HP Diagnostics

Windows®，UNIX，および Linux オペレーティング・システム用

ソフトウェア・バージョン : 9.20

インストールおよび設定ガイド

ドキュメント・リリース日 : 2012 年 5 月（英語版）
ソフトウェア・リリース日 : 2012 年 5 月（英語版）

ご注意

保証

HP 製品，またはサービスの保証は，当該製品，およびサービスに付随する明示的な保証文によってのみ規定されるものとします。ここでの記載で追加保証を意図するものは一切ありません。ここに含まれる技術的，編集上の誤り，または欠如について，HP はいかなる責任も負いません。

ここに記載する情報は，予告なしに変更されることがあります。

権利の制限

機密性のあるコンピュータ・ソフトウェアです。これらを所有，使用，または複製するには，HP からの有効な使用許諾が必要です。商用コンピュータ・ソフトウェア，コンピュータ・ソフトウェアに関する文書類，および商用アイテムの技術データは，FAR12.211 および 12.212 の規定に従い，ベンダーの標準商用ライセンスに基づいて米国政府に使用許諾が付与されます。

著作権について

商標について

Java は，Oracle Corporation およびその関連会社の登録商標です。

Adobe® および Acrobat® は，Adobe Systems Incorporated の商標です。

Microsoft®, Windows®, Windows® NT, Windows® XP, および Windows Vista® は，Microsoft Corporation の米国登録商標です。

Oracle® は，Oracle Corporation およびその関連会社の登録商標です。

UNIX® は，The Open Group の登録商標です。

謝辞

本製品には，Apache Software Foundation（http://www.apache.org/）（英語サイト）によって開発されたソフトウェアが含まれています。

本製品には，Spice Group（http://spice.codehaus.org）（英語サイト）によって開発されたソフトウェアが含まれています。

オープン・ソースおよびサード・パーティの使用許諾契約の詳細については，製品のインストール・メディアの Documentation ディレクトリを参照してください。

ドキュメントの更新情報

このマニュアルの表紙には，以下の識別番号が記載されています。

- ソフトウェアのバージョン番号は，ソフトウェアのバージョンを示します。
- ドキュメント・リリース日は，ドキュメントが更新されるたびに変更されます。
- ソフトウェア・リリース日は，このバージョンのソフトウェアのリリース期日を表します。

最新の更新のチェック，またはご使用のドキュメントが最新版かどうかのご確認には，次のサイトをご利用ください。

http://support.openview.hp.com/selfsolve/manuals

このサイトを利用するには，HP Passport への登録とサインインが必要です。HP Passport ID を登録するには，以下の Web サイトにアクセスしてください。

http://h20229.www2.hp.com/passport-registration.html （英語サイト）

または，HP Passport のログイン・ページの［**New users - please register**］リンクをクリックします。

適切な製品サポート・サービスをお申し込みいただいたお客様は，更新版または最新版をご入手いただけます。詳細は，HP の営業担当にお問い合わせください。

サポート

次の HP ソフトウェアのサポート Web サイトを参照してください。

http://support.openview.hp.com/

HP ソフトウェアが提供する製品，サービス，サポートに関する詳細情報をご覧いただけます。

HP ソフトウェア・オンラインではセルフソルブ機能を提供しています。お客様の業務の管理に必要な対話型の技術支援ツールに素早く効率的にアクセスいただけます。HP ソフトウェア・サポート Web サイトのサポート範囲は次のとおりです。

- 関心のある技術情報の検索
- サポート・ケースとエンハンスメント要求の登録とトラッキング
- ソフトウェア・パッチのダウンロード
- サポート契約の管理
- HP サポート窓口の検索
- 利用可能なサービスに関する情報の閲覧
- 他のソフトウェア・カスタマとの意見交換
- ソフトウェア・トレーニングの検索と登録

一部を除き，サポートのご利用には，HP Passport ユーザとしてご登録の上，ログインしていただく必要があります。また，多くのサポートのご利用には，サポート契約が必要です。HP Passport ID を登録するには，以下の Web サイトにアクセスしてください。

http://h20229.www2.hp.com/passport-registration.html （英語サイト）

アクセス・レベルに関する詳細は，以下の Web サイトにアクセスしてください。

http://support.openview.hp.com/access_level.jsp

目次

第 III 部：JAVA および .NET AGENT のインストールおよびセットアップ

第 VI 部 : DIAGNOSTICS メトリックス・コレクタの設定

第 VII 部 : ほかの HP ソフトウェア製品との統合のセットアップ

第 VIII 部 : DIAGNOSTICS サーバおよび JAVA と .NET AGENT の詳細設定

はじめに

『HP Diagnostics インストールおよび設定ガイド』へようこそ。本書では HP Diagnostics コンポーネントをインストールして設定する方法について説明します。また，ほかの HP ソフトウェア製品との統合の概要についても説明します。

本書の構成

本書は，次の各部で構成されています。

第 I 部　インストールの準備

Diagnostics コンポーネントのインストールおよび設定について計画を立てたり，準備したりするための情報と手順を示します。

第 II 部　Diagnostics サーバと Collector のインストール

HP Diagnostics Server および HP Diagnostics Collector をインストールして設定する方法について説明します。

第 III 部　Java および .NET Agent のインストールおよびセットアップ

Diagnostics Agent のインストールおよび設定のプロセスについて説明します。

第 IV 部　Java および .NET アプリケーションを監視するためのカスタム・インストルメンテーション

HP Diagnostics がパフォーマンス・メトリックスの収集を可能にするために監視対象アプリケーションのクラスとメソッドに適用するインストルメンテーションの制御方法について説明します。

第 V 部　Diagnostics サーバおよび Java と .NET Agent の詳細設定

Diagnostics サーバ，および Diagnostics の .NET Agent と Java Agent の詳細設定について説明します。

第 VI 部　プロキシおよびファイアウォールを介した 通信の設定

さまざまな通信チャネルを使用して Diagnostics デプロイメントをセットアップする方法について説明します。

第 VII 部　Diagnostics メトリックス・コレクタの設定

メトリックスのキャプチャ，および .NET Agent と Java Agent の測定値コレクタの設定方法について説明します。

第 VIII 部　ほかの HP ソフトウェア製品との統合のセットアップ

HP Diagnostics と統合するための LoadRunner，Performance Center，および Business Service Management のセットアップ方法について概要を示します。

第 IX 部　付録

次のような，Diagnostics 管理者の管理タスクについて説明し，技術的なデータ・フロー図を示します。

- 管理 UI を使用した Diagnostics の設定および管理
- ユーザ，権限，承認，および認証のセットアップ
- コンポーネント間のセキュアな HTTPS 通信の有効化
- システム状況 UI の使用
- データの管理およびバックアップと回復の実行
- Diagnostics のアップグレードおよび更新パッチのインストール
- データ・エクスポート機能の使用
- トラブルシューティングおよびその他の参照情報の検索

HP Diagnostics オンライン・ドキュメント

ご使用の HP Diagnostics アプリケーションには次のドキュメントが付属しています。

- **『Diagnostics ユーザ・ガイド』およびオンライン・ヘルプ :** HP Diagnostics を使用してエンタープライズ・アプリケーションのパフォーマンスを分析する方法について説明します。オンライン・ヘルプにアクセスして HP Diagnostics の使用法を参照するには，Diagnostics UI の［**ヘルプ**］ボタン，または統合 HP ソフトウェア製品のヘルプ・メニューを使用します。ユーザー・ガイドの PDF バージョンには，Diagnostics のオンライン・ヘルプのホーム・ページ，Windows の［スタート］メニュー（［**スタート**］>［**プログラム**］>［**HP Diagnostics Server**］>［**User Guide**］), HP Diagnostics インストール・ディスクの **Documentation** ディレクトリ，Diagnostics サーバのインストール・ディレクトリからアクセスできます。
- **『Diagnostics インストールおよび設定ガイド』:** Diagnostics コンポーネントをインストールして設定する方法，およびほかの HP ソフトウェア製品と統合するための Diagnostics の設定方法について説明します。このガイドの PDF には，Diagnostics のオンライン・ヘルプのホーム・ページ，Diagnostics インストール・ディスクの **Documentation** ディレクトリ，Diagnostics サーバ・インストール・ディレクトリ，Windows の［スタート］メニュー（［**スタート**］>［**プログラム**］>［**HP Diagnostics Server**］>［**Install Guide**］）からアクセスできます。
- **Diagnostics の FAQ :** FAQ の回答が記載されています。Diagnostics のオンライン・ヘルプから PDF にアクセスできます。
- **Diagnostics のデータ・モデルおよびクエリ API :** データにアクセスするために使用できる Diagnostics のデータ・モデルおよびクエリ API について説明します。Diagnostics のオンライン・ヘルプから PDF にアクセスできます。
- **Readme :** HP Diagnostics の最新の技術情報やトラブルシューティング情報が記載されています。Readme ファイルは HP Diagnostics インストール・ディスクのルート・ディレクトリに保管されています。また，アップグレード・リリースまたはパッチ・リリースをインストールするための詳細情報が記載された『アップグレードとパッチ・インストールの手順』ドキュメントもあります。
- **Diagnostics Java Agent Guide :** Diagnostics Java Agent および Diagnostics Profiler for Java のインストール，設定，および使用方法を示します。このガイドの PDF には，Agent システムの ¥docs ディレクトリ，Java Diagnostics Profiler UI のオンライン・ヘルプ・リンク，HP Diagnostics インストール・ディスクの Documentation ディレクトリからアクセスできます。

➤ **Diagnostics .NET Agent Guide :** Diagnostics .NET Agent および Diagnostics Profiler for .NET のインストール，設定，および使用方法を示します。このガイドの PDF には, Agent システムの ¥docs ディレクトリ, .NET Diagnostics Profiler UI のオンライン・ヘルプ・リンク，HP Diagnostics インストール・ディスクの Documentation ディレクトリからアクセスできます。

注 : Diagnostics Agent ガイドの情報は,『**Diagnostics インストールおよび設定ガイド**』と『**Diagnostics ユーザ・ガイド**』の情報に基づいています。

その他のオンライン・リソース

HP ソフトウェア Web サイト : HP ソフトウェア Web サイトにアクセスします。このサイトでは, HP ソフトウェア製品の最新情報をご覧になれます。新しいソフトウェアのリリース，セミナー，展示会，カスタマ・サポートなどの情報も含まれています。［**ヘルプ**］＞［**HP ソフトウェア Web サイト**］を選択します。この Web サイトの URL は，www.hp.com/go/software（英語サイト）です。

HP ソフトウェア・サポート : HP ソフトウェア・サポート Web サイトにアクセスします。このサイトで, セルフ・ソルブ・ナレッジベースを参照できます。また, ユーザ・ディスカッション・フォーラムへの投稿や検索，サポート依頼の送信, パッチや更新されたドキュメントのダウンロードなども行えます。［**ヘルプ**］＞［**HP ソフトウェア サポート**］を選択します。この Web サイトの URL は, www.hp.com/go/hpsoftwaresupport（英語サイト）です。

一部を除き，サポートのご利用には，HP Passport ユーザとしてご登録の上，ログインしていただく必要があります。また, 多くのサポートのご利用には, サポート契約が必要です。

アクセス・レベルに関する詳細は, 以下の Web サイトにアクセスしてください。

http://support.openview.hp.com/access_level.jsp

HP Passport ユーザ ID を登録するには，以下の Web サイトにアクセスしてください。

http://h20229.www2.hp.com/passport-registration.html（英語サイト）

ドキュメントの更新情報

HP ソフトウェアでは，製品ドキュメントを常に新しい情報で更新しています。

最新の更新のチェック，またはご使用のドキュメントが最新版かどうかのご確認には，HP ソフトウェア製品マニュアル Web サイト（http://support.openview.hp.com/selfsolve/documents）に進みます。

第 I 部

インストールの準備

本項の内容

➤ HP Diagnostics のインストールの準備

1

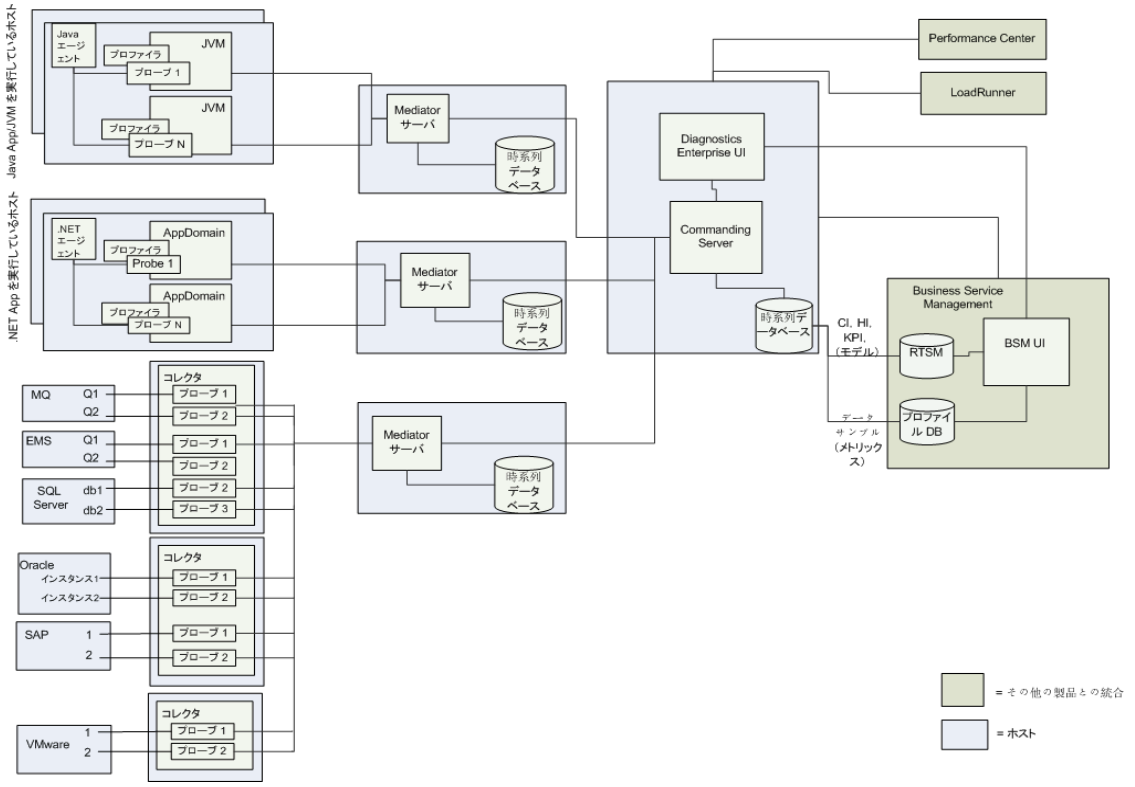
HP Diagnostics のインストールの準備

HP Diagnostics をインストールする前に，Diagnostics コンポーネントのインストールと設定を計画して準備するのに役立つ次の情報と手順をお読みください。

本章の内容

HP Diagnostics のコンポーネントとデータ・フロー

次の図は，Diagnostics のコンポーネント間のデータ・フローおよびほかの HP ソフトウェア製品との統合を示しています。



HP Diagnostics は，次のコンポーネントで構成されます。

- **Diagnostics Agents :** メソッド呼び出し，ビジネス・トランザクションとサーバ要求の開始と終了のような J2EE および .NET アプリケーションからのイベントをキャプチャし，次いで Diagnostics サーバに送信されるパフォーマンス・メトリックスの累積を行います。

  Diagnostics Agent ソフトウェアは監視対象のシステムにインストールされます。Java Agent で，監視用にアプリケーション・サーバをインストルメントします。.NET Agent で，監視用にアプリケーション・ドメインをインストルメントします。

インストルメントされた各アプリケーション・サーバまたはアプリケーション・ドメインの結果は，プローブ・エンティティで表わされる Agent インスタンスになります。Agent インストール・フォルダの多数のさまざまな設定ファイルを使用して，これらのプローブ・エンティティに関するデータ収集設定を制御できます。

- **Diagnostics Collector** Oracle データベース，SQL Server システム，IBM WebSphere MQ メッセージング・システム, TIBCO Enterprise Message Service, Software AG webMethods Broker，VMware vCenter または VMware ESX Server，SAP NetWeaver - ABAP システムを含む外部環境からデータを収集します。Diagnostics Collector をインストールし, 監視対象のこれらシステムの固有インスタンスを定義します。監視対象の各インスタンスは，Diagnostics ユーザ・インタフェースのプローブ・エンティティとして表わされます。

- **Diagnostics サーバ :** Agent，コレクタ，ほかの HP ソフトウェア製品と連携して機能し，アプリケーションのパフォーマンス・メトリックスをキャプチャ，処理し，表示します。

  Diagnostics サーバは，受信したデータを処理，集計し，ユーザ・インタフェースのビューで表示できるように情報をフォーマットします。

  Diagnostics のデプロイメントは, 1 台または多数の Diagnostics サーバで構成されることがあります。デプロイメントに Diagnostics サーバ 1 台しかない場合，それは Diagnostics コマンダ・サーバとして設定され, Commander と Mediator 両方の役割を行う必要があります。デプロイメントに複数の Diagnostics サーバがある場合，1 台は Diagnostics コマンダ・サーバとして設定され，その他はすべて（分散）Mediator として実行される必要があります。

  一般的なデプロイメントでは, Mediator として実行中の複数のサーバに接続された Diagnostics コマンダ・サーバがあります。各 Diagnostics メディエータ・サーバは，Agent とコレクタがインストールされているシステムからデータを受信するよう設定されています。Diagnostics メディエータ・サーバは，受信したイベントをフィルタし，集計します。この情報は，Diagnostics コマンダ・サーバに送信され，UI に処理されたメトリックスが表示されます。

  Diagnostics コマンダ・サーバは，命令を行い，さまざまな Diagnostics コンポーネントと，Diagnostics が動作しているほかの製品のコンポーネントの間で機能を制御します。

  コマンダ・サーバは，ほかの Diagnostics コンポーネントの場所と状態を追跡し，ほかのコンポーネント間の通信ハブの役割を果たします。

さらに，Diagnostics ユーザ・インタフェースに監視対象アプリケーションのパフォーマンス情報を表示します。

ユーザ・インタフェース : Diagnostics のメイン・ユーザ・インタフェース（Diagnostics Enterprise UI）は，複雑なパフォーマンス上の問題を解決するため，パフォーマンスの監視，問題の特定，原因の分析に使用するパフォーマンス・データを表やグラフで表示します。

ほかの HP ソフトウェア製品で Diagnostics を使用している場合，ほかの製品のユーザ・インタフェースから Diagnostics Enterprise UI にアクセスできます。たとえば，HP Business Service Management から Diagnostics Enterprise UI にアクセスできます。また，実稼動前の負荷テストの際，HP LoadRunner または HP Performance Center から Diagnostics Enterprise UI にアクセスできます。

Diagnostics では，別のユーザ・インタフェース（Diagnostics Profiler UI）に表示される Java Profiler および .NET Profiler を Agent システムで直接使用するか，Diagnostics のメイン・ユーザ・インタフェースからのドリルダウンとして使用することができます。

- **統合 :** Diagnostics は次のほかの HP ソフトウェア製品とも統合されています。詳細については，第 VII 部，「ほかの HP ソフトウェア製品との統合のセットアップ」を参照してください。また，オンラインおよびユーザーズ・ガイドの「ほかの HP ソフトウェア製品との統合」の項も参照してください。
  - HP Business Service Management
  - HP LoadRunner
  - HP Performance Center
  - HP SiteScope
  - HP Continuous Delivery Automation（CDA）

サポートされているアプリケーション・サーバおよび環境

HP Diagnostics では，次の監視をサポートしています。

- **Java EE ベースのアプリケーション・サーバ :** WebLogic, WebSphere, Oracle, Sun Java Enterprise Server，JBoss など。
- **.NET ベースのアプリケーション・サーバ :** HP Diagnostics では Microsoft IIS .NET Framework をサポートしています。

- **SAP NetWeaver – ABAP システム**
- **Oracle データベース**
- **SQL Server データベース**
- **IBM WebSphere MQ システム**
- **TIBCO Enterprise Message Service（EMS）システム**
- **VMware vCenter または VMware ESX Server**
- **Software AG webMethods Broker**

サポートされている環境の最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

Diagnostics コンポーネントのシステム要件

次のセクションでは，HP Diagnostics のコンポーネントをホスティングするための推奨システム設定について説明します。前セクションのデプロイメント図を参照して，この項で説明するコンポーネント・ホストについて理解しておいてください。

Diagnostics コンポーネントをホストするマシンを選択した場合，マシンのシステム設定が処理の負荷と，監視するアプリケーションの数をサポートしていることを確認する必要があります。

本項の内容

➤ 37 ページ「Diagnostics .NET Profiler ユーザ・インタフェースのホストの 要件」

➤ 37 ページ「Diagnostics Collector ホストの要件」

Diagnostics コンポーネントのサポートされている環境

Diagnostics コンポーネントのサポートされている環境の最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

Diagnostics サーバと Diagnostics Collector では Java 1.6 JVM が使用されます。

重要 : Diagnostics Linux インストーラ（サーバ，エージェント，コレクタの 32 ビットと 64 ビット両方）では，グラフィック・モードでインストーラを実行するには，64 ビットの Linux システムにパッチ libXtst-1.0.1-3.1 がインストール済みである必要があります。

Diagnostics Enterprise UI の要件

Diagnostics Enterprise UI は，Java アプレットを使って Web ブラウザに表示されます。Java アプレットでは，UI にアクセスするクライアント・システムに JRE 1.6 以上をインストールする必要があります。サポートされるブラウザには，Microsoft Internet Explorer 7，8，9 および Mozilla Firefox 3.5，3.6，5，6 があります。サポートされているブラウザの最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

Diagnostics サーバ・ホストの要件

Diagnostics サーバのホストのシステム設定要件は，Diagnostics サーバにレポートするプローブとメディエータ・サーバの数に応じて異なります。サーバが Diagnostics コマンダ・サーバとして指定されている場合，通常プローブ・データはそのサーバにレポートする各メディエータ・サーバに保存されます。

Diagnostics サーバを SAN ストレージ・デバイスにインストールする場合，その SAN の読み取り速度と書き込み速度はミッド・エンドからハイ・エンドのドライブと同等である必要があります（32 ページ「スケーラビリティ情報」を参照）。

注：表内の要件は，サーバ要求の数と要求の深さが平均的なアプリケーションを監視しているプローブで実行されたテストに基づいた目安です。必要な実際のシステム要件とサポートされるプローブの数は，サーバ要求の数，サーバ要求の深さ（呼び出しプロファイル内のメソッド数），トレンド・メソッドの数，発信呼び出しの数など，監視対象の環境のいくつかの特徴によって影響されます。サーバ要求のタイプも，要件に影響します。たとえば，Web サービスにはより多くのリソースが必要であり，トリミングは適用されません。

次の表は，Java プローブからデータを受信する Diagnostics サーバ（通常はメディエータ・サーバ）のホストに対する必要なシステム要件を示します。

プラットフォーム	項目	Java プローブ 50 台以内プローブ	Java プローブ 100 台以内	Java プローブ 200 台以内
Windows	CPU	2x 2.4 GHz	2x 2.8 GHz	2x 3.4 GHz
Windows	メモリ	4 GB	4 GB	4 GB
Solaris	CPU	2x Ultra Sparc 3	2x Ultra Sparc 4	2x Ultra Sparc 4
Solaris	メモリ	4 GB	4 GB	4 GB
Linux	CPU	2x 2.0 GHz	2x 2.4 GHz	2x 2.8 GHz
Linux	メモリ	2 GB	4 GB	4 GB
すべて	ヒープ・サイズ	512 M	750 M	1280 M
すべて	ディスク	プローブあたり 4 GB		

テスト環境に関する注意

- サーバ要求あたりの呼び出しプロファイル : 5
- プローブあたりの一意のサーバ要求数 : 23

次の表は，.NET プローブからデータを受信する Diagnostics サーバ（通常はメディエータ・サーバ）のホストに対する必要なシステム要件を示します。

プラットフォーム	項目	.NET プローブ 10 台以内	.NET プローブ 20 台以内	.NET プローブ 50 台以内
Windows	CPU	1x 1.0 GHz	1x 2.0 GHz	2x 2.4 GHz
Windows	メモリ	768 MB	1 GB	3 GB
Solaris	CPU	1x Ultra Sparc 2	2x Ultra Sparc 2	2x Ultra Sparc 3
Solaris	メモリ	1 GB	1.5 GB	3 GB
Linux	CPU	1x 1.0 GHz	1x 2.0 GHz	2x 2.4 GHz
Linux	メモリ	768 MB	1 GB	3 GB
すべて	ヒープ・サイズ	350 M	700 M	1400 M
すべて	ディスク	プローブあたり 3 GB		

スケーラビリティ情報

次のスケーラビリティ数は次の参照ハードウェア設定から導かれたものです。

プラットフォーム： Windows

オペレーティング・システム： Windows Server 2008，64 ビット

CPU： Intel Xeon 5160 3.00Ghz（クアッド・コア）

メモリ： 8 GB

ディスク I/O： Smart Array P400i，RAID 0 の SCSI ドライブ 2 台（136 GB，130M B/S，順次読み取りおよび書き込み）

Java ヒープ： 5.9 GB（-Xmx6096m）；64 ビット JVM

ディスク容量： 2 ～ 4 GB，プローブあたり（ディスク容量全体は，保存期間を変更することで調整できます）

ネットワーク： 1 Gbps

注 : Diagnostics で最適なパフォーマンスを得るには，64 ビットの OS と JVM を使用することをお勧めします。

上記のリファレンス・ハードウェアのスケーラビリティに関する数値です。

Java プローブ 100 台以内 : プローブあたり 100 個のサーバ要求, 呼び出しプロファイルあたり 78 個のメソッドを 45 秒ごとに取得（標準設定）

Java プローブ 400 台以内 : プローブあたり 25 個のサーバ要求, 呼び出しプロファイルあたり 78 個のメソッドを 45 秒ごとに取得（標準設定）

Java プローブ 150 台以内 : プローブあたり 150 個のサーバ要求, 呼び出しプロファイルあたり 25 個のメソッドを 240 秒ごとに取得

Java プローブ 230 台以内 : プローブあたり 100 個のサーバ要求, 呼び出しプロファイルあたり 25 個のメソッドを 240 秒ごとに取得

Java プローブ 40 台以内 : 75 個の Web サービス操作，Web サービス操作あたり 10 個の一意のコンシューマ, 呼び出しプロファイルあたり 25 個のメソッドを 45 秒ごとに取得（標準設定）

この読み込み設定では, プローブあたり 7 GB のディスク容量が必要です。

428 ページ「大規模インストールの Diagnostics サーバの設定」も参照してください。

注 :

- プローブが多数存在する環境では，Diagnostics サーバの複数のインスタンスを作成し，サーバ・インスタンス間でプローブを分散させることにより，通常はパフォーマンス向上を達成できます。
- Diagnostics コマンダ・サーバのホストに保存される Diagnostics パフォーマンス・データに関する設定については，840 ページ「インストール前のデータ管理に関する留意点」を参照してください。
- Diagnostics サーバを最適化して処理するプローブの数を増やす方法については，453 ページ「実運用環境で処理できるプローブを増やすための Diagnostics サーバの最適化」を参照してください。

Diagnostics Java Agent ホストの要件

監視対象のシステムで Diagnostics Java Agent によって生じるオーバーヘッドは，極めて低いものです。次は，Agent の処理をサポートするための推奨空きメモリ容量とディスク容量です。

プラットフォーム :	すべてのプラットフォーム
メモリ :	50 MB の追加 RAM
ハード・ディスク空き容量 :	Java プローブの最初のインストールには 200 MB の空きディスク容量が必要です。ログ・ファイルとクラスマップの作成のために，実行時により大きい容量が必要になる可能性があります。大規模アプリケーションの場合は，ログ・ファイルとクラスマップのデータ用に，プローブごとにさらに 200 MB を利用できるようにしておくことをお勧めします。

注 : アプリケーションの起動スクリプトの Java 設定に **–Xmx???m** を追加して，JVM の最大ヒープに追加メモリを割り当てる必要があります。

ヒープ・サイズの調整 : Java Agent の最大ヒープの設定については，235 ページ「アプリケーション・サーバ内の Java Agent のヒープ・サイズの調整」を参照してください。

permgen サイズの調整 : 通常，Diagnostics Agent の追加によって使用される permgen サイズは大きくありません。ただし，Agent を使用しない一部のアプリケーションでは，permgen 制限サイズのほぼすべてを使用する場合があります。このような場合は，permgen サイズを調整する必要があります。たとえば，既存の制限サイズ x 1.05 + 5MB のようにサイズを増やすことができます。ホットスポット JVM の permgen を調整するには，-XX:MaxPermSize オプションを使用します（例 : -XX:MaxPermSize=240m）。

Diagnostics Java Profiler ユーザ・インタフェースのホストの要件

Diagnostics Profiler for Java のユーザ・インタフェースは，Java アプレットを使って Web ブラウザに表示されます。Java アプレットでは，UI にアクセスするクライアント・システムに JRE 1.6 以上をインストールする必要があります。このマシンは，次の Diagnostics Profiler URL にアクセスできる必要があります。http://<probe_host>:<probeport>/profiler。デフォルトでは，プローブは 35000 で始まる最初の空きポートに割り当てられます。サポートされるブラウザには，Microsoft Internet Explorer 7，8，9 および Mozilla Firefox 3.5，3.6，5，6 があります。サポートされているブラウザの最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

Diagnostics .NET Agent ホストの要件

監視しているシステムで .NET Agent によって生じるオーバーヘッドは, 極めて低いものです。次は, Agent の処理をサポートするための推奨空きメモリ容量とディスク容量です。

プラットフォーム	サポートされているすべてのプラットフォーム
メモリ	60 MB の追加 RAM
ハード・ディスクの空き容量	200 MB の追加容量
.NET Framework	2.0 以降

重要 : .NET Framework 1.1 をサポートする必要がある場合は，古いバージョンの .NET Agent（8.x）を使用してください。このバージョンは引き続きサポートされ，パッチによって更新されます。

WCF の要件と制限事項 : .NET Windows Communication Foundation (WCF) サービスを監視するには, .NET Framework 3.0 SP1 以降が必要です。次のバインディングのみがサポートされています。

- BasicHttpBinding
- WSHttpBinding
- NetTcpBinding

サポートされていないバインドがアプリケーションで使用されている場合，.NET プローブは各 WCF メソッドに対する一般的なサーバ要求のみを作成します。この要求は Web サービスではなく，XVM 相関は発生しません。

Diagnostics .NET Profiler ユーザ・インタフェースのホストの要件

.NET Diagnostics Profiler のユーザ・インタフェースは，Internet Explorer 7 以降を必要とする DHTML/XML/XSLT/JScript テクノロジを使って表示されます。UI を表示するのに使われるマシンは，.NET Diagnostics Profiler URL（http:// ＜プローブのホスト＞：＜プローブのポート＞ /profiler）にアクセス可能である必要があります。プローブは，プローブのインストール時に定義された範囲内の最初の空きポートに割り当てられます。標準設定のポート範囲は 35000 ～ 35100 です。

Diagnostics Collector ホストの要件

Collector は，データを収集している SAP NetWeaver-ABAP，Oracle，SQL Server，IBM WebSphere MQ，TIBCO EMS，Software AG webMethods Broker または VMware アプリケーションのホスト・マシンとやり取り可能なサポート対象システムにインストールできます。

Collector のインストールには 350MB の空きディスク容量が必要です。ログ・ファイルを作成するため，または環境が大規模であるため，実行時により大きい容量が必要になる可能性があります。

Diagnostics コンポーネントのサポートされている環境の最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

インストールに必要な情報

Diagnostics コンポーネントのインストールの前に，Diagnostics コンポーネントと，それらのコンポーネントをホストするマシンの設定を慎重に計画する必要があります。また，ネットワーク・トポグラフィ内でのコンポーネント・ホストの場所も考慮する必要があります。

次のセクションの表は，Diagnostics コンポーネントのインストール時に必要な情報の収集に役立ててください。

注：Diagnostics に Business Service Management をインストールしている場合，Diagnostics コンポーネントのホストの名前を入力する際，完全修飾ホスト名，つまりマシン名とドメイン名を使用することを強くお勧めします。

Diagnostics サーバ

必要な情報	説明
コマンダ・サーバと 1 つ以上のメディエータを使用するかどうか	Diagnostics デプロイメントの計画時に，監視する環境のサイズと複雑さに基づいてこれを判断します。
Diagnostics コマンダ・サーバの場合，サーバをホストするマシン用に生成された HP Diagnostics ライセンスの場所	HP ソフトウェア・サポート担当者に問い合わせてライセンスを請求し，Diagnostics サーバ・インストーラからアクセス可能な場所に置いてください。
Diagnostics メディエータ・サーバの場合，Diagnostics コマンダ・サーバの URL	Diagnostics コマンダ・サーバをインストールした後，URL を使用して Diagnostics ビューを開くことができます。
SaaS 環境で Diagnostics を使用するかどうか	SaaS（HP ホスト）環境に Diagnostics をデプロイする場合，異なるインストーラ・オプションが表示されます。
Business Service Management 環境で Diagnostics を統合するかどうか	Business Service Management 環境に Diagnostics をデプロイする場合，インストーラでこのオプションを選択する必要があります。

Java Agent

➤ **HP ソフトウェア製品および Diagnostics サーバ情報**

必要な情報	調べる場所	値
Agent をインストールするモード	製品ライセンスに従って選択します。	➤ **Profiler 専用（サーバに接続しない）** ➤ **LoadRunner/ Performance Center のみで使用（AD ライセンス）** ➤ **次のいずれか，または両方で使用する Enterprise モード（AM ライセンス）** 　➤ Diagnostics 　➤ TransactionVision
Diagnostics サーバの名前	Diagnostics サーバのホストの完全修飾したホスト名または IP アドレス。 スタンドアロン・モードで Java Diagnostics Profiler を使う場合，これは必要ありません。	Agent が実行されるデプロイメントに 1 つの Diagnostics サーバしかない場合，これは Diagnostics コマンダ・サーバです。 コマンダ・サーバと メディエータ・サーバのある分散環境では，これは Agent からイベントを受信する Diagnostic メディエータ・サーバです。
Diagnostics サーバ・ポート	標準設定の 2006 を使用するか，Diagnostics へのアクセス用に設定したポートを使用します。 スタンドアロン・モードで Java Diagnostics Profiler を使う場合，これは必要ありません。	**標準設定値** : 2006

➤ インストルメントされたアプリケーション・サーバと Agent 情報

必要な情報	調べる場所	値
Java Agent の名前	一意の文字列。 ユーザが作成したもの。 Agent 名は標準設定のプローブ・エンティティ名になるように割り当てられます。 Agent の名前を，監視するアプリケーションおよびプローブ・インストルメンテーションのタイプを示すものにすると，さまざまなアプリケーションとプローブのタイプを区別するのに便利です。 1 つの Java Agent 設定を使用する複数のプローブがあることがあります。この場合，後で，監視対象アプリケーションごとに一意のプローブ名を設定できます。	例： WebLogic_MedRec_java
Java agent グループ	これは Agent のインストール時にユーザが定義します。 入力する Agent グループ名はプローブグループ名として使用されます。 プローブ・グループは，同じ Diagnostics サーバに報告するプローブの論理グループです。	**標準設定値：** **標準設定**
監視用にインストルメントされるアプリケーション・サーバのタイプ	ホスト・システムの管理者。	

必要な情報	調べる場所	値
アプリケーション・サーバの設定プロパティ	ホスト・システムの管理者。 詳細は，監視しているアプリケーション・サーバによって異なります。	
JRE 実行可能ファイルの場所	ホスト・システムの管理者。 監視するアプリケーション・サーバのタイプによって異なります。詳細については，161 ページ「Java Agent で監視するためのアプリケーション・サーバの準備」を参照してください。	

.NET Agent

➤ **Diagnostics サーバ情報**

必要な情報	調べる場所	値
Agent をインストールするモード	製品ライセンスに従って選択します。	➤ **Profiler 専用（サーバに接続しない）** ➤ **LoadRunner/ Performance Center のみで使用（AD ライセンス）** ➤ **次のいずれか，または両方で使用する Enterprise モード（AM ライセンス）** 　➤ Diagnostics 　➤ TransactionVision
Diagnostics サーバの名前	Diagnostics サーバのホストの完全修飾したホスト名または IP アドレス。 スタンドアロン・モードで .NET Diagnostics Profiler を使う場合，これは必要ありません。	Agent が実行されるデプロイメントに 1 つの Diagnostics サーバしかない場合，これは Diagnostics コマンダ・サーバです。 コマンダ・サーバとメディエータ・サーバのある分散環境では，これは Agent からイベントを受信する Diagnostic メディエータ・サーバです。
Diagnostics サーバ・ポート	標準設定の 2006 を使用するか，Diagnostics へのアクセス用に設定したポートを使用します。 スタンドアロン・モードで .NET Diagnostics Profiler を使う場合，これは必要ありません。	**標準設定値 :** 2612

➤ **Agent およびポート情報**

必要な情報	調べる場所	値
Agent グループ	これは Agent のインストール時にユーザが定義します。 入力する Agent グループ名はプローブ・グループ名として使用されます。 プローブ・グループは, 同じ Diagnostics サーバに報告するプローブの論理グループです。	**標準設定値 :** 標準設定
最小 Web ポート	システム管理者。 プローブに割り当てることができる Agent システムのポート範囲の最小ポート番号。	**標準設定値 :** 35000
最大 Web ポート	システム管理者。 プローブに割り当てることができる Agent システムのポート範囲の最大ポート番号。	**標準設定値 :** 35100

プレインストールについて

注 : Windows マシンに Diagnostics コンポーネントをインストールする前に, ［管理ツール］からアクセスできる［**サービス**］ウィンドウが開いていないことを確認してください。

Diagnostics サーバ

➤ Diagnostics コマンダ・サーバに有効なライセンスが与えられるまで, HP Diagnostics のパフォーマンス・メトリックスは表示できません。ライセンスの取得およびその他のライセンスの問題については, 第 3 章,「HP Diagnostics ライセンスの有効化」を参照してください。

注 : Diagnostics ビューの最適な表示のために, 画面解像度が 1024 x 768 以上になっている必要があります。

Diagnostics Java Agent

➤ テスト時に, Java Agent を Java アプリケーションと同じシステムにインストールする必要があります。

➤ Diagnostics Profiler for Java は, 正しくライセンスが与えられた Diagnostics コマンダ・サーバに接続可能になるまで, 読み込み制限付きで非ライセンス・モードで動作します。ライセンスの取得およびその他のライセンスの問題については, 第 3 章,「HP Diagnostics ライセンスの有効化」を参照してください。

➤ Diagnostics では, Agent 名のローカリゼーションはサポートしていません。

Diagnostics .NET Agent

- テスト時に, .NET Agent を .NET アプリケーションと同じシステムにインストールする必要があります。
- Diagnostics Profiler for .NET は，正しくライセンスが与えられた Diagnostics コマンダ・サーバに接続可能になるまで, 読み込み制限付きで非ライセンス・モードで動作します。ライセンスの取得およびその他のライセンスの問題については，第 3 章,「HP Diagnostics ライセンスの有効化」を参照してください。
- Diagnostics では，Agent 名のローカリゼーションはサポートしていません。

