しましたというダイアログがポップアップします。

2. 「スキーマ」ページを開き、「送信」をクリックします。

   Identity Manager サーバーでサポートされているスキーマの階層ビューが表示されます。

   正常な接続を確立できない場合は、次の操作を行います。

   - 入力した URL をダブルクリックします。
   - 受信したエラーに no response や connection refused といったフレーズが含まれている場合は、接続 URL で使用されているホストかポートに問題があると考えられます。
   - 接続は確率したが、Web アプリケーションかサーブレットが見つからなかったというエラーの場合は、WEB-INF/web.xml ファイルに問題があると考えられます。

アップグレードのトラブルシューティング

アップグレード中に問題が発生した場合は、$WSHOME/patches/ logs でぃれくとりにあるアップグレードログファイルを確認します。このログのファイル名は、タイムスタンプとアップグレードのステージに基づいています。

アップグレード後、Identity Manager が次の例外で起動できない場合は、使用中の JDK/JRE に問題がある可能性があります。

```
java.lang.IllegalStateException: Error attempting to decrypt:
Given final block not properly padded
```

以前使用していたベンダーと同じベンダー製の JDK/JRE を使用していることを確認します。たとえば、以前 IBM 製の JDK を使用していた場合は、Sun JDK にアップグレードできません。この問題を解決するには、JDK/JRE をアンインストールして、以前のベンダー製の JDK または JRE をインストールします。

◆ ◆ ◆ 第 6 章

エラーと例外

この章では、Identity Manager から生成されるエラーメッセージと例外メッセージについて説明します。

この章のトピックは、次の通りです。

Identity Manager のメッセージを使用する前に

Identity Manager のエラーメッセージと例外メッセージを使用する前に、次をお読みください。

Identity Manager のメッセージについての重要な注意点

Identity Manager のエラーメッセージと例外メッセージを使用する前に、必ず次をお読みください。

- この章の例では、例示専用のロケール (*xx_XX* ロケール) を使用しています。
- エラーメッセージを解釈する際には、次の点にご注意ください。
  - メッセージによっては、文面が違っても同じエラーメッセージ本文を指していることがあります。
  - メッセージによっては、複数のコンポーネントで使用されているものもあります。
  - 例外は通常、例外種類かコンポーネント、あるいはその両方で記載されます。

  - 例外の中には内部プログラミングエラーが原因のものもあり、Identity Managerユーザーには表示されないものもあります。
  - 例外の中には、ラッパーだけのものもあり、そのパラメータも含めて例外のものもあります。例えば、Identity Manager例外では、パスワードポリシー違反などのリソースメッセージ、アダプターコード、複合例外がラップされます。
- メッセージ内に一重引用符 (') や二重引用符 (") が付いたパラメータ化メッセージには、さらに一重引用符 (') や二重引用符 (") を付けてメッセージを拡張してください。(システムにメッセージを出力した時に、一重引用符 (') だけが表示されます。)

  例えば、次のメッセージは2つの一重引用符で囲まれます。

  ```
  "'0}"
  ```
- 問題の診断にお困りの場合は、次のSunのテクニカルサポートまでご連絡ください。

  http://www.sun.com/service/online/us

関連ドキュメントとWebサイト

展開に役立つIdentity Managerのチューニングについては、ほかにもここで紹介するドキュメントとWebサイトを参照してください。

推奨ドキュメント

Product_IDMgr; のエラーメッセージと例外メッセージについては、次のドキュメントを参照してください。

- カスタマイズしたメッセージカタログを作成する方法については、『Sun Identity Manager Deployment Guide』の第8章「Customizing Message Catalogs」を参照してください。
- システムログレポートの作成方法と編集方法については、『Sun Identity Manager 8.1 ビジネス管理者ガイド』の「システムログレポート」を参照してください。

有用なWebサイト

次の表に、有用なWebサイトの一部を紹介します。

表 6–1　有用な Web サイト

Web サイト URL	説明
http://sharespace.sun.com/gm/document-1.26.2296	Sun の Share Space の Identity Manager FAQ 注意: この FAQ にアクセスするには、Share Space ID を登録する必要があります。
http://sunsolve.sun.com/	診断ツール、フォーラム、特集記事と記事、セキュリティ情報、およびパッチデータが記載されている Sun の Web サイト。 注意: このサイトの情報は、次の 3 分野に分かれています。 ■ 社内。Sun の従業員のみ。 ■ 契約。契約アクセス権をお持ちのお客様のみ。 ■ 公開。全員。
http://www.sun.com/service/online/us	Sun テクニカルサポート Web サイト