LoadRunner および Performance Center のホスト・マシン

- LoadRunner がすでにインストールされている場合，LoadRunner Diagnostics Add-in をインストールする前に，Controller および LoadRunner のメイン・ウィンドウを閉じる必要があります。
- Performance Center では, LoadRunner Diagnostics Add-in は必要ありません。
- Diagnostics コンポーネントの時刻およびタイムゾーン設定が一致している必要があります。時刻が正しく設定されていないと，時間の誤差問題が生じます。

推奨するインストール順序

HP Diagnostics のコンポーネントをインストールする上で，慎重に計画し，準備すると，混乱やエラーを回避するのに役立ち，インストールおよび設定手順を短時間で完了できます。

注 : 製品とコンポーネントについては，次のインストール順をお勧めします。この順序を守らないと，インストール・プロセスの煩雑性が増したり，予期しない結果が生じることがあります。

始める前に，次の情報を見直して，インストールおよび設定プロセスの全容を把握してください。

推奨するインストール順序：

1 システム要件とインストール時の注意事項を確認します。

詳細については，29 ページ「Diagnostics コンポーネントのシステム要件」を参照してください。

2 Diagnostics サーバをインストールします。

詳細については，第 2 章，「Diagnostics サーバのインストール」と第 3 章，「HP Diagnostics ライセンスの有効化」を参照してください。

3 Diagnostics Agent とコレクタをインストールします。

Java 環境の場合，第 5 章，「Java Agent のインストール」を参照してください。

.NET 環境の場合，第 8 章，「.NET Agent のインストール」を参照してください。

Oracle データベース，SAP NetWeaver-ABAP，SQL Server データベース，VMware vCenter または VMware ESX Server，WebSphere MQ，TIBCO EMS，Software AG webMethods Broker 環境の場合，第 4 章，「Diagnostics Collector のインストール」を参照してください。

4 Java Agent の場合，Diagnostics によって監視するアプリケーション・サーバをインストルメントします（この結果，Agent インスタンスが Diagnostics のプローブ・エンティティとして表示されます）。

詳細については，第 6 章，「Java Agent で監視するためのアプリケー ション・サーバの準備」を参照してください。

5 Agent インストール・フォルダの多数のさまざまな設定ファイルを使用して，インストルメンテーションをカスタマイズし，データ収集設定を制御します。

詳細については，「Java および .NET アプリケーションを監視するためのカスタム・インストルメンテーション」と「Diagnostics サーバおよび Java と .NET Agent の詳細設定」に関するセクションを参照してください。

6 HP Diagnostics が LoadRunner, Performance Center または Business Service Management と統合されている場合，HP Diagnostics を使用するためにこれらの各製品を設定する必要があります。

Business Service Management については，第 21 章，「Business Service Management と Diagnostics 間の統合のセットアップ」を参照してください。

LoadRunner の統合については, LoadRunner Diagnostics Add-in をインストールし（第 22 章,「LoadRunner Diagnostics Add-in のインストール」を参照), Diagnostics を使うように LoadRunner をセットアップします（第 23 章,「HP LoadRunner と HP Diagnostics の統合のセットアップ」を参照)。

Performance Center については，第 24 章,「Diagnostics を使用するための Performance Center のセットアップ」を参照してください。

HP Diagnostics のライセンス

Diagnostics ビューでアプリケーションの測定値を表示するには, Diagnostics コマンダ・サーバの有効なライセンスを取得する必要があります。ライセンスの取得およびその他のライセンスの問題については，第 3 章,「HP Diagnostics ライセンスの有効化」を参照してください。

前バージョンの Diagnostics からのアップグレード

前バージョンの製品がインストールされている環境，または Diagnostics の機能にアクセスするためにほかの HP ソフトウェア製品をアップグレードする必要がある環境に HP Diagnostics をインストールする場合，付録 G,「アップグレードとパッチ・インストールの手順」の手順で行います。これらの手順では，現在の HP ソフトウェア製品および Diagnostics コンポーネントをアップグレードするための適切な方法を紹介します。

第 II 部

Diagnostics サーバと Collector のインストール

本項の内容

- Diagnostics サーバのインストール
- HP Diagnostics ライセンスの有効化
- Diagnostics Collector のインストール

2

Diagnostics サーバのインストール

本章では, Windows および UNIX マシンに Diagnostics サーバをインストールする方法について説明します。

本章の内容

Diagnostics サーバのインストール

本章では，次の環境に適用される，Diagnostics サーバの詳細なインストール手順を示します。

- Windows 環境
- グラフィック・インストーラを使用するほとんどの UNIX 環境。コンソール・モードのコマンドライン・インタフェースを使用して UNIX でインストールすることもできます。

サポートされているプラットフォームの最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。サポート早見表に記載されていないその他のプラットフォームのインストールの詳細については，HP サポートにお問い合わせください。

注 : 以前のバージョンの Diagnostics サーバがマシンにインストールされている場合は，付録 G,「アップグレードとパッチ・インストールの手順」を参照してください。

ルート・アクセス要件 : Diagnostics コマンダ・サーバを Business Service Management 9.00 以降と統合する場合は，Diagnostics サーバのインストール時にルート・アクセスが必要です。ルート・アクセスは，OM エージェントと IAPA コンポーネントのインストールに必要です。

ルート・アクセスなしで Diagnostics サーバをインストールする必要がある場合は, OM エージェントと IAPA コンポーネントをインストールしないで後から手動でインストールしてもかまいません。OM エージェントおよび IAPA コンポーネントのインストール・ダイアログ・ボックスでチェックボックスをオフのまま残して，後からインストールします（72 ページ「OM エージェントおよび IAPA コンポーネントの手動インストール」を参照）。

ルート権限は，Solaris および Linux システムでブート時にサーバを自動起動するよう設定する場合にも必要です。

HP Software-as-a-Service（SaaS）: HP Diagnostics は HP Software-as-a-Service（SaaS）環境にデプロイできます。SaaS デプロイメントでは，Diagnostics Java Agent が企業の IT 環境にインストールされ，Diagnostics コマンダ・サーバとメディエータ・サーバが HP によって社内の SaaS システムにインストールされます。このため，SaaS 環境で Diagnostics を使用する顧客は企業のシステムで Diagnostics サーバをインストールしないのが一般的であるため，本章を無視してもかまいません。SaaS デプロイメントでは，顧客は HP の SaaS 管理者によって社内で設定されたサーバに接続する Java Agent をインストールするのみです。詳細については，SaaS 管理者にお問い合わせください。

本項の内容

➤ 53 ページ「Diagnostics サーバ・インストーラの起動」

➤ 56 ページ「インストールの実行」

Diagnostics サーバ・インストーラの起動

使用環境に応じて，Windows インストーラまたは UNIX インストーラを起動します。66 ページ「Diagnostics サーバのサイレント・インストール」も参照してください。

注 : temp ディレクトリに対して約 400 MB の空き領域を使用可能にします。

Windows インストーラにアクセスするには，次の手順を実行します。

1 Diagnostics DVD（Autorun.exe）からインストール・メニュー・ページが表示されます。32 ビット Windows バージョンの Diagnostics サーバをインストールするには，メニューから［**Diagnostics Server 32 ビット**］を選択します。64 ビット Windows バージョンの Diagnostics サーバをインストールするには，［**Diagnostics Server 64 ビット**］を選択します。

2 **Diagnostics_Servers** ディレクトリにある実行ファイル **HPDiagServer_＜リリース番号＞ _win32.exe**（32 ビット用）または **HPDiagServer_ ＜リリース番号＞ _win64.exe**（64 ビット JVM で実行される 64 ビット用）をダブルクリックすると，適切なインストーラを実行できます。

56 ページ「インストールの実行」に進みます。

UNIX インストーラにアクセスするには，次の手順を実行します。

1 Diagnostics のインストール先から **Diagnostics_Servers** ディレクトリにアクセスします。Diagnostics サーバをインストールするマシンに適切なインストーラである **HPDiagServer_ ＜リリース番号＞ _ ＜プラットフォーム＞ .bin** をコピーします。

2 インストーラ・ファイルのモードを変更して実行可能にします。

3 インストーラを実行します。

- グラフィック・モードでインストーラを実行するには，次のように UNIX コマンド・プロンプトでインストーラのファイル名 **HPDiagServer_ ＜リリース番号＞ _ ＜プラットフォーム＞ .bin** を入力します。

```
./HPDiagServer_9.00_linux.bin
```

- コンソール・モードでインストーラを実行するには，次のように UNIX コマンド･プロンプトでインストーラのファイル名 **HPDiagServer_ ＜リリース番号＞ _ ＜プラットフォーム＞ .bin** を入力し，**-console** オプションを指定します。

```
./HPDiagServer_9.00_linux.bin -console
```

56 ページ「インストールの実行」に進みます。

HP ソフトウェアのダウンロード・センターからインストーラをダウンロードするには，次の手順を実行します。

1 HP ソフトウェア Web サイトのソフトウェア・ダウンロード・センターにアクセスします。

2 **Diagnostics** 情報を見つけ，適切なリンクを選択して Diagnostics サーバ・ソフトウェアをダウンロードします。

3 Web サイトのダウンロード手順を行います。

56 ページ「インストールの実行」に進みます。

Business Service Management の Diagnostics [ダウンロード] ページからインストーラをダウンロードするには，次の手順を実行します。

1 Business Service Management のトップ・メニューから，［**管理**］＞［**Diagnostics**］を選択し，［**ダウンロード**］タブをクリックします。

2 ［ダウンロード］ページでリンクをクリックして，適切な Diagnostics サーバ・インストーラをダウンロードします。

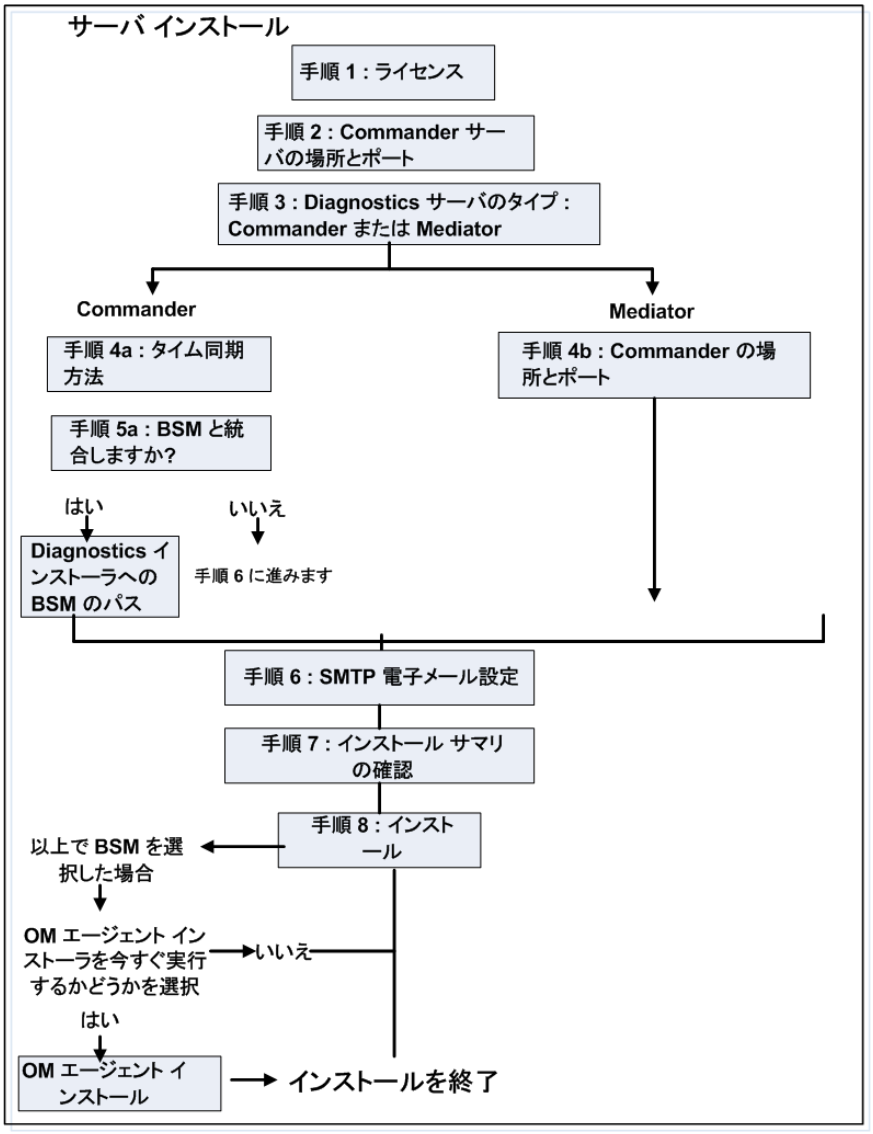
注 : Java Agent インストーラが Business Service Management で利用できるのは，インストーラが Business Service Management のアクセスできるディレクトリにある場合のみです。Diagnostics サーバのインストール時にこれを有効にするか，またはサーバ・インストーラをインストール・ディスクから必要な場所に手動でコピーできます。

---

56 ページ「インストールの実行」に進みます。

インストールの実行

下の図は，Diagnostics サーバのインストール手順の概要を示しています。各手順の詳細については，本項のその後の内容を参照してください。



HP 社内におけるサーバの SaaS デプロイメントの場合，HP の SaaS 管理者による追加手順が必要です（HP 内部ドキュメント）。

重要 : ホスト・マシンにすでに Java Agent がインストールされている場合，次のインストール手順ではなく Agent システムのアップグレード手順に従う必要があります。詳細については，851 ページ「アップグレードとパッチ・インストールの手順」を参照してください。

インストーラを起動すると，ソフトウェア使用許諾契約書が開きます。

Ubuntu または CentOS Linux で，アーカイブ（インストール・ファイル）の自己解凍に失敗した場合，32 ビット glibc ライブラリをインストールする必要があります。

32 ビット glibc パッケージを RHEL 6 / CentOS 6 にインストールするには，コマンド **yum install glibc.i686** を実行します。

32 ビット glibc パッケージを Ubuntu 11 にインストールするには，コマンド **sudo apt-get install libc6-i386** を実行する必要があります。

その後，もう一度インストールを試します。

インストールを開始してインストール先とモードを選択するには，次の手順を実行します。

1 ソフトウェア使用許諾契約書に同意します。

ソフトウェア使用許諾契約書が表示されます。

使用許諾契約書を読み，規約に同意します。

続行するには［**次へ**］を選択します。

注 : UNIX のコンソール・モード・インストーラの場合は，読んでいるときに，ENTER キーを押すと次のページに進み，q を入力すると使用許諾契約書の末尾にジャンプします。

---

2 Diagnostics サーバをインストールする場所を指定します。

デフォルトのディレクトリを受け入れるか，ほかの場所のパスを入力します。Windows インストーラ（または UNIX のグラフィック・モード・インストーラ）の場合は，［**参照**］をクリックして別のディレクトリを指定できます。

続行するには［**次へ**］を選択します。

注 : UNIX のコンソール・モード・インストーラでは，次に進むには 1，前に戻るには 2，キャンセルするには 3，画面を再表示するには 4 を押します。

3 インストールする Diagnostics サーバの Diagnostics サーバ・モードを指定します。

セットアップしている Diagnostics デプロイメントは，1 つ以上の Diagnostics サーバで構成できます。デプロイメントに Diagnostics サーバ 1 台しかない場合，Diagnostics サーバは Commander モードでインストールされ，Commander と Mediator の両方の役割を実行できます。デプロイメントに複数の Diagnostics サーバがある場合，1 台は Commander モードに，その他はすべてコマンダ・サーバにレポートする Mediator モードに設定されます。

- デプロイメントに Diagnostics サーバが 1 台だけある場合は，[**Commander モード**] を選択してください。
- デプロイメントに Diagnostics サーバが複数あり，現在インストールしているものを Commander モードに設定する場合は，[**Commander モード**] を選択します。その他の場合は，[**Mediator モード**] を選択します。

[**このサーバは HP Software-as-a-Service（SaaS）環境で使用されます**] チェックボックスは，HP 社内で Diagnostics サーバ（Commander または Mediator）をインストールする HP の SaaS 管理者用であるため無視します。

ここで，Commander モードと Mediator モードのどちらで Diagnostics サーバをインストールしているかによってインストールが異なります。

- Diagnostics コマンダ・サーバをインストールするには，59 ページ「Diagnostics サーバ（Commander モード）のインストール」に進みます。
- Diagnostic メディエータ・サーバをインストールするには，63 ページ「Diagnostics サーバ（Mediator モード）のインストール」に進みます。

Diagnostics サーバ（Commander モード）のインストール

Diagnostics サーバを Commander モードでインストールしている場合は，次の手順に進みます。

4 時間の同期方法を選択します。

診断データを正しく相関させるには，Diagnostics デプロイメントのすべてのコンポーネントが時刻と同期している必要があります。時間の同期方法を次の中から選択します。

- **NTP サーバと同期 :** このオプションは，Diagnostics サーバがファイアウォールの外の NTP Server にアクセスできる場合のみ適用されます。これはデフォルトの方法です。
- **登録されている Business Service Management サーバと同期 :** Diagnostics サーバを Business Service Management 環境で稼働させる場合，このオプションを選択して，Business Service Management サーバと同期させます。
- **システム時刻と同期 :** Diagnostics サーバを Business Service Management 以外の環境で運用し，NTP サーバにアクセスできない場合は，このオプションを選択します。

続行するには［**次へ**］を選択します。

5 Diagnostics サーバに関する任意指定の構成を選択します。

このサーバは HP Business Service Management（BSM）用です : Diagnostics コマンダ・サーバを Business Service Management と統合する場合は，このボックスをオンにします。

Business Service Management 9.00 以降と統合する場合，このオプションをオンにすると，追加の OM エージェント・コンポーネントと IAPA コンポーネントがインストールされて，Business Service Management に状況インジケータを送信できるようになります。IAPA は，Business Service Management と通信するために Diagnostics で使用する OMi エージェントの Integration Adapter Policy Activation コンポーネントです。

この Diagnostics コマンダ・サーバのインストールが終了するまでに，これらのコンポーネントをインストールするかどうかを確認するプロンプトが表示されます。

BSM との統合に必要なインストール後のその他の設定については，693 ページ「Business Service Management と Diagnostics 間の統合のセットアップ」を参照してください。また，OM サーバおよび BSM サーバへのレポートを設定する手順については，712 ページ「Diagnostics および OM サーバの共存」を参照してください。

この Diagnostics サーバに適用するオプションを選択し，［**次へ**］を選択して続行します。

HP Business Service Management オプションを選択した場合は追加のダイアログが表示され，**Diagnostics インストーラが格納されている，HP Diagnostics インストール DVD のディレクトリのパスを入力します**。

注 : この手順では，Diagnostics インストール・ディスクが必要です。

Business Service Management の Diagnostics の設定ページから Diagnostics Agent および Collector インストーラをダウンロードするには, それらのインストーラがある Diagnostics インストール・ディスクのディレクトリへのパスを指定する必要があります（**¥Diagnostics_Installers**）。

Diagnostics インストール・ディスクの Diagnostics インストーラへのパスを入力し，［**次へ**］を選択して続行します。

インストーラが Diagnostics サーバのインストール・ディレクトリに自動的にコピーされ，Business Service Management からアクセスできます。**¥Diagnostics_Installers** ディレクトリは約 1.85 GB あるため, コピーが終わるまでに数分かかる場合があります。

注 : この手順をスキップして，Diagnostics インストール・ディスクから直接 Agent および Collector インストーラにアクセスすることもできます。または，Agent および Collector インストーラを Diagnostics インストール・ディスク（**/Diagnostics_Installers**）から Business Service Management がアクセスできる Diagnostics サーバのインストール・ディレクトリ（**< Diagnostics サーバのインストール・ディレクトリ> /html/opal/downloads**）にコピーして，後でこの手順を手動で実行することも可能です。

---

6 電子メール警告用の SMTP 設定を指定します（任意）。

インストール時の以下の SMTP 設定は任意です。SMTP 設定は, Diagnostics サーバで問題がある場合に電子メール警告を送信するために設定します。この設定（または設定の変更）を後で行う場合は，このダイアログをスキップし，Diagnostics サーバの警告のプロパティ・ページを使用して設定を行います（詳細については,『HP Diagnostics ユーザー・ガイド』の警告に関するセクションを参照）。

- **SMTP サーバ :** SMTP サーバのホスト名または IP アドレス。
- **SMTP ポート :** SMTP サーバのポート番号。
- **送信元電子メール・アドレス :** 電子メール・メッセージ送信者の電子メール・アドレス。
- **管理警告電子メール・アドレス :** この Diagnostics サーバで問題がある場合に Diagnostics 管理者が電子メール警告を受け取るようにするには，管理者の電子メール・アドレスのカンマ区切りリストを指定します。サーバに対して大量のサーバ要求を生成するプローブやサーバのディスク容量に関する問題，またはコマンダ・サーバからのライセンス・チェックに対して警告を発行できます。

  Diagnostics 管理者のためのこれらの警告タイプを決定するしきい値は，出荷時にサーバの **server.properties** ファイルで設定されています。詳細については，このファイルの各種 **watchdog** プロパティのコメントを参照してください。840 ページ「サーバのディスク容量の問題」も参照してください。

7 インストール前にサマリ情報を確認します。

選択したインストール設定が表示されます。表示された情報が正確か確認します。

注 : Diagnostics サーバ（Commander モード）のインストールで使用する予想合計サイズには，Agent インストーラのサイズは含まれません。これは，Agent インストーラを Business Service Management で使用できる場合も同様です。

前のインストール手順に戻って設定を変更できます。Windows の場合は,［**戻る**］をクリックします。UNIX の場合は［**前へ**］を選択します。

Diagnostics サーバのインストールを開始するには,［**次へ**］を選択します。

8 インストールが開始します。

サーバのインストールが開始します。インストールが完了すると，インストール後のサマリ情報が表示されます。上記の手順 5a で Business Service Management との統合を選択した場合は，追加のダイアログが表示されます。

インストール後のサマリを確認し［**完了**］を選択してインストールを終了するか，BSM と統合する場合は次の手順に進みます。

9 OM エージェントおよび IAPA コンポーネントのインストール・チェックボックス

Business Service Management で Diagnostics コマンダ・サーバを使用するように選択した場合は，OM エージェントおよび IAPA コンポーネントのインストール用に追加のダイアログ・ボックスが表示されます。上記コンポーネントをインストールする場合は，チェックボックスをオンにします。チェックボックスをオフのまま残して，これらのコンポーネントを後から手動でインストールしてもかまいません。詳細については，72 ページ「OM エージェントおよび IAPA コンポーネントの手動インストール」を参照してください。

Diagnostics と Business Service Management 9.00 以降を統合する場合は，OM エージェントおよび IAPA コンポーネントを Diagnostics コマンダ・サーバにインストールする必要があります。これらのコンポーネントは Diagnostics で使用されて，Business Service Management ゲートウェイ・サーバに状況インジケータ・ステータス・イベントを送信します。以前のバージョンの Business Availability Center と統合する場合は，これらのコンポーネントをインストールする必要はありません。

＜Diagnostics のインストール・ディレクトリ＞ /server/log.txt ファイルにエラーが報告されます。インストール中に問題が発生した場合は，875 ページ「OM エージェントのトラブルシューティング」を参照してください。

ルート・アクセスは，OM エージェントと IAPA コンポーネントのインストールに必要です。ルート・アクセスなしで Diagnostics サーバをインストールする必要がある場合は，これら 2 つのコンポーネントをインストールしないで後から手動でインストールしてもかまいません。

OM エージェントがすでにシステムにインストールされている場合，OM エージェントのコンポーネントが古いバージョンであれば，インストーラによって更新されます。

Windows システムの場合，これらのコンポーネントはインストールに時間がかかります。

チェックボックスをオフのまま残して，OM エージェントおよび IAPA コンポーネントのインストールをスキップし，後からインストールすることができます。

チェックボックスをオンにするかどうかに関係なくインストーラの一部はいつでも配置されている状態になっているため，**＜Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-agent** および **/ovoiapa** から必要に応じて，後でコンポーネントをインストールできます。詳細については，72 ページ「OM エージェントおよび IAPA コンポーネントの手動インストール」を参照してください。

これらのコンポーネントをインストールした後，Business Service Management では追加の設定手順が必要です。詳細については，696 ページ「Business Service Management での Diagnostics サーバの登録」を参照してください。

インストールが完了したら，インストール後のサマリ情報でインストールが正常に完了したことを確認します。

インストールを終了するには，［**完了**］を選択してください。

注 : Windows マシンの場合，Diagnostics サーバは自動的に起動を試みます。その他のアプリケーションが標準設定の Diagnostics サーバ・ポートを使用している場合，Diagnostics サーバは起動しません。Diagnostics サーバを手動で起動する方法については，68 ページ「Diagnostics サーバの起動および停止」を参照してください。

Diagnostics サーバ（Mediator モード）のインストール

Diagnostic メディエータ・サーバをインストールしている場合は，次の手順に進みます。

Diagnostics サーバを Mediator モードでインストールしている場合は，次の手順に進みます。

4 Commander モードの Diagnostics サーバの場所を指定します。

Diagnostic メディエータ・サーバを Diagnostics コマンダ・サーバに接続できるようにするための情報を入力します。

➤ Diagnostics コマンダ・サーバ のホスト名または IP アドレスを入力します。

➤ Diagnostics コマンダ・サーバのポートを入力します。

Diagnostics コマンダ・サーバのデフォルトのポートは **2006** です。Diagnostics サーバをインストールした後にポートを変更した場合は，標準設定のポートではなく，変更後のポート番号を指定します。Diagnostics サーバのポートの変更については，433 ページ「デフォルトの Diagnostics サーバ・ポートの変更」を参照してください。

➤ インストーラで，指定したホストとポートへの接続性をチェックできるようにするには，［**Diagnostics サーバへの接続を確認する**］を選択します。

ここで接続性の問題をチェックしない場合は，［**Diagnostics サーバへの接続を確認する**］オプションをクリアします。

続行するには［**次へ**］を選択します。

インストーラに接続性のチェックを実行するように指定した場合，この時点で接続性のテストが実行されます。結果が好ましくない場合は，次のインストール手順に進む前に報告されます。

5 電子メール警告用の SMTP 設定を指定します（任意）。

インストール時の以下の SMTP 設定は任意です。SMTP 設定は，Diagnostics サーバで問題がある場合に電子メール警告を送信するために設定します。この設定（または設定の変更）を後で行う場合は，このダイアログをスキップし，Diagnostics サーバの警告のプロパティ・ページを使用して設定を行います（詳細については，『HP Diagnostics ユーザー・ガイド』の警告に関するセクションを参照）。

➤ **SMTP サーバ**：SMTP サーバのホスト名または IP アドレス。

➤ **SMTP ポート**：SMTP サーバのポート番号。

➤ **送信元電子メール・アドレス**：電子メール・メッセージ送信者の電子メール・アドレス。

➤ **管理警告電子メール・アドレス**：この Diagnostics サーバで問題がある場合に Diagnostics 管理者が電子メール警告を受け取るようにするには，管理者の電子メール・アドレスのカンマ区切りリストを指定します。サーバに対して大量のサーバ要求を生成するプローブやサーバのディスク容量に関する問題，またはコマンダ・サーバからのライセンス・チェックに対して警告を発行できます。

Diagnostics 管理者のためのこれらの警告タイプを決定するしきい値は，出荷時にサーバの **server.properties** ファイルで設定されています。詳細については，このファイルの各種 **watchdog** プロパティのコメントを参照してください。840 ページ「サーバのディスク容量の問題」も参照してください。

6 インストール前にサマリ情報を確認します。

選択したインストール設定が表示されます。表示された情報が正確か確認します。

前のインストール手順に戻って設定を変更できます。Windows の場合は，［**戻る**］をクリックします。UNIX の場合は［**前へ**］を選択します。

Diagnostics サーバのインストールを開始するには，［**次へ**］を選択します。

7 インストールが開始します。

インストールが完了したら，インストール後のサマリ情報でインストールが正常に完了したことを確認します。

インストールを終了するには，［**完了**］を選択してください。

注 : Windows マシンの場合，Diagnostics サーバは自動的に起動を試みます。その他のアプリケーションが標準設定の Diagnostics サーバ・ポートを使用している場合，Diagnostics サーバは起動しません。Diagnostics サーバを手動で起動する方法については，68 ページ「Diagnostics サーバの起動および停止」を参照してください。

Diagnostics サーバのインストールの確認

Diagnostics サーバが正しくインストールされたことを確認するには，**＜ Diagnostics サーバのインストール・ディレクトリ＞ /log/server.log** ファイルでエラーおよび警告を調べます。

また，Diagnostics Enterprise UI を起動して，サーバが実行していることも確認できます。**http:// ＜ Diagnostics コマンダ・サーバ＞ :2006/** に移動します。標準設定のユーザ / パスワード（**admin/admin**）を使用するか，異なるログイン情報が設定されている場合はそのログイン情報を使用できます。ただし，Diagnostics は企業内でデプロイされているため，admin および admin のデフォルト・ログインを変更する必要があります（ユーザ，ロール，権限および認証を設定する準備ができたら付録 B,「ユーザの認証と承認」を参照）。

エージェントからのデータを UI に表示するには，Diagnostics エージェントと Collector ソフトウェアをインストールおよび設定して，UI に表示するパフォーマンス・データを収集してサーバにレポートする必要もあります。

また，システムの状況ビューをチェックして，Diagnostics サーバおよびそれをホストするマシンに関する情報も確認できます。

システム・ビューにアクセスするには，次の手順を実行します。

1. http:// ＜ Diagnostics コマンド・サーバ名＞ :2006/query/ から Mercury System カスタマとして Diagnostics UI を開きます。
2. クエリ・ページで，リスト内から Mercury System カスタマを見つけて，Diagnostics を開くリンクを選択します。
3. Diagnostics にログインして，［アプリケーション］ウィンドウで［全組織］を選択し，Diagnostics ビューを開くためのリンクを選択します。
4. ［ビュー］表示枠にシステム・ビューのビュー・グループが表示されます。ビュー・グループを開き，システムの状況ビューまたはシステム・キャパシティ・ビューを選択します。

Diagnostics サーバのサイレント・インストール

サイレント・インストールは，ユーザが操作することなく自動的に実行されます。サイレント・インストールでは，ユーザ入力の代わりに，各インストール手順の応答ファイルからの入力を使用します。

たとえば，複数のマシンにコンポーネントをデプロイする必要のあるシステム管理者は，必要な設定情報がすべて含まれる応答ファイルを作成し，複数のマシン上でサイレント・インストールを実行します。これにより，インストール中に手動で入力する必要がなくなります。

複数のマシン上でサイレント・インストールを実行する前に，インストール中に入力を提供する応答ファイルを作成する必要があります。この応答ファイルは，インストール時に同じ入力を必要とするすべてのサイレント・インストールで使用できます。

重要 : Diagnostics の新しいリリースごとに, 複数のマシン上でサイレント・インストールを実行する前に，Diagnostics サーバのサイレント・インストール応答ファイルを再度記録する必要があります。

応答ファイルには拡張子 **.rsp** が付いています。標準的なテキスト・エディタで応答ファイルを編集できます。

応答ファイルを作成するには，次の手順を実行します。

➤ 次のコマンド・ライン・オプションを使って通常のインストールを実行します。

```
＜インストーラ＞ -options-record ＜ responseFileName ＞
```

これにより，インストール中に送信されたすべての情報が含まれる応答ファイルが作成されます。

サイレント・インストールを実行するには，次の手順を実行します。

➤ 関連する応答ファイルを使って，サイレント・インストールを実行します。

-silent コマンド・ライン・オプションを使って，次のようにサイレント・インストールを実行します。

```
＜インストーラ＞ -silent -options ＜ responseFileName ＞
```

サイレント・インストールを実行する場合は，応答ファイル名の後に次の 2 つの追加オプションを指定できます。

➤ **-is:log ＜ログ・ファイルのパス＞** オプションを指定して，ログ・ファイルを作成する

➤ **-is:tempdir ＜一時ディレクトリのパス＞** オプションを指定して，一時ディレクトリをユーザ指定ディレクトリに変更する

Diagnostics サーバの起動および停止

Diagnostics サーバは自動的に起動します。Diagnostics サーバを手動で起動または停止する必要がある場合は，次の指示に従います。

Windows マシンの場合

Windows マシンで Diagnostics サーバを起動するには，次の手順を実行します。

［**スタート**］>［**すべてのプログラム**］>［**HP Diagnostics Server**］>［**Start HP Diagnostics Server**］を選択します。

Windows マシンで Diagnostics サーバを停止するには，次の手順を実行します。

［**スタート**］>［**すべてのプログラム**］>［**HP Diagnostics Server**］>［**Stop HP Diagnostics Server**］を選択します。

Solaris または Linux マシンの場合（nanny を使用）

nanny は，デーモンとして実行して，Diagnostics サーバが常時作動するようにするプロセスです。nanny は，LoadRunner または Performance Center 用のオフライン･データ収集ができるように, LoadRunner エージェントも起動します。

次の手順では，nanny を使って Diagnostics サーバを起動および停止します。

m_daemon_setup スクリプトは，システムのブート後に自動的に再起動するようにサーバを設定しません。これをサポートするには，スタートアップをブート・シーケンスに統合するか，システムをブートするたびに手動で実行する必要があります。

Solaris または Linux マシンで Diagnostics サーバを起動するには，次の手順を実行します。

1 **M_LROOT** 環境変数が，Diagnostics サーバ nanny のルート・ディレクトリとして定義されていることを確認します。たとえば，*ksh* で次のように入力します。

```
export M_LROOT= ＜ Diagnostics サーバのインストール・ディレクトリ＞ /nanny/
＜プラットフォーム＞
```

この例では，＜プラットフォーム＞は solaris，linux，または hpux です。**M_LROOT** 環境変数がルート・ディレクトリとして定義されていない場合，次のエラーが発生します。

```
Warning : MDRV: cannot find lrun root directory . Please check your M_LROOT
Unable to format message id [-10791]
m_agent_daemon ( is down )
```

2 ディレクトリを **$M_LROOT/bin** に変更します。

3 次の例のように，**-install** オプションを使って **m_daemon_setup** を実行します。

```
cd $M_LROOT/bin
$./m_daemon_setup -install
```

Linux でエラーが発生した場合は，libstdc++.so.5 共有ライブラリのインストールが必要になる可能性があります。

UNIX または Linux マシンで Diagnostics サーバを停止するには，次の手順を実行します。

1 ディレクトリを **$M_LROOT/bin** に変更します。

2 次の例のように，**-remove** オプションを使って **m_daemon_setup** を実行します。

```
cd $M_LROOT/bin
$./m_daemon_setup -remove
```

Solaris または Linux マシンの場合（nanny を使用しない）

次の手順では，nanny を使わないで Diagnostics サーバを起動および停止します。

Solaris または Linux マシンで Diagnostics サーバを起動するには，次の手順を実行します。

＜ Diagnostics サーバのインストール・ディレクトリ＞ /bin/server.sh を実行します。

Solaris または Linux マシンで Diagnostics サーバを停止するには，次の手順を実行します。

kill などのユーティリティを使ってプロセスを終了します。

Diagnostics ソフトウェアのライセンス

Diagnostics ソフトウェアには簡易ライセンスが付属しており，すぐに使用を開始できます。ただし，Diagnostics コマンダ・サーバを使用して，最終的には永久ライセンス・キーをインストールする必要があります。ライセンス・ファイルの要求およびアップロードの方法については，第 3 章，「HP Diagnostics ライセンスの有効化」を参照してください。

Diagnostics サーバの設定の詳細

Diagnostics サーバは，すぐに動作を開始できる標準設定でインストールされています。

しかし，設定を変更することで Diagnostics サーバのパフォーマンスが向上したり，異常または複雑な状況で動作できるようになることがあります。Diagnostics サーバの詳細設定については，第 11 章，「Diagnostics サーバの詳細設定」を参照してください。

BSM との統合に必要なインストール後のその他の設定については，691 ページ「ほかの HP ソフトウェア製品との統合のセットアップ」を参照してください。また，OM サーバおよび BSM サーバへのレポートを設定する手順については，712 ページ「Diagnostics および OM サーバの共存」を参照してください。

Diagnostics サーバのバージョン確認

サポートを要求する場合は，Diagnostics サーバのバージョン情報が必要です。Diagnostics Enterprise UI の［バージョン情報］ダイアログ・ボックスに，Diagnostics サーバのバージョンが表示されます。［バージョン情報］ダイアログ・ボックスにアクセスするには，Diagnostics Enterprise UI ツールバーのヘルプ・メニューから［**HP Diagnostics のバージョン情報**］を選択します。

Diagnostics サーバのアンインストール

次の項では，Diagnostics サーバをアンインストールする方法について説明します。

重要 : OM エージェントがほかの製品で使用されている場合は, Diagnostics サーバでアンインストールされないことに注意してください。OM エージェントおよび IAPA コンポーネントのアンインストーラは Diagnostics サーバ・ディレクトリ内にあるため，これらをアンインストールする場合は，Diagnostics サーバより先にアンインストールする必要があります。

Windows マシンで Diagnostics サーバをアンインストールするには，次の手順を実行します。

1 ［**スタート**］>［**すべてのプログラム**］>［**HP Diagnostics Server**］>［**Uninstall HP Diagnostics Server**］を選択して，Diagnostics サーバをアンインストールします。

または，**< Diagnostics サーバのインストール・ディレクトリ> ¥_uninst** ディレクトリにある **uninstaller.exe** を実行することもできます。

2 アンインストール・プロセス中，特定のファイルを削除するかどうかを尋ねるメッセージが表示されます。次の操作を実行します。

- Diagnostics サーバをプロパティ設定も含めて完全にアンインストールするには，［**はい**］または［**すべてはい**］をクリックします。
- Diagnostics サーバを再インストールする予定があり，アンインストールしている Diagnostics サーバのカスタム・プロパティ設定を保持する場合は，**etc** ディレクトリにあるプロパティ・ファイルを新しい場所にバックアップする必要があります。

  これらのファイルをバックアップした場合は，［**はい**］または［**すべてはい**］をクリックします。

  これらのファイルをバックアップしていない場合は，［**いいえ**］または［**すべていいえ**］を選択します。

UNIX マシンで Diagnostics サーバをアンインストールするには，次の手順を実行します。

コンソール・モードまたはグラフィック・モードで Diagnostics サーバをアンインストールできます。

1 Diagnostics サーバを停止します。詳細については，68 ページ「Diagnostics サーバの起動および停止」を参照してください。

2 ディレクトリをルート・ディレクトリに変更します。

3 UNIX コマンド・プロンプトで次のように入力します。

➤ **コンソール・モード：**

```
＜Diagnostics サーバのインストール・ディレクトリ＞ /Server/_uninst/uninstaller.bin
-console
```

➤ **グラフィック・モード**

グラフィック・モードで実行する前に，画面表示をエクスポートします。

```
export DISPLAY= ＜ホスト名＞ .0.0
```

```
＜Diagnostics サーバのインストール・ディレクトリ＞ /Server/_uninst/uninstaller.bin
```

OM エージェントおよび IAPA コンポーネントの手動インストール

Diagnostics コマンダ･サーバのインストーラには，Business Service Management 9.00 以降に状況インジケータ・ステータスのイベント送信に使用される，OM エージェントと IAPA コンポーネントのインストールが含まれています（以前のバージョンの Business Availability Center と統合する場合は，これらのコンポーネントをインストールする必要はありません）。

Windows での OM エージェントおよび IAPA コンポーネントのインストールには時間がかかることがあります。Diagnostics サーバのインストール中にこれらのコンポーネントのインストールをスキップして，後で Diagnostics コマンダ・サーバに手動でインストールするように選択できます（次を参照）。

インストール中に問題が発生した場合は，875 ページ「OM エージェントのトラブルシューティング」を参照してください。

注 : Diagnostics サーバがコマンダ・サーバか Mediating Server かに関係なく，Diagnostics サーバのインストール中に OM エージェント・インストーラおよび IAPA インストーラの一部が配置されます。これらのコンポーネントを後でインストールするように選択した場合も同様です。

Windows システムに OM エージェントを手動でインストールするには，次の手順を実行します。

1 Windows システムの場合は，ディレクトリを**＜ Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-agent/ ＜プラットフォーム＞**に変更します。＜プラットフォーム＞は **win32** または **win64**. のいずれかです。

2 このディレクトリで，コマンド・ラインから次のコマンドを実行します。

cscript.exe opc_inst.vbs

Linux または Solaris システムに OM エージェントを手動でインストールするには，次の手順を実行します。

1 Linux または Solaris システムの場合は，ディレクトリを**＜ Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-agent/ ＜プラットフォーム＞** に変更します。＜プラットフォーム＞は **Linux32** や **Linux64**, または **solaris** です。

2 このディレクトリで，コマンド・ラインからルート・ユーザとして次のコマンドを実行します。

./opc_inst

Windows システムに IAPA コンポーネントを手動でインストールするには，次の手順を実行します。

1 Windows システムの場合は，ディレクトリを**＜ Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-iapa/ ＜プラットフォーム＞**に変更します。＜プラットフォーム＞は **win32** または **win64** のいずれかです。

2 win32 ディレクトリで，コマンド・ラインから次のコマンドを実行します。

cscript.exe ＜インストール・ディレクトリ＞ /server/bin/install_ovo_iapa.vbs /i HPOprIAPA-09.00.111-WinNT4.0-release.msi ＜ログ・ファイル＞

ここで＜ログ・ファイル＞は，インストール結果が記録されるファイルです。パスは任意です。

または，win64 ディレクトリで，コマンド・ラインから次のコマンドを実行します。

cscript.exe ＜インストール・ディレクトリ＞ /server/bin/install_ovo_iapa.vbs /i HPOprIAPA-09.00.111-Win5.2_64-release.msi ＜ログ・ファイル＞

ここで＜ログ・ファイル＞は，インストール結果が記録されるファイルです。パスは任意です。

Linux または Solaris システムに IAPA コンポーネントを手動でインストールするには，次の手順を実行します。

1 Linux または Solaris システムの場合は，ディレクトリを**＜ Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-iapa/ ＜プラットフォーム＞**に変更します。＜プラットフォーム＞は **Linux32** や **Linux64**，または **solaris** です。

2 Linux32 ディレクトリで，コマンド・ラインからルート・ユーザとして次のコマンドを実行します。

rpm -ivh HPOprIAPA-09.00.111-Linux2.6-release.rpm

または，Linux32 ディレクトリで，コマンド・ラインからルート・ユーザとして次のコマンドを実行します。

rpm -ivh HPOprIAPA-09.00.111-Linux2.6_64-release.rpm

または，solaris ディレクトリで，コマンド・ラインからルート・ユーザとして次のコマンドを実行します。

pkgadd -a ./noask_pkgadd -d HPOprIAPA-09.00.111-SunOS5.10-release.sparc HPOprIAPA

OM エージェントのインストールを完了するには，次の手順を実行します。

➤ OM エージェントの設定を完了するには，Diagnostics に Business Service Management を登録する手順を完了する必要があります。OM エージェントに関する詳細については，696 ページ「Business Service Management での Diagnostics サーバの登録」を参照してください。

OM エージェントおよび IAPA コンポーネントの手動アンインストール

Diagnostics サーバをアンインストールしても，OM エージェントおよび IAPA コンポーネントはアンインストールされません。OM エージェントおよび IAPA コンポーネントのアンインストーラは Diagnostics サーバ・ディレクトリ内にあるため，これらをアンインストールする場合は，Diagnostics サーバより先にアンインストールする必要があります。また，これらのコンポーネントは，最小に IAPA コンポーネント，次に OM エージェントという順番でアンインストールする必要があります。

Windows システムから IAPA コンポーネントを手動でアンインストールするには，次の手順を実行します。

1 Windows システムの場合は，ディレクトリを**＜ Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-iapa/ ＜プラットフォーム＞**に変更します。＜プラットフォーム＞は **win32** または **win64** のいずれかです。

2 win32 ディレクトリで，コマンド・ラインから次のコマンドを実行します。

cscript.exe ＜インストール・ディレクトリ＞ ¥server¥bin¥install_ovo_iapa.vbs /x HPOprIAPA-09.00.111-WinNT4.0-release.msi uninstall.log

または，win64 ディレクトリで，コマンド・ラインから次のコマンドを実行します。

cscript.exe ＜インストール・ディレクトリ＞ ¥server¥bin¥install_ovo_iapa.vbs /x HPOprIAPA-09.00.111-Win5.2_64-release.msi uninstall.log

Linux または Solaris システムから IAPA コンポーネントを手動でアンインストールするには，次の手順を実行します。

1 Linux または Solaris システムの場合は，ディレクトリを**＜ Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-iapa/ ＜プラットフォーム＞**に変更します。＜プラットフォーム＞は **Linux32** や **Linux64**，または **solaris** です。

2 Linux32 または Linux64 ディレクトリで，コマンド・ラインからルート・ユーザとして次のコマンドを実行します。

rpm -e HPOprIAPA

または，solaris ディレクトリで，コマンド・ラインからルート・ユーザとして次のコマンドを実行します。

pkgadd HPOprIAPA

Windows システムから OM エージェントを手動でアンインストールするには，次の手順を実行します。

1 Windows システムの場合は，ディレクトリを**＜ Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-agent/ ＜プラットフォーム＞**に変更します。＜プラットフォーム＞は **win32** または **win64**. のいずれかです。

2 このディレクトリで，コマンド・ラインから次のコマンドを実行します。

cscript.exe opc_inst.vbs -r

Linux または Solaris システムから OM エージェントを手動でアンインストールするには，次の手順を実行します。

1 Linux または Solaris システムの場合は，ディレクトリを**＜ Diagnostics のインストール・ディレクトリ＞ /server/setup/ovo-agent/ ＜プラットフォーム＞** に変更します。＜プラットフォーム＞は **Linux32** や **Linux64**, または **solaris** です。

2 このディレクトリで，コマンド・ラインからルート・ユーザとして次のコマンドを実行します。

./opc_inst -r

3

HP Diagnostics ライセンスの有効化

HP Diagnostics では，Diagnostics コマンド・サーバに有効なライセンスをアップロードする必要があります。

本章の内容

HP Diagnostics ライセンスの有効化

Diagnostics は，Diagnostics コマンド・サーバにアップロードしたファイルを使ってライセンスが付与されます。このライセンス・ファイルは，HP ソフトウェアのカスタマ・サポート担当者に要求します。

Java Agent と .NET Agent および Diagnostic メディエータ・サーバが Diagnostics コマンド・サーバに初めて接続すると，Diagnostics コマンド・サーバにインストールされているライセンスに基づいてライセンスが与えられます。

Diagnostics 管理者がライセンス・チェック警告を受け取るようにするには，コマンダ・サーバのインストール時に SMTP 設定インストール・ダイアログで管理警告電子メール・アドレスのカンマ区切りリストを指定します。またはコマンダ・サーバの［警告プロパティ］ページを使用して，インストール後に管理者のアドレスを設定することもできます。

ライセンスの種類

インストール時には，製品に付属している**簡易ライセンス**が付与されます。この簡易ライセンスを使って，Diagnostics コンポーネントをインストールし，アプリケーションの監視とパフォーマンス・メトリックスの処理を開始できます。簡易ライセンスは，インストールまたは製品の最初の使用時点から一定期間有効です。

この期間内に，**永久ライセンス**を入手するか，評価ライセンスを要求して評価期間を延長する必要があります。評価ライセンスは，顧客による製品の評価を延長する手段としてライセンス・キーを提供するために Diagnostics で使用できます。評価ライセンスは，一定期間有効です。

永久ライセンスを取得する前に簡易ライセンス（または延長された評価ライセンス）の有効期限が切れる場合は，Diagnostics サーバからリマインダ・メッセージが発行されます。

注 : 完全な Diagnostics 製品には，簡易ライセンスが付属しています。スタンドアロンの Diagnostics Profiler は，有効なライセンス・ファイルが提供されるまで，読み込みが制限されます。

永久ライセンスでは通常，キャパシティが限定されています（83 ページ「現在接続されているプローブに基づくライセンス情報」を参照）。ライセンス・キーをインストールすると，このキャパシティに対して Diagnostics で使用数がカウントされます。

Diagnostics には 2 種類の LTU（使用ライセンス）があります。

- AM ライセンス - 製品を Application Management/Enterprise モードで使用する場合，実運用環境で使用するのが一般的です。LoadRunner / Performance Center では AM ライセンスが付与されたエージェントも使用できます。
- AD ライセンス - 製品を LoadRunner / Performance Center の Diagnostics モードで使用する場合，実運用前の負荷テスト環境で実行します。

  Diagnostics で受け取る簡易ライセンスには，期間およびキャパシティに関して次のような制限があります。AM - 60 日間，キャパシティ 50。AD - 14 日間，キャパシティ 50。

  この制限を超えると，リマインダ・メッセージが表示されます。AM ライセンスと AD ライセンスの詳細については，83 ページ「現在接続されているプローブに基づくライセンス情報」を参照してください。

Diagnostics サーバ（Commander モード）のライセンスの有効化

Diagnostics のライセンスは，HP ソフトウェアのカスタマ・サポート担当者から入手します。以下で説明している［ライセンス管理］ページには，各システムから手動で情報を取得しなくても，必要なライセンス数を判断できる有益な情報が含まれています。この情報は，Diagnostics 8.00 以降のプローブでのみ有効です。

ライセンス購入の条件を確認するライセンス証明書を HP から受け取ります。ライセンス・キーとパスワードは，ソフトウェア製品の購入に関連する販売注文番号の入力後に発行されます。この番号は注文ごとに一意です。この番号は，ライセンス引き換えフォーム，および注文の出荷と梱包に関連するすべての書類に表示されます。

Diagnostics コマンド・サーバの［ライセンス管理］からアクセス可能なディレクトリにライセンス・ファイルを保存します。次に，以下の手順に従ってライセンス・ファイルを Diagnostics コマンダ・サーバにアップロードします。

重要 : Diagnostics 9.10 より前のバージョンのライセンスを持つ顧客の場合, 9.10 でも古いライセンスは動作しますが，次の項では Diagnostics 9.10 以降を新規購入して新しいライセンスを有効化するプロセスの使用方法について説明しています。

Diagnostics デプロイメントにライセンスを与えるには

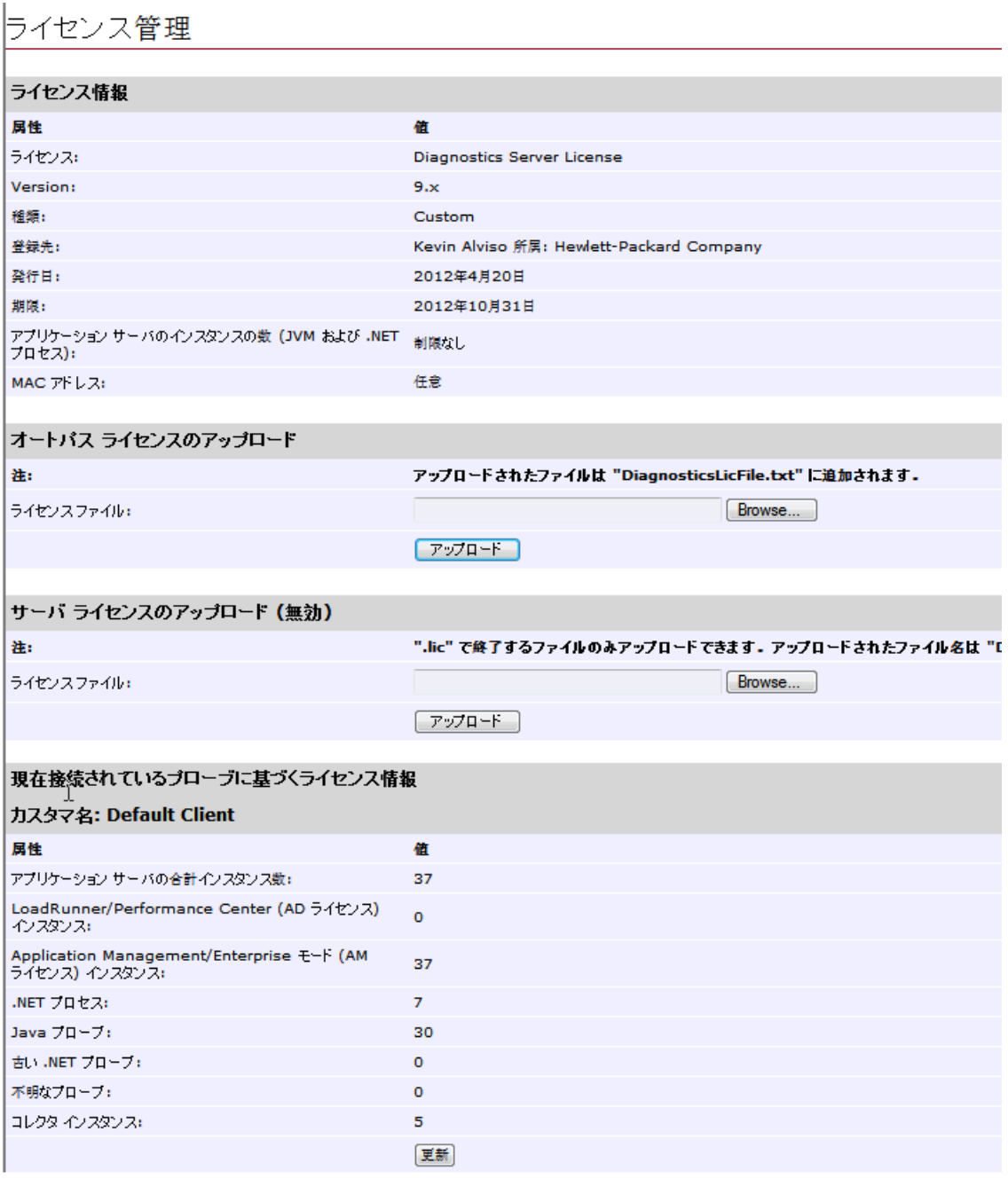
1 Diagnostics Enterprise UI（http:// < Diagnostics サーバ> :2006）にアクセスして，Diagnostics コマンド・サーバの［ライセンス管理］ページにアクセスします。

2 ログイン名とパスワードを入力します。標準設定を使用するか，作成および割り当て済みの設定を使用します。標準設定のログイン名およびパスワードはどちらも admin です。

3 ［**Diagnostics の設定**］を選択します。

4 ［**license**］リンクを選択します。［ライセンス管理］ページが開き，次の情報が表示されます。

- 現在のライセンスに関する情報
- HP ソフトウェア・サポートから受け取ったライセンスをアップロードするためのユーティリティ
- 監視対象環境内のアプリケーション・サーバ・インスタンスの総数や，現在接続されているプローブの数に関する情報。Diagnostics の AD および AM ライセンスのキャパシティに対する使用数に関する情報も確認できます。



ライセンス管理

ライセンス情報

属性	値
ライセンス:	Diagnostics Server License
Version:	9.x
種類:	Custom
登録先:	Kevin Alviso 所属: Hewlett-Packard Company
発行日:	2012年4月20日
期限:	2012年10月31日
アプリケーション サーバのインスタンスの数 (JVM および .NET プロセス):	制限なし
MAC アドレス:	任意

オートパス ライセンスのアップロード

注:	**アップロードされたファイルは "DiagnosticsLicFile.txt" に追加されます。**
ライセンスファイル:	Browse...
	アップロード

サーバ ライセンスのアップロード (無効)

注:	**".lic" で終了するファイルのみアップロードできます。アップロードされたファイル名は "D**
ライセンスファイル:	Browse...
	アップロード

現在接続されているプローブに基づくライセンス情報

カスタマ名: Default Client

属性	値
アプリケーション サーバの合計インスタンス数:	37
LoadRunner/Performance Center (AD ライセンス) インスタンス:	0
Application Management/Enterprise モード (AM ライセンス) インスタンス:	37
.NET プロセス:	7
Java プローブ:	30
古い .NET プローブ:	0
不明なプローブ:	0
コレクタ インスタンス:	5
	更新

5 Diagnostics デプロイメント用のライセンス・ファイルを受け取ったら，［ライセンス管理］ページの［**オートパス ライセンスのアップロード**］セクションを使用して，そのファイルをアップロードします。

［**サーバ ライセンスのアップロード（無効）**］セクションは廃止されており，Diagnostics で以前に使用されていたタイプのライセンス・キー（.lic ファイル）または簡易ライセンスのみがサーバにインストールされている場合にのみ表示されます。このアップロードは，9.10 より前のバージョンの Diagnostics からライセンスを取得済みの既存の顧客が古いライセンスをアップロードできるようにするためのものです。

注 : ライセンス・ファイルを Diagnostics サーバ・インストール・ディレクトリに直接コピーしないでください。ファイルをアップロードする際は，［ライセンス管理］ページの［オートパス ライセンスのアップロード］セクションを使用します。

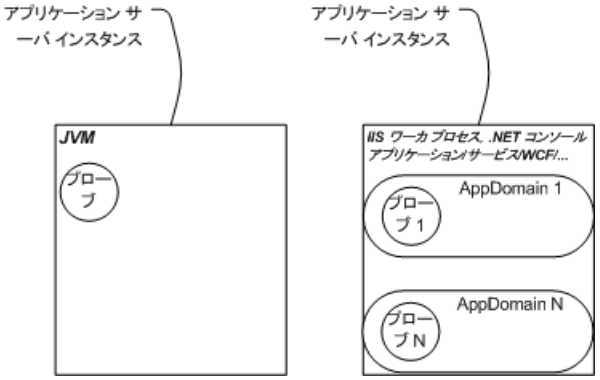
ライセンス・ファイルの格納場所へのパスを入力するか，［**Browse**］をクリックして，ライセンス・ファイルの場所に移動します。［**アップロード**］をクリックして，Diagnostics サーバにライセンス・ファイルを適用します。

これに成功すると（ライセンス・ファイルのキーが有効であり期限切れでない），アップロード・プロセスによりライセンスが **DiagnosticsLicFile.txt** に追加され，Diagnostics コマンダ・サーバの**＜ Diagnostics のインストール・ディレクトリ＞ /etc** ディレクトリに格納されます。オートパス ライセンスでは，ライセンス・ファイルに追加されるインクリメンタル・ライセンスをアップロードできます（古いライセンスが混在している場合は，この操作ができません）。

ライセンス情報の表示

現在のライセンスに関する情報が［ライセンス管理］ページに表示されます。ライセンスの種類，有効期限（ある場合），およびライセンスで扱われるアプリケーション・サーバ（JVM および .NET プロセス）やコレクタ・インスタンスの最大数が表示されます。

現在接続されているプローブに基づくライセンス情報も表示できます。この情報には，現在 Diagnostics で監視されているアプリケーション・サーバ・インスタンス，.NET プロセス，JVM インスタンス，およびコレクタ・インスタンスの個数が含まれます。



現在接続されているプローブに基づくライセンス情報

ライセンス情報セクションには，現在接続されているプローブに基づく件数が表示されます（以下の例を参照）。これを使用すれば，各システムから手動で情報を取得しなくても，必要なライセンス数を判断できます。この情報は，Diagnostics 8.00 以降のプローブでのみ有効です。

現在接続されているプローブに基づくライセンス情報

カスタマ名: Default Client

属性	値
アプリケーション サーバの合計インスタンス数:	36
LoadRunner/Performance Center (AD ライセンス) インスタンス:	0
Application Management/Enterprise モード (AM ライセンス) インスタンス:	36
.NET プロセス:	6
Java プローブ:	30
古い .NET プローブ:	0
不明なプローブ:	0
コレクタ インスタンス:	5
	更新
	詳細

➤ **アプリケーション・サーバの合計インスタンス数** : アプリケーション・サーバ・インスタンスは, Java Agent インスタンスまたは .NET Agent インスタンスです。Java Agent インスタンスは「プローブ」であり，.NET Agent インスタンスは .NET ワーカ・プロセスです。この値は,「LoadRunner/Performance Ceter インスタンス」と「Application Management/Enterprise モード・インスタンス」の合計です。ライセンス・キャパシティは，この合計値に対応できる必要があります。

LoadRunner/Performance Ceter（AD ライセンス）インスタンス : AD モードで設定され LoadRunner または Performance Center の実行にアクティブに関係する Java プローブと .NET プロセスのインスタンス数。HP Diagnostics の AD ライセンス・キャパシティに対して，AD モードのアクティブなプローブまたはプロセスのみがカウントされます。実行中でないものはカウントされません。

Agent のインストール時には, LoadRunner および Performance Center を実行する AD モードで Agent を設定するかどうかを指定するように求められます。AD ライセンス・オプションを選択すると, Diagnostics で次の値が設定されます。

Java Agent の場合 - **etc/probe.properties** ファイルの **active.products** プロパティの値が, Java Agent のインストール時に **AD** モードに設定されます（461 ページ「アクティブ・プロダクト・モードの設定」を参照）。このプロパティを変更して，インストール後にモードの値を変更できます。

.NET Agent の場合 - **probe_config.xml ＜モード＞**要素の値が，.NET Agent のインストール時に **ad** モードに設定されます（548 ページ「<modes> 要素」を参照）。この要素を変更して，インストール後にモードの値を変更できます。

プローブを AD モードで実行すると，LoadRunner または Performance Center のテスト実行に現在含まれているプローブ数に対するライセンス・キャパシティで十分であるという利点があります。したがって，たとえばテスト・システムに 20 個の Agent がインストールされていてもテスト実行で一度に必要とするプローブが 5 個しかない場合は，5 個のプローブに対する AD ライセンス・キャパシティしか必要ありません。

➤ **Application Management/Enterprise モード（AM ライセンス）インスタンス** : 実運用環境でインストールされ Enterprise モードまたは AM モードに設定された Java Agent と .NET プロセスのインスタンス数。HP Diagnostics の AM ライセンス・キャパシティに対してカウントされます。

Agent のインストール時には，実運用環境の Diagnostics サーバで動作するように Agent を Application Management/Enterprise モード（AM ライセンス）で設定するかどうかを指定するように求められます。このモードを選択すると，Diagnostics で次の値が設定されます。

Java Agent の場合 - **etc/probe.properties** ファイルの **active.products** プロパティの値が，Java Agent のインストール時に **Enterprise** モードに設定されます（461 ページ「アクティブ・プロダクト・モードの設定」を参照）。このプロパティを変更して，インストール後にモードの値を変更できます。

.NET Agent の場合 - **probe_config.xml ＜モード＞**要素の値が，.NET Agent のインストール時に **enterprise** モードに設定されます（548 ページ「<modes> 要素」を参照）。この要素を変更して, インストール後にモードの値を変更できます。

Enterprise モードが設定された Agent の場合, HP Diagnostics の AM ライセンス・キャパシティに対して，プローブがカウントされます。

- **.NET プロセス**：1 つ以上の .NET プローブで監視するためにインストルメントされる任意のプロセス（アプリケーション・ドメイン）。たとえば，IIS ワーカ・プロセスまたは .NET コンソール・アプリケーション / サービス / WCF などです。ライセンス・レポートには，バージョンが 8.00 より前の古い .NET プローブの個数が表示されることがあります。
- **Java プローブ**：監視対象の Java や javaw プロセス，または JVM に埋め込まれたその他のプロセス。これは，Java プローブに相当します。
- **コレクタ・インスタンス**：Collector インスタンス数はライセンス・ページに表示されますが，ライセンスに対してカウントされません。Collector インスタンスに含まれる内容は，次のとおりです。
  - Oracle -（実行された）Oracle ソフトウェア内のインスタンス（Oracle プロセス）および使用メモリ（SGA）。インスタンスは SID で識別されます。**oracle-config.xml** 内の＜ oracleInstance ＞エントリで監視するように設定されているインスタンスが含まれます。
  - SQL Server - インスタンスは主にデータベース・エンジンおよびサポート対象コンポーネントに適用されます。**sqlserver-config.xml** 内の＜sqlserverInstance＞エントリで監視するように設定されているインスタンスが含まれます。
  - WebSphere MQ - **mq-config.xml** 内の＜ mqInstance ＞エントリで監視するように設定されているインスタンスが含まれます。
  - TIBCO EMS - **tibco-ems-config.xml** 内の＜ emsInstance ＞エントリで監視するように設定されているインスタンスが含まれます。
  - WebMethods Broker - **wm-broker-config.xml** 内の＜ WmBrokerInstace ＞エントリで監視するように設定されているインスタンスが含まれます。

- SAP/ABAP - 検出された各ダイアログ・インスタンス（SAP ABAP プローブ）が含まれます。
- VMware - **vmware-config.xml** ファイルで指定された vSphere サーバの数が含まれます。

- 8.0x より前のプローブは，古いプローブに表示されます。

ライセンスの詳細

ライセンス・ページの下部にある［詳細］リンクを選択すると，.NET プローブ，Java プローブ，Collector の各詳細が表示されます。詳細には，プローブ名，ホスト名とポートまたは PID，実行 ID（LoadRunner/Performance Center の負荷テストの実行用），プローブのバージョン，および製品モードが含まれます。

次に［ライセンス管理］の詳細ページを示します。

Diagnostics

イセンス管理

に対する詳細: Default Client

ET Probes

ET processes (hostname:pid)	Probe Name	Mode	Run ID	Version
ntt153.ovrtest.adapps.hp.com:2452	L81_1ROOTMSPetShop4.NET_OVRNTT153_W2k3	Enterprise,PRO		9.20.104.44902
svm46.ovrtest.adapps.hp.com:3896	bsavm46_1ROOTTrade.NET	Enterprise,PRO		9.20.104.44902
svm46.ovrtest.adapps.hp.com:4200	bsavm46_1ROOTBasicWCF_WebClient.NET	Enterprise,PRO		9.20.104.44902
esx1- 10.ovrtest.adapps.hp.com:576	L91_1ROOTTestService2.WebService.NET_OVRESX1-VM10_W2k3x64	Enterprise,PRO		9.20.104.44902
esx1- 10.ovrtest.adapps.hp.com:576	L91_1ROOTJavaTrader2.WebClient.NET_OVRESX1-VM10_W2k3x64	Enterprise,PRO		9.20.104.44902
esx1- 10.ovrtest.adapps.hp.com:576	L91_1ROOTCallChain2_0.NET_OVRESX1-VM10_W2k3x64	Enterprise,PRO		9.20.104.44902
esx1- 10.ovrtest.adapps.hp.com:576	L91_1ROOTTestService2.WebServiceChain.NET_OVRESX1-VM10_W2k3x64	Enterprise,PRO		9.20.104.44902
esx1- 10.ovrtest.adapps.hp.com:576	L91_1RootMSPetShop4.NET_OVRESX1-VM10_W2k3x64	Enterprise,PRO		9.20.104.44902
esx1- 10.ovrtest.adapps.hp.com:576	L91_1ROOTTestService2.WebClient.NET_OVRESX1-VM10_W2k3x64	Enterprise,PRO		9.20.104.44902
RNTT209.ovrtest.adapps.hp.com:1244	L81_1ROOTTestService2.WebService_DefaultWebSite.NET_OVRNTT209_W2k3	Enterprise,PRO		9.20.104.44902
RNTT209.ovrtest.adapps.hp.com:1244	L81_1ROOTJavaTrader2.WebClient_DefaultWebSite.NET_OVRNTT209_W2k3	Enterprise,PRO		9.20.104.44902
RNTT209.ovrtest.adapps.hp.com:1244	L81_1ROOTCallChain2_0_DefaultWebSite.NET_OVRNTT209_W2k3	Enterprise,PRO		9.20.104.44902
RNTT209.ovrtest.adapps.hp.com:1244	L81_1ROOTTestService2.WebClient_DefaultWebSite.NET_OVRNTT209_W2k3	Enterprise,PRO		9.20.104.44902
RNTT209.ovrtest.adapps.hp.com:1244	L81_1RootTestService2.WebServiceChain_DefaultWebSite.NET_OVRNTT209_W2k3	Enterprise,PRO		9.20.104.44902
ntt153.ovrtest.adapps.hp.com:548	L81_1ROOTJavaTrader2.WebClient.NET_OVRNTT153_W2k3	Enterprise,PRO		9.20.104.44902
ntt153.ovrtest.adapps.hp.com:548	L81_1ROOTTestService2.WebClient.NET_OVRNTT153_W2k3	Enterprise,PRO		9.20.104.44902
ntt153.ovrtest.adapps.hp.com:548	L81_1ROOTTestService2.WebServiceChain.NET_OVRNTT153_W2k3	Enterprise,PRO		9.20.104.44902
ntt153.ovrtest.adapps.hp.com:548	L81_1ROOTCallChain2_0.NET_OVRNTT153_W2k3	Enterprise,PRO		9.20.104.44902
ntt153.ovrtest.adapps.hp.com:548	L81_1ROOTTestService2.WebService.NET_OVRNTT153_W2k3	Enterprise,PRO		9.20.104.44902

va Probes:

obe Name	Mode	Run ID	HostName:Port	Version
gInx01_WAS80	Enterprise, PRO		diaglnx01.hpswlabs.adapps.hp.com:35000	9.20.60.1257
tClinic_hpsw-vm139_W2k8	Enterprise, PRO		hpsw-vm139.ovrtest.adapps.hp.com:35000	9.20.58.1236
avm52_BSMServerMon_marble1	Enterprise, PRO		bsavm52.ovrtest.adapps.hp.com:35000	9.20.41.776
esx1-vm15_JavaREST_BankService	Enterprise, PRO		ovresx1-vm15.ovrtest.adapps.hp.com:35000	9.20.104.1323
gwin01_WAS80	Enterprise, PRO		diagwin01.ovrtest.adapps.hp.com:35000	9.20.60.1257
iseNativeAmkibld05	Enterprise, PRO		amkibld05.rose.hp.com:35000	9.00.77.1123
avm51_BSMServerMon_mercuryAS	Enterprise, PRO		bsavm51.ovrtest.adapps.hp.com:35001	9.20.44.864
avm52_BSMServerMon_offline_eng	Enterprise, PRO		bsavm52.ovrtest.adapps.hp.com:35002	9.20.41.776
iseNativeAmkibld06	Enterprise, PRO		amkibld06.rose.hp.com:35000	9.00.76.1115
iseNativeHpswbld02	Enterprise, PRO		hpswbld02.rose.hp.com:35000	9.00.77.1123
msmoke1_mercuryAS	Enterprise, PRO		bsmsmoke1.ovrtest.adapps.hp.com:35000	9.20.58.1236
iseNativeAmkibld07	Enterprise, PRO		amkibld07.rose.hp.com:35000	9.00.77.1123
S61_TV75_AMKILAB02_W2k3	Enterprise, PRO, TV		amkilab02.ovrtest.adapps.hp.com:35000	9.20.104.1323
avm52_BSMServerMon_marble_matcher	Enterprise, PRO		bsavm52.ovrtest.adapps.hp.com:35001	9.20.41.776
avm51_BSMServerMon_odb	Enterprise, PRO		bsavm51.ovrtest.adapps.hp.com:35000	9.20.44.864
1_Bac8_0_J2EE_ovrntt158_W2k3	Enterprise, PRO		ovrntt158.ovrtest.adapps.hp.com:35000	9.20.104.1323
M920Engine_hpsw-vm129_W2k3r2x64	Enterprise, PRO		hpsw-vm129.ovrtest.adapps.hp.com:35000	9.20.58.1236
msmoke1_wde	Enterprise, PRO		bsmsmoke1.ovrtest.adapps.hp.com:35002	9.20.58.1236
iseNativeOvrbld01	Enterprise, PRO		ovrbld01.rose.hp.com:35000	9.00.77.1123
avm46_websphere_plants	Enterprise, PRO		bsavm46.ovrtest.adapps.hp.com:35000	9.20.104.1323
S6_Plants_ovrntt105_W2k3_probe8_x	Enterprise, PRO		ovrntt105.ovrtest.adapps.hp.com:35000	8.07.03.247
iseNativeOvresx4-vm31	Enterprise, PRO		ovresx4-vm31.rose.hp.com:35000	9.00.77.1123
avm52_BSMServerMon_OPR	Enterprise, PRO		bsavm52.ovrtest.adapps.hp.com:35003	9.20.41.776
avm51_BSMServerMon_wde	Enterprise, PRO		bsavm51.ovrtest.adapps.hp.com:35002	9.20.44.864
msmoke1_odb	Enterprise, PRO		bsmsmoke1.ovrtest.adapps.hp.com:35001	9.20.58.1236
iseMasterAmkibld03	Enterprise, PRO		amkibld03.rose.hp.com:35000	9.00.77.1123
i_hpswos008	Enterprise, PRO		hpswos008.ovrtest.adapps.hp.com:35000	9.20.60.1257
iseNativeGerstner	Enterprise, PRO		gerstner.rose.hp.com:35000	9.00.77.1123
iseNativeOvruxt59	Enterprise, PRO		ovruxt59.rose.hp.com:35000	9.00.77.1123
iseNativeRainier	Enterprise, PRO		rainier.rose.hp.com:35000	9.00.77.1123
i_rosbtmss1120poc	Enterprise, PRO		rosbtmss1120poc.ovrtest.adapps.hp.com:35000	9.20.57.1197

.NET probes not reporting their PID:

obe Name	Mode	Run ID	CLR Info	Version

known Probes (no PID or VM info missing):

obe Name	Mode	Run ID	VM Info	Version

llectors:

me	Instances	Mode	Run	Version
llector_ovrntt152_W2k3	5	Enterprise		9.20.104.1323

Diagnostic サーバ "server-ovrntt150", バージョン 9.20.104.1323

他の Diagnostics コンポーネントのライセンスの有効化

Mediator として稼働している Diagnostics サーバ，および Diagnostics エージェントには，独立したライセンスがありません。それらのライセンスは，Diagnostics コマンド・サーバのライセンスに基づきます。これらのコンポーネントがライセンス付きの Diagnostics コマンド・サーバに初めて接続すると，Diagnostics エージェントおよび Diagnostic メディエータ・サーバは自動的にライセンスが設定されます。

Java または .NET Agent をインストールするときに，Diagnostics Profiler も自動的にインストールされます。プローブ・エンティティのコンテキストでは，Diagnostics Profiler が表示されます。Diagnostics Profiler は独立した UI です。Agent がインストールされているシステムから直接アクセスしたり，HP Diagnostics UI を通してアクセスしたりできます。

Diagnostics Profiler は, 正しくライセンスが与えられた Diagnostics コマンド・サーバにプローブが接続可能になるまで，読み込み制限付きで非ライセンス・モードで作動します。非ライセンス・モードの場合，Profiler がデータをキャプチャできる同時スレッド数は 5 つに制限されます。

4

Diagnostics Collector のインストール

Diagnostics Collector は Windows および UNIX マシンにインストールできます。

本章の内容

➤ Diagnostics Collector のバージョンの確認（128 ページ）

➤ Diagnostics Collector のアンインストール（128 ページ）

Diagnostics Collector のインストールについて

Diagnostics Collector はリモート・システムからデータを収集します。次のタイプのアクティブ・システムからパフォーマンス・データを収集するように Collector を設定できます。

➤ SAP NetWeaver–ABAP

➤ Oracle データベース（Oracle RAC を含む）

➤ IBM WebSphere MQ

➤ TIBCO Enterprise Message Service（EMS）

➤ Software AG webMethods Broker

➤ SQL Server データベース

➤ VMware vCenter または VMware ESX Server

Collector のインストール時に，上記アクティブ・システムのいずれかを監視するように選択できます。インストール後，監視する Oracle データベース，SQL Server システム，VMware vCenter または VMware ESX Server，IBM WebSphere MQ メッセージング・システム，TIBCO EMS システム，Software AG webMethods Broker，SAP NetWeaver–ABAP システムのインスタンスを定義します。監視対象のインスタンスはそれぞれ，プローブ・エンティティで表されます。コレクタごとに複数のプローブを設定できます。

注 : Collector は，どのマシンにもインストールできます。必ずしも，SAP，Oracle，MQ，Tibco EMS，webMethods Broker，VMware または SQL Server アプリケーションのホスト・マシンにインストールする必要はありません。Collector のホストの要件については，37 ページ「Diagnostics Collector ホストの要件」を参照してください。

Collector インストーラへのアクセス

Diagnostics インストール・ディスクからインストールを実行することも，実行可能インストール・ファイルをほかの場所にコピーして実行することも，または Business Service Management の Diagnostics［ダウンロード］ページからインストールを選択することもできます。

Diagnostics インストール元からインストーラにアクセスするには，次の手順を実行します。

➤ Windows の場合，Diagnostics インストール DVD（Autorun.exe）からインストール・メニュー・ページが表示されます。メニューから，［**Diagnostics Collector**］を選択してインストーラを起動します。

➤ または，インストール元から **HPDiagCollector_＜リリース番号＞_win.exe** ファイル（Windows の場合），または **HPDiagCollector_ ＜リリース番号＞ _ ＜プラットフォーム＞ .bin** ファイル（Unix の場合）を見つけ，そのファイルを新しいインストール先にコピーして .exe ファイルをダブルクリックするか，.bin インストーラを実行することもできます。

92 ページ「Collector のインストール」に進みます。

HP ソフトウェアのダウンロード・センターからインストーラをダウンロードするには，次の手順を実行します。

1 HP ソフトウェア Web サイトのソフトウェア・ダウンロード・センターにアクセスします。

2 **Diagnostics** ダウンロードを見つけて，Diagnostics Collector ソフトウェアのダウンロードに適切なリンクを選択します。

3 Web サイトのダウンロード手順を行います。

92 ページ「Collector のインストール」に進みます。

Business Service Management の Diagnostics [ダウンロード] ページからインストーラをダウンロードするには，次の手順を実行します。

1 **Business Service Management** のトップ・メニューから，[**管理**] > [**Diagnostics**] を選択し，[**ダウンロード**] タブをクリックします。

2 [ダウンロード] ページで，適切なリンクをクリックして該当する Collector インストーラをダウンロードします。

注 : Collector インストーラを Business Service Management で使用するには，Business Service Management からアクセスするために必要なディレクトリに配置する必要があります。これを可能にするには，Diagnostics サーバのインストール時に，Diagnostics Agent および Collector インストーラのパスを指定するか，ファイルをインストール・ディスクから Diagnostics サーバ・インストール・ディレクトリの **< Diagnostics サーバのインストール・ディレクトリ> /html/opal/downloads** フォルダに手動でコピーします。第 2 章，「Diagnostics サーバのインストール」の 60 ページ手順「」を参照してください。

次の「Collector のインストール」に進みます。

Collector のインストール

Windows または UNIX システムに Collector をインストールするための詳細な手順は次のとおりです。

ほかのタイプのインストールの詳細については，次を参照してください。

- Linux および Solaris 以外のプラットフォームに対応する標準 UNIX インストーラの使用手順については，102 ページ「標準インストーラを使用した Diagnostics Collector のインスト ール」を参照してください。
- サイレント・インストールの詳細については，101 ページ「Diagnostics Collector のサイレント・インストール」を参照してください。