Identity Manager のエラーと例外について

ここでは、Identity Manager のエラーメッセージと例外メッセージのシステムのほか、エラーと例外が発生する可能性のあるコンポーネントについて説明します。

これらの情報は、次のように構成されています。

- 165 ページの「メッセージの保存場所」
- 166 ページの「メッセージの表示の仕方」
- 167 ページの「エラーの重要度」

メッセージの保存場所

エラーメッセージの保存場所は次の通りです。

- Identity Manager のメッセージは、`WPMessages.properties` ファイルの `idmcommon.jar` および `RAMessages.properties` ファイルの `idmadapter.jar` に保存されます。
- `WPMessages.properties` ファイルは、プロジェクトディレクトリ `/waveset/idm/common/src/com/waveset/msgcat` にあります。
- `RAMessages.properties` ファイルは、プロジェクトディレクトリ `/waveset/idm/adapter/src/com/waveset/adapter` にあります。

Identity Manager Service Providerは、ユーザーインタフェースに文字列を表示するのに、標準Javaのメッセージリソースバンドルを使用します。このリソースバンドルは通常、idmspe.jarファイルに含まれているので、メッセージ文字列をカスタマイズする場合は展開する必要があります。

```
$ cd $WSHOME/WEB-INF/classes
$ jar xvf ../lib/idmspe.jar com/sun/idm/idmx/msgcat/I
DMXMessages.properties
```

- Identity Auditorメッセージは、AUMessages.propertiesファイルに保存されます。このファイルは、次の場所にあります。

  project_directory/waveset/idm/auditor/src/com/sun/idm/auditor/msgcat/AUMessages.properties

- Data Exporterのデフォルト実装には、WICMessages.propertiesと呼ばれる独自のメッセージバンドルが含まれています。WICMessages.propertiesファイルは、次の場所にあるexporter.jarファイルの中にあります。

  com/sun/idm/warehouse/msgcat/WICMessages.properties

注– カスタマイズしたメッセージカタログを作成すれば、メッセージを追加したり、システムに用意されているメッセージを修正することができます。

この手順については、『Sun Identity Manager Deployment Guide』の第8章「Customizing Message Catalogs」を参照してください。

メッセージの表示の仕方

わかりやすいように、Identity Managerではページの最上部にエラーアイコンが付いた枠内にエラーメッセージと例外メッセージがページごとに表示されます。

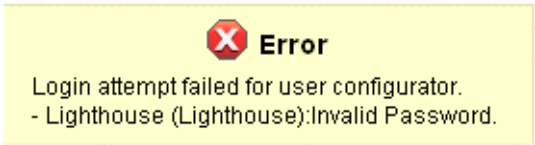


図6–1 ログイン認証エラーの例

注 – Identity Manager メッセージシステムのうち \{0\} または \{1\} のいずれかで表示される項目は、コードから提供されたパラメータを指します。たとえば、次のようにします。

```
The file is larger than the maximum supported length of
 {0} bytes.
```

この例外では、コードによって \{0\} が最大バイト数を表すパラメータ値に置き換えられます。

エラーの重要度

Identity Manager では、エラーの重要度が次のように定められています。

- 致命的。システムクラッシュの原因となる深刻なエラー。保存されていないデータは紛失するか壊れます。
- エラー。深刻なエラー。保存されていないデータは紛失するか壊れる危険性があります。データの紛失を防ぐには、早急な措置が必要です。
- 警告。今後深刻なエラーが発生しないように、いずれ措置を講じる必要があります。
- 情報。情報メッセージ。通常、サーバーの活動を表します。何もする必要はありません。

注 – エラーメッセージと例外メッセージの詳細は、Identity Manager のシステムログをチェックするとわかります。167 ページの「システムログレポートのエラー表示」を参照してください。)

システムログレポートのエラー表示

システムログレポートには、Identity Manager から生成されたエラーが説明されています。システムログレポートは、エラーのタイムスタンプ、重要度、サーバー名、コンポーネント名、エラーコードまたはエラー ID、スタックトレース (プログラム期間のうち、その時点の実行スタックの構成)、およびエラー本文から構成されています。