注 : temp ディレクトリに対して約 400 MB の空き領域を使用可能にします。

注 : ホスト・マシンに Collector の既存インストールがある場合は，上記のインストール手順の代わりに，851 ページ「アップグレードとパッチ・インストールの手順」を参照して Collector のアップグレード手順に従ってください。

グラフィック・モードでの Windows インストーラと UNIX インストーラには同じ画面が表示されます。コンソール・モードで UNIX インストーラを実行する場合，画面の代わりにプロンプトが表示されますが，フローは本項に記載されている内容と同じです。

UNIX インストーラの場合は，必要に応じて，インストーラ・ファイルのモードを変更して実行可能にします。

➤ グラフィック・モードで UNIX インストーラを実行するには, UNIX コマンド・プロンプトで< installer >実行可能ファイルを入力します。< installer >は次のようになります。

HPDiagCollector_ ＜リリース番号＞ _sol.bin

HPDiagCollector_ ＜リリース番号＞ _linux.bin

インストーラには, Windows インストーラのものと同じ画面が表示されます。

➤ コンソール・モードで UNIX インストーラを実行するには, UNIX コマンド・プロンプトで< installer > -console を入力します。

次の手順は，UNIX コンソールの画面とコマンドを理解していることを前提としています。UNIX の画面とコマンドについては，881 ページ「UNIX コマンドの使用」を参照してください。インストーラはコンソール・モードで実行され，一連のプロンプトが表示されます。

インストーラを起動したら，ソフトウェア使用許諾契約書が開きます。

Collector をインストールするには，次の手順を実行します。

1 ソフトウェア使用許諾契約書に同意します。

使用許諾契約書を読み，［**使用条件の条項に同意します**］を選択します。コンソール・モードで使用許諾契約書を続けるには **Enter** キーを押します。プロンプトが表示されたら，**1** を入力して契約書に同意します。

続行するには［**次へ**］を選択します。

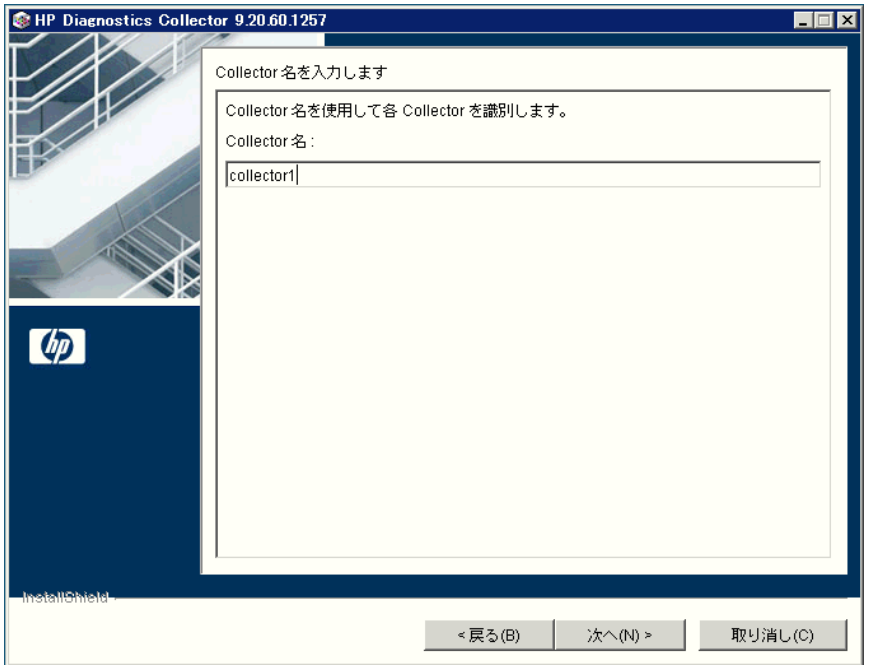
2 Collector のインストール場所を指定します。

［**インストール先ディレクトリ名**］ボックスで，標準設定のディレクトリ **C:¥MercuryDiagnostics¥Collector** を受け入れるか，Collector をインストールするディレクトリの名前を入力します。［**参照**］をクリックして別のディレクトリに移動します。このドキュメントでは，このディレクトリは< Collector のインストール・ディレクトリ >と表されます。

対象のディレクトリに, アップグレードする必要のある Collector の既存のインストールが含まれている場合は，このインストールをキャンセルし，付録 G, 「アップグレードとパッチ・インストールの手順」で説明されている Collectors のアップグレード手順に従う必要があります。

続行するには［**次へ**］を選択します。

3 Collector に一意の名前を割り当てます。



この特定の Collector を一意に識別する名前を Collector に割り当てます。名前には，-，_，すべての英数字を使用できます。

続行するには［**次へ**］を選択します。

4 監視する環境を選択します。

この Collector に適用するオプションを選択します。1 つ以上のオプションを選択できます。

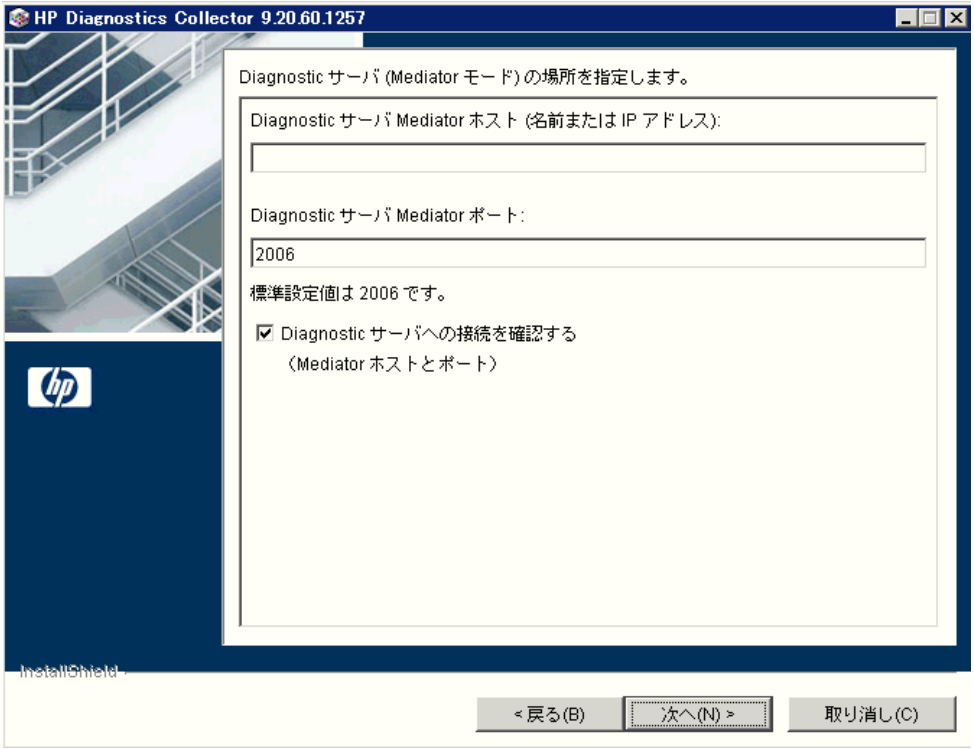
- SAP NetWeaver – ABAP 環境でデータを収集するには，［**SAP NetWeaver – ABAP**］を選択します。
- Oracle 10g データベース・サーバでデータを収集するには，［**Oracle**］を選択します。
- MQ Series 環境でデータを収集するには，［**MQ**］を選択します。
- SQL Server データベースでデータを収集するには，［**SQL Server**］を選択します。
- VMware vCenter または VMware ESX Server でデータを収集するには，［**VMware**］を選択します。
- TIBCO EMS 環境でデータを収集するには，［**TIBCO EMS**］を選択します。
- webMethods Broker システムでデータを収集するには，［**WebMethods Broker**］を選択します。

重要 : インストール後，監視する SAP NetWeaver–ABAP，Oracle，MQ，TIBCO EMS，SQL Server，webMethods Broker，VMware のインスタンスをそれぞれ指定します。これらのインスタンスは，インストーラに付属の XML ファイルに手動で定義できます。詳細については，104 ページ「アクティブ・システムのプロパティ・ファイルの設定」を参照してください。

続行するには［**次へ**］を選択します。

5 Diagnostic メディエータ・サーバの情報を入力します。

Diagnostic メディエータ・サーバとの通信を可能にする詳細情報を入力します。



Collector が実行する Diagnostics デプロイメントに Diagnostics サーバが 1 つしかない場合，その Diagnostics サーバのホスト名とイベント・ポート情報を入力します。

デプロイメントに Diagnostics サーバが複数ある場合は，Collector からイベントを受信する Diagnostic メディエータ・サーバの情報を入力します。

a ［**Diagnostics サーバ Mediator ホスト**］ボックスに，Diagnostic メディエータ・サーバのホストのホスト名と IP アドレスを入力します。

注 : 完全修飾ホスト名を指定する必要があります。OS が混在し，その 1 つが UNIX の環境で，これは正確なネットワーク・ルーティングのために不可欠です。

---

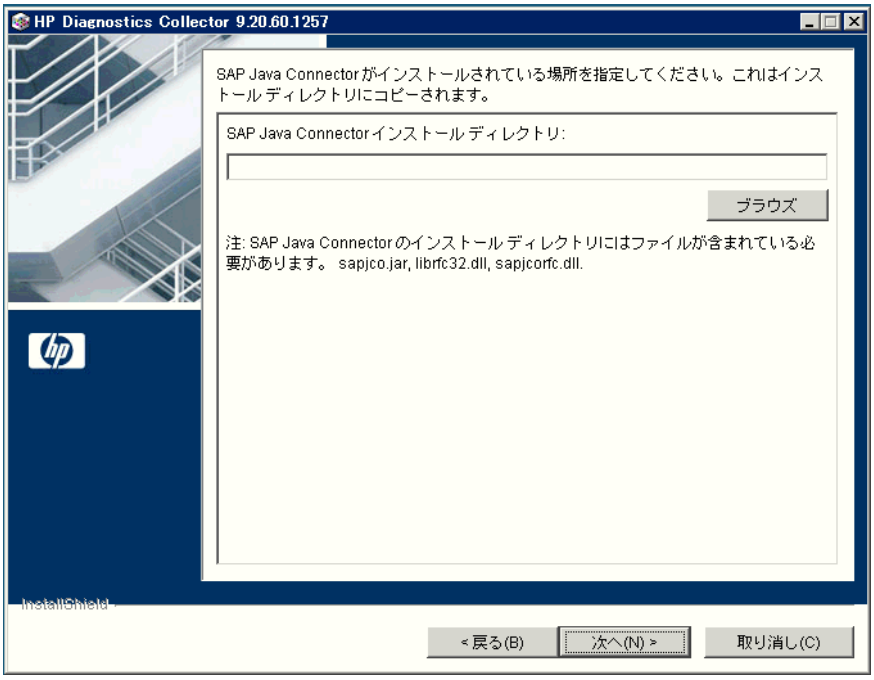
b ［**Diagnostics サーバ Mediator ポート**］ボックスに，Diagnostics サーバが Collector との通信のためにリッスンしているポート番号を入力します。デフォルトのポート番号は **2006** です。Diagnostics サーバをインストールした後にポートを変更した場合は，標準設定のポート番号ではなく，変更後のポート番号を指定します。

c Diagnostics サーバが実行中で，インストール・ホストからアクセス可能なことを確認するには，［**Diagnostics サーバへの接続を確認する（Mediator ホストとポート）**］を選択します。

続行するには［**次へ**］を選択します。

［**Diagnostics サーバへの接続を確認する（Mediator ホストとポート）**］を選択していて，接続の問題が生じた場合，インストーラは接続チェックの結果を通知します。ここで問題を解決しない場合は，［**Diagnostics サーバへの接続を確認する（Mediator ホストとポート）**］チェック・ボックスをクリアして，インストールに進み，後で問題を解決します。

6 手順 4 で［SAP NetWeaver – ABAP］を選択した場合，SAP Java Connector の場所を指定します。



［**SAP Java Connector インストール ディレクトリ**］ボックスに，SAP Java Connector がインストールされているディレクトリへのパスを入力します。インストーラによって，Collector がインストールされているシステムの＜**Collector のインストール・ディレクトリ＞ ¥lib** ディレクトリに必要なファイルがコピーされます。

このディレクトリには，次のファイルが含まれている必要があります。

- sapjco.jar
- librfc.dll，librfc32.dll，または librfccm.so
- sapjcorfc.dll または libsapjcorfc.so

SAP Java Connector のディレクトリ名がわからない場合，またはディレクトリ内にこれらのファイルがない場合は，SAP 窓口にお問い合わせください。

7 インストール後に必要なファイルを Collector システムにコピーしてください。

Tibco EMS を選択した場合，インストール後にサードパーティの jar である **tibjms.jar** と **tibjmsadmin.jar** のコピーを求めるリマインダが表示されます。通常，これらのファイルは TIBCO EMS インストールの**＜ Tibco EMS ＞ /ems/ ＜ バージョン＞ /lib** ディレクトリ内にあり，Collector がインストールされているシステムの**＜ Collector のインストール・ディレクトリ＞ ¥lib** ディレクトリにコピーします。

webMethods Broker を選択した場合，インストール後にサードパーティの jar である **wm-brokerclient.jar**，**wm-g11nutils.jar** のコピーを求めるリマインダが表示されます。通常，これらのファイルは Software AG インストールの**＜SoftwareAG＞/common/lib** ディレクトリ内にあり，Collector がインストールされているシステムの**＜ Collector のインストール・ディレクトリ＞ ¥lib** ディレクトリにコピーします。

8 インストール前にサマリを確認します。

選択したインストール設定が表示されます。表示された情報が正確か確認します。

異なるインストール設定を選択する場合は，［**戻る**］をクリック（またはコンソール・モードで［前へ］を選択）します。

インストールを開始するには，［**次へ**］を選択します。

9 インストールが完了します。

インストールが完了すると，Collector が正常にインストールされたことを伝えるメッセージが表示されます。インストーラを終了するには，［**完了**］を選択します。

10 アクティブ・システムの XML ファイルを設定します。

手順 4 で，監視するアクティブ・システムを選択しました。各アクティブ・システムに，Collector のホストとアクティブ・システムのホストが通信できるようにするプロパティを設定する必要があります。

関連するアクティブ・システム・プロパティの設定については，104 ページ「アクティブ・システムのプロパティ・ファイルの設定」を参照してください。

11 Collector が正常にインストールされ，実行していることを確認します。

インストールが完了すると，Collector は自動的に実行を開始します。Collector のインストールを検証するには，**collector.log** ファイルでエラーを確認します。詳細については，125 ページ「Diagnostics Collector のインストールの確認」を参照してください。

Diagnostics UI では，各 Collector インスタンスはシステム・タイプのプローブ・エンティティ（Oracle Probe，SAP Probe，MQ Probe，EMS probe，WM probe または SQL Server Probe）として表されます。

Diagnostics Collector のサイレント・インストール

サイレント・インストールは，ユーザが操作することなく自動的に実行されます。サイレント・インストールでは，ユーザ入力の代わりに，各インストール手順の応答ファイルからの入力を使用します。

たとえば，複数のマシンにコンポーネントをデプロイする必要のあるシステム管理者は，必要な設定情報がすべて含まれる応答ファイルを作成し，複数のマシン上でサイレント・インストールを実行します。これにより，インストール中に手動で入力する必要がなくなります。

複数のマシン上でサイレント・インストールを実行する前に，インストール中に入力を提供する応答ファイルを作成する必要があります。この応答ファイルは，インストール時に同じ入力を必要とするすべてのサイレント・インストールで使用できます。

重要 : Diagnostics の新しいリリースごとに，複数のマシン上でサイレント・インストールを実行する前に，Diagnostics Collector のサイレント・インストール応答ファイルを再度記録する必要があります。

応答ファイルには拡張子 **.rsp** が付いています。標準的なテキスト・エディタで応答ファイルを編集できます。

応答ファイルを作成するには，次の手順を実行します。

➤ 次のコマンド・ライン・オプションを使って通常のインストールを実行します。

```
＜インストーラ＞ -options-record ＜ responseFileName ＞
```

これにより，インストール中に送信されたすべての情報が含まれる応答ファイルが作成されます。

サイレント・インストールを実行するには，次の手順を実行します。

➤ 関連する応答ファイルを使って，サイレント・インストールを実行します。

-silent コマンド・ライン・オプションを使って，次のようにサイレント・インストールを実行します。

```
＜インストーラ＞ -silent -options ＜ responseFileName ＞
```

サイレント・インストールを実行する場合は，2 つの追加オプションを指定できます。

➤ 応答ファイル名の後に **-is:log ＜ログ・ファイルのパス＞**オプションを指定して，ログ・ファイルを作成する

➤ 応答ファイル名の後に **-is:tempdir ＜一時ディレクトリのパス＞**オプションを指定して，一時ディレクトリをユーザ指定のディレクトリに変更する

標準インストーラを使用した Diagnostics Collector のインストール

Diagnostics Collector のインストーラでは，Windows，Linux，および Solaris システムへの Collector のインストールをサポートしています。汎用 UNIX インストーラがインストール・ディスクに収録されていて，HP-UX や AIX など，その他のプラットフォームに Collector をインストールできます。

標準インストーラを使用して Diagnostics Collector をインストールするには，次の手順を実行します。

1 HP Diagnostics インストール・ディスクの Diagnostics Installers フォルダ内で，**HPDiagCollector_ ＜リリース番号＞ _unix.zip** を探しします。

2 Collector をインストールするシステム上にファイルを展開します。

3 監視するアクティブ・システムごとに，Collector のホストとアクティブ・システムのホストが通信できるようにするプロパティを設定する必要があります。アクティブ・システム・プロパティの設定については，104 ページ「アクティブ・システムのプロパティ・ファイルの設定」を参照してください。

Collector のインストール後に別のコレクション・タイプを手動で追加する方法

Collector の初期インストール中に，監視する異なるコレクション・タイプまたはアクティブ・システム・タイプ（SAP や Oracle など）を選択します。Collector のインストール後，別のコレクション・タイプまたはアクティブ・システム・タイプを手動で追加できます。

別のアクティブ・システム・タイプを手動で追加するには，次の手順を実行します。

1 必要なファイルを手動で**＜ Collector のインストール・ディレクトリ＞ ¥lib** ディレクトリにコピーします。これらのファイルは，SAP，Tibco EMS，webMethods Broker アクティブ・システムで必要なファイルです。必要なファイルの詳細については，インストール手順を参照してください。ほかのタイプのコレクション（Oracle または SQL Server など）ではこの手順は不要です。

2 Collector をインストールしたシステム上の**＜ Collector のインストール・ディレクトリ＞ ¥etc¥collector.properties** ファイルで，**active.systems** プロパティを編集してほかのコレクション・タイプを追加します。有効な値（大文字と小文字を区別しない）は，SAP_R3，Oracle，MQ，SQL_Server，VMWARE，EMS，WM_BROKER です。

アクティブ・システムのプロパティ・ファイルの設定

Collector をインストールする際，コレクション・タイプ（Collector で監視するアクティブ・システム）を指定するように指示されます。インストール後，監視するアクティブ・システムのインスタンスを定義します。これらのインスタンスは，Collector のインストールに付属の XML ファイルに手動で定義できます。XML ファイルのインスタンス定義は，アクティブ・システムのプローブ・エンティティとして表示されます。設定方法については，次の項を参照してください。

- 104 ページ「SAP NetWeaver – ABAP の設定」
- 108 ページ「Oracle の設定」
- 111 ページ「SQL Server の設定」
- 115 ページ「MQ の設定」
- 118 ページ「TIBCO EMS の設定」
- 119 ページ「webMethods Broker の設定」
- 121 ページ「VMware の設定」

SAP NetWeaver – ABAP の設定

SAP NetWeaver – ABAP システム・デプロイメントには 1 つ以上の SAP NetWeaver – ABAP アプリケーション・インスタンスを含めることができます。これらのインスタンスは，同時に SAP NetWeaver – ABAP システムを構成します。

ユーザ権限に応じて，システムのシステム・インスタンスまたはアプリケーション・インスタンスに直接アクセスできる場合もあれば，SAP Message Server を介して接続するように要求されることもあります。SAP NetWeaver – ABAP Probe エンティティごとに，使用されている接続オプションを把握する必要があります。

監視対象のアクティブな SAP NetWeaver – ABAP システムのインスタンスごとにデータを収集するように，Collector を設定します。**＜Collector のインストール・ディレクトリ＞¥etc¥r3config.xml** ファイルで，監視対象の SAP NetWeaver – ABAP を設定します。XML ファイルのレイアウト，要素，および属性は，**＜Collector のインストール・ディレクトリ＞¥etc¥r3config.xsd** に記述されています。

SAP NetWeaver – ABAP モニタリングを設定するには，次の手順を実行します。

1 **Collector¥etc¥r3config.xml** を開きます。

2 SAP Message Server を通して SAP NetWeaver – ABAP インスタンスにアクセスする SAP NetWeaver – ABAP Probe を定義している場合は，コードの中で次のコメントで始まる部分を見つけます。

```
<!--
Template to be used with the message server connection option.
-->
```

SAP NetWeaver – ABAP インスタンスに直接アクセスする SAP NetWeaver – ABAP Probe を定義している場合は，コードの中で次のコメントで始まる部分を見つけます。

```
<!--
Template to be used with the direct connection option.
-->
```

3 コメントと，コメントの下のテンプレート・コードをコピーして，ファイルの末尾に貼り付けます。

4 テンプレート・コードの上の空行に **!--** と入力し，その後の空行に **-->** と入力して，元のテンプレート・コードをコメント・アウトします。

5 ファイル末尾にあるコピーしたコードで，次の表どおりに各プロパティの値を変更し，ファイルを保存します。

プロパティ	説明	値
r3system name	Diagnostics UI にこの SAP NetWeaver–ABAP Probe エンティティを表示するためのプローブ・グループの論理名。	ユーザ定義。
systemId	SAP NetWeaver–ABAP システムの ID。3 文字だけで構成されます。	形式：[XXX] SAP システム管理者から取得可能。

プロパティ	説明	値
client	SAP NetWeaver–ABAP システムのクライアント名。	SAP システム管理者から取得可能。
user	SAP NetWeaver–ABAP システムに接続しているユーザの名前。 このユーザは，ダイアログ情報をクエリするために少なくとも S_RFC 認証オブジェクトが必要です。ターゲット・システムのユーザは，RFC を使用してターゲット・システムに接続できる認証プロファイルにこのオブジェクトが含まれている必要があります。 R/3 4.7 以前のシステムでは不十分です。この問題に対処するには，ABAP ホストと時間が同期したマシンに Collector をインストールし，（Collector¥etc¥r3.properties で）プロパティ timesynch.interval.secs を 0 に設定して Collector の時間同期を無効にします。	SAP システム管理者から取得可能。
password	SAP NetWeaver-ABAP システムに接続しているユーザのパスワード（プレーンテキスト）。	SAP システム管理者から取得可能。
encrypted-password	SAP NetWeaver-ABAP システムに接続しているユーザのパスワード（暗号化されたもの）。	EncryptPassword.jsp ユーティリティを使って（123 ページ「パスワードの暗号化」を参照），パスワードを暗号化します。
messageServerHost （Message Server 接続のみ）	SAP Message Server のホスト・マシンの名前。	SAP システム管理者から取得可能。

プロパティ	説明	値
r3Name （Message Server 接続のみ）	3 文字だけで構成されます。	形式：[XXX] SAP システム管理者から取得可能。
group （Message Server 接続のみ）	SAP アプリケーション・サーバのグループ。	SAP システム管理者から取得可能。
dialogInstance	監視するダイアログ・インスタンスのリストを指定します。 標準設定では，ABAP システム（クラスタ）内のすべてのダイアログ・インスタンスは，自動的に検出，監視されます。 ただし，ダイアログ・インスタンスが 1 つの Collector で処理するには多すぎる（ビジー状態になりメモリ不足になる可能性がある）場合，このプロパティを使用して一部のダイアログ・インスタンスのみを監視し，残りのダイアログ・インスタンスを異なる Collector で監視できます。	SAP ダイアログ・インスタンス

Oracle の設定

監視対象のアクティブな Oracle システムのインスタンスごとにデータを収集するように，Collector を設定します。**＜ Collector のインストール・ディレクトリ＞ ¥etc¥oracle-config.xml** ファイルで Oracle モニタリングを設定します。XML ファイルのレイアウト，要素，および属性は，**＜ Collector のインストール・ディレクトリ＞ ¥etc¥oracle-config.xsd** に記述されています。

Oracle モニタリングを設定するには，次の手順を実行します。

1 **＜ Collector のインストール・ディレクトリ＞ ¥etc¥oracle-config.xml** を開きます。

2 コメント・タグ（<!-- と -->）で囲まれたテンプレート・コードをコピーして，ファイルの末尾に貼り付けます。

テンプレートの **oracleInstance** 要素を使用して，Oracle 10g および 11g インスタンスから収集します。複数のインスタンスから収集する場合は，oracleInstance 要素の別のエントリを追加します。

Oracle RAC（Real Application Cluster）から収集するには，**oracleRac** 要素を指定します。oracleRac 設定は，**oracle-config.xml** ファイルの oracleInstance 設定の後に指定する必要があります。

3 コピーしたコードで，次の表どおりに各プロパティの値を変更し，ファイルを保存します。

プロパティ	説明	値
hostName	Oracle データベース・サーバのホスト・マシンの名前。完全修飾ホスト名を使用する必要があります。 oracleRAC 設定では，これはクラスタの別名です。	Oracle 管理者から取得可能。
portNumber	Oracle データベース・サーバが要求をリッスンするポート。	標準設定値 : **1521**
instanceName	oracleInstance 要素に使用します（oracleRAC 要素には適用されません）。Oracle データベース・サーバのインストール時に Oracle インスタンスに指定された名前。	標準設定値 : **Orcl** Oracle 管理者から取得可能。

プロパティ	説明	値
serviceName	oracleRac 要素に使用します（oracleInstance 要素には適用されません）。serviceName およびクラスタの別名 hostName の組み合わせによって，クライアントが RAC インストールの変更から分離されます。	Oracle 管理者から取得可能。
userId	Oracle データベース・サーバに接続しているユーザの ID。 **注 :** パフォーマンス・メトリックスを収集するには，少なくともユーザには **CREATE SESSION** および **SELECT ANY DICTIONARY** が必要です。	Oracle 管理者から取得可能。
password	Oracle データベース・サーバに接続しているユーザのパスワード（プレーンテキスト）。	Oracle 管理者から取得可能。
encrypted-password	Oracle データベース・サーバに接続しているユーザのパスワード（暗号化されたもの）。	EncryptPassword.jsp ユーティリティを使って（123 ページ「パスワードの暗号化」を参照），パスワードを暗号化します。

プロパティ	説明	値
probeName	oracleInstance 用。Diagnostics UI にこの Oracle インスタンスを表示するための論理名。この名前は一意である必要があります。 oracleRac 設定には複数のプローブがあるため，probeName は入力しません。	ユーザ定義。この値が定義されていない場合は，**instanceName** と同じ値が使われる。 Oracle RAC 設定内の各 Oracle インスタンスのプローブ名は，GV$INSTANCE ビューの INSTANCE_NAME 列から実行時に取得されます。
probeGroupName	Diagnostics UI にこのプローブを表示するためのプローブ・グループの論理名。既存のプローブ・グループを使うことも，または新しいプローブ・グループを定義することもできます。 oracleRac 要素では省略可能で，省略した場合はプローブ・グループが serviceName に設定されます。	ユーザ定義。たとえば，次のようになる。 既存 : **Default** 新規 : **Oracle**

追加メトリックスのコレクションを有効にするには，次の手順を実行します。

1 Collector で収集するように設定されていないメトリックスが検出された場合，認識できないメトリックスの ID と名前を含む警告がログに記録されます。メトリックスが count，percent，byte，または centisecond メトリックスの場合，必要に応じてメトリックス ID を**＜ Collector のインストール・ディレクトリ＞¥etc¥oracle.properties** に追加することで，そのメトリックスを収集できます。

2 Collector で収集するメトリックスのタイプに対応するプロパティ名を特定し，メトリックスを追加します。プロパティ名は次のとおりです。

- oracle.metrics.count
- oracle.metrics.percent

- oracle.metrics.bytes
- oracle.metrics.centiseconds（Collector によってミリ秒に変換されます）

3 Collector を再起動します。

SQL Server の設定

監視対象のアクティブな SQL Server システムのインスタンスごとにデータを収集するように，Collector を設定します。**＜ Collector のインストール・ディレクトリ＞ ¥etc¥sqlserver-config.xml** ファイルで SQL モニタリングを設定します。XML ファイルのレイアウト，要素，および属性は，**＜ Collector のインストール・ディレクトリ＞ ¥etc¥sqlserver-config.xsd** に記述されています。

SQL Server モニタリングを設定するには，次の手順を実行します。

1 **＜ Collector のインストール・ディレクトリ＞ ¥etc¥sqlserver-config.xml** を開きます。

2 テンプレート・コードをコピーして，ファイルの末尾に貼り付けます。

3 テンプレート・コードの上の空行に **<!--** と入力し，その後の空行に **-->** と入力して，テンプレート・コードをコメント・アウトします。

4 コピーしたコードで，次の表どおりに各プロパティの値を変更し，ファイルを保存します。

プロパティ	説明	値
hostName	SQL Server データベースのホスト・マシンの名前。完全修飾ホスト名を使用する必要があります。	SQL Server 管理者から取得可能。
portNumber	SQL Server データベースが要求をリッスンするポートの番号。	標準設定値 : **1433**

プロパティ	説明	値
instanceName	SQL Server データベースのインストール時に SQL Server インスタンスに指定された名前。 インスタンス名を指定すると，**インスタンス内のすべての SQL Server データベースが Diagnostics によって自動的に検出されます**。一部のデータベース（システム・データベースなど）を収集対象から除外する場合は，**＜Collector のインストール・ディレクトリ＞¥etc¥sqlserver.properties** ファイルの **exclude.db.list** プロパティに，カンマで区切られたリストを指定します。	標準設定値 : **Default** SQL Server 管理者から取得可能。

プロパティ	説明	値
integratedSecurity	**true** に設定されている場合は，ユーザ名 / パスワードは指定しないでください。ローカル・コンピュータの資格情報キャッシュ内で，コンピュータまたはネットワークのログオン時に指定された資格情報が JDBC ドライバによって検索されます。 サービス **HP Diagnostics Collector** から Collector を実行した場合は，そのサービスの **logon** プロパティに，SQL Server との接続に使用される Windows ユーザ資格情報を設定する必要があります。このためには，Windows Services Manager（実行ダイアログで **services.msc** を選択するか，[**マイ コンピュータ**] > [**管理**] > [**サービスとアプリケーション**] > [**サービス**] を選択）を実行します。サービス **HP Diagnostics Collector** の [プロパティ] ダイアログが開きます。[ログオン] タブを選択して，[**Log on as:**] を SQL Server インスタンスへのアクセスが許可されたユーザに設定します。これは，ドメイン・アカウントである必要があります。サービスを再起動します。 Windows 認証を使用している場合は，SQL Server インスタンスとの接続にドメイン・アカウント（ローカル・アカウントでない）を使用する必要があります。 **false** に設定されている場合は，ユーザ名およびパスワードを指定する必要があります。指定しなかった場合の標準設定値は，false です。	標準設定値 : **false**

プロパティ	説明	値
userId	SQL Server データベースに接続しているユーザの ID。 **注 :** パフォーマンス・メトリックスを収集するには，少なくともユーザには **VIEW SERVER STATE** が必要です。 次の手順で VIEW SERVER STATE のユーザを作成する。 ➤ ログイン Diagnostics を作成（パスワード =「<pwd>」） ➤ マスタを使用 ➤ Diagnostics に VIEW SERVER STATE を付与 ➤ 実行 SQL Server Management Studio GUI でインスタンス名を右クリックして，プロパティを選択できる。	SQL Server 管理者から取得可能。
password	SQL Server データベースに接続しているユーザのパスワード（プレーンテキスト）。	SQL Server 管理者から取得可能。
encrypted-password	SQL Server データベースに接続しているユーザのパスワード（暗号化）。	EncryptPassword.jsp ユーティリティを使って（123 ページ「パスワードの暗号化」を参照），パスワードを暗号化します。

プロパティ	説明	値
probeName	HP Diagnostics UI で，このインスタンスをプローブとして表すのに使う名前。 インスタンス内に n 個のデータベースがある場合，実際は n+1 個のプローブがある。待機イベントなどのメトリックスを含む，インスタンス全体に関する追加プローブが 1 つ存在する。 追加プローブは，probeName として UI に表示されます。データベース単位のプローブは，probeName_databaseName と表示されます。	ユーザ定義。 この値が定義されていない場合は，**instanceName** と同じ値が使われる。
probeGroupName	Diagnostics UI にこのプローブ・エンティティを表示するためのプローブ・グループの論理名。 既存のプローブ・グループを使うことも，または新しいプローブ・グループを定義することもできます。	ユーザ定義。たとえば，次のようになる。 既存 : **Default** 新規 : **SQL Server**

MQ の設定

監視対象のアクティブな MQ システムのインスタンスごとにデータを収集するように，Collector を設定します。**＜ Collector のインストール・ディレクトリ＞¥etc¥mq-config.xml** ファイルで MQ モニタリングを設定します。XML ファイルのレイアウト，要素，および属性は，**＜ Collector のインストール・ディレクトリ＞ ¥etc¥mq-config.xsd** に記述されています。

MQ Probe には，次の権限が必要です。

setmqaut -m <queue_manager_name> -n ** -t queue -g <OS_group_name> +dsp +get

setmqaut -m <queue_manager_name> -n SYSTEM.ADMIN.COMMAND.QUEUE -t queue -g <OS_group_name> +dsp +get +put

setmqaut -m <queue_manager_name> -n ** -t channel -g <OS_group_name> +dsp
setmqaut -m <queue_manager_name> -t qmgr -g <OS_group_name> +connect +dsp +inq

MQ プローブがメトリックスを収集するキューのタイプを限定して，アプリケーションで最も関心のあるメトリックスを取り出すことができます。標準設定では，MQ Probe は定義済み（非動的）キューのみからメトリックスを収集します。**＜ Collector のインストール・ディレクトリ＞ ¥etc¥mq.properties** ファイルでプロパティを設定することによって，収集または無視するキューのタイプを指定します。

収集するメトリックスのキューを限定するには，次の手順を実行します。

1 **＜ Collector のインストール・ディレクトリ＞ ¥etc¥mq.properties** ファイルを開きます。

2 Collector でメトリックスを収集しない MQ Queue 定義タイプに対応するプロパティ名を探します。次の表は，プロパティ名と対応する MQ Queue 定義タイプを示します。

プロパティ	MQ Queue 定義タイプ
collect.predefined.queues	MQQDT_PREDEFINED
collect.permanent.dynamic.queues	MQQDT_PERMANENT_DYNAMIC
collect.temporary.dynamic.queues	MQQDT_TEMPORARY_DYNAMIC
collect.shared.dynamic.queues	MQQDT_SHARED_DYNAMIC

3 **collect.predefined.queues** プロパティは標準設定で true に設定されています。その他の 3 つのプロパティは標準設定で false に設定されています。Collector でメトリックスを収集したくないタイプに対して **false** を指定し，**mq.properties** ファイルを保存します。

注 : これらのプロパティは，MQ 6.x およびそれ以降のバージョンのみサポートされます。

MQ jar ファイルは Diagnostics Collector に含まれていますが，必要に応じてこれらのファイルを上書きすることもできます。Collector に付属された MQ jar ファイルは**＜ Collector のインストール・ディレクトリ＞ ¥lib** ディレクトリにあり，ローカル MQ インストールに付属された MQ jar ファイルで上書きできます。通常，これらの jar ファイルはローカル WebSphere の MQ インストールの **com.ibm.mq.jar** ファイルが含まれる，**¥lib** ディレクトリ内にあります。これらのファイルが見つからない場合は，WebSphere MQ 管理者にお問い合わせください。

MQ モニタリングを設定するには，次の手順を実行します。

1 **＜Collector のインストール・ディレクトリ＞¥etc¥mq-config.xml** を開きます。

2 テンプレート・コードをコピーして，ファイルの末尾に貼り付けます。

3 テンプレート・コードの上の空行に **<!--** と入力し，その後の空行に **-->** と入力して，テンプレート・コードをコメント・アウトします。

4 コピーしたコードで，次の表どおりに各プロパティの値を変更し，ファイルを保存します。

プロパティ	説明	値
hostName	ホスト名。	MQ 管理者から取得可能。
portNumber	ポートの番号。	(この手順は省略可能です)。
queueManagerName	接続する MQ Manager。	MQ 管理者から取得可能。
channelName	キュー・マネージャへの接続に使用するチャネル。	MQ 管理者から取得可能。
securityExit	プラグ可能なセキュリティ・プロバイダに関する IBM の用語（MQ にセキュアなインタフェースを提供するコード)。 MQ へのゲートウェイとして securityExit を使用している場合，パラメータとして完全なクラス名を指定し，クラスパスに securityExit クラスがあることを確認します。	
probeName	HP Diagnostics UI で，このインスタンスをプローブとして表すのに使う名前。 この名前は一意である必要があります。	ユーザ定義。定義されていない場合，標準設定でキュー・マネージャの名前が使用されます。
probeGroupName	Diagnostics UI にこのプローブエンティティを表示するためのプローブ・ループの論理名。既存のプローブ・グループを使うことも，または新しいプローブ・グループを定義することもできる。	ユーザ定義。たとえば，次のようになる。 既存 : **Default** 新規 : **MQ**

TIBCO EMS の設定

監視対象のアクティブな TIBCO Enterprise Message Service（EMS）システムのインスタンスごとにデータを収集するように，Collector を設定します。

＜Collector のインストール・ディレクトリ＞¥etc¥tibco-ems-config.xml ファイルで TIBCO EMS モニタリングを設定します。XML ファイルのレイアウト，要素，属性は，**＜Collector のインストール・ディレクトリ＞¥etc¥tibco-ems-config.xsd** に記述されています。

下記の設定に加えて，次の TIBCO EMS jar ファイルを TIBCO EMS インストールの**＜Tibco EMS＞/ems/＜バージョン＞/lib** ディレクトリから，Collector がインストールされているシステムの**＜Collector のインストール・ディレクトリ＞¥lib** ディレクトリにコピーする必要があります。

- tibjms.jar
- tibjmsadmin.jar

TIBCO EMS モニタリングを設定するには，次の手順を実行します。

1. **＜Collector のインストール・ディレクトリ＞¥etc¥tibco-ems-config.xml** を開きます。
2. テンプレート・コードをコピーして，ファイルの末尾に貼り付けます。
3. テンプレート・コードの上の空行に **<!--** と入力し，その後の空行に **-->** と入力して，テンプレート・コードをコメント・アウトします。
4. コピーしたコードで，次の表どおりに各プロパティの値を変更し，ファイルを保存します。

プロパティ	説明	値
emsServerUrl	EMS サーバの URL。	標準設定は tcp://localhost:7222。
username	EMS サーバのユーザ名。ユーザには「view-destination」および「view-server」権限が必要です。	
password	EMS サーバのパスワード（プレーンテキスト）。	

プロパティ	説明	値
obfuscated-password	EMS サーバの暗号化されたパスワード（任意）。両方が定義されている場合，このプロパティはプレーンテキストのパスワードよりも優先されます。いずれも定義されていない場合，パスワードには空白が使用されます。	EncryptPassword.jsp ユーティリティを使って（123 ページ「パスワードの暗号化」を参照），パスワードを暗号化します。
probeName	HP Diagnostics UI で, このインスタンスをプローブとして表すのに使う名前。 この名前は一意である必要があります。	ユーザ定義。
probeGroupName	Diagnostics UI にこのプローブ・エンティティを表示するためのプローブ・グループの論理名。 既存のプローブ・グループを使うことも, または新しいプローブ・グループを定義することもできる。	次のようなユーザ定義。 既存 : **Default** 新規 : **TIBCO**

＜Collector のインストール・ディレクトリ＞¥etc¥tibco-ems.properties ファイルでプロパティを設定することで，TIBCO データ収集をカスタマイズできます。

- データを収集する頻度
- 接続が確立されていない場合に再接続を試みる頻度
- サーバ・レベル，キュー・レベル，トピック・レベルのメトリックス収集の有効化または無効化
- グローバル，静的，または一時キューおよびトピックの包含または除外
- 個々のメトリックスの選択

webMethods Broker の設定

監視対象の webMethods Broker システムのデータを収集するように Collector を設定できます。

＜Collector のインストール･ディレクトリ＞¥etc¥wm-broker-config.xml ファイルで webMethods Broker モニタリングを設定します。XML ファイルのレイアウト，要素，属性は，**＜Collector のインストール・ディレクトリ＞¥etc¥wm-broker-config.xsd** に記述されています。

下記の設定に加えて，次の webMethods Broker jar ファイルを webMethods Broker インストールの**＜ Software AG ＞ /common/lib** ディレクトリから，Collector がインストールされているシステムの**＜ Collector のインストール・ディレクトリ＞¥lib** ディレクトリにコピーする必要があります。

- wm-brokerclient.jar
- wm-g11nutils.jar

webMethods Broker モニタリングを設定するには，次の手順を実行します。

1. **＜ Collector のインストール・ディレクトリ＞ ¥etc¥wm-broker-config.xml** を開きます。
2. テンプレート・コードをコピーして，ファイルの末尾に貼り付けます。
3. テンプレート・コードの上の空行に **<!--** と入力し，その後の空行に **-->** と入力して，テンプレート・コードをコメント・アウトします。
4. コピーしたコードで，次の表どおりに各プロパティの値を変更し，ファイルを保存します。

プロパティ	説明	値
hostname	Broker サーバのホスト名。	必須。例 : localhost
brokerName	接続するブローカの名前。省略した場合，Broker サーバ内で定義された標準設定のブローカに接続します。	省略可能

プロパティ	説明	値
clientGroup	使用するクライアント・グループの名前。省略した場合,「admin」クライアント・グループに接続します。	省略可能
probeGroupName	Diagnostics UI にこのプローブ・エンティティを表示するためのプローブ・グループの論理名。 既存のプローブ・グループを使うことも, または新しいプローブ・グループを定義することもできる。	省略可能。標準設定は **Default**。 ユーザ定義の名前を入力できます。

＜ Collector のインストール・ディレクトリ＞ ¥etc¥wm-broker.properties ファイルでプロパティを設定することで，webMethods データ収集をカスタマイズできます。

- データを収集する頻度
- 接続が確立されていない場合に再接続を試みる頻度
- サーバ・レベル，キュー・レベルのメトリックス収集の有効化または無効化
- 個々のメトリックスの選択

VMware の設定

監視対象の各 VMware ノードのデータを収集するように Collector を設定できます。**＜ Collector のインストール・ディレクトリ＞ ¥etc¥vmware-config.xml** ファイルで VMware モニタリングを設定します。XML ファイルのレイアウト，要素，属性は，**＜ Collector のインストール・ディレクトリ＞ ¥etc¥vmware-config.xsd** に記述されています。**vmware-config.xml** ファイルへの変更は動的に反映されます。

Collector では，vCenter Server にパッチをインストールする必要があります。詳細については，<http://kb.vmware.com/selfservice/microsites/search.do?cmd=displayKC&docType=kc&externalId=1024596&sliceId=1&docTypeID=DT_KB_1_1&dialogID=139216791&stateId=1 0 139218894>（英語サイト）を参照してください。

VMware ゲストに最新の VMware Tools がインストールされている必要があります。これらのツールは，vSphere Client を使用してインストールできます。VMware Tools は vCenter 経由でゲストの FQDN を VMware Collector で使用可能にするため，Diagnostics UI で VMware ゲストからホストにドリルダウンするために最新のツールが必要です。

VMware ホスト / ゲストの関連付けは，Collector の起動時に作成されます。新しい VMware ホストと VMware ゲストが Diagnostics に表示されるまで, 最大で 15 分かかることがあります。削除または移行された VMware ゲストが Diagnostics に表示されるまで，最大で 5 分かかることがあります。

VMware モニタリングを設定するには，次の手順を実行します。

1 **＜ Collector のインストール・ディレクトリ＞ ¥etc¥vmware-config.xml** を開きます。

2 テンプレート・コードをコピーして，ファイルの末尾に貼り付けます。

3 テンプレート・コードの上の空行に **<!--** と入力し，その後の空行に **-->** と入力して，テンプレート・コードをコメント・アウトします。

4 コピーしたコードで，次の表どおりに各プロパティの値を変更し，ファイルを保存します。

プロパティ	説明	値
serverURL	VMware インフラストラクチャの vSphere Web サービス API で VMware ESX または VCenter Server に接続するために使用される URL。 次に例を示します。 https://<myVM.myCo.com>/sdk	
userId	VMware ESX または vCenter のユーザ ID。 ユーザには少なくとも読み取り専用アクセス権が必要で，ユーザ・グループに属している必要があります。	

プロパティ	説明	値
encrypted-password	userId に対応する暗号化された VMware パスワード。最初に暗号化されたパスワードを確認して, 空白でない場合はそれを使用し, 空白の場合はプレーンテキスト・パスワードを使用します。プレーンテキスト・パスワードが存在しないか空白の場合は, パスワードに空白を使用します。	省略可能
password	userId に対応するプレーンテキストの VMware パスワード。	省略可能

＜Collector のインストール・ディレクトリ＞¥etc¥vmware.properties ファイルでプロパティを設定することで，VMware データ収集を次のようにカスタマイズできます。

- クエリ間隔および再接続時間を制限できます。サンプリング間隔は実際には VMware Server で設定された間隔の倍数である必要があるため，Collector にとってクエリ間隔は単なるヒントにすぎません。
- VMware ホスト（ESX Server）および VMware ゲスト（仮想マシン）でフィルタすることもできます。VMware Collector が vCenter 全体の負荷を処理できない場合，ホストおよびゲストのフィルタによって，vCenter の最も重要な部分で VMware Collector を使用するか, 複数の VMware Collector 間で vCenter をパーティションで区切ることができます。

パスワードの暗号化

Diagnostics 付属の Web アプリケーションを使用して，暗号化されたパスワードを作成します。[セキュリティ] ページ（http:// ＜ホスト名＞ :2006/security）にアクセスして，ページの下部で［**パスワードの暗号化**］を選択します。＜ホスト名＞は，Diagnostics サーバがインストールされているコンピュータ名で置き換えます。

作成した暗号化パスワードは，異なるコレクション・タイプの次の xml ファイルで使用できます。

- SAP NetWeaver–ABAP Collector の設定に使用する **r3config.xml** ファイル

- Oracle Collector の設定に使用する **oracle-config.xml** ファイル
- VMware Collector の設定に使用する **vmware-config.xml** ファイル
- TIBCO EMS Collector の設定に使用する **tibco-ems-config.xml** ファイル
- SQL Server Collector の設定に使用する **sqlserver-config.xml** ファイル

プレーンテキストのパスワードを入力し，パスワードの再入力で確認して，［**パスワードの暗号化**］ボタンを選択します。暗号化パスワードが表示されます。このページから，先頭の OBF: を含む暗号化パスワード全体をコピーして，該当するプロパティ・ファイルに貼り付けます（**r3config.xml**，**oracle-config.xml**，**vmware-config.xml**，**tibco-ems-config.xml**，または **sqlserver-config.xml**）。

注 : プレーンテキストのパスワード・プロパティを引き続き使用できます。

---

security.encrypted-password プロパティは，**collector.properties**，**dispatcher.properties**，および **server.properties** プロパティ・ファイルの mercury ユーザ・パスワードにも使用できます。mercury ユーザは，さまざまな Diagnostics コンポーネント間で認証に使用されます。次は，これらのプロパティ・ファイルの関連セクションのコピーです。

```
##########################################################################
# Remote Server Authentication Properties
##########################################################################

#
# This user name and password is used for communication between Diagnostics
# components (probes, and servers). You may want to change this password
# every so often to keep your system secure inside your enterprise. If you
# do change this password, you must first use
# http://< ホスト名 >:2006/security and select Encrypt Password to encrypt the
# password.
# Plaintext passwords can be used by replacing the security.encrypted-password
# with security.password. You must also change the encrypted password in the
# <install-dir>/etc/.htaccess file, as well as all the Diagnostics probe, and
# servers, that communicate with each other in your enterprise.
#
security.username=mercury
security.encrypted-password=OBF:1c431jg81hv41k1d1l161wu81z0d1pyl1wmt1n6h1y
m71n511wnd1pw11z0h1wu61kxw1jyl1hse1jd21c2z
```

Diagnostics Collector のインストールの確認

インストールが完了すると，Collector は自動的に実行を開始します。Collector のインストールを検証するには，**collector.log** ファイルでエラーを確認します。

Collector のプローブ・インスタンスが開始されたら，Diagnostics Enterprise UI を起動してプローブが動作していることを確認できます。
http:// ＜ Diagnostics コマンダ・サーバ＞ :2006/ に移動します。標準設定のユーザ / パスワード（**admin/admin**）を使用するか，異なるログイン情報が設定されている場合はそのログイン情報を使用できます。

または，システムの状況ビューで，Collector デプロイメントおよび Collector をホストするマシンに関する情報を確認することもできます。

システム・ビューにアクセスするには，次の手順を実行します。

1. http://＜Diagnostics コマンド・サーバ名＞:2006/query/ から Mercury System カスタマとして Diagnostics UI を開きます。
2. クエリ・ページで，リスト内から Mercury System カスタマを見つけて，Diagnostics を開くリンクを選択します。
3. Diagnostics にログインして，［アプリケーション］ウィンドウで［全組織］を選択し，Diagnostics ビューを開くためのリンクを選択します。
4. ［ビュー］表示枠にシステム・ビューのビュー・グループが表示されます。ビュー・グループを開き，システムの状況ビューまたはシステム・キャパシティ・ビューを選択します。

Diagnostics Collector の起動および停止

Windows マシンの場合

Windows マシンで Collector を起動するには，次の手順を実行します。

➤ ［**スタート**］>［**すべてのプログラム**］>［**HP Diagnostics Collector**］>［**Start HP Diagnostics Server**］を選択します。または，コマンド・ラインで **net start "HP Diagnostics Collector"** と入力します。

Windows マシンで Collector を停止するには，次の手順を実行します。

➤ ［**スタート**］>［**すべてのプログラム**］>［**HP Diagnostics Collector**］>［**Stop HP Diagnostics Collector**］を選択します。または，コマンド・ラインで **net stop "HP Diagnostics Collector"** と入力します。

UNIX マシンの場合（nanny を使用）

nanny は，デーモンとして実行して，Collector が常時作動するようにするプロセスです。次の手順では，nanny を使って Collector を起動および停止します。

UNIX マシンで Collector を起動するには，次の手順を実行します。

1 **M_LROOT** 環境変数が，Collector のルート・ディレクトリとして定義されていることを確認します。たとえば，ksh で次のように入力します。

```
export M_LROOT= < Collector のインストール・ディレクトリ > /nanny/solaris
```

M_LROOT 環境変数がルート・ディレクトリとして定義されていない場合，次のエラーが発生します。

```
Warning : MDRV: cannot find lrun root directory . Please check your M_LROOT
Unable to format message id [-10791]
m_agent_daemon ( is down )
```

2 ディレクトリを **$M_LROOT/bin** に変更します。

3 次の例のように，**-install** オプションを使って **m_daemon_setup** を実行します。

```
cd $M_LROOT/bin
./m_daemon_setup -install
```

UNIX マシンで Collector を停止するには，次の手順を実行します。

1 ディレクトリを上記の起動手順で設定した **$M_LROOT/bin** に変更します。

2 次の例のように，**-remove** オプションを使って **m_daemon_setup** を実行します。

```
cd $M_LROOT/bin
./m_daemon_setup -remove
```

UNIX マシンの場合（nanny を使用しない）

次の手順では，nanny を使わないで Collector を起動および停止します。

UNIX マシンで Collector を起動するには，次の手順を実行します。

➤ **< Collector のインストール・ディレクトリ > /bin/collector.sh** を実行します。

UNIX マシンで Collector を停止するには，次の手順を実行します。

➤ **kill** などのユーティリティを使ってプロセスを終了します。

Diagnostics Collector のバージョンの確認

サポートを依頼する際，Diagnostics Collector のバージョンを把握しておくと便利です。Collector のバージョン番号は，**＜Collector のインストール・ディレクトリ＞¥version.txt** で確認できます。

Diagnostics Collector のアンインストール

Collector をアンインストールするには，次の手順を実行します。

➤ Windows マシンの場合，［**スタート**］＞［**プログラム**］＞［**HP Diagnostics Collector**］＞［**Uninstall Diagnostics Collector**］を選択します。

または，**＜Collector のインストール・ディレクトリ＞¥_uninst** ディレクトリにある **uninstaller.exe** を実行することもできます。

➤ Linux または Solaris UNIX マシンの場合，**＜Collector のインストール・ディレクトリ＞/_uninst** ディレクトリにある **uninstall*** を実行します。

➤ それ以外の UNIX マシンの場合，1.5 以降の JVM を選択し，**java -jar ＜Collector のインストール・ディレクトリ＞ /_uninst/uninstall.jar** を実行して Collector をアンインストールします。

第 III 部

Java および .NET Agent のインストールおよびセットアップ

本項の内容

- Java Agent のインストール
- Java Agent で監視するためのアプリケー ション・サーバの準備
- Java Agent でクライアント監視のためのアプリケーション・サーバの準備
- .NET Agent のインストール

5

Java Agent のインストール

本項では，Java Agent のインストール方法について説明し，Java Agent のセットアップや設定の情報を示します。

本章の内容

Java Agent のインストールの概要

HP Diagnostics / TransactionVision Java Agent インストーラは, Diagnostics または TransactionVision のどちらかのために，または両方のためにデータを収集するために Java Agent をインストールします。TransactionVision は, Business Service Management のトランザクション管理アプリケーションにデータを提供します。TransactionVision 用に Java Agent をセットアップする方法の詳細については，Business Service Management 文書ライブラリの『TransactionVision Deployment Guid』を参照してください。

Agent は，監視するアプリケーションのホスト・マシンにインストールします。

HP Diagnostics で Java Agent を使用してアプリケーションを監視する前に，次を行う必要があります。

- Java Agent をインストールする。
- インストーラの後で自動的に起動する Java Agent セットアップ･モジュールを実行する。
- 次の手順では，アプリケーション・サーバによって使用される JRE をインストルメントし，Java Agent を呼び出すようにアプリケーション・サーバの JVM パラメータを設定します。アプリケーション・サーバによっては，JRE Instrumenter ユーティリティを手動で実行する代わりに自動 JRE インストルメンテーション・オプションを使用する場合があります。

  temp ディレクトリに対して約 400 MB の空き領域を使用可能にします。Java Agent のホスティングのための推奨システム設定については，34 ページ「Diagnostics Java Agent ホストの要件」を参照してください。Windows 用および複数の UNIX プラットフォーム用に Java Agent インストーラが用意されています。グラフィック・モード（Windows インストーラのようなインストール画面が表示される）またはコンソール・モードのコマンド・ライン・インタフェースでインストーラを実行できます。通常の UNIX インストーラを使用できない場合，154 ページ「標準のインストーラを使用した Java Agent のインストール」の説明に従って標準インストーラを使用できます。

重要：標準設定では，**<プローブのインストール・ディレクトリ> /log** ディレクトリは 777 に設定されます。これにより，Java Agent はユーザが実行している監視対象アプリケーションからメトリックスを収集できます。
組織のセキュリティ要件に応じて，このディレクトリへのアクセスをさらに制限する必要が生じることもあります。次に例を示します。
chmod 775 /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/log

注：ホスト・マシンにすでに Java Agent がインストールされている場合，Agent システムをアップグレードする方法に関する重要な説明については，851 ページ「アップグレードとパッチ・インストールの手順」を参照してください。

HP Software-as-a-Service（SaaS）：HP Diagnostics は HP Software-as-a-Service（SaaS）環境にデプロイできます。SaaS デプロイメントでは，Diagnostics Java Agent が企業の IT 環境にインストールされ，Diagnostics コマンダ・サーバとメディエータ・サーバが HP によって社内の SaaS システムにインストールされます。SaaS でホストされる Mediator が HP の施設にインストールされる場合，Java Agent のセットアップ中に Diagnostics の Agent を設定するためのオプションを選択します。

最初に Java Agent インストーラへのアクセスを参照してください。

Java Agent インストーラへのアクセス

Diagnostics インストール・ディスクから Java Agent をインストールすることも，実行可能インストール・ファイルをほかの場所にコピーして実行することも，または Business Service Management の Diagnostics［ダウンロード］ページからの Java Agent のインストールを選択することもできます。

Profiler trial software のみをインストールする場合は，HP ソフトウェア Web サイトからインストーラを起動します。

Diagnostics インストール元からインストーラにアクセスするには，次の手順を実行します。

➤ Windows の場合，Diagnostics インストール DVD（Autorun.exe）からインストール・メニュー・ページが表示されます。メニューで，[**Diagnostics Agent for Java**] を選択してインストーラを起動します。

または

➤ 適切なインストーラを直接実行できます。これを行うには，インストール元の **HPDiagTVJavaAgt_ ＜リリース番号＞ _ ＜プラットフォーム＞ .bin** ファイル（Unix の場合）または **HPDiagTVJavaAgt_ ＜リリース番号＞ _win.exe**（Windows の場合）を見つけ，これらのファイルを新しいインストール先にコピーし, .exe ファイルをダブルクリックするか, .bin インストーラを実行します。

135 ページ「Java Agent のインストール」に進みます。

HP ソフトウェアのダウンロード・センターからインストーラをダウンロードするには，次の手順を実行します。

1 HP ソフトウェア Web サイトのソフトウェア・ダウンロード・センターにアクセスします。

2 **Diagnostics**（または TransactionVision）のダウンロードを見つけ，Diagnostics Agent ソフトウェアをダウンロードするための適切なリンクを選択します。ダウンロード・センターを使用して，Diagnostics Profiler の試用版 / 評価版を入手することもできます。

3 Web サイトのダウンロード手順を行います。

135 ページ「Java Agent のインストール」に進みます。

Business Service Management の Diagnostics [ダウンロード] ページからインストーラをダウンロードするには，次の手順を実行します。

1 Business Service Management のメイン・メニューから，[**管理**]>[**Diagnostics**] を選択し，[**ダウンロード**] タブをクリックします。

2 [ダウンロード] ページで，リンクをクリックして適切な Java Agent インストーラをダウンロードします。

注 : Java Agent インストーラが Business Service Management で利用できるのは，インストーラが Business Service Management のアクセスできるディレクトリにある場合のみです。Diagnostics サーバのインストール時にこれを行うか，または Java Agent インストーラをインストール・ディスクから必要な場所に手動でコピーできます。

Java Agent のインストールに進みます。

Java Agent のインストール

本項では，Java Agent を Windows または UNIX システムに初めてインストールする場合の詳細な手順について説明します。

重要 : ホスト・マシンにすでに Java Agent がインストールされている場合，次のインストール手順ではなく Agent システムのアップグレード手順に従う必要があります。詳細については，851 ページ「アップグレードとパッチ・インストールの手順」を参照してください。

ほかのタイプのインストールの詳細については，次を参照してください。

- z/OS については，152 ページ「z/OS メインフレームへの Java Agent のインストール」を参照してください。
- 標準インストーラを使用したインストールについては，154 ページ「標準のインストーラを使用した Java Agent のインストール」を参照してください。

- サイレント・インストールの詳細については，155 ページ「Java Agent のサイレント・インストール」を参照してください。
- 必要に応じて，UNIX インストーラ・ファイルのモードを変更して実行可能にします。UNIX コマンドの詳細については，881 ページ「UNIX コマンドの使用」を参照してください。
- コンソール・モードでインストーラを実行するには，コマンド・プロンプトで次のように入力します。

```
./ <インストーラ> -console
```

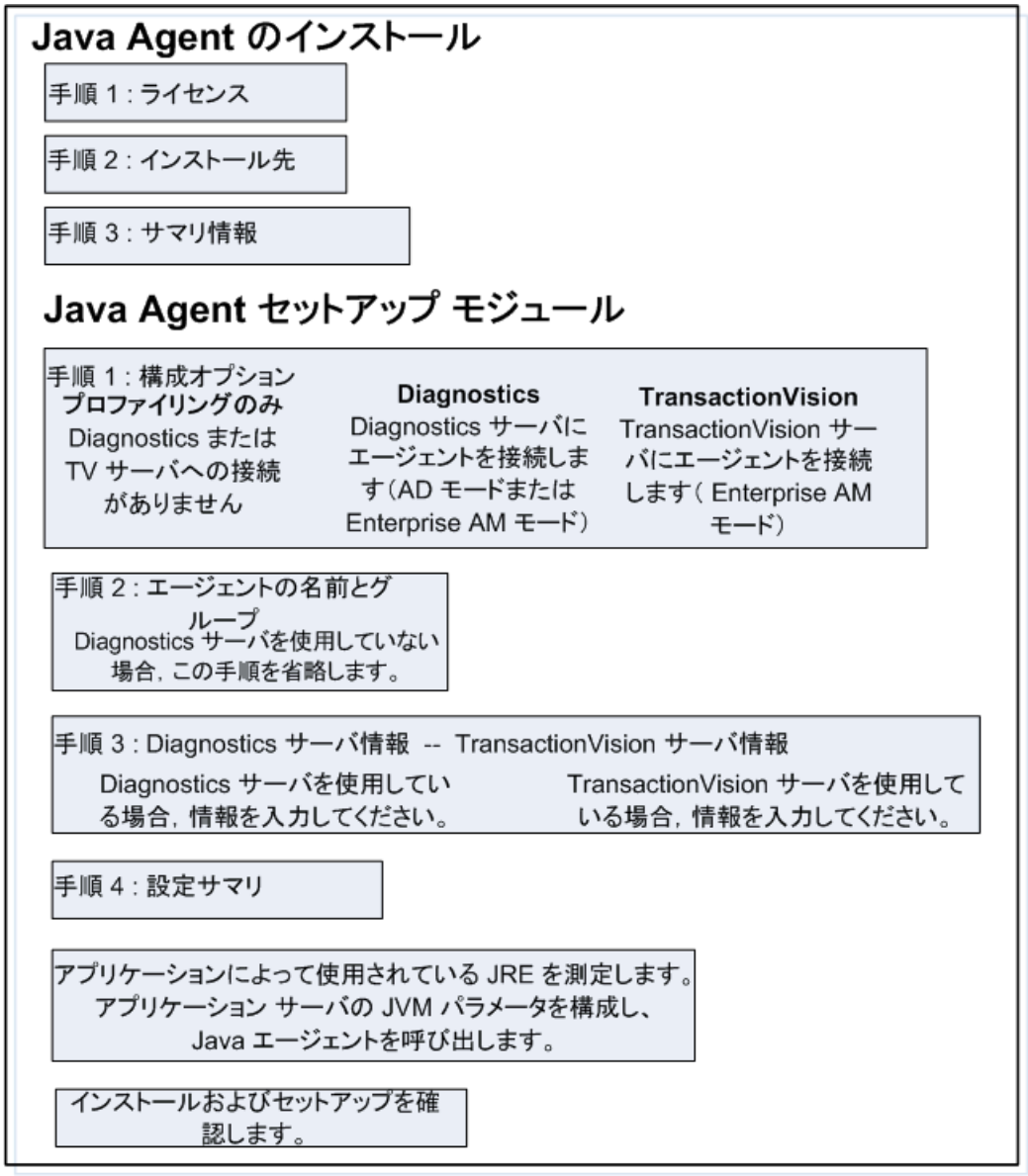
インストーラに，コンソール・モードでインストール・プロンプトが表示されます。

- グラフィック・モードでインストーラを実行するには，コマンド・プロンプトで次のように入力します。

```
./< installer>
```

インストーラには，Windows インストーラのものと同じ画面が表示されます。

下の図に，Java Agent のインストール手順の概要を示します。各手順の詳細については，本項のその後の内容を参照してください。



138 ページ「手順 1. エンド・ユーザ使用許諾契約書」で，Java Agent のインストールを開始します。

手順 1. エンド・ユーザ使用許諾契約書

エンド・ユーザ使用許諾契約書に同意します。

使用許諾契約書を読み，［**使用条件の条項に同意します**］を選択します。

コンソール・モード・インタフェースで，［次へ］を選択せずに **Enter** キーを押すと次のページに進み，q を入力すると使用許諾契約書の末尾に移動します。

［**次へ**］（コンソール・モードでは **Enter** キー）を選択して，次の手順に進みます。

手順 2. インストール先の指定

Agent のインストール先を指定します。

標準設定のインストール・ディレクトリを受け入れるか，ほかの場所を指定します。ほかの場所を指定するには，［**インストール先ディレクトリ名**］ボックスにインストール・ディレクトリのパスを入力するか，または［**参照**］をクリックしてインストール・ディレクトリを指定します。

コンソール・モード・インタフェースの［**インストール先ディレクトリ名**］プロンプトで，括弧内に表示される標準設定のインストール先を受け入れか，またはほかの場所を入力します。

注 : この場所が**＜プローブのインストール・ディレクトリ＞**になります。標準設定では，Windows の場合は **C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent**，UNIX の場合は **/opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent** です。

エラーが発生した場合，JavaAgent ディレクトリがすでに存在しているかどうかを確認して削除します。これは，以前に Java Agent をインストールし，JavaAgent サブディレクトリを削除せずにアンインストールした場合に発生することがあります。

［**次へ**］（コンソール・モードでは **Enter** キー）を選択して，次の手順に進みます。

手順 3. インストール前のサマリ情報の確認

インストールのサマリ情報を確認します。

インストール先ディレクトリとサイズの要件がリストされます。

これらが問題なければ，[**次へ**]（コンソール・モードでは **Enter** キー）を選択してインストールを開始します。インストールには数分かかることがあります。

インストールが完了したら，Java Agent セットアップ・モジュールが起動します。Java Agent セットアップ・モジュールの実行の次の項に進みます。

Java Agent セットアップ・モジュールの実行

Java Agent は，Diagnostics サーバに接続しない Profiler（または Diagnostics サーバおよび TransactionVision Server と連携する Agent）として設定できます。Agent が Profiler 専用として初期設定されている場合，後で Java Agent セットアップ・モジュールを再実行して Diagnostics サーバと連携するように Agent を設定できます。

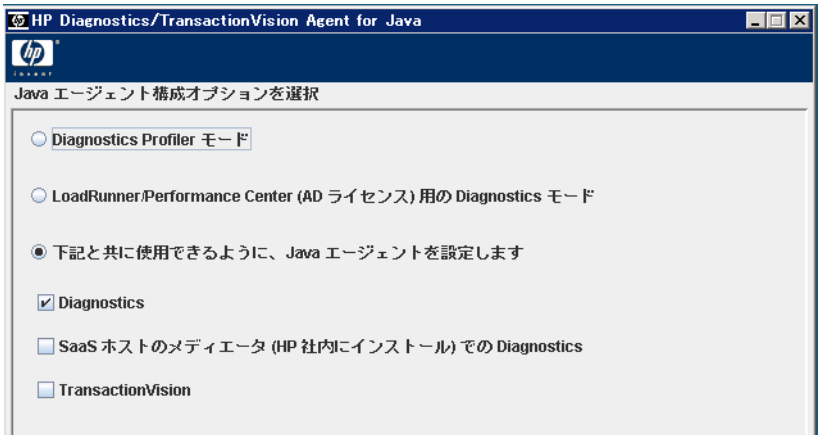
Java Agent のインストール後に自動的に起動する Java Agent セットアップ・モジュールを使用して Java Agent を設定します。また，これは［**スタート**］>［**すべてのプログラム**］>［**HP Java Agent**］>［**Setup Module**］を選択すればいつでも起動できます。UNIX の場合，**＜プローブのインストール・ディレクトリ＞/bin/setupModule.sh** を実行すればいつでも起動できます。

Java Agent セットアップ・モジュールには次の手順があり，まずは手順 1. 構成オプションを選択します。

- 140 ページ「手順 1. 構成オプション」
- 142 ページ「手順 2. Agent の名前とグループ」
- 143 ページ「手順 3. Diagnostics サーバ情報」
- 147 ページ「手順 4. ポスト・セットアップのサマリ」

手順 1. 構成オプション

Java Agent をサーバに接続しないスタンドアロンの Profiler としてインストールするか（Diagnostics Java Profiler の評価版ソフトウェアをインストールする場合など），LoadRunner/Performance Center や Diagnostics および TransactionVision Server と連携するように Agent をインストールするかを指定します。



Agent を使用する環境に適した項目を選択します。

- **Application Management/Enterprise モード（AM ライセンス）**：このオプションは，エンタープライズ（または実運用）環境の Diagnostics サーバや TransactionVision Server で使用するために Agent をインストールする場合に選択します。

  Agent を設定する対象を指定します。

  - （ローカルにインストールされている）Diagnostics サーバまたは HP の施設にある HP SaaS システムでホストされる Diagnostics サーバのいずれか
  - TransactionVision Server
  - ローカルにインストールされている Diagnostics サーバと TransactionVision Server の両方

  HP SaaS モードを選択した場合，HP SaaS でホストされる Diagnostics メディエータ・サーバに Java Agent を接続するための情報が HP SaaS 管理者から提供されます。

  TransactionVision を選択した場合，TransactionVision 固有のセットアップ・オプションの詳細については，Business Service Management 文書ライブラリの『HP TransactionVision Deployment Guid』を参照してください。

Diagnostics サーバを選択した場合,［Application Management/Enterprise モード (AM ライセンス)］オプションでは **etc/probe.properties** ファイルの **active.properties** プロパティの値が **Enterprise** モードに設定されます。Java Agent のインストール時に TransactionVision Server を選択すると，**TV** モードに設定されます（461 ページ「アクティブ・プロダクト・モードの設定」を参照）。

Enterprise モードが設定された Agent は, HP Diagnostics AM ライセンス・キャパシティに対してカウントされます。

➤ Diagnostics サーバに接続しない Diagnostics Java Profiler として Agent を設定するには,［**Diagnostics Profiler モード**］を選択します。［Diagnostics Profiler モード］は, 一般的に HP Diagnostics 製品を購入する前に Diagnostics Java Profiler の評価版ソフトウェアをインストールする場合に使用します。

［Diagnostics Profiler モード］を選択すると，Java Agent のインストール時に **etc/probe.properties** ファイルの **active.products** プロパティの値が **PRO** モードに設定されます（461 ページ「アクティブ・プロダクト・モードの設定」を参照）。

［Diagnostics Profiler モード］を選択する場合，ほかの設定オプションはないため，［**完了**］を選択して設定を完了できます。これにより，ポスト・セットアップのサマリ・ダイアログが表示されます。

➤ **LoadRunner/Performance Center（AD ライセンス）用の Diagnostics モード :** このオプションは，負荷テスト（または実稼動前）環境の Diagnostics サーバで使用するために Agent をインストールする場合に選択します。この場合, プローブが LoadRunner または Performance Center の実行でのみ使用されます。

Agent は AD ライセンス・モードでインストールされます。つまり，Agent が LoadRunner または Performance Center のテスト実行に含まれる場合にのみ, HP Diagnostics AD ライセンス・キャパシティに対して Agent がカウントされます。AD ライセンス・キャパシティの詳細については，83 ページ「現在接続されているプローブに基づくライセンス情報」を参照してください。

AD モードの場合, Agent は LoadRunner または Performance Center で実行しているときのみデータをキャプチャします。結果は，Default Client:21 など，この実行のための専用 Diagnostics データベースに格納されます。AD モードでは, プローブが LoadRunner または Performance Center の実行の一部である場合を除いて，Agent はサーバにデータを送信しません。

この AD ライセンス・オプションを選択すると，Java Agent のインストール時に **etc/probe.properties** ファイルの **active.properties** プロパティの値が **AD** モードに設定されます（461 ページ「アクティブ・プロダクト・モードの設定」を参照）。

AD モードでプローブを実行する利点は, AD モードのプローブが LoadRunner または Performance Center のテスト実行に含まれる場合にのみライセンス・キャパシティに対してプローブがカウントされるという点にあります。たとえば, 20 個のプローブが LoadRunner/Performance Center AD モードでインストールされていても，一度に 5 個のみが実行に含まれる場合は 5 個のプローブの AD ライセンス・キャパシティだけが必要になります。

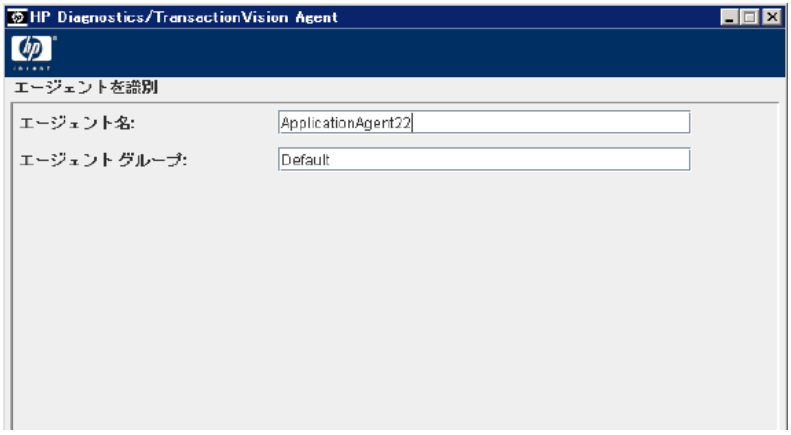
コンソール・モード・インタフェースで，X と入力してインストールのモードを選択します。

［**次へ**］（コンソール・モードでは **Enter** キー）を選択して，次の手順に進みます。

手順 2. Agent の名前とグループ

Agent から Diagnostics サーバにレポートしない場合，この手順はスキップします。

Java Agent に名前を割り当て，それが属するグループを指定します。



➤ Java Agent 名には, HP Diagnostics 内で Agent を一意に特定する名前を入力します。名前には，-，_，すべての英数字を使用できます。Agent 名は標準設定のプローブ・エンティティ名になるように割り当てられます。システムに 1 つの Agent しかインストールしておらず，しかも複数のアプリケーション・サーバまたはアプリケーション・ドメインを監視する予定の場合，後で，監視対象アプリケーションごとに一意のプローブ名を設定できます。

Agent に名前を割り当てる際は，監視されるアプリケーションと Agent がインストールされるシステムを簡単に識別できる名前を選択します（たとえば，WebLogic アプリケーション・サーバのあるシステム ovrserver130 に Agent をインストールする場合，WL10_MedRec_ovrserver130 という Agent 名を使用できます)。

➤ Java Agent グループ名に対しては，既存のグループの名前か新規作成するグループの名前を入力します。Agent グループ名は大文字と小文字を区別します。Agent グループ名は，プローブ・グループ名として使用されます。

プローブ・グループは，同じ Diagnostics サーバに報告するプローブの論理グループです。プローブ・グループのパフォーマンス測定値は追跡され，さまざまな Diagnostics ビューに表示できます。

たとえば，グループ・レベルのパフォーマンスと個別のプローブ・エンティティに基づいたパフォーマンスの両方をを監視するために，1 つのプローブ・グループに特定のエンターラプライズ・アプリケーションのすべてのプローブを割り当てることができます。

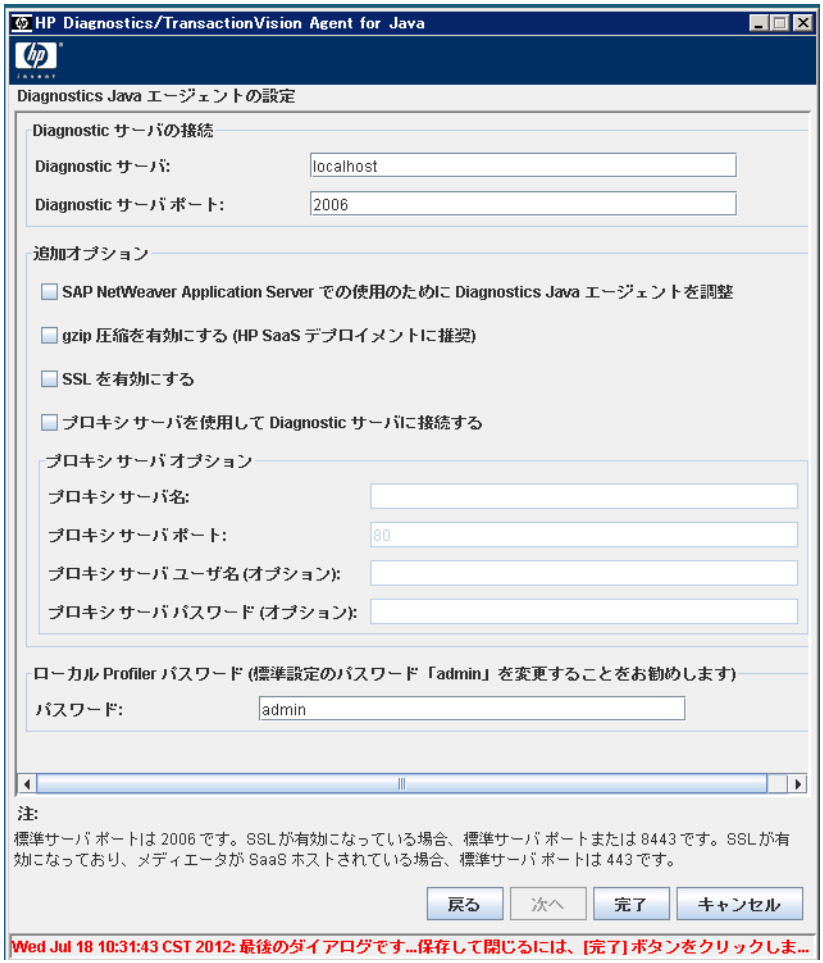
［**次へ**］（コンソール・モードでは **Enter** キー）を選択して，次の手順に進みます。

手順 3. Diagnostics サーバ情報

Agent から Diagnostics サーバにレポートしない場合，この手順はスキップします。

Diagnostics サーバの設定情報と追加オプションを入力します。

コンソール・モード・インタフェースで，オプションごとに **X**（はい）および **O**（いいえ）を入力します。



[**Diagnostics サーバ**]ボックスに，この Agent の接続先となる Diagnostics サーバホストのホスト名と IP アドレスを入力します。単純なホスト名ではなく，**完全修飾ホスト名**を指定する必要があります。OS が混在し，その 1 つが UNIX の環境で，これは正確なネットワーク・ルーティングのために不可欠です。

コマンダ・サーバ : Agent が実行する Diagnostics デプロイメントに Diagnostics サーバが 1 台だけある場合は，Diagnostics コマンダ・サーバのホスト名とポート情報を入力します。