システムログレポートを実行して表示するには、Identity Manager の管理者インタフェースまたはコマンドラインインタフェースのいずれからでも行えます。

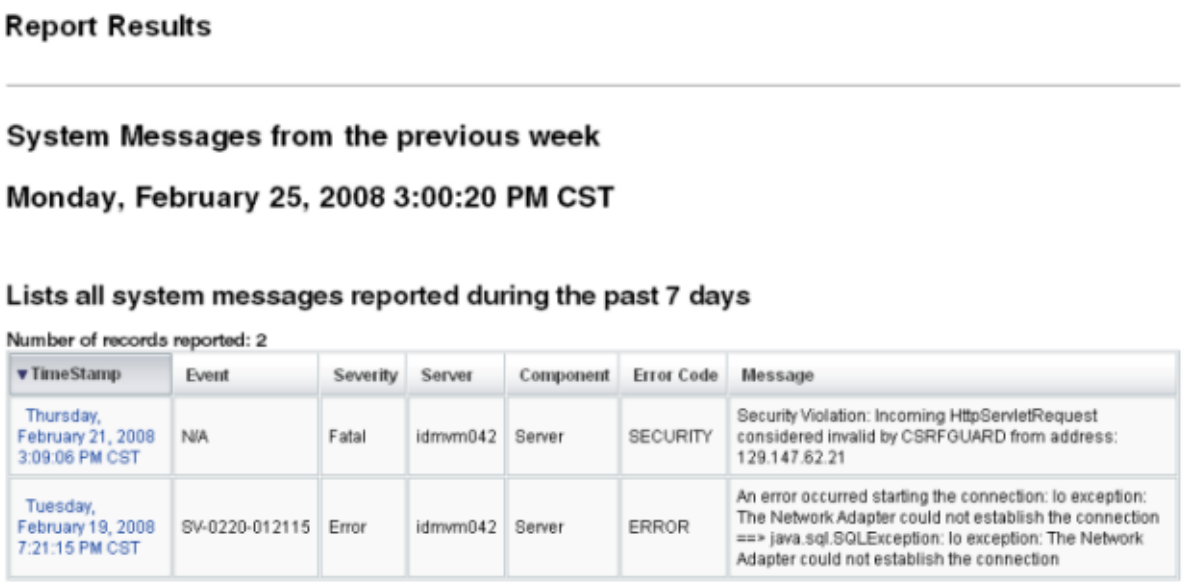
注 – システムログレポートの作成手順と編集手順は、『Sun Identity Manager 8.1 ビジネス管理者ガイド』に記載されています。

▼ 管理者インタフェースからシステムログレポートを実行する

管理者インタフェースからシステムログレポートを実行するには、次の項目を実行します。

1. **Identity Manager** 管理者インタフェースにログインします。
2. 「レポート」→「レポートの実行」の順に選択し、「レポートの実行」ページを開きます。
3. レポート種類列から該当するシステムログレポートのエントリを見つけ、その行にある「実行」ボタンをクリックします。

   「レポート結果」ページが表示され、指定した期間内にレポートされたシステムメッセージが一覧表示されます。例えば図 6–2 には、2 つのシステムメッセージが表示されています。



Report Results

System Messages from the previous week

Monday, February 25, 2008 3:00:20 PM CST

Lists all system messages reported during the past 7 days

Number of records reported: 2

▼TimeStamp	Event	Severity	Server	Component	Error Code	Message
Thursday, February 21, 2008 3:09:06 PM CST	N/A	Fatal	idmvm042	Server	SECURITY	Security Violation: Incoming HttpServletRequest considered invalid by CSRFGUARD from address: 129.147.62.21
Tuesday, February 19, 2008 7:21:15 PM CST	SV-0220-012115	Error	idmvm042	Server	ERROR	An error occurred starting the connection: Io exception: The Network Adapter could not establish the connection ==> java.sql.SQLException: Io exception: The Network Adapter could not establish the connection

図 6–2 「レポート結果」ページの例

このレポート結果表には、次の情報が表示されています。

- 「タイムスタンプ」。エラーが発生した日、曜日、時刻が表示されます。

  このシステムログレコードの詳細を表示するには、「タイムスタンプ」のリンクをクリックします。例えば、図 6–2 の「タイムスタンプ」リンクをクリックすると、次の除法が表示されます。