メディエータ・サーバ : コマンダ・サーバとメディエータ・サーバのある分散環境では，Agent からデータを受信する Diagnostics メディエータ・サーバの情報を入力します。

HP Software-as-a-Service（SaaS）を使用する場合，Diagnostics Mediator は HP によって HP の施設にある HP SaaS システムにインストールされます。使用するホスト名およびポートの情報は，HP SaaS 管理者によって提供されます。また，HP SaaS 環境では，［gzip を有効にする］オプションが自動的にオンになります。[SSL を有効にする]オプションは HP の施設の Diagnostics Commander/ Mediator で設定されるため表示されません。

➤ ［**Diagnostics サーバ ポート**］ボックスに，Diagnostics サーバのポート番号を入力します。

Diagnostics サーバの標準設定のポートは **2006** です。サーバと SSL 接続する場合，ローカルにインストールされているサーバのポートは一般的に **8443** に設定されます。

SaaS 環境に Agent をインストールする場合，標準設定ポートは **443** です（詳細情報は，SaaS 管理者から提供される）。

Diagnostics サーバをインストールした後にポートを変更した場合は，標準設定ポートではなく，新しいポート番号を指定する必要があります。

➤ この Agent が SAP NetWeaver アプリケーション・サーバをサポートできるようにするには，［**SAP NetWeaver Application Server での使用のために、Diagnostics Java エージェントを調整**］チェック・ボックスを設定します。

➤ Java Agent と Mediator 間でデータを圧縮する必要がある場合，［**gzip 圧縮を有効にする**］チェックボックスを設定します。これは，帯域幅とプローブのパフォーマンス・オーバーヘッドのトレードオフです。通常，HP SaaS 環境では gzip 圧縮を有効にするかどうかを尋ねられます。詳細については，SaaS 管理者にお問い合わせください。

- ［SSL を有効にする］がオンになっている場合または Diagnostics サーバが HP の施設の SaaS でホストされる場合, Java Agent と Diagnostics サーバは SSL 経由で接続されます。［**SSL を有効にする**］チェックボックスをオンにすると, Agent が SSL モードで Diagnostics サーバに接続して必要な証明書チェーンをサーバからダウンロードします。その結果，**server.properties** 信頼済み証明書にその証明書が含まれるようになります。セキュアな通信の詳細については, 797 ページ「コンポーネント間での HTTPS 有効化」を参照してください。
- プロキシ・サーバを使用して Diagnostics メディエータ・サーバと通信する場合，［**プロキシ サーバを使用して Diagnostic サーバに接続する**］チェックボックスをオンにして適切なオプションを入力します。HP SaaS 環境では，会社で外部のサーバへの通信にプロキシが必要になる場合にこのオプションを選択します。これらのオプションは，Agent システムの **dispatcher.properties** ファイルで設定することもできます。これを行うには，proxy.enabled を true に設定してその他のオプションを入力します。627 ページ「HTTP プロキシ用の Diagnostics サーバおよび Agent の設定」を参照してください。

  プロキシ・サーバのオプション：

  - **プロキシ・サーバ名**：プロキシ・サーバのホスト名。
  - **プロキシ・サーバ・ポート**：プロキシ・サーバのポート。
  - **プロキシ・サーバ・ユーザ名（オプション）**：プロキシ・サーバの認証に使用するユーザ。
  - **プロキシ・サーバ・パスワード（オプション）**： プロキシ・サーバの認証に使用するパスワード。
- ［**ローカル Profiler パスワード**］を標準設定のパスワード（admin）から変更することをお勧めします。

TransactionVision の情報

この Agent を TransactionVision 用に設定する場合，TransactionVision の Agent を設定するための追加のダイアログ・ボックスが表示されます。これらのインストール・オプションの詳細については，『HP TransactionVision Deployment Guid』を参照してください。

セットアップ・プロセスの開始

Java Agent セットアップ・プロセスが開始します。グラフィック・モードでは，進行状況バーに設定の進行状況が表示されます。

Diagnostics サーバへの接続性がテストされます。接続の問題が生じた場合, セットアップ・プログラムは接続性チェックの結果を通知します。

次の手順に進みます。

手順 4. ポスト・セットアップのサマリ

ポスト・セットアップのサマリを確認して［**OK**］をクリックします。

アプリケーション・サーバによっては，JRE Instrumenter ユーティリティを手動で実行する代わりに自動 JRE インストルメンテーション・オプションを使用します。そのため，JRE Instrumenter ユーティリティを実行するためのチェック・ボックスは標準設定で空白のままになっています。続行するには，監視用のアプリケーション・サーバの準備についてを参照してください。

監視用のアプリケーション・サーバの準備について

次の手順では，アプリケーション・サーバによって使用される JRE をインストルメントし，Java Agent を呼び出すようにアプリケーション・サーバの JVM パラメータを設定します。

アプリケーション・サーバによって使用される JRE をインストルメントする方法や，特定のアプリケーション・サーバで Java Agent を呼び出すように JVM パラメータを設定する方法については，161 ページ「Java Agent で監視するためのアプリケー ション・サーバの準備」の指示に従ってください。

Java Agent で監視できるようにアプリケーション・サーバを準備したら，アプリケーション・サーバを再起動します。Java Agent が呼び出されてアプリケーションの監視が開始されます。

クライアント監視の詳細については，第 7 章，「Java Agent でクライアント監視のためのアプリケーション・サーバの準備」を参照してください。

Diagnostics サーバへの Agent の登録

Agent を設定して，Diagnostics コマンダ・サーバに登録します。

Diagnostics コマンダ・サーバの機能の 1 つでは，Diagnostics コンポーネントを追跡し，それらコンポーネント間の通信を促進し，コンポーネントの状態や状況に関する情報をに知らせ続けます。

Agent を設定して Diagnostics サーバに登録するには，プロパティ・ファイル **＜プローブのインストール・ディレクトリ＞ ¥etc¥dispatcher.properties** の **registrar.url** プロパティを使用してホスト名とポートを設定します。

次は，**registrar.url** プロパティを示す **dispatcher.properties** の抜粋です。

```
## the URL of the registrar
registrar.url=http://host01.company.com:2006/registrar/
```

Java Agent のインストールの検証

Agent は，起動するまで Diagnostics サーバに登録されません。プローブは，インストルメント対象のアプリケーション・サーバが起動したときに立ち上がります。そのため，アプリケーション・サーバによって使用される JRE をインストルメントし，Java Agent を呼び出すようにアプリケーション・サーバの JVM パラメータを設定するまで，Agent が正常に動作しているかどうかを確認できません。

Java プローブ・インスタンスが起動したら，Diagnostics Enterprise UI を起動してプローブが動作しているかどうかを確認できます。
http:// ＜ Diagnostics コマンダ・サーバ＞ :2006/ に移動します。標準設定のユーザ / パスワード（**admin/admin**）を使用するか，異なるログイン情報が設定されている場合はそのログイン情報を使用できます。

また，［システムの状況］ビューを確認して，Java Agent デプロイメントおよびそれらをホストするマシンに関する情報を確認することもできます。

システム・ビューにアクセスするには，次の手順を実行します。

1. http://＜Diagnostics コマンド・サーバ名＞:2006/query/ から Mercury System カスタマとして Diagnostics UI を開きます。
2. クエリ・ページで，リスト内から Mercury System カスタマを見つけて，Diagnostics を開くリンクを選択します。
3. Diagnostics にログインして，［アプリケーション］ウィンドウで［全組織］を選択し，Diagnostics ビューを開くためのリンクを選択します。
4. ［ビュー］表示枠にシステム・ビューのビュー・グループが表示されます。ビュー・グループを開き，システムの状況ビューまたはシステム・キャパシティ・ビューを選択します。

また，**＜プローブのインストール・ディレクトリ＞ ¥log¥ ＜ Probe ID ＞ ¥probe.log** ファイルでエントリを確認することもできます。ファイルにエントリがない場合は，JRE をインストルメントしていないか，**Xbootclasspath** などの Java パラメータを正しく入力しなかったかのいずれかです。**probe.log** ファイルで，エラーを探すか，プローブとサーバ間の通信が確立されていることを示す「Successfully downloaded first command」と表示されているエントリを探します。

次の項に進み，インストール / セットアップ後の作業を行います。

追加設定およびカスタム・インストルメンテーションについて

実行できる追加の設定やオプションのカスタム・インストルメンテーションがあります。次を参照してください。

- 150 ページ「SOAP メッセージ・ハンドラの設定」
- 151 ページ「Java システム・プロパティとしてのプローブ・プロパティの設定」
- 151 ページ「任意の詳細設定」
- 152 ページ「任意のカスタム・インストルメンテーション」
- プロキシを使用した環境での設定については 627 ページ「HTTP プロキシ用の Diagnostics サーバおよび Agent の設定」を，ファイアウォールについては 631 ページ「ファイアウォール環境で動作するための Diagnostics の設定」を，HTTPS の有効化については 797 ページ「コンポーネント間での HTTPS 有効化」を参照してください。

SOAP メッセージ・ハンドラの設定

Diagnostics SOAP メッセージ・ハンドラは，次の機能をサポートするために必要です。

- SOAP 失敗のペイロードを収集する。
- SOAP ヘッダ, ボディまたはエンベロープから SOA コンシューマ ID を確認する。

  ほとんどのアプリケーション・サーバでは，インストルメンテーション・ポイントとコード・スニペットが書き込まれ，自動的に監視対象の Web サービスの Diagnostics ハンドラが設定されます。

WebSphere 5 JAX-RPC と Oracle 10g JAX-RPC の場合は，手動ステップにより SOAP ハンドラを設定する必要があります。詳細については，237 ページ「Diagnostics SOAP メッセージ・ハンドラの読み込み」を参照してください。

Java システム・プロパティとしてのプローブ・プロパティの設定

dynamic.properties プロパティ・ファイルで定義されたものを除いて，プローブ・プロパティはすべて，アプリケーション・サーバの起動コマンド・ラインで Java システム・プロパティとして指定できます。これは，1 つのプローブ・インストールを使用する JVM が複数ある場合に非常に便利です。

プロパティを Java システム・プロパティとして指定するには，プロパティ名の先頭に - およびプロパティ・ファイル名の最初の部分を付けます。次の例でこれについて説明します。

➤ 起動コマンドから，**probe.properties** に **id** プロパティを設定するには，次のように - とプロパティ・ファイル名の**プローブ**を連結して，指定しているプロパティの名前，つまり **id** に付加します。

```
-Dprobe.id=SomeId
```

➤ 起動コマンドから，**probe.properties** に **active.products** プロパティを設定するには，次のように - とプロパティ・ファイル名の**プローブ**を連結して，指定しているプロパティの名前，つまり **id** に付加します。

```
-Dprobe.active.products=Enterprise,TV
```

➤ 起動コマンドから，**dispatcher.properties** に **registrar.url** プロパティを設定するには，次のように - とプロパティ・ファイル名の **dispatcher** を連結して，指定しているプロパティの名前，つまり **registrar.url** に付加します。

```
-Ddispatcher.registrar.url=http:/</host01.company.com>:2006/commander/registrar
```

任意の詳細設定

環境に適用するプローブの詳細設定を確認します。第 12 章，「Java Agent およびアプリケーション・サーバの詳細構成」を参照してください。

任意のカスタム・インストルメンテーション

必要に応じてカスタム・インストルメンテーションを設定することもできます。詳細については，第 9 章，「Java アプリーションのカスタム・インストルメンテーション」を参照してください。

z/OS メインフレームへの Java Agent のインストール

本項では，Diagnostics インストール・ディスクに含まれる .tgz ファイルから Java Agent をインストールする方法について説明します。

z/OS 環境で Java Agent をインストールして Java Agent として設定する場合は，次のことを考慮してください。

- Diagnostics Java Agent を z/OS にインストールして，z/OS の Unix System Services 環境（USS）を大いに利用します。
- z/OS 環境にインストールすると，Java Agent は，Diagnostics プロパティ・ファイルが ASCII ではなく EBCDIC 形式であると予測します。EBCDIC エディタを使って，プロパティ・ファイルを更新し，更新ファイルを同じ形式で保存します。
- システム・メトリックスは，z/OS に対してキャプチャされません。Diagnostics Java Agent は，制限された数のシステム・レベル・メトリックスをキャプチャするように設定できます。

  z/OS におけるシステム・メトリックスのキャプチャについては，
  677 ページ「z/OS システム測定値のキャプチャの有効化」を参照してください。

Diagnostics インストール・ディスクから z/OS への Java Agent のインストール

Java Agent ファイルが含まれる .tgz ファイルは, Diagnostics インストール・ディスクにあり, z/OS メインフレームに Java Agent をインストールするために使用されます。

z/OS メインフレームに Java Agent をインストールするには，次の手順を実行します。

1 HP Diagnostics インストール・ディスクの **Diagnostics_Installers** フォルダから，インストーラを展開する z/OS システムのディレクトリに **HPDiagTVJavaAgt_＜リリース番号＞ _zos.tgz** をアップロードします。

2 次の例のように gzip を使って **HPDiagTVJavaAgt_ ＜リリース番号＞ _zos.tgz** を展開します。

```
gzip -d HPDiagTVJavaAgt_9.10_zos.tgz
```

このコマンドでは, 展開ファイル **HPDiagTVJavaAgt_＜リリース番号＞_zos.tar** が作成されます。

3 .tar ファイルを展開するには，次の例のように tar コマンドを実行します。

```
tar -xfp HPDiagTVJavaAgt_9.10_zos.tar
```

このコマンドでは，展開ディレクトリ **JavaAgent** が作成されます。

4 Java 実行可能ファイルがパス上に存在していることを確認し，Java Agent セットアップ・モジュールを実行して Profiler 専用として Java Agent を設定するか, Diagnostics サーバおよび TransactionVision 処理サーバと連携するように Java Agent を設定します。詳細については，139 ページ「Java Agent セットアップ・モジュールの実行」を参照してください。例 (各自の shell に対応必要) :

```
setenv PATH /u/Java6_31/J6.0/bin:/bin
```

例 :

```
＜プローブのインストール・ディレクトリ＞ /bin/setupModule.sh
```

5 Agent をインストールしてセットアップ・モジュールを実行したら，アプリケーション・サーバで使用される JRE をインストルメントし，Java Agent を呼び出すようにアプリケーション・サーバの JVM パラメータを設定する必要があります。第 6 章，「Java Agent で監視するためのアプリケー ション・サーバの準備」を参照してください。

6 149 ページ「Java Agent のインストールの検証」の説明に従って，Agent のインストールを確認します。

7 必要に応じてインストール後の設定を行います。詳細については，150 ページ「追加設定およびカスタム・インストルメンテーションについて」を参照してください。

複数の z/OS マシンへの Java Agent のインストール

複数の z/OS マシンに Java Agent をインストールする場合，最初のマシンに Agent 実装の pax アーカイブを作成し，その pax アーカイブを使ってほかのマシンに Agent をインストールすることもできます。詳細については，システム管理者にお問い合わせください。

標準のインストーラを使用した Java Agent のインストール

Java Agent のインストーラは，コンポーネントが認定されたすべてのプラットフォームへの Agent のインストールをサポートするように構築されました。ただし，認定されていないほかのプラットフォームで Agent が機能することがあります。標準インストーラは製品インストール・ディスクに収録されており，それらの認定されていないプラットフォームに Agent をインストールできます。

標準インストーラでサポートされていないプラットフォームで Agent を機能させるには，標準インストーラを実行し，ほかの Diagnostics コンポーネントと通信して，アプリケーションの処理を監視できるように Agent を Java プローブとして手動で設定します。

認定されていないプラットフォームに Java Agent をインストールおよび設定するには，次の手順を実行します。

1 HP Diagnostics インストール・ディスクの **Diagnostics_Installers** フォルダで **HPDiagTVJavaAgt_ ＜リリース番号＞ _unix.tgz** を見つけます。

2 次の例のように gzip を使って **HPDiagTVJavaAgt_＜リリース番号＞__unix.tgz** を展開します。

```
gzip -d HPDiagTVJavaAgt_9.10_unix.tgz
```

このコマンドが完了すると，**HPDiagTVJavaAgt_ ＜リリース番号＞ _unix.tar** という展開ファイルが作成されます。

3 tar ファイルを展開するには，次の tar コマンドを実行します。

```
tar -xfp HPDiagTVJavaAgt_9.10_unix.tar
```

このコマンドでは，展開ディレクトリ **JavaAgent** が作成されます。

4 Java Agent セットアップ・モジュールを実行して Profiler 専用として Java Agent を設定するか，Diagnostics サーバおよび TransactionVision 処理サーバと連携するように Java Agent を設定します。詳細については，139 ページ「Java Agent セットアップ・モジュールの実行」を参照してください。

```
＜プローブのインストール・ディレクトリ＞ /bin/setupModule.sh
```

5 Agent をインストールしてセットアップ・モジュールを実行したら，アプリケーション・サーバで使用される JRE をインストルメントし，Java Agent を呼び出すようにアプリケーション・サーバの JVM パラメータを設定する必要があります。第 6 章，「Java Agent で監視するためのアプリケー ション・サーバの準備」を参照してください。

6 149 ページ「Java Agent のインストールの検証」の説明に従って，Agent のインストールを確認します。

7 詳細については，150 ページ「追加設定およびカスタム・インストルメンテーションについて」を参照してください。

Java Agent のサイレント・インストール

Java Agent のサイレント・インストールがサポートされています。サイレント・インストールは，ユーザが操作することなく自動的に実行されます。サイレント・インストールでは，ユーザ入力の代わりに，各インストール手順の応答ファイルからの入力を使用します。

複数のマシン上でサイレント・インストールを実行する前に，インストール中に入力を提供する応答ファイルを作成する必要があります。この応答ファイルは，インストール時に同じ入力を必要とするすべてのサイレント・インストールで使用できます。

重要 : Diagnostics の新しいリリースごとに，複数のマシン上でサイレント・インストールを実行する前に応答ファイルを記録しなおす必要があります。

応答ファイルには拡張子 **.rsp** が付いています。標準的なテキスト・エディタで応答ファイルを編集できます。

サイレント・インストールは 2 つの応答ファイルを使用します。1 つは Java Agent インストール用，もう 1 つは Java Agent セットアップ・モジュール用です。

Agent のインストール用に応答ファイルを作成するには，次の手順を実行します。

➤ 次のコマンド・ライン・オプションを使って通常のインストールを実行します。

```
＜インストーラ＞ -options-record ＜インストール用応答ファイル名＞
```

＜インストール用応答ファイル名＞は完全修飾ファイルです。これにより，インストール中に送信されたすべての情報が含まれる応答ファイルが作成されます。

Java Agent セットアップ・モジュールの応答ファイルを作成するには，次の手順を実行します。

➤ 次のコマンドライン・オプションを使用して Java Agent セットアップ・モジュールを実行します。

Windows の場合 :

```
＜プローブのインストール・ディレクトリ＞ \bin\setupModule.cmd  -createBackups
-console  -recordFile<>
   <JASMResponseFileName>
```

UNIX の場合 :

```
＜プローブのインストール・ディレクトリ＞ /bin/setupModule.sh  -createBackups
-console  -recordFile<>
   <JASMResponseFileName>
```

＜ JASM 用応答ファイル名＞は完全修飾ファイルです。どちらのコマンドでも，セットアップ中に送信されたすべての情報が含まれる応答ファイルが作成されます。

Java Agent のサイレント・インストールを実行するには，次の手順を実行します。

➤ Java agent インストール応答ファイルを使って，サイレント・インストールを実行します。

まず環境変数を設定し，次のコマンド・ライン・オプション **-silent** を使ってインストーラを実行します。

```
set HP_JAVA_AGENT_SETUP=-DoNotRun
<installer>  -silent  -options <installResponseFileName>
```

UNIX システムでは，環境変数を指定する場合に引用符で囲みます。

```
set HP_JAVA_AGENT_SETUP="-DoNotRun"
```

Java Agent セットアップ・モジュールを使用してサイレント設定を実行するには，次の手順を実行します。

➤ Java agent セットアップ・モジュール応答ファイルを使って，サイレント・インストールを実行します。

環境変数を設定解除し，次の **-silent** コマンドライン・オプションを使って Java Agent セットアップ・モジュールを実行します。

```
set HP_JAVA_AGENT_SETUP=
<setupModule>  - silent  -createBackups  -console  -installFile <JASMResponseFileName>
```

UNIX システムでは，空の引用符を使用して環境変数を設定解除します。

```
set HP_JAVA_AGENT_SETUP=""
```

サイレント・インストールを実行するとき，応答ファイル名の後に 2 つの追加オプションを指定するには，次の手順を実行します。

➤ **is:log ＜ログ・ファイルのパス＞**オプションを指定して，ログ・ファイルを作成します。

➤ **is:tempdir <tempDirPath>** オプションを指定して，temp ディレクトリをユーザ指定ディレクトリに変更できます。

ファイル権限の設定（UNIX 専用）

Java Agent のインストール後，Agent の「グループ」をアプリケーション・サーバの「グループ」と同じにします（UNIX 専用）。プローブのインストール・ディレクトリにあるグループのファイルに次の権限を割り当てます。

- **＜プローブのインストール・ディレクトリ＞**ディレクトリおよびファイルへの読み取りアクセス。
- **＜プローブのインストール・ディレクトリ＞ /bin** ディレクトリへの実行アクセス。
- **＜プローブのインストール・ディレクトリ＞ /log** ディレクトリへの読み取り / 書き込みアクセス。

組織のセキュリティ要件に応じて，このディレクトリへのアクセスをさらに制限することもできます。次に例を示します。

```
chmod 775 /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/log
```

Java Agent のバージョン確認

サポートを依頼する際，質問のある Diagnostics コンポーネントのバージョンを把握しておくと便利です。

Java Agent のバージョンは，次のいずれかの方法で確認できます。

- バージョン・ファイル **＜プローブのインストール・ディレクトリ＞ ¥version.txt** を確認する。このファイルには，4 桁のバージョン番号とビルド番号があります。
- プローブ・ログ・ファイル（**＜プローブのインストール・ディレクトリ＞ /log/ ＜ Probe ID ＞ /probe.log**）でバージョン番号を確認する。
- Diagnostics UI の［システムの状況］ビューでバージョン番号を確認する（付録 D,「管理者用のシステム・ビューの使用」を参照）。

Java Agent のアンインストール

Java Agent をアンインストールするには，次の手順を実行します。

- Windows マシンの場合は，［スタート］＞［すべてのプログラム］＞［HP Java Agent］＞［Uninstaller］を選択します。

または，**＜プローブのインストール・ディレクトリ＞¥_uninst** ディレクトリにある **uninstaller.exe** を実行することもできます。

- Linux または Solaris マシンの場合，**＜プローブのインストール・ディレクトリ＞/_uninst** ディレクトリにある **uninstaller.bin** を実行します。
- それ以外の UNIX マシンの場合，1.4 以降の JVM を選択し，**java -jar ＜プローブのインストール・ディレクトリ＞/_uninst/uninstall.jar** を実行して Java Agent をアンインストールします。

  アプリケーション・サーバの起動スクリプトに追加した Java Agent パラメータも削除します。

6

Java Agent で監視するためのアプリケーション・サーバの準備

本章では，HP Diagnostics Java Agent でアプリケーションを監視できるようにアプリケーション・サーバを準備する方法について説明します。

本章の内容

監視用のアプリケーション・サーバの準備について

HP Diagnostics Java Agent をインストールしたら，Java Agent でアプリケーションを監視できるようにアプリケーション・サーバを準備（インストルメント）する必要があります。この準備では，通常，アプリケーション・サーバによって使用される **JRE をインストルメント**し，Java Agent を呼び出すように**アプリケーション・サーバの JVM パラメータを設定**します。

インストールされている JRE が Diagnostics の JRE のインストルメンテーションによって変更されることはありません。インストルメントされたクラスのコピーが Java Agent のインストール・ディレクトリの下に配置されます。次に，適切な JVM パラメータを使用することで，アプリケーション・サーバを実行する JVM にこれらのインストルメントされたクラスがロードされます。インストルメンテーションは Diagnostics JRE Instrumenter ユーティリティを使用して行われ，このユーティリティはさまざまなオプションを使用して自動的に呼び出すことも手動で呼び出すこともできます。

次の 2 つのレベルのインストルメンテーションがあります。

- 基本インストルメンテーション

  アプリケーション・サーバの起動時に Java Agent を起動することで，アプリケーション・サーバがインストルメントし，監視されます。これを行うには，アプリケーション・サーバの JVM パラメータに -javaagent オプションを追加します。

- 推奨インストルメンテーション

  基本インストルメンテーションに加えて，Java Agent に付属の JRE Instrumenter ユーティリティを使用して，アプリケーション・サーバで使用する JRE（Java Runtime Environment）もインストルメントすることをお勧めします。この追加インストルメンテーションにより，特許出願中のコレクション・リークの指摘（CLP）などの高度な機能を Java Agent で使用できるようになります。CLP はリーク中のコレクションを自動的に検出し，リークの発生場所に関するスタック・トレースを提供します。これは問題を早期に識別するのに役立ち，問題（結果的なメモリ不足のエラーやサーバのクラッシュなど）を軽減する時間的な余裕が生まれるだけでなく，ヒープ・ダンプの分析という退屈な作業を回避することで開発者が時間を節約することもできます（337 ページ「コレクション・リークの指摘の設定」を参照）。さらに，追加インストルメンテーションには WebSphere 6.1 などの特定のアプリケーション・サーバでのパフォーマンス上の利点もあります。

さまざまな JRE インストルメンテーション・モードの使用に関する一般的な手順については，217 ページ「JRE Instrumenter とさまざまな起動オプションについて」を参照してください。

JRE 1.4 を使用する WebLogic 8.1 や WebSphere 5.1 / 6.0 などの旧アプリケーション・サーバでは，基本インストルメンテーションは使用できず，推奨インストルメンテーションを使用する必要があります。

続行するには，下記のリストでお使いのアプリケーション・サーバを見つけ，手順に従ってインストルメンテーションと設定を行います。

アプリケーション・サーバの設定例

本項では，監視に一般的に使用されるさまざまなアプリケーション・サーバの設定方法の例を紹介します。JRE Instrumenter を呼び出すさまざまな方法の説明については，217 ページ「JRE Instrumenter とさまざまな起動オプションについて」の項を参照してください。

重要 : アプリケーション・サーバの設定を変更する前に，起動スクリプトの構造，プロパティ値の設定方法，環境変数の使い方を理解しておいてください。変更する前に，更新するファイルのバックアップ・コピーを必ず作成してください。

---

164 ページ「例 1 : GlassFish の設定」

167 ページ「例 2 : JBoss の設定」

171 ページ「例 3 : Oracle の設定」

177 ページ「例 4 : SAP NetWeaver の設定」

181 ページ「例 5 : TIBCO ActiveMatrix/BusinessWorks の設定」

185 ページ「例 6 : Tomcat の設定」

190 ページ「例 7 : WebLogic の設定」

194 ページ「例 8 : WebSphere の設定」

210 ページ「例 9 : webMethods の設定」

特定のプラットフォームでサポートされているアプリケーション・サーバの最新情報については，HP Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照するか, HP カスタマ・サポートにお問い合わせください。

注 :

- この章で示しているスクリプト例には，読みやすくするために改行があるものがあります。ただし，実際のスクリプトに改行はありません。コマンドのテキストは，必要に応じて画面でラップします。
- 指定するパスにスペースがある場合は引用符を使用します。

例 1 : GlassFish の設定

GlassFish アプリケーション・サーバの設定では，その設定ファイルを変更して JVM パラメータを追加します。一般的な GlassFish 3.x アプリケーション・サーバの実装については，次の手順を実行してください。サイト管理者は，これらの手順を使用して，特定の環境に適した変更を行う方法を示すことができます。

GlassFish アプリケーション・サーバについては，暗黙モードを使用して JRE のインストルメンテーションを設定します（222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照）。

GlassFish アプリケーション・サーバを設定するには，次の手順を実行します。

1 GlassFish 設定ファイルでプロパティ **org.osgi.framework.bootdelegation** を見つけて，その最後に（区切り記号としてカンマを使用して）**com.mercury.opal.capture.proxy** を追加します。

GlassFish 3.1.2 では，このプロパティは**＜ GlassFish のインストール・ディレクトリ＞ /glassfish/config/osgi.properties** にあります。

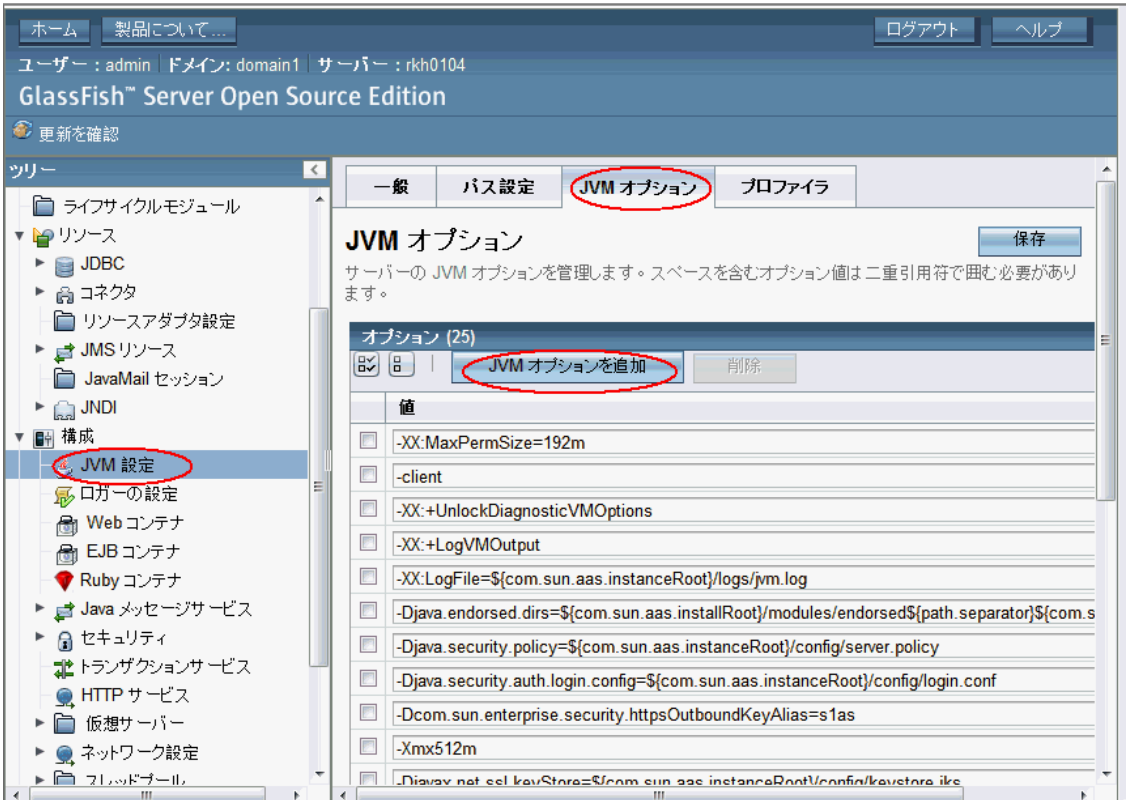
以前のバージョンの GlassFish では，このプロパティは次の 2 つのファイル内に存在する可能性があります。

＜GlassFish のインストール･ディレクトリ＞ /osgi/equinox/configuration/config.ini

＜ GlassFish のインストール・ディレクトリ＞ /osgi/felix/conf/config.properties

2 GlassFish 管理コンソールにログインし，次の手順で［**JVM オプション**］ページに移動します。

GlassFish 3.1.2 の場合，左側のツリーで，［**構成**］＞［**{config_name}**］＞［**JVM 設定**］に移動します。{config_name} は，サーバ設定の名前（**server-config** など）です。次に，［**JVM オプション**］タブを選択します。次のスクリーンショットを参照してください。



より古いバージョンの GlassFish を使用している場合，左側のツリーで［**Application Server**］をクリックし，上部の［JVM 設定］タブを選択します。次に，［**JVM オプション**］タブを選択します。

3 [**JVM オプションを追加**] ボタンをクリックして，2 つの JVM パラメータを 1 つずつ追加します。最初のパラメータ（-javaagent）では，アプリケーション・サーバの JVM の起動時に Java Agent を起動します。最初の呼び出しで，2 番目のパラメータ（-Xbootclasspath）では，アプリケーション・サーバの JRE をインストルメントします。-Xbootclasspath パラメータで名前を入力し，インストルメントされるクラスを格納するディレクトリの名前を指定します。次の例では，**MyServer** を自分で選択した名前に置き換えてください。

Windows 環境での例を次に示します。

```
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥auto¥MyServer¥instr.jre
```

UNIX 環境での例を次に示します。

```
-javaagent:/opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/probeagent.jar
-Xbootclasspath/p:/opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/classes/auto/MyServer/instr.jre
```

注 : クラスタ・セットアップの場合，次の JVM パラメータも追加します。
-Dprobe.id=<ProbeID>_%0
＜ ProbeID ＞は，このアプリケーション・サーバ・クラスタに割り当てるプローブ名です。文字列「%0」は，異なるプローブ・インスタンスを区別できるように一意の ID に置き換えられます。

---

4 GlassFish アプリケーション・サーバを再起動します。

Java Agent が起動し，JRE Instrumenter が暗黙的に実行されて JRE がインストルメントされます。

GlassFish アプリケーション・サーバが起動しない場合，**＜ GlassFish のインストール・ディレクトリ＞ /glassfish/domains/ ＜ドメイン名＞ /config/domain.xml** ファイルで JVM パラメータを確認，変更し，問題を解決します。＜ドメイン名＞はドメインの名前（**domain1** など）です。

GlassFish アプリケーション・サーバの初期化に時間がかかり，次のエラーが発生して起動に失敗することがあります。

Could not load Logmanager
"com.sun.enterprise.server.logging.ServerLogManager"

この場合は，次の JVM オプションを追加します。
-Ddiag.agent.init.delay.ms= ＜遅延（ミリ秒）＞
＜遅延（ミリ秒）＞は，ミリ秒数（6000 など）です。エラーがなくなるまで＜遅延（ミリ秒）＞の値を増やします。

5 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞/DiagnosticsAgent/log/ ＜Probe ID＞/probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合, JVM パラメータを正しく設定していない可能性があります。GlassFish サーバ・ログでエラー・メッセージを確認してください。

6 必要に応じて, インストルメントされた JRE が使用されるように, アプリケーション・サーバを再起動します。

重要 : 今後アプリケーション・サーバで使用する JRE を更新する場合（アプリケーション・サーバのパッチを適用する場合など），アプリケーション・サーバを再起動する前に，新しい JRE がインストルメントされるように**＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/classes/auto/MyServer** ディレクトリ（MyServer は自分のディレクトリ名に置き換えます）を削除する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。詳細については，222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照してください。

例 2 : JBoss の設定

JBoss アプリケーション・サーバは，シェルまたはコマンド・スクリプトで起動します。したがって，起動スクリプトを変更して JBoss アプリケーション・サーバを設定することをお勧めします。JBoss が提供する起動スクリプトは，サイト管理者によって頻繁にカスタマイズされるため，すべての状況に適合する詳細な設定手順を説明するのは不可能です。したがって，次の項では，一般的な実装に JBoss アプリケーション・サーバを設定するための具体的な例を含む手順を紹介します。サイト管理者は，これらの手順を使用して，カスタマイズされた環境で変更を行う方法を示すことができます。

JBoss アプリケーション・サーバについては，明示モードで JRE のインストルメンテーションを設定します（219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照）。

自動明示モードを使用するには，次の 2 つのタスクを行う必要があります。

- アプリケーション・サーバで使用されるのと同じ JRE を使用して JRE Instrumenter を実行するように，アプリケーション・サーバの起動スクリプトを変更します。JRE Instrumenter からの出力を見れば，次のタスクに必要な JVM パラメータがわかります。
- JRE Instrumenter からの出力で見つかるアプリケーション・サーバの JVM パラメータを設定します。

JBoss アプリケーション・サーバを設定するには，次の手順を実行します。

1 アプリケーションに対して JBoss を起動するのに使う起動スクリプトを特定します。次に例を示します。

- 7.0 未満のバージョンの JBoss の場合

  通常，起動スクリプト・ファイルは，次のようなパスにあります。

  ＜ JBoss ホーム＞ ¥bin¥run.[bat|sh]

  ＜ JBoss ホーム＞は，**C:¥jboss-4.2.3.GA** または **C:¥jboss-6.0.0.Final** などの JBoss のインストール・ディレクトリへのパスです。

- JBoss 7.0 以降の場合

  通常，起動スクリプト・ファイルは，次のいずれかのようなパスにあります。

  ＜ JBoss ホーム＞ ¥bin¥domain.[bat|sh]

  ＜ JBoss ホーム＞ ¥bin¥standalone.[bat|sh]

  ＜ JBoss ホーム＞は，**C:¥jboss-as-7.1.0.Final** などの JBoss のインストール・ディレクトリへのパスです。

注 : JBoss アプリケーション・サーバが Windows サービスとして起動されている場合，次の手順に進む前に，起動スクリプトを使用してアプリケーション・サーバを起動できることを確認します。これは，起動スクリプトがバックグラウンドで Windows サービスによって起動されるためです。

---

2 起動スクリプトのバックアップ・コピーを作成し，エディタで起動スクリプトを開きます。

3 アプリケーション・サーバを起動する Java コマンド・ライン（またはコード・ブロック）を特定します。

.bat ファイルの Java コマンド・ラインの例を次に示します。

```
"%JAVA%" %JAVA_OPTS% ^
-Djava.endorsed.dirs="%JBOSS_ENDORSED_DIRS%" ^
-classpath "%JBOSS_CLASSPATH%" ^   org.jboss.Main %*
```

.sh ファイルの Java コマンド・ラインの例を次に示します。

```
if [ "x$LAUNCH_JBOSS_IN_BACKGROUND" = "x" ]; then
  # Execute the JVM in the foreground
  "$JAVA" $JAVA_OPTS ¥
    -Djava.endorsed.dirs="$JBOSS_ENDORSED_DIRS" ¥
    -classpath "$JBOSS_CLASSPATH" ¥
    org.jboss.Main "$@"
  JBOSS_STATUS=$?
else
  # Execute the JVM in the background
  "$JAVA" $JAVA_OPTS ¥
    -Djava.endorsed.dirs="$JBOSS_ENDORSED_DIRS" ¥
    -classpath "$JBOSS_CLASSPATH" ¥
    org.jboss.Main "$@" &
  JBOSS_PID=$!
  . ……
```

4 Java コマンド・ライン（またはコード・ブロック）の直上に 2 行を追加します。1 行目は JRE Instrumenter を実行する Java コマンドを呼び出します。2 行目は必要な JVM パラメータを追加します。使用する JVM パラメータが不明な場合は，後の手順 6 で追加できます。この 2 つの行で名前を入力し，自動的にインストルメントされる JRE クラスを格納するためのディレクトリを指定します。次の例では，**MyServer** を自分で選択した名前に置き換えてください。

JRE 5 以降を使用する JBoss アプリケーション・サーバの .bat ファイルに追加される 2 つの行の例を次に示します。

```
"%JAVA%" -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f MyServer

set JAVA_OPTS=%JAVA_OPTS% -Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
```

JRE 5 以降を使用する JBoss アプリケーション・サーバの .sh ファイルに追加される 2 つの行の例を次に示します。

```
"$JAVA" -jar /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/jreinstrumenter.jar -f MyServer

JAVA_OPTS="$JAVA_OPTS -Xbootclasspath/p:/opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/classes/
MyServer/instr.jre -javaagent:/opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/probeagent.jar"
```

JRE 1.4 を使用する JBoss アプリケーション・サーバの .bat ファイルに追加される 2 つの行の例を次に示します。

```
"%JAVA%" -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f MyServer

set JAVA_OPTS=%JAVA_OPTS% -Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre;C:¥MercuryDiagnostics¥
JavaAgent¥DiagnosticsAgent¥classes¥boot
```

注 : Java コマンド・ラインで JVM パラメータの定義に JAVA_OPTS 変数を使用しない場合，これらの例で示している変数名 JAVA_OPTS を正しい名前に変更する必要があります。

重要 :

- JBoss 6 以降の場合，次の JVM パラメータを JAVA-OPTS に追加します。
  -Djava.util.logging.manager=org.jboss.logmanager.LogManager
- JBoss 7 以降の場合，次の JVM パラメータを JAVA-OPTS に追加します。
  -Djboss.modules.system.pkgs=org.jboss.byteman,com.mercury.opal

---

5 変更を起動スクリプトに保存し, 変更されたスクリプトを使用してアプリケーション・サーバを再起動します。

6 エラーや入力ミスを見つけやすくするために，起動スクリプトを実行し，JRE Instrumenter からの出力を検索します（Xbootclasspath を検索）。手順 4 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加した場合，それを JRE Instrumenter の出力にある JVM パラメータと比較します。これらが同じでない場合，JRE Instrumenter で出力された正しい JVM パラメータを使用して起動スクリプトを更新し，アプリケーション・サーバを再起動します。手順 4 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加していない場合，ここで追加してアプリケーション・サーバを再起動します。例については手順 4 を参照してください。

7 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞¥DiagnosticsAgent¥log¥＜Probe ID＞¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，JRE のインストルメンテーションに失敗したか，JVM パラメータを正しく設定しなかったかのいずれかです。詳細については，219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照してください。

例 3 : Oracle の設定

本項では，Oracle 9i または 10g アプリケーション・サーバを設定するための方法を紹介します。

Oracle9i アプリケーション・サーバの設定について

Oracle9i アプリケーション・サーバは，アプリケーション・サーバの起動に使用される XML ファイルに JRE Instrumenter で出力された JVM パラメータを追加することで設定します。

Oracle9i アプリケーション・サーバを設定するには，次の手順を実行します。

1 サーバの起動時にアプリケーション・サーバの設定を制御するのに使われる XML ファイルを見つけます。通常，このファイルは **＜ Oracle 9iAS のインストール・ディレクトリ＞ /opmn/conf/opmn.xml** に置かれています。

2 **opmn.xml** ファイルのバックアップ・コピーを作成し，エディタで編集する **opmn.xml** ファイルを開きます。

3 JRE Instrumenter を手動で実行して，Oracle アプリケーション・サーバで使用する JRE をインストルメントします（JRE Instrumenter を手動で実行する手順については，224 ページ「マニュアル・モードでの JRE Instrumenter の使用」を参照してください）。

4 JRE Instrumenter の結果からコピーした Java パラメータ（**Xbootclasspath** プロパティなど）を java-option value プロパティにコピーします。

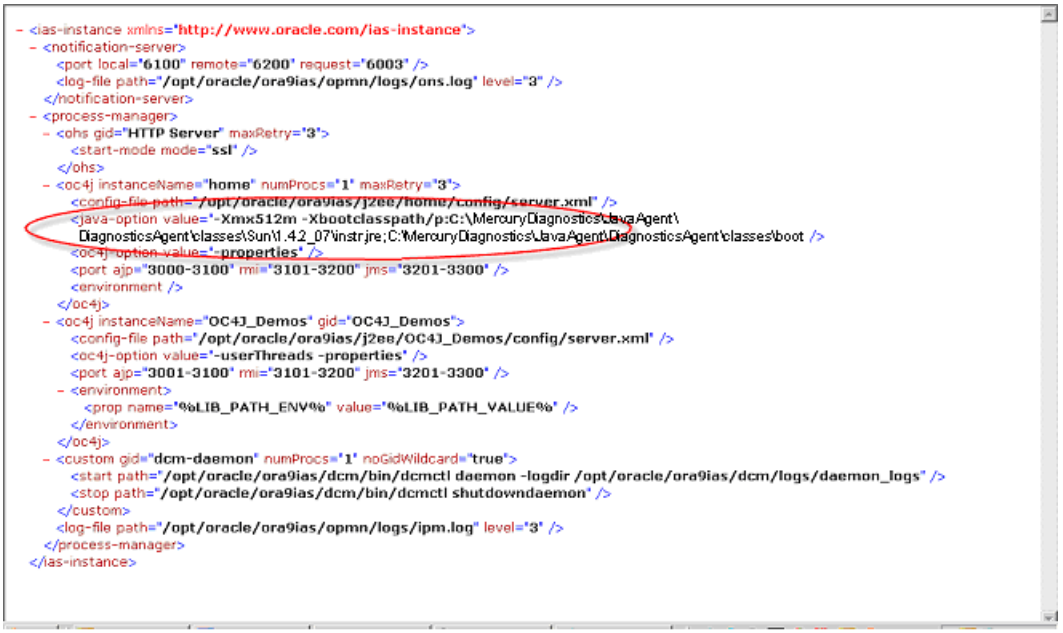
次は，**Xbootclasspath** パラメータの例です。

```
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥Sun¥1.4.2_04¥instr.jre;
C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥boot
```

注：-Xbootclasspath パラメータを変更する際, 指定するパスにスペースがある場合は引用符を使用してください。

次の図は，**Xbootclasspath** パラメータを追加する前の Oracle 9iAS 起動スクリプトの例です。

次の図は，**Xbootclasspath** パラメータを追加した後の Oracle 9iAS 起動スクリプトの例です。



5 変更を XML ファイルに保存して Oracle アプリケーション・サーバを再起動します。

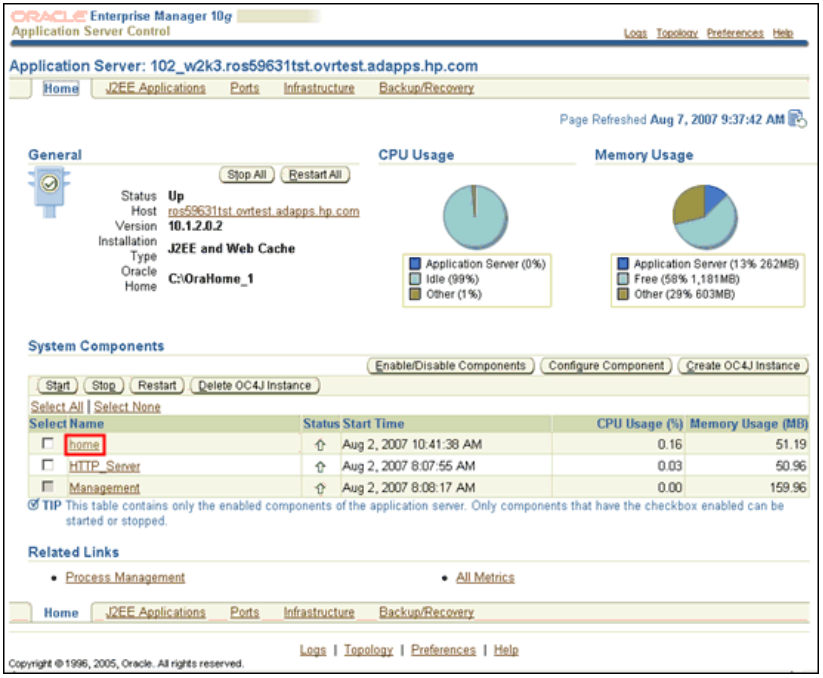
6 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞¥DiagnosticsAgent¥log¥＜Probe ID＞¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，JRE Instrumenter を実行していないか，Xbootclasspath などの Java パラメータを正しく入力しなかったかのいずれかです。

重要：今後アプリケーション・サーバで使用する JRE を更新する場合（アプリケーション・サーバのパッチを適用する場合など），JRE Instrumenter を再度実行して新しい JRE をインストルメントし，JVM パラメータを適宜変更する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。詳細については，224 ページ「マニュアル・モードでの JRE Instrumenter の使用」を参照してください。

Oracle 10g アプリケーション・サーバを設定するには，次の手順を実行します。

1 JRE Instrumenter を手動で実行して，Oracle アプリケーション・サーバで使用する JRE をインストルメントします （詳細については，224 ページ「マニュアル・モードでの JRE Instrumenter の使用」を参照)。JRE Instrumenter によって，後で使用する JVM パラメータ（-Xbootclasspath パラメータなど）が出力されます。

2 Oracle の**アプリケーション・サーバ管理コンソール**を開きます。



3 ［**home**］（または［**MyOC4J**］）システム・コンポーネントをクリックします。

4 ［**OC4J: home**］ページで，［**Administration**］を選択します。

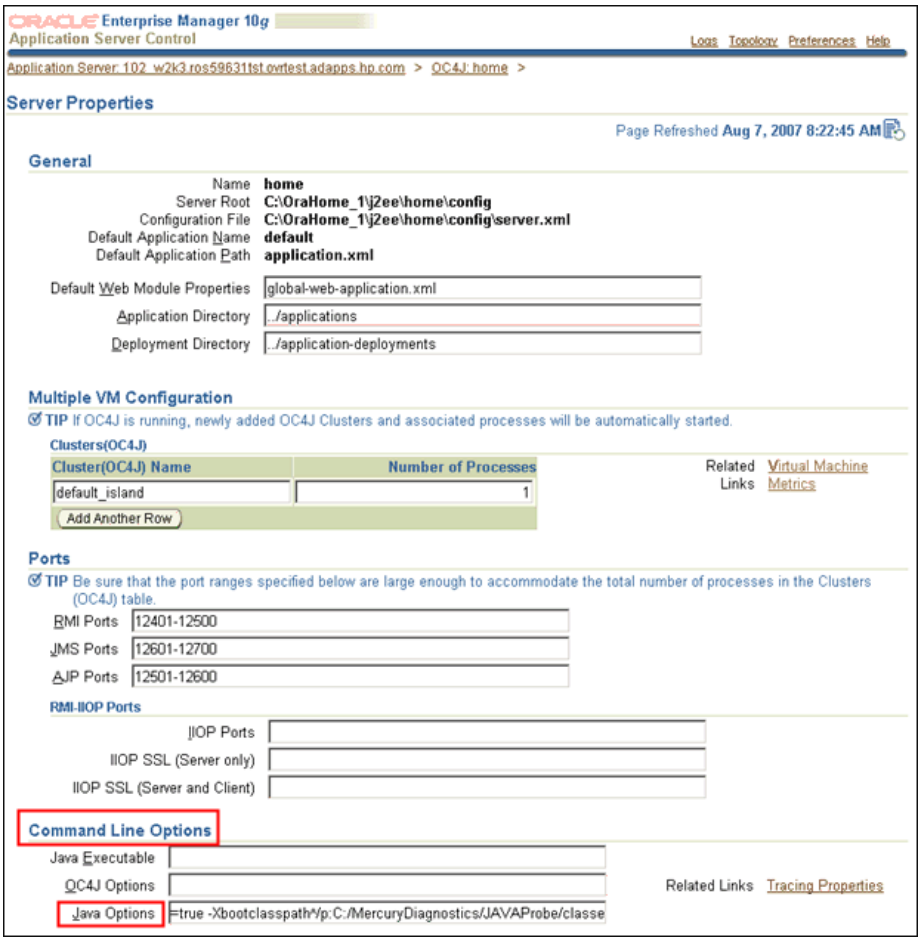
5 ［Administration］ページで，［**Server Properties**］を選択します。

6 ［Server Properties］ウィンドウの［**Command Line Options**］で，［**Java Options**］ボックスに JRE Instrumenter の結果からコピーした JVM パラメータを追加します。

注 : /p スイッチの前に（^）を追加する必要があります。追加しない場合，Oracle は（/）スイッチ・オプションを（¥）に変更します。

次は，（^）を挿入した Xbootclasspath パラメータの例です。

```
-Xbootclasspath^/ p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥Sun¥1.4.2_07¥instr.jre;
C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥boot
```



7 変更を適用して Oracle サーバを再起動します。

8 プローブが正しく設定されていることを確認するには，**＜Java Agent のインストール・ディレクトリ＞¥DiagnosticsAgent¥log¥＜Probe ID＞¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，JRE Instrumenter を実行していないか，**Xbootclasspath** などの Java パラメータを正しく入力しなかったかのいずれかです。

重要 : 今後アプリケーション・サーバで使用する JRE を更新する場合（アプリケーション・サーバのパッチを適用する場合など），JRE Instrumenter を再度実行して新しい JRE をインストルメントし，JVM パラメータを適宜変更する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。詳細については，224 ページ「マニュアル・モードでの JRE Instrumenter の使用」を参照してください。

例 4 : SAP NetWeaver の設定

SAP NetWeaver アプリケーション・サーバで使用している JRE のバージョンによって，設定の手順が異なります。次の 2 つの項では，JRE のバージョンが 5 以降の一般的な NetWeaver 実装と, JRE のバージョンが 1.4 の一般的な NetWeaver 実装についての設定手順を説明します。サイト管理者は，これらの手順を使用して，特定の環境に適した変更を行う方法を示すことができます。

JRE のバージョンが 5 以降の SAP NetWeaver アプリケーション・サーバの場合，暗黙モードを使用して JRE のインストルメンテーションを設定します（222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照）。JRE のバージョンが 1.4 の SAP NetWeaver アプリケーション・サーバの場合，手動で JRE Instrumenter を実行します（224 ページ「マニュアル・モードでの JRE Instrumenter の使用」を参照）。

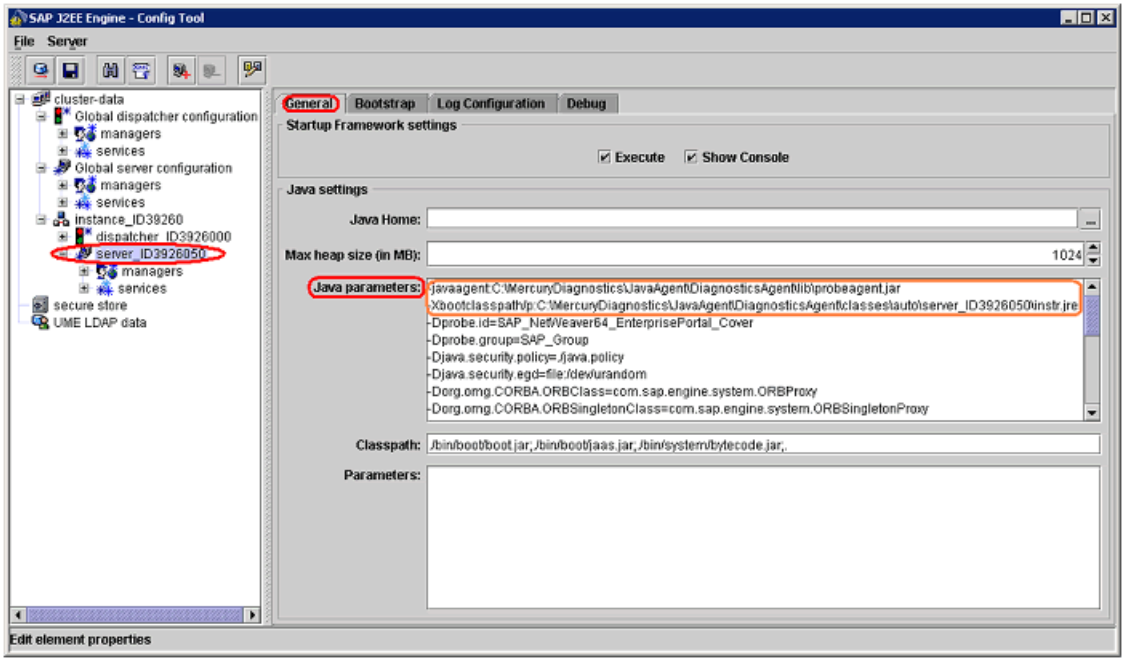
JRE バージョン 5 以降を使用する SAP NetWeaver アプリケーション・サーバを設定するには，次の手順を実行します。

1. NetWeaver J2EE Engine 設定ツールを実行します。このツールを実行するスクリプト名は **configtool.bat** で, **usr¥sap¥CR2¥JC00¥j2ee¥configtool** ディレクトリにあります。CR2 は，SAP システムの名前の一例です。
2. 設定ツール UI の左側のツリーで，監視するサーバを選択します。たとえば，下のスクリーンショットでは［**cluster-data**］＞［**instance_ID39260**］＞［**server_ID3926050**］を選択します。次に，右側で［**General**］タブを選択し，2 つの JVM パラメータを［**Java parameters**］テキスト・ウィンドウに追加します。

最初のパラメータ（**-javaagent**）では，アプリケーション・サーバの JVM の起動時に Java Agent を呼び出します。最初の呼び出しで，2 番目のパラメータ（**-Xbootclasspath**）では，アプリケーション・サーバの JRE をインストルメントします。-Xbootclasspath パラメータで名前を入力し，インストルメントされるクラスを格納するディレクトリを指定します。下の例では，server_ID3926050 という名前を使用しています。server_ID3926050 を自分で選択した名前に置き換えてください。

```
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥auto¥server_ID3926050¥instr.jre
```

次のスクリーンショットは，JVM パラメータの例です。



3 変更を保存して，設定ツールを終了します。

4 **＜Java Agent のインストール・ディレクトリ＞¥etc¥capture.properties** ファイルを編集して，次の値をこれらのプロパティに割り当てます。

event_buffer.size = 10000
event_buffer.flush.level = 1000

5 SAP NetWeaver アプリケーション・サーバを再起動します。

6 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/log/ ＜ Probe ID ＞ /probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，JRE のインストルメンテーションに失敗したか，JVM パラメータを正しく入力しなかったかのいずれかです。NetWeaver ブートストラップ・ログでエラー・メッセージを確認します。

重要 : 今後アプリケーション・サーバで使用する JRE を更新する場合（アプリケーション・サーバのパッチを適用する場合など），アプリケーション・サーバを再起動する前に，新しい JRE がインストルメントされるように**＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/classes/auto/server_ID3926050** ディレクトリ（server_ID3926050 は自分のディレクトリ名に置き換えます）を削除する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。詳細については，222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照してください。

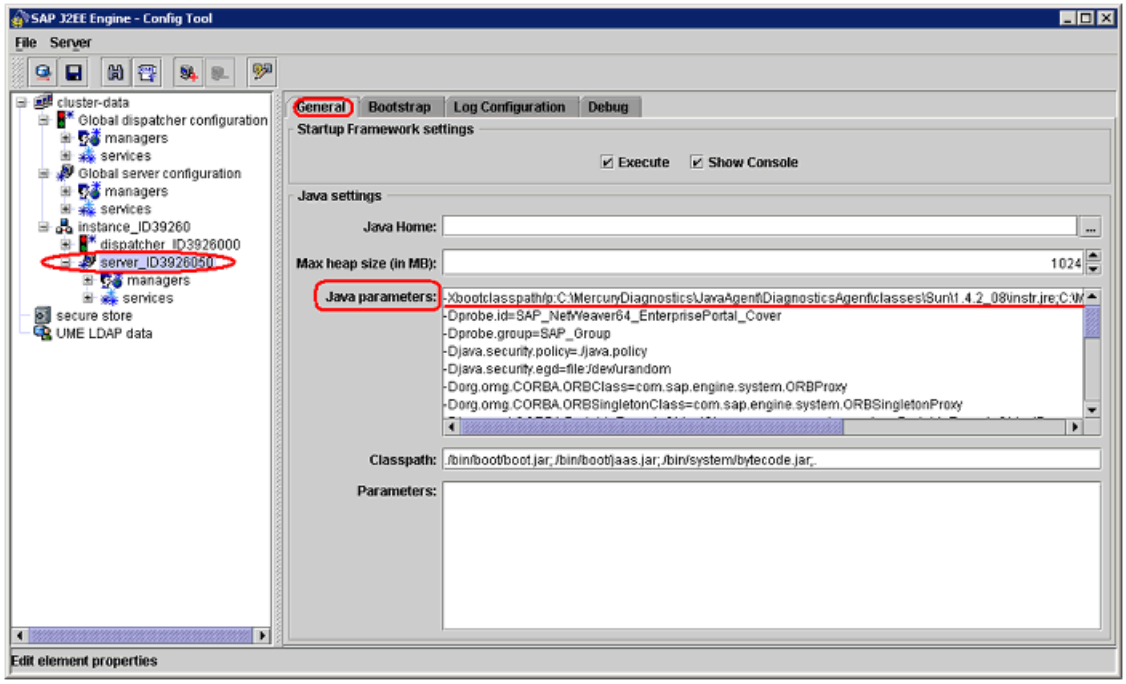
JRE バージョン 1.4 を使用する SAP NetWeaver アプリケーション・サーバを設定するには，次の手順を実行します。

1 NetWeaver アプリケーション・サーバの実行に使用する JRE を特定します。

2 JRE Instrumenter を実行してこの JRE をインストルメントします。JRE Instrumenter によって，後で使用する JVM パラメータ（**-Xbootclasspath** パラメータなど）が出力されます JRE Instrumenter を実行して，結果として生成される JVM パラメータをクリップボードにコピーして以下の手順 4 で使用する方法については，224 ページ「マニュアル・モードでの JRE Instrumenter の使用」を参照してください。

3 NetWeaver J2EE Engine 設定ツールを実行します。このツールを実行するスクリプト名は **configtool.bat** で，**usr¥sap¥CR2¥JC00¥j2ee¥configtool** ディレクトリにあります。CR2 は，SAP システムの名前の一例です。

4 設定ツール UI の左側のツリーで，監視するサーバを選択します。たとえば，［**cluster-data**］>［**instance_ID39260**］>［**server_ID3926050**］のように選択します。［General］タブで，Java パラメータ・テキスト・ウィンドウに移動します。JRE Instrumenter で出力された JVM パラメータをテキスト・ウィンドウに追加します。次のスクリーンショットを参照してください。



5 変更を保存して，設定ツールを終了します。

6 **＜Java Agent のインストール・ディレクトリ＞¥etc¥capture.properties** ファイルを編集して，次の値をこれらのプロパティに割り当てます。

event_buffer.size = 10000
event_buffer.flush.level = 1000

7 NetWeaver アプリケーション・サーバを再起動します。

8 プローブが正しく設定されていることを確認するには，**＜Java Agent のインストール・ディレクトリ＞/DiagnosticsAgent/log/＜Probe ID＞/probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合，JVM パラメータを正しく設定していない可能性があります。NetWeaver ブートストラップ・ログでエラー・メッセージを確認してください。

例 5 : TIBCO ActiveMatrix/BusinessWorks の設定

次の項では，アプリケーションを監視できるように TIBCO ActiveMatrix と BusinessWorks を設定する方法について説明します。

TIBCO ActiveMatrix アプリケーション・サーバと BusinessWords アプリケーション・サーバの場合，暗黙モードを使用して JRE のインストルメンテーションを設定します（222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照）。

重要 : 今後アプリケーション・サーバで使用する JRE を更新する場合（アプリケーション・サーバのパッチを適用する場合など），アプリケーション・サーバを再起動する前に，新しい JRE がインストルメントされるように**＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/classes/auto/Tibco_Node1** ディレクトリ（Tibco_Node1 は自分のディレクトリ名に置き換えます）を削除する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。

TIBCO サーバをインストルメントするには，次の手順で示しているように異なる TIBCO バージョンについて 2 つの JVM パラメータを追加します。最初のパラメータ（-javaagent）では，アプリケーション・サーバの JVM の起動時に Java Agent を起動します。最初の呼び出しで，2 番目のパラメータ（-Xbootclasspath）では，アプリケーション・サーバの JRE をインストルメントします。-Xbootclasspath パラメータで，サーバを識別する名前を入力します。次の手順では，太字で示された名前の例を自分で選択した名前に置き換えます。

TIBCO BusinessWorks を設定するには，次の手順を実行します。

TIBCO BusinessWorks の場合，アプリケーションのデプロイ方法と監視するアプリケーションに応じて，次のいずれかの TIBCO BusinessWorks ファイルの **java.extended.properties** に対して次の例のように 2 つの JVM パラメータを追加します。**java.extended.properties** が存在しない場合，追加します。そのための方法の 1 つを下の例で示しています。

```
java.extended.properties=-javaagent:C:/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/probeagent.jar
-Xbootclasspath/p:C:/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/classes/auto/TIBCO_BW1/instr.jre
```

以前にデプロイしたアプリケーションを監視するか，TIBCO BusinessWorks の特定のアプリケーションのみを監視する場合，**＜ BusinessWorks のインストール・ディレクトリ＞ ¥tra¥domain¥ ＜ドメイン名＞ ¥application¥ ＜アプリケーション名＞ ¥ ＜アプリケーション名＞ .tra** を JVM パラメータで更新します。インストルメントする JRE が .tra ファイルのユーザ・プロパティ **tibco.env.TIB_JAVA_HOME** で指定されます。

- TIBCO Administrator を使用してアプリケーションをデプロイした場合：**bwengine.tra** を JVM パラメータで更新します。
- TIBCO Designer を使用してアプリケーションをデプロイした場合：**designer.tra** を JVM パラメータで更新します。

JMX メトリックス収集のために TIBCO BusinessWorks を設定するには，次の手順を実行します。

TIBCO BusinessWorks JMX メトリックス収集の場合，BusinessWorks プロセスへの JMX アクセスを有効にします。これを行うには，Java Agent のインストルメンテーションが設定されているのと同じ BusinessWorks の **.tra** ファイルに次のプロパティを追加します。

JmxEnabled=true

追加 JMX メトリックスは，Apache Tomcat および Pramati J2EE サーバなどの，TIBCO で使用する特定のコンポーネントによって公開されます。これらのメトリックスの一部は，標準設定では収集されません（こうしたメトリックスは **metric.config** ファイルではコメント・アウトされています）。これらのメトリックスは，**metrics.config** ファイルでコメントアウトを解除することでアクティブ化できます。これらのメトリックスは主にパフォーマンス調整設定のために存在するパラメータで，アプリケーションの存続中に変更することはほとんどないため，非アクティブになっています。

JMX メトリックス収集の詳細については，679 ページ「Java Agent - JMX メトリックス・キャプチャ」を参照してください。

TIBCO ActiveMatrix Service Bus 2.0 および 2.3 を設定するには，次の手順を実行します。

1 TIBCO ActiveMatrix Service Bus（AMSB）2.0 および 2.3 の場合，AMSB **.tra** ファイルを見つけて，**java.extended.properties** に JVM パラメータを追加します。java.extended.properties が存在しない場合，追加します。

次の Windows での例では，名前として **TIBCO_Node1** を使用しています。バックスラッシュ（¥）の代わりにスラッシュ（/）を使用している点に注意してください。

```
java.extended.properties=-javaagent:jarC:/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/probeagent.jar
-Xbootclasspath/p:C:/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/classes/auto/TIBCO_Node1/instr.jre
```

2 さらに，AMSB の発信 JMS Web サービス操作を表示するため，**etc¥inst.properties** の **details.conditional.properties** を適切なバージョンに更新する必要がある場合があります。現在，2.0 と 2.3 の 2 つのバージョンの AMSB がサポートされています。1 つのバージョンのみが「true」に設定されて有効になっていてその他が「false」に設定されていることを確認します。次の例では，標準設定の AMSB バージョン 2.3 が有効になっています。

```
details.conditional.properties= ¥
 mercury.enable.SOAPHandler=true, ¥
 mercury.enable.autoLoadSOAPHandler=true, ¥
 mercury.enable.resourcemonitor.jdbcConnection=false, ¥
 mercury.enable.resourcemonitor.jdbcStatement=false, ¥
 mercury.enable.resourcemonitor.jdbcResultSet=false, ¥
 mercury.enable.tibco.amsb2.0=false, ¥
 mercury.enable.tibco.amsb2.3=true
```

TIBCO ActiveMatrix Service Bus 3.1.2 実運用環境を設定するには，次の手順を実行します。

1 実運用環境の TIBCO ActiveMatrix Service Bus（AMSB）3.1.2 の場合，次の AMSB ファイルを見つけます。

＜TIBCO AMX の設定ディレクトリ＞¥data¥tibcohost¥＜エンタープライズ・サーバ名＞ ¥tools¥machinemodel¥machine.xmi

2 監視する**各ノード**のファイルの **runtimes** セクションを更新します。次に例を示します。

<runtimes xsi:type="machinemodel:OSGiRuntime" **name**="**Node1**"

各ノードの runtimes セクションで，**frameworkProperties** のキー **org.osgi.framework.bootdelegation** を見つけて，プロパティの値に **com.mercury.*** を追加します。

次に例を示します。

<frameworkProperties key="org.osgi.framework.bootdelegation" value="com.ibm.*, ...,sun.*,**com.mercury.***"/>

3 次に，**各ノード**で .tra ファイルを見つけます。

＜TIBCO AMX の設定ディレクトリ＞¥data¥tibcohost¥＜エンタープライズ・サーバ名＞ ¥nodes¥ ＜ノード名＞ ¥bin¥tibams_ ＜ノード名＞ .tra

各ファイルの **java.extended.properties** に 2 つの JVM パラメータを追加します。次の例では，名前として **AMSB_Node1** を使用しています。

```
java.extended.properties=-javaagent:C:/ MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/probeagent.jar
-Xbootclasspath/p:C:/MercuryDiagnostics/ JavaAgent/DiagnosticsAgent/classes/auto/AMSB_Node1/instr.jre
```

TIBCO ActiveMatrix Service Bus 3.1.2 開発環境を設定するには，次の手順を実行します。

1 開発環境の TIBCO ActiveMatrix Service Bus（AMSB）3.1.2 の場合，ブートの委任を更新します。

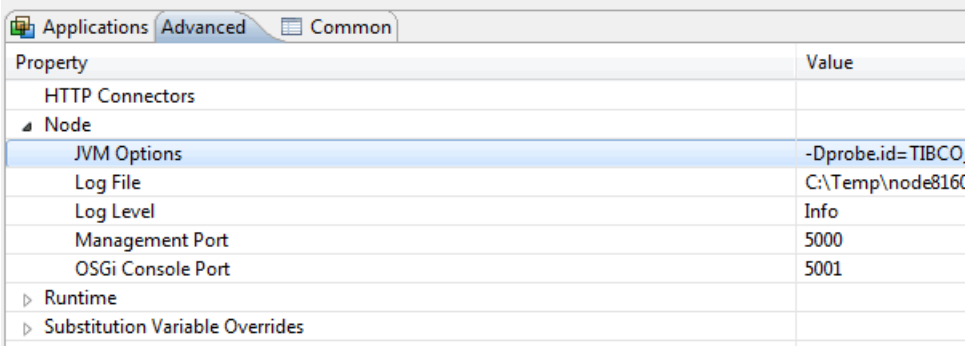
```
<TIBCO AMX の設定ディレクトリ >¥components¥shared¥1.0.0¥plugins¥com.tibco.
metadata.hpa.tibcohost.nodetype.integration_3.1.200.000¥META-INF¥com.tibco.amf.
node.types¥com.tibco.amf.hpa.tibcohost.node.integration.feature¥3.1.200¥provisioning.
properties
```

org.osgi.framework.bootdelegation の最後に **com.mercury.*** を追加します。

次に例を示します。

```
# OSGi framework properties
org.osgi.framework.bootdelegation=¥
...
 org.xml.*,¥
 sun.*,¥
 com.mercury.*
```

2 次に，実行設定の［**Advanced**］タブの［JVM Options］を更新し，2 つの JVM パラメータを追加します。



次の例では，名前として **AMSB_DevNode** を使用しています。

```
-javaagent:C:/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/probeagent.jar
-Xbootclasspath/p:C:/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/classes/auto/AMSB_DevNode/instr.jre
```

例 6 : Tomcat の設定

Apache Tomcat は，頻繁にほかのアプリケーションまたはサーバに組み込まれます。したがって，インストルメントする方法は異なる可能性があります。以降の項では，単純なシナリオで，特定の状況でも十分参考になる一般的な Tomcat サーバの設定方法について説明します。

Tomcat サーバをシェルまたは Windows コマンド・スクリプトで起動している場合，インストルメンテーションを実行するように起動スクリプトを変更することをお勧めします。185 ページ「起動スクリプトを使用した Tomcat サーバの設定」を参照してください。

Windows 環境で Tomcat が Windows サービスとしてインストールされていてスクリプトが存在しない場合，インストルメンテーションを実行するように Java オプションを変更することをお勧めします。詳細については，189 ページ「起動スクリプトを使用しない Tomcat サーバの設定」を参照してください。

起動スクリプトを使用した Tomcat サーバの設定

Tomcat が提供する起動スクリプトは，ほかのアプリケーションまたはサイト管理者によって頻繁にカスタマイズされるため，すべての状況に適合する詳細な設定手順を説明するのは不可能です。したがって，次の項では，一般的な Tomcat サーバを実装するための具体的な例を含む手順を紹介します。サイト管理者は，これらの手順を使用して，カスタマイズされた環境で変更を行う方法を示すことができます。

起動スクリプトを使用して Tomcat サーバを設定するには，自動明示インストルメンテーション・モードを使用します（219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照）。これには 2 つのタスクが伴います。

- アプリケーション・サーバで使用されるのと同じ JRE を使用して JRE Instrumenter を実行するように，アプリケーション・サーバの起動スクリプトを変更します。JRE Instrumenter からの出力を見れば，次のタスクに必要な JVM パラメータがわかります。
- JRE Instrumenter からの出力で見つかるアプリケーション・サーバの JVM パラメータを設定します。

起動スクリプトを使用して Tomcat サーバを設定するには，次の手順を実行します。

1 アプリケーションに対して Tomcat を起動するのに使う起動スクリプトを特定します。

シナリオによっては, 起動スクリプトが次のスクリプトを呼び出して Tomcat を起動する場合があります。

＜ Tomcat のインストール・ディレクトリ＞ /bin/catalina.[bat|sh]

＜ Tomcat のインストール・ディレクトリ＞は，**C:¥apache-tomcat-7.0.8** などの Tomcat のインストール・ディレクトリへのパスです。

2 起動スクリプトのバックアップ・コピーを作成し，エディタで起動スクリプトを開きます。

3 Tomcat サーバを起動する **Java** コマンド・ライン（またはコード・ブロック）を特定します。

catalina.bat ファイルからの例を次に示します。

```
rem Execute Java with the applicable properties
if not "%JPDA%" == "" goto doJpda
if not "%SECURITY_POLICY_FILE%" == "" goto doSecurity
%_EXECJAVA% %JAVA_OPTS% %CATALINA_OPTS% %DEBUG_OPTS%
-Djava.endorsed.dirs="%JAVA_ENDORSED_DIRS%" -classpath "%CLASSPATH%"
-Dcatalina.base="%CATALINA_BASE%" -Dcatalina.home="%CATALINA_HOME%"
-Djava.io.tmpdir="%CATALINA_TMPDIR%" %MAINCLASS% %CMD_LINE_ARGS%
%ACTION%
……
```

catalina.sh ファイルからの例を次に示します。

```
if [ "$1" = "-security" ] ; then
  if [ $have_tty -eq 1 ]; then
    echo "Using Security Manager"
  fi
  shift
  eval ¥"$_RUNJAVA¥" ¥"$LOGGING_CONFIG¥" $JAVA_OPTS $CATALINA_OPTS ¥
    -Djava.endorsed.dirs=¥"$JAVA_ENDORSED_DIRS¥" -classpath ¥"$CLASSPATH¥" ¥
    -Djava.security.manager ¥
    -Djava.security.policy==¥"$CATALINA_BASE/conf/catalina.policy¥" ¥
    -Dcatalina.base=¥"$CATALINA_BASE¥" ¥
    -Dcatalina.home=¥"$CATALINA_HOME¥" ¥
    -Djava.io.tmpdir=¥"$CATALINA_TMPDIR¥" ¥
    org.apache.catalina.startup.Bootstrap "$@" start
else
  eval ¥"$_RUNJAVA¥" ¥"$LOGGING_CONFIG¥" $JAVA_OPTS $CATALINA_OPTS ¥
    -Djava.endorsed.dirs=¥"$JAVA_ENDORSED_DIRS¥" -classpath ¥"$CLASSPATH¥" ¥
    -Dcatalina.base=¥"$CATALINA_BASE¥" ¥
    -Dcatalina.home=¥"$CATALINA_HOME¥" ¥
    -Djava.io.tmpdir=¥"$CATALINA_TMPDIR¥" ¥
    org.apache.catalina.startup.Bootstrap "$@" start
fi
```

4 Java コマンド・ライン（またはコード・ブロック）の直上に 2 行を追加します。1 行目は JRE Instrumenter を実行する Java コマンドを呼び出します。2 行目は必要な JVM パラメータを追加します。使用する JVM パラメータが不明な場合は，後の手順 6 で追加できます。この 2 つの行で名前を入力し，自動的にインストルメントされる JRE クラスを格納するためのディレクトリを指定します。次の例では，**MyServer** を自分で選択した名前に置き換えてください。

JRE バージョン 5 以降を使用する Tomcat サーバの **catalina.bat** ファイルに追加される 2 行の例を次に示します。

```
%_EXECJAVA% -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f MyServer

set JAVA_OPTS=%JAVA_OPTS% -Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
```

JRE バージョン 5 以降を使用する Tomcat サーバの **catalina.sh** ファイルの例を次に示します。

```
"$_RUNJAVA" -jar /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/ jreinstrumenter.jar -f MyServer

JAVA_OPTS="$JAVA_OPTS -Xbootclasspath/p:/opt/MercuryDiagnostics/JavaAgent/ DiagnosticsAgent/classes/
MyServer/instr.jre -javaagent:/opt/MercuryDiagnostics/ JavaAgent/DiagnosticsAgent/lib/probeagent.jar"
```

JRE 1.4 を使用する Tomcat サーバの **catalina.bat** ファイルの例を次に示します。

```
%_EXECJAVA% -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f MyServer

set JAVA_OPTS=%JAVA_OPTS% -Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre;C:¥Mercur
yDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥boot
```

注 : Java コマンド・ラインで JVM パラメータの定義に JAVA_OPTS 変数を使用しない場合, これらの例で示している変数名 JAVA_OPTS を正しい名前に変更する必要があります。

---

5 変更を保存してアプリケーション・サーバを再起動します。

6 起動スクリプトの実行結果で，JRE Instrumenter からの出力を検索します（Xbootclasspath を検索）。手順 4 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加した場合，それを JRE Instrumenter の出力にある JVM パラメータと比較します。これらが同じでない場合，JRE Instrumenter で出力された正しい JVM パラメータを使用して起動スクリプトを更新し, アプリケーション・サーバを再起動します。手順 4 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加していない場合，ここで追加してアプリケーション・サーバを再起動します。例については手順 4 を参照してください。

7 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞ ¥log¥ ＜ Probe ID ＞ ¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，JRE のインストルメンテーションに失敗したか，JVM パラメータを正しく設定しなかったかのいずれかです。詳細については，219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照してください。