- 「イベント」。対象となるエントリに syslog ID があれば、syslog ID が表示されます。
- 「重要度」。エラーの重要度が表示されます。

  重要度には、次の種類があります。

  - 致命的。システムクラッシュの原因となる深刻なエラー。保存されていないデータは紛失するか壊れます。
  - エラー。深刻なエラー。保存されていないデータは紛失するか壊れる危険性があります。データの紛失を防ぐには、早急な措置が必要です。
  - 警告。今後深刻なエラーが発生しないように、いずれ措置を講じる必要があります。
  - 情報。情報メッセージ。通常、サーバーの活動を表します。何もする必要はありません。
- 「サーバー」。エラーが発生したサーバーを示します。
- 「コンポーネント」。エラーが生成されたシステムコンポーネントを示します。
- 「エラーコード」。そのエラーに関連付けられているエラーコードが表示されます。
- 「メッセージ」。実際のエラーメッセージ本文が表示されます。

▼ コマンドラインインタフェースからシステムログレポートを実行する

注 – この手順の説明は、Identity Manager コマンドラインインタフェースと lh コマンドを使い慣れた方を対象としています。詳細は、『『Sun Identity Manager 8.1 ビジネス管理者ガイド』』の『Sun Identity Manager 8.1 ビジネス管理者ガイド』の付録 A「lh リファレンス」をお読みください。

コマンドラインからシステムログレポートを実行して表示するには、次の項目を実行します。

1. コマンドウィンドウを開きます。
2. **Identity Manager** のデフォルトインストール先ディレクトリに移動します。
3. プロンプトで、`lh syslog [オプション ]` コマンドを入力します。
   次のオプションを使用して、情報を含めるか除外します。
   - `-d`: 数値 – 何日前かのレコードが表示されます (デフォルトは 1 です)。
   - `-F` – 重要度が致命的なレコードだけが表示されます。
   - `-E` – 重要度がエラー以上のレコードだけが表示されます。
   - `-i` *logid* – 指定した Syslog ID のレコードだけが表示されます。
     syslog ID は一部のエラーメッセージに表示され、特定のシステムログエントリを参照するものです。
   - `-W` – デフォルトで重要度が警告以上のレコードだけが表示されます。
   - `-X` – エラーの原因が報告されていれば、エラーの原因が表示されます。

デフォルトエラーメッセージのカスタマイズ

注 – メッセージカタログエントリを追加したり、システムに用意されているエントリを修正するには、カスタマイズしたメッセージをカタログを作成する必要があります。この手順については、『Sun Identity Manager Deployment Guide』の第 8 章「Customizing Message Catalogs」を参照してください。

Identity Manager のデフォルトのエラーメッセージは、`WPMessages.properties` ファイルの `idmcommon.jar` ファイル、および `R AMessages.properties` ファイルの `idmadapter.jar` に保存されます。

- `WPMessages.properties` ファイルは、プロジェクトディレクトリ `/waveset/idm/common/src/com/waveset/msgcat` にあります。
- `RAMessages.properties` ファイルは、プロジェクトディレクトリ `/waveset/idm/adapter/src/com/waveset/adapter` にあります。

Identity Manager &Product_IDM_SPE のデフォルトエラーメッセージは、`IDMXMessages.properties` ファイルにあります。

これらのデフォルトエラーメッセージをカスタマイズするには、そのメッセージに用意されている `ErrorUIConfig` オブジェクトの属性を修正します。

▼ `ErrorUIConfig` オブジェクトを修正する

1. **Identity Manager** 管理者インタフェースにログインします。
2. ブラウザに `http://` *host:port*`/idm/debug` と入力して、「**System Settings**」ページを開きます。
3. 「**List Objects**」ボタンの隣にある「**Type**」メニューを見つけます。メニューから「**Configuration**」を選択します。

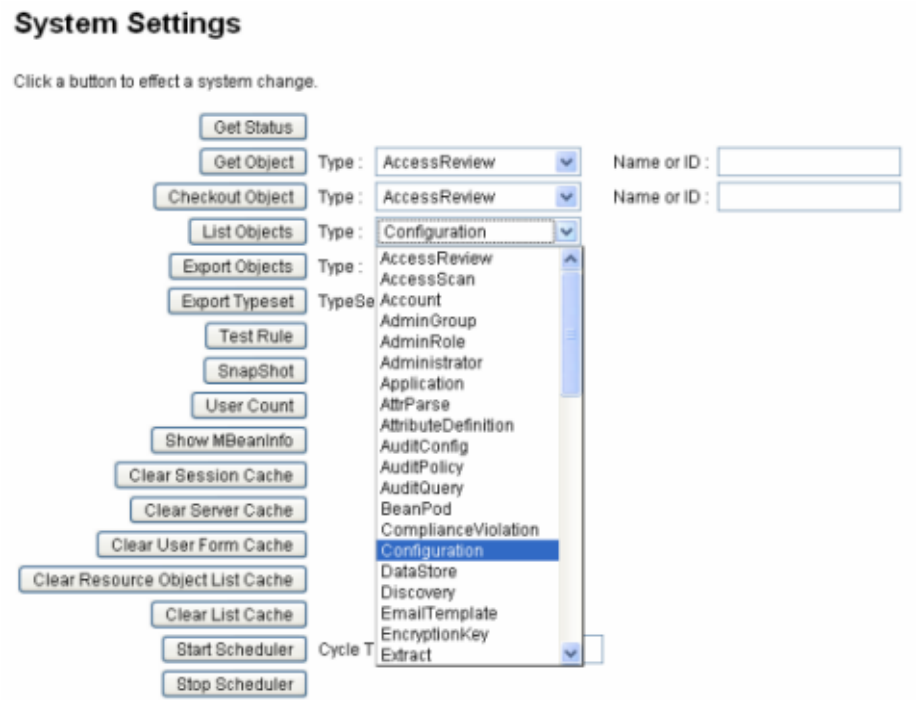


図 6–3 「List Objects」の「Type」メニュー

4 「**List Objects**」ボタンをクリックします。

5 「**List Objects of type: Configuration**」ページで、`ErrorUIConfig` **Edit** リンクをクリックします。

次の例は、「Checkout Object: Configuration」ページの `ErrorUIConfig` オブジェクトに使用する XML を示したものです。

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<!--  MemberObjectGroups="#ID#Top" extensionClass="GenericObject"
id="#ID#9787BA467E01441B:178655:1156195C008:-7FFE"
lastModifier="com.waveset.object.ErrorUIConfig" name="ErrorUIConfig"-->
<Configuration id='#ID#9787BA467E01441B:178655:1156195C008:-7FFE' name='ErrorUIConfig'
lock='Configurator#1200600790328' creator='com.waveset.object.ErrorUIConfig'
createDate='1191343145328' lastModifier='com.waveset.object.ErrorUIConfig'
lastModDate='1191343145328'>
  <Extension>
    <Object>
      <Attribute name='Enabled'>
        <Boolean>true</Boolean>
      </Attribute>
```

```
      <Attribute name='ErrorMsgID' value='UI_ERROR_DEFAULT_FATAL_MESSAGE'/>

    </Object>
  </Extension>
  <MemberObjectGroups>
    <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
  </MemberObjectGroups>
</Configuration>
```

6 ErrorUIConfig オブジェクト属性のうち、修正できるものは次の通りです。

- 「**Enabled**」。メッセージを有効 (true) または無効 (false) にします。

  注 – この属性を無効にできる機能は、下位互換性のために用意されていますが、この属性を無効にすると不可解なメッセージとなり、システムログに通常の拡張メッセージが表示されなくなります。

- 「**ErrorMsgID**」。表示対象のエラーメッセージを指します。

  表示対象とするテキストメッセージを用意するには、この ErrorMsgID 属性値を変更します。

  この属性の現在の設定は次の通りです。また、メッセージカタログ内で UI_ERROR_DEFAULT_FATAL_MESSAGE メッセージを参照しています。

  ```
  <Attribute name='ErrorMsgID' value='UI_ERROR_DEFAULT_FATAL_MESSAGE'/>
  ```

  注 – メッセージ内に一重引用符 (') や二重引用符 (") が付いたパラメータ化メッセージには、さらに一重引用符 (') や二重引用符 (") を付けてメッセージを拡張してください。(システムにメッセージを出力した時に、一重引用符 (') だけが表示されます。)

  例えば、次のメッセージは 2 つの一重引用符で囲まれます。

  ```
  "'{0}"
  ```

7 完了したら、「**Save**」をクリックして変更内容を保存します。

注 – デフォルトエラーメッセージの作成にお困りの場合は、貴社の管理者または管理者ホットラインまでご連絡ください。

索引

テ

デ

ト