起動スクリプトを使用しない Tomcat サーバの設定

次の手順では，Windows サービスとして実行している Tomcat サーバを設定する方法について説明します。

起動スクリプトを使用しないで Tomcat サーバを設定するには，次の手順を実行します。

1 Windows タスク・バーで，Apache Tomcat サービス・アイコンを右クリックし，［**設定**］を選択します。または，［スタート］メニューからも選択できます。たとえば，［**プログラム**］＞［**Apache Tomcat 7.0**］＞［**Configure Tomcat**］。

2 ［Apache Tomcat Properties］ダイアログ・ボックスで，［**Java**］タブを選択します。

3 ［**Java Options**］ボックスで，次のように 2 つの JVM パラメータを追加します。最初のパラメータ（-javaagent）では，アプリケーション・サーバの JVM の起動時に Java Agent を起動します。最初の呼び出しで，2 番目のパラメータ（-Xbootclasspath）では，アプリケーション・サーバの JRE をインストルメントします。-Xbootclasspath パラメータで名前を入力し，インストルメントされるクラスを格納するディレクトリの名前を指定します。次の例では，**MyServer** を自分で選択した名前に置き換えてください。

```
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥auto¥MyServer¥instr.jre
```

重要 : 各 JVM パラメータは独自の行で指定する必要があります。

---

4 Tomcat サービスを再起動します。

Java Agent が起動し，JRE Instrumenter が暗黙的に実行されて JRE がインストルメントされます。

5 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/log/ ＜Probe ID＞ /probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合，JVM パラメータを正しく設定していない可能性があります。**＜ Tomcat のインストール・ディレクトリ＞ /logs/catalina. ＜日付＞ .log** ファイルでエラー・メッセージを確認します。＜日付＞は今日の日付です。

6 必要に応じて，インストルメントされた JRE が使用されるように，アプリケーション・サーバを再起動します。

重要 : 今後 Tomcat サーバで使用する JRE を更新する場合, Tomcat サーバを再起動する前に, 新しい JRE がインストルメントされるように**< Java Agent のインストール・ディレクトリ> /DiagnosticsAgent/classes/auto/MyServer** ディレクトリ（MyServer は自分のディレクトリ名に置き換えます）を削除する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。使用するインストルメンテーション・モードに関する一般的な情報については, 222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照してください。

例 7 : WebLogic の設定

WebLogic アプリケーション・サーバは，シェルまたはコマンド・スクリプトで起動します。したがって，インストルメンテーションを実行するように起動スクリプトを変更することをお勧めします。

WebLogic が提供する起動スクリプトは, サイト管理者によって頻繁にカスタマイズされるため，すべての状況に適用される詳細な設定手順を説明するのは不可能です。代わりに，次の項では，WebLogic アプリケーション・サーバの一般的な実装の具体的な例を含む手順を紹介します。サイト管理者は，これらの手順を使って，カスタマイズされた環境で変更を行う方法を示すことができます。

WebLogic サーバを設定するには，自動明示インストルメンテーション・モードを使用します（219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照）。これには 2 つのタスクが伴います。

- アプリケーション・サーバで使用されるのと同じ JRE を使用して JRE Instrumenter を実行するように，アプリケーション・サーバの起動スクリプトを変更します。JRE Instrumenter からの出力を見れば，次のタスクに必要な JVM パラメータがわかります。
- JRE Instrumenter からの出力で見つかるアプリケーション・サーバの JVM パラメータを設定します。

WebLogic アプリケーション・サーバを設定するには，次の手順を実行します。

1 ドメインに対して WebLogic を起動するのに使う起動スクリプトを特定します。

➤ WebLogic 9.0 以降の場合

通常，起動スクリプト・ファイルは，次のようなパスにあります。

＜ドメイン・ホーム＞ ¥bin¥startWebLogic.[cmd|sh]

＜ドメイン・ホーム＞は，**C:¥bea¥user_projects¥domains¥ ＜ドメイン名＞** または **C:¥bea¥wlserver_10.0¥samples¥domains¥ ＜ドメイン名＞**（**＜ドメイン名＞**はドメインの名前）などのドメイン・ディレクトリへのパスです。

たとえば，ドメイン名が **MedRec** の場合，パスは次のようになります。

C:¥bea¥wlserver_10.0¥samples¥domains¥medrec¥bin¥startWebLogic.cmd

➤ WebLogic 8.1 の場合

通常, 起動スクリプト・ファイルは, 次のいずれかのようなパスにあります。

＜ WLS ホーム＞ ¥server¥bin¥startWLS.[cmd|sh]

＜ WLS ホーム＞は，**C:¥bea¥weblogic81** などの WebLogic のインストール・ディレクトリへのパスです。

＜ドメイン・ホーム＞ ¥start ＜ドメイン名＞ Server.[cmd|sh]

＜ドメイン・ホーム＞は，**C:¥bea¥user_projects¥domains¥ ＜ドメイン名＞**または **C:¥bea¥weblogic81¥samples¥domains¥ ＜ドメイン名＞**（**＜ドメイン名＞**はドメインの名前）などのドメイン・ディレクトリへのパスです。

たとえば，ドメイン名が **MedRec** の場合，パスは次のようになります。

C:¥bea¥weblogic81¥samples¥domains¥medrec¥startMedRecServer.cmd

2 起動スクリプトのバックアップ・コピーを作成し，エディタで起動スクリプトを開きます。

3 アプリケーション・サーバを起動する **Java** コマンド・ラインを特定します。.cmd ファイルからの例を次に示します。

```
%JAVA_HOME%¥bin¥java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dweblogic.Name=%SERVER_NAME%
-Djava.security.policy=%WL_HOME%¥server¥lib¥weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS%
```

4 Java コマンド・ライン（またはコード・ブロック）の直上に 2 行を追加します。1 行目は JRE Instrumenter を実行する Java コマンドを呼び出します。2 行目は必要な JVM パラメータを追加します。使用する JVM パラメータが不明な場合は，後の手順 6 で追加できます。この 2 つの行で名前を入力し，自動的にインストルメントされる JRE クラスを格納するためのディレクトリを指定します。次の例では，**MedRec** を自分で選択した名前に置き換えてください。

WebLogic 9.x 以降の .cmd ファイルに追加される 2 行の例を次に示します。

```
%JAVA_HOME%¥bin¥java -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f
MedRec

set JAVA_OPTIONS=%JAVA_OPTIONS% -Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MedRec¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
```

WebLogic 9.x 以降の .sh ファイルに追加される 2 行の例を次に示します。

```
${JAVA_HOME}/bin/java -jar /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/jreinstrumenter.jar -f
MedRec

JAVA_OPTIONS="$JAVA_OPTIONS -Xbootclasspath/p:/opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/
classes/MedRec/instr.jre -javaagent:/opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/probeagent.jar"
```

WebLogic 8.1 の .cmd ファイルに追加される 2 行の例を次に示します。

```
%JAVA_HOME%¥bin¥java -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f
MedRec

set JAVA_OPTIONS=%JAVA_OPTIONS% -Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MedRec¥instr.jre;C:¥MercuryDiagnostics¥Jav
aAgent¥DiagnosticsAgent¥classes¥boot
```

注 :

- Java コマンド・ラインで JVM パラメータの定義に JAVA_OPTIONS 変数を使用しない場合，これらの例で示している変数名 JAVA_OPTIONS を正しい名前に変更する必要があります。
- JRockit JRE を使用する WebLogic 8.1 では，次の JVM パラメータを 2 行目の最後に追加します。
  -Xmanagement:class=com.mercury.opal.capture.proxy.JRockitManagement

---

5 変更を起動スクリプトに保存し，変更されたスクリプトを使用してアプリケーション・サーバを再起動します。

6 起動スクリプトの実行結果で，JRE Instrumenter からの出力を検索します（Xbootclasspath を検索）。手順 4 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加した場合，それを JRE Instrumenter の出力にある JVM パラメータと比較します。これらが同じでない場合，JRE Instrumenter で出力された正しい JVM パラメータを使用して起動スクリプトを更新し，アプリケーション・サーバを再起動します。手順 4 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加していない場合，ここで追加してアプリケーション・サーバを再起動します。例については手順 4 を参照してください。

注 : JRE Instrumenter の出力が見つからない場合，またはエラー・メッセージがある場合，JRE は適切にインストルメントされない可能性があります。この場合は起動スクリプトを確認し，問題を解決します。

---

7 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞¥DiagnosticsAgent¥log¥＜Probe ID＞¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，JRE のインストルメンテーションに失敗したか，JVM パラメータを正しく設定しなかったかのいずれかです。詳細については，219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照してください。

例 8 : WebSphere の設定

WebSphere アプリケーション・サーバは，UNIX 環境ではシェル・スクリプトまたは Node Agent で起動します。Windows 環境では，Windows サービスとしてインストールできますが，Windows コマンド・スクリプトからも起動できます。したがって，JRE Instrumenter を自動明示モードで実行してアプリケーション・サーバで使用する JRE をインストルメントするように，起動スクリプトを変更することをお勧めします。

起動スクリプトを変更できない場合，または WebSphere アプリケーション・サーバが z/OS で実行されている場合，手動で JRE Instrumenter を実行するか，WebSphere バージョンが 7.0 以降の場合は自動暗黙モードで JRE Instrumenter を使用します。後のほうのシナリオでは，JVM パラメータのみを追加する必要があります。詳細については，222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照してください。

その一方で，WebSphere アプリケーション・サーバの JVM パラメータが起動スクリプトではなく設定ファイルによって制御されるため，統合ソリューション・コンソール（古いバージョンでは WebSphere アプリケーション・サーバ管理コンソールとも呼ばれています）を使用して，Java Agent の起動およびインストルメントされた JRE の使用に必要な JVM パラメータを追加することをお勧めします。

WebSphere のバージョンによって，コンソールの外観が異なる場合があります。その結果，次の例はお使いの WebSphere のバージョンに正確に対応していない可能性がありますが，必要な情報は備えています。コンソールの該当する場所に，Diagnostics の監視に必要なパラメータを入力します。

WebSphere 5.1 / 6.0（以下を参照）と WebSphere 6.1 以降（203 ページ「WebSphere 6.1 以降を設定するには，次の手順を実行します。」を参照）のための手順を示します。209 ページ「WebSphere 6.1/7.0 サーバに JMX メトリックスの収集を設定するには，次の手順を実行します。」も参照してください。

WebSphere 5.1 または 6.0 を設定するには，次の手順を実行します。

1 WebSphere アプリケーション・サーバの起動に使用するスクリプトを特定します。

例 : **＜ WAS のインストール・ディレクトリ＞ ¥bin¥startServer.bat**

＜ WAS のインストール・ディレクトリ＞は，**C:¥Program Files¥IBM ¥WebSphere¥AppServer** などの WebSphere のインストール・ディレクトリへのパスです。

注 : システムによっては，サーバを起動するプロファイルの bin ディレクトリで startServer.[bat|sh] スクリプトを使用できます。ただし，通常このスクリプトは**＜ WAS のインストール・ディレクトリ＞ /bin** ディレクトリの **startServer.[bat|sh]** スクリプトを呼び出す単なるラッパーです。

---

2 起動スクリプトのバックアップ・コピーを作成し，エディタで起動スクリプトを開きます。

3 WebSphere 5.1 または 6.0 では，アプリケーション・サーバ・ランチャを実行する Java コマンド・ラインを特定します。**startServer.bat** ファイルからの例を次に示します。

```
"%JAVA_HOME%¥bin¥java" -Dcmd.properties.file=%TMPJAVAPROPFILE%
%WAS_TRACE% %WAS_DEBUG% %CONSOLE_ENCODING% "%CLIENTSAS%"
"%CLIENTSSL%" %USER_INSTALL_PROP% "-Dwas.install.root=%WAS_HOME%"
com.ibm.ws.bootstrap.WSLauncher
com.ibm.ws.management.tools.WsServerLauncher "%CONFIG_ROOT%"
"%WAS_CELL%" "%WAS_NODE%" %* %WORKSPACE_ROOT_PROP%
```

識別された Java コマンド・ラインの上に，JRE Instrumenter を起動する行を追加します。この行では，インストルメントされるクラスを格納するディレクトリの名前を指定する必要があります。Windows 環境での WebSphere 6.0 の例を次に示します。**MyServer** を自分で選択した名前に置き換えてください。

```
"%JAVA_HOME%¥bin¥java" -jar
C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f
MyServer
```

4 変更を保存し，変更されたスクリプトを使用してアプリケーション・サーバを再起動します。

5 起動スクリプトの実行結果で，JRE Instrumenter からの出力を検索します（Xbootclasspath を検索）。

Windows での WebSphere 6.0（または 5.1）の JRE Instrumenter の出力例（-Xbootclasspath）を示します。この例では，**-f MyServer** を使用してインストルメントされるクラスを格納するディレクトリを指定しています。上記の手順 3 を参照してください。MyServer を自分で選択した名前に置き換えてください。

```
-Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre;C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥boot
```

JRE Instrumenter の出力が見つからない場合，またはエラー・メッセージがある場合，JRE は適切にインストルメントされない可能性があります。この場合は起動スクリプトを確認し，問題を解決します。詳細については，219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照してください。

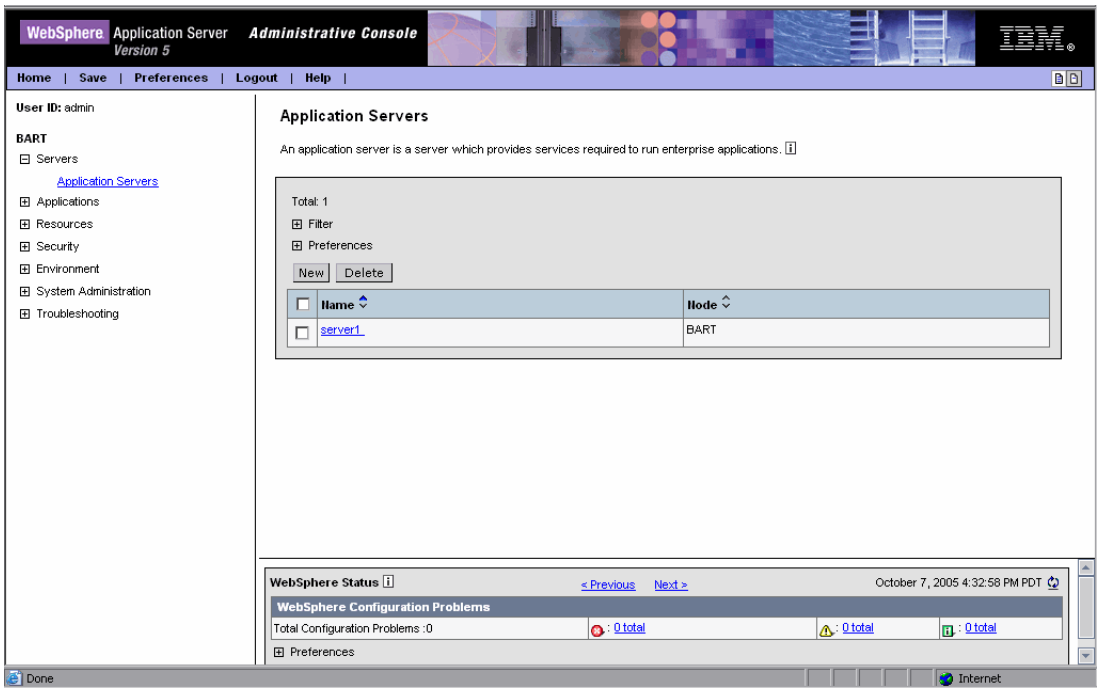
JRE Instrumenter からの出力を取得したら，その出力をアプリケーション・サーバの JVM パラメータに追加する必要があります。

6 Web ブラウザを使用して，プローブが監視されているアプリケーション・サーバ・インスタンスの WebSphere 管理コンソールにアクセスします。

```
http:// <アプリケーション・サーバのホスト名> :9090/admin
```

<アプリケーション・サーバのホスト名>をアプリケーション・サーバ・ホストのマシン名に置き換え，必要に応じて 9090 を正しい管理ポート番号（9060，9061 など）に置き換えます。

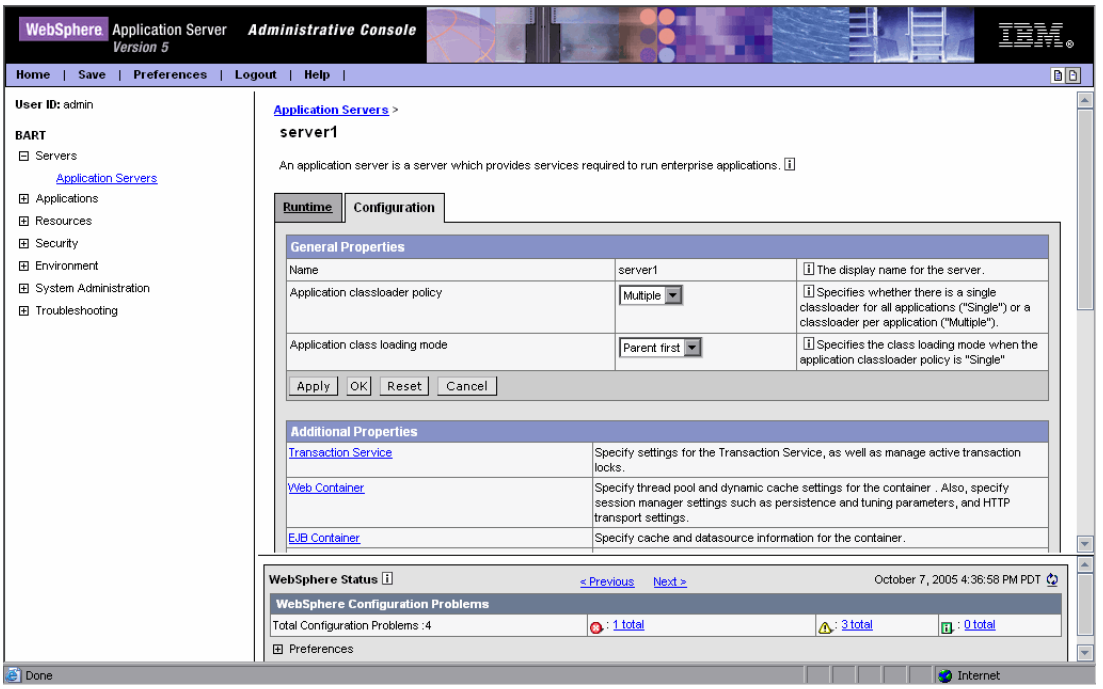
WebSphere アプリケーション・サーバ管理コンソールが開きます。



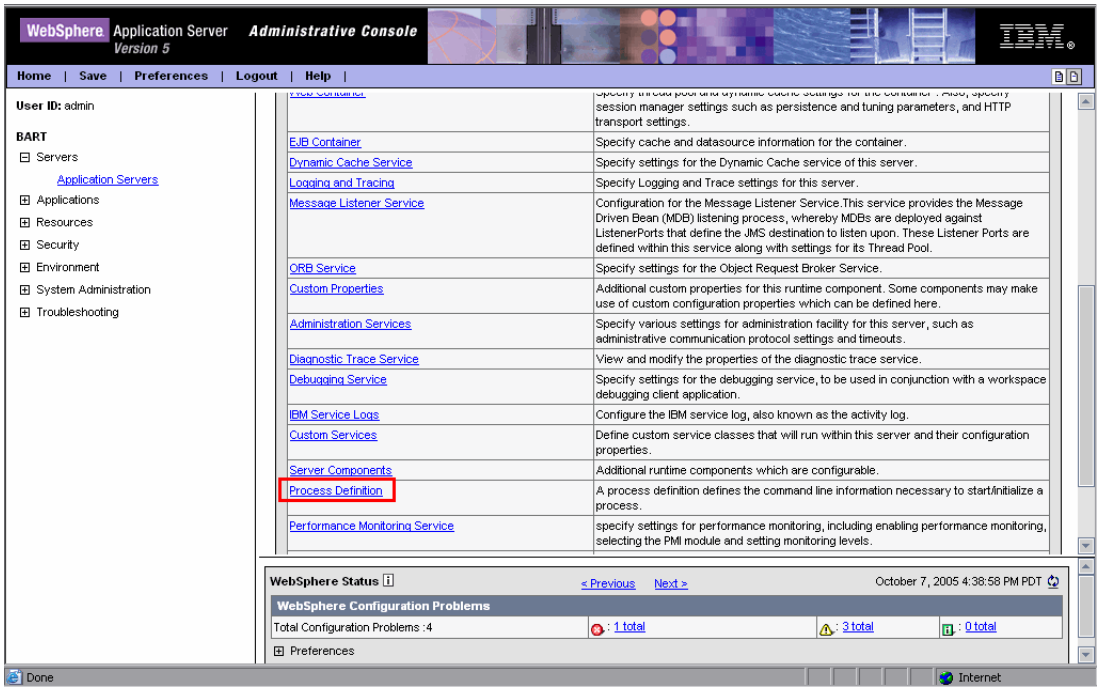
7 左のパネルで，［**Servers**］>［**Application Servers**］を選択します。

8 右のパネルのアプリケーション・サーバのリストから，プローブによって監視されるように設定するサーバの名前を選択します。

選択したアプリケーション・サーバの［Configuration］タブが表示されます。

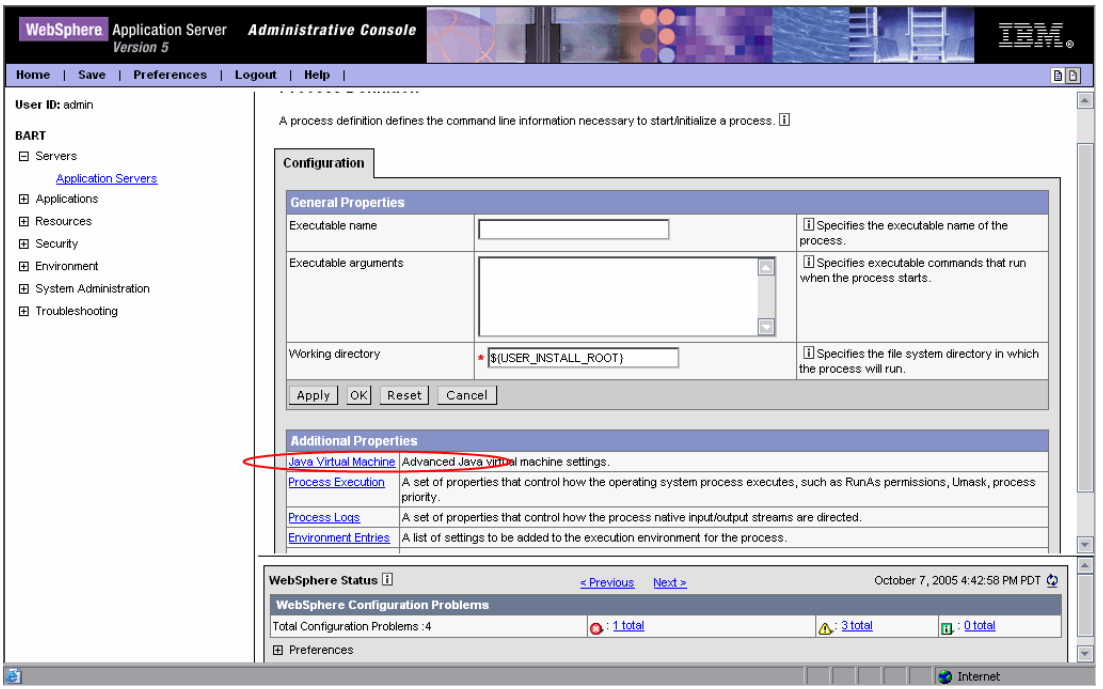


9［Configuration］タブを下にスクロールし，［**General Properties**］列で，［**Process Definition**］プロパティを探します。



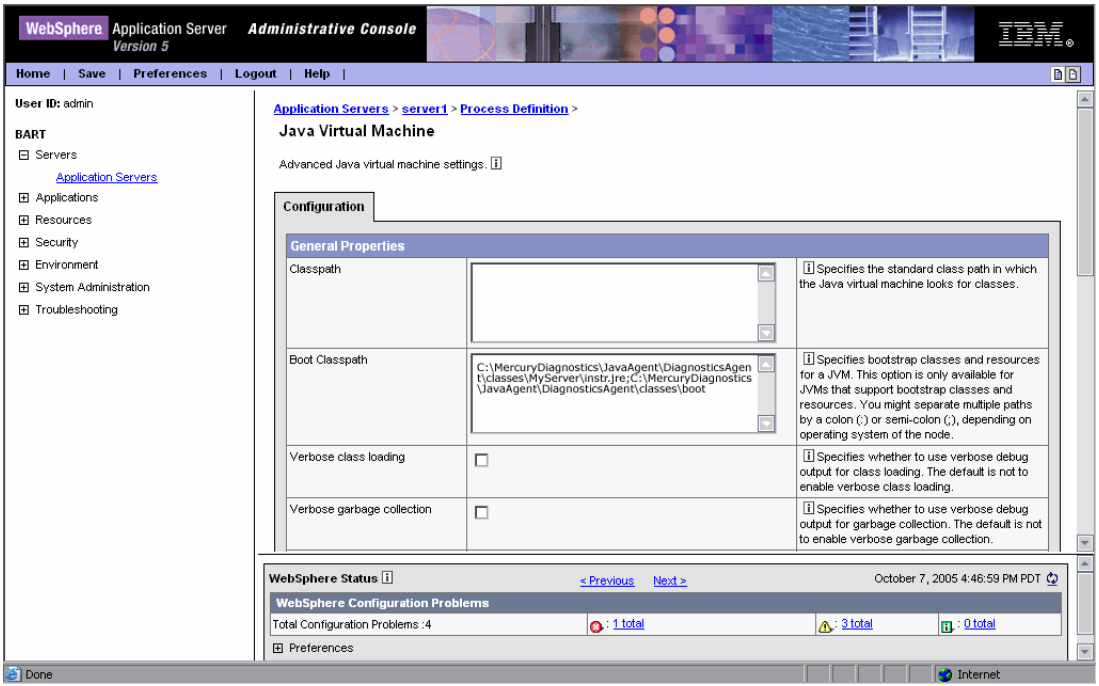
10［**Process Definition**］をクリックします。

11 右のパネルを下にスクロールして，**Java Virtual Machine** を探します。



12［**Java Virtual Machine**］をクリックします。

13 Java 仮想マシンの［Configuration］タブが表示されます。



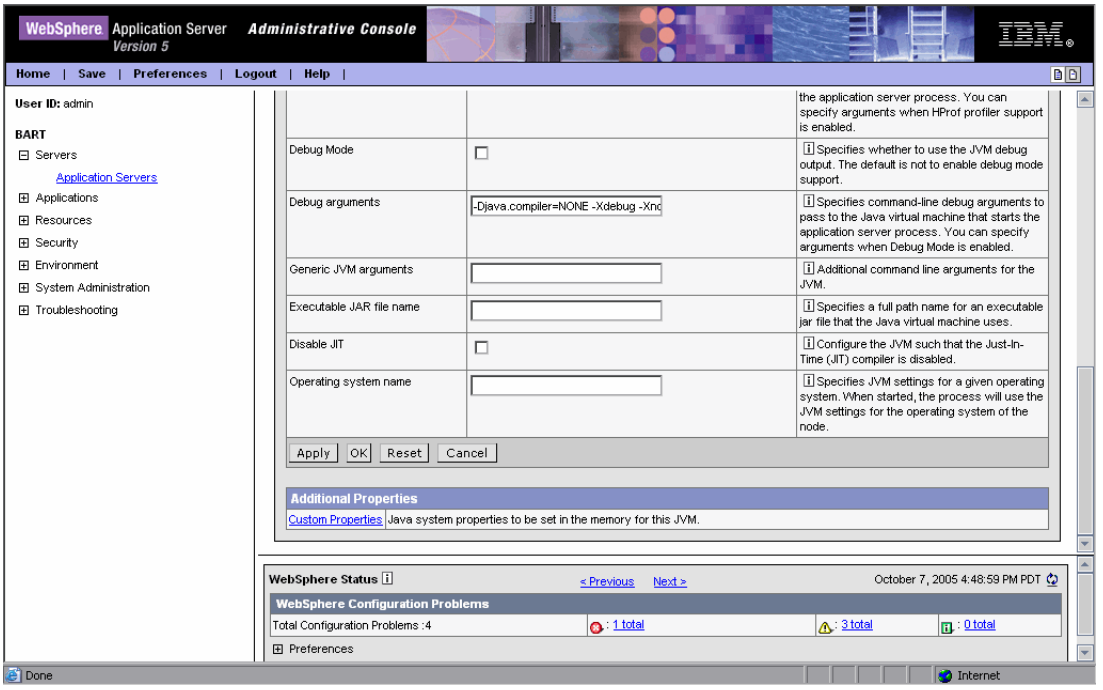
14 ［**Boot Classpath**］ボックスで，JRE Instrumenter で出力された JVM パラメータからのブート・クラス・パス（-Xbootclasspath/p: パラメータの後の文字列）を入力します。JRE Instrumenter からの JVM パラメータ出力は，前の手順 3 で生成されたものです。

-f MyServer を使用してインストルメントされるクラスを格納するディレクトリの名前を指定する WebSphere 6.0（または 5.1）の例を次に示します。

```
C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre;C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥boot
```

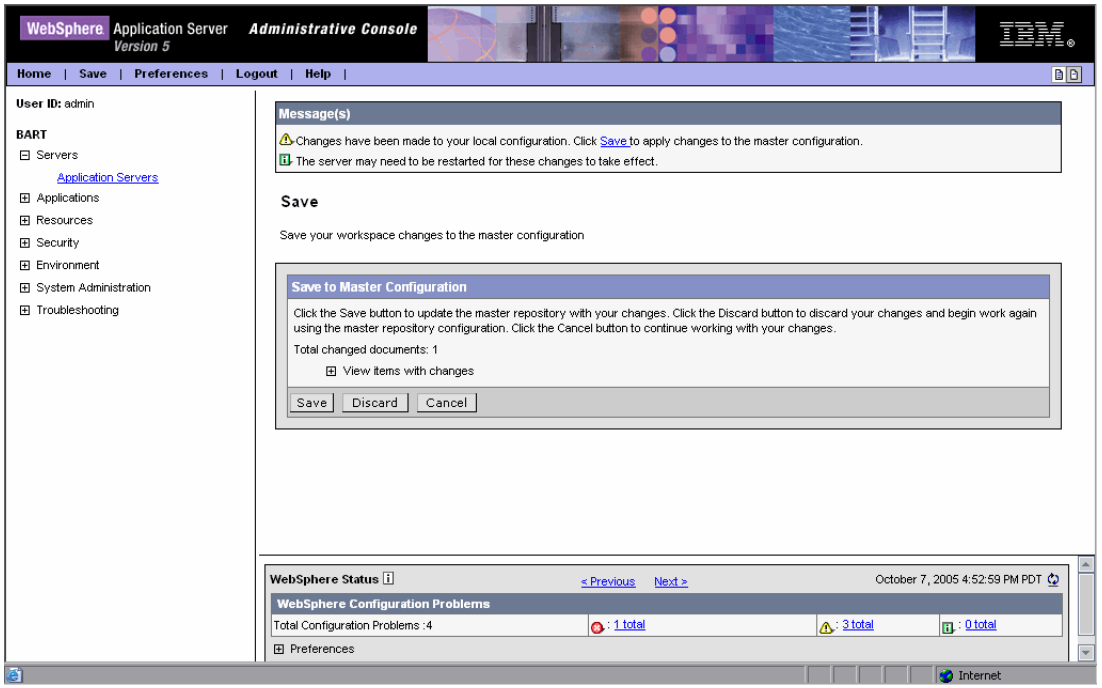
注 : 32 ビットの WebSphere 6.0 JRE を使用する 64 ビットのプラットフォーム（OS）では，マニュアル・モードで JRE Instrumenter を使用した場合，［**Generic JVM arguments**］ボックスに **-Xj9** も追加する必要があります。

15 コマンド・ボタンが表示されるまで［Configuration］タブを一番下にスクロールします。



［**Apply**］をクリックします。

16 変更が適用されたことを伝えるメッセージが表示されます。[**Save to Master Configuration**] エリアで，[**Save**] をクリックします。



17 [**Save**] をクリックして，マスター設定に変更を適用します。確認を求められた場合，再び [**Save**] をクリックします。

18 WebSphere アプリケーション・サーバを再起動します。

19 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞¥DiagnosticsAgent¥log¥＜Probe ID＞¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，Java Agent を正しく起動していないか，JRE Instrumenter を実行していないか，Xbootclasspath などの Java パラメータを正しく入力しなかったかのいずれかです。詳細については，217 ページ「JRE Instrumenter とさまざまな起動オプションについて」を参照してください。

WebSphere 6.1 以降を設定するには，次の手順を実行します。

1 WebSphere アプリケーション・サーバの起動に使用するスクリプトを特定します。

例 : **＜ WAS のインストール・ディレクトリ＞ ¥bin¥startServer.bat**

＜WAS のインストール・ディレクトリ＞は，**C:¥Program Files¥IBM ¥WebSphere¥AppServer** などの WebSphere のインストール・ディレクトリへのパスです。

注 : システムによっては，サーバを起動するプロファイルの bin ディレクトリで startServer.[bat|sh] スクリプトを使用できます。ただし，通常このスクリプトは**＜WAS のインストール・ディレクトリ＞ /bin** ディレクトリの **startServer.[bat|sh]** スクリプトを呼び出す単なるラッパーです。

---

2 起動スクリプトのバックアップ・コピーを作成し，エディタで起動スクリプトを開きます。

3 WebSphere 6.1 以降では，JAVA_EXE 変数を定義するコード・ブロックを特定します。

```
if exist "%JAVA_HOME%¥bin¥java.exe" (
   set JAVA_EXE="%JAVA_HOME%¥bin¥java"
) else (
   set JAVA_EXE="%JAVA_HOME%¥jre¥bin¥java"
)
```

上記のコード・ブロックの下に，JRE Instrumenter を起動する行を追加します。この行では，インストルメントされるクラスを格納するディレクトリの名前を指定する必要があります。Windows 環境での WebSphere 6.1 の例を次に示します。**MyServer** を自分で選択した名前に置き換えてください。

```
%JAVA_EXE% -jar
C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f
MyServer
```

4 変更を保存し，変更されたスクリプトを使用してアプリケーション・サーバを再起動します。

5 起動スクリプトの実行結果で，JRE Instrumenter からの出力を検索します（Xbootclasspath を検索）。

WebSphere 6.1 : Windows での WebSphere 6.1 の JRE Instrumenter の出力例（-Xbootclasspath）を示します。この例では，**-f MyServer** を使用してインストルメントされるクラスを格納するディレクトリを指定しています（上記の手順 3 を参照）。MyServer を自分で選択した名前に置き換えてください。

```
-Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre;C:¥MercuryDiagnostics
¥JavaAgent¥DiagnosticsAgent¥classes¥boot -Xshareclasses:none
```

WebSphere 7.0 または 8.0 : Windows での WebSphere 7.0（または 8.0）の JRE Instrumenter の出力例（-Xbootclasspath）を示します。この例では，**-f MyServer** を使用してインストルメントされるクラスを格納するディレクトリを指定しています（上記の手順 3 を参照）。MyServer を自分で選択した名前に置き換えてください。

```
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar -Xshareclasses:none
```

JRE Instrumenter の出力が見つからない場合，またはエラー・メッセージがある場合，JRE は適切にインストルメントされない可能性があります。この場合は起動スクリプトを確認し，問題を解決します。詳細については，219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照してください。

JRE Instrumenter からの出力を取得したら，その出力をアプリケーション・サーバの JVM パラメータに追加する必要があります。

6 WebSphere アプリケーション・サーバ管理コンソールを開きます。次に例を示します。

http:// ＜アプリケーション・サーバのホスト名＞ :9060/ibm/console

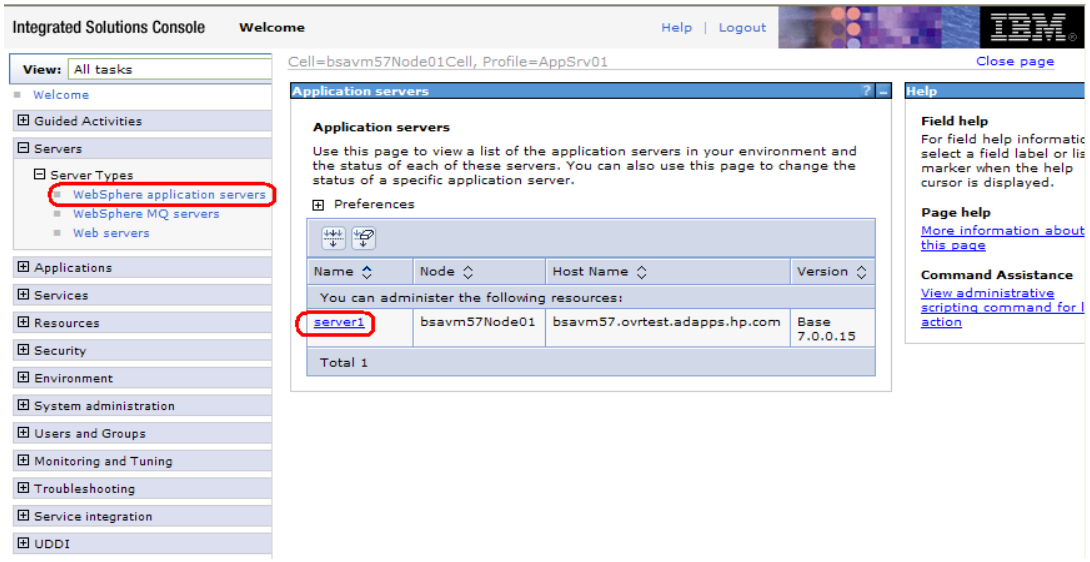
＜アプリケーション・サーバのホスト名＞をアプリケーション・サーバ・ホストのマシン名に置き換え，9060 を正しい管理ポート番号（9060，9061 など）に置き換えます。

7 ［Java 仮想マシン］ページに移動します。次に例を示します。

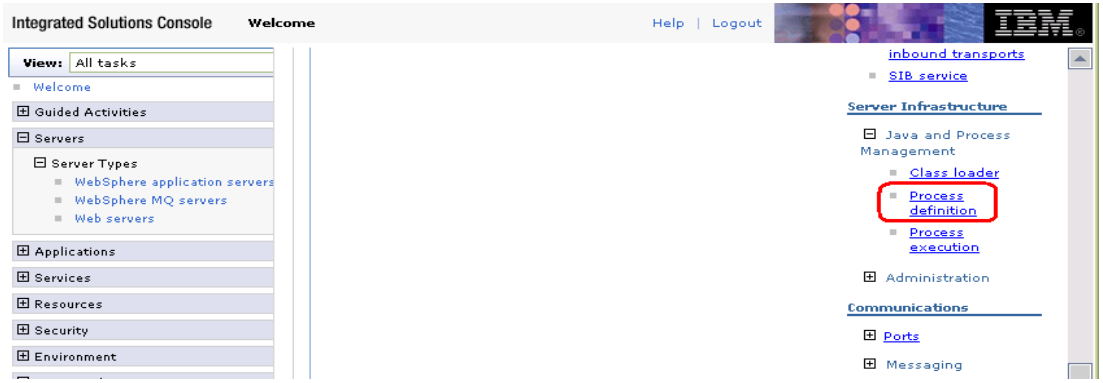
WebSphere 6.1 の場合，［**Servers**］>［**Application servers**］に移動します。

WebSphere 7.0 の場合，［**Servers**］>［**Server Types**］>［**WebSphere Application servers**］に移動します。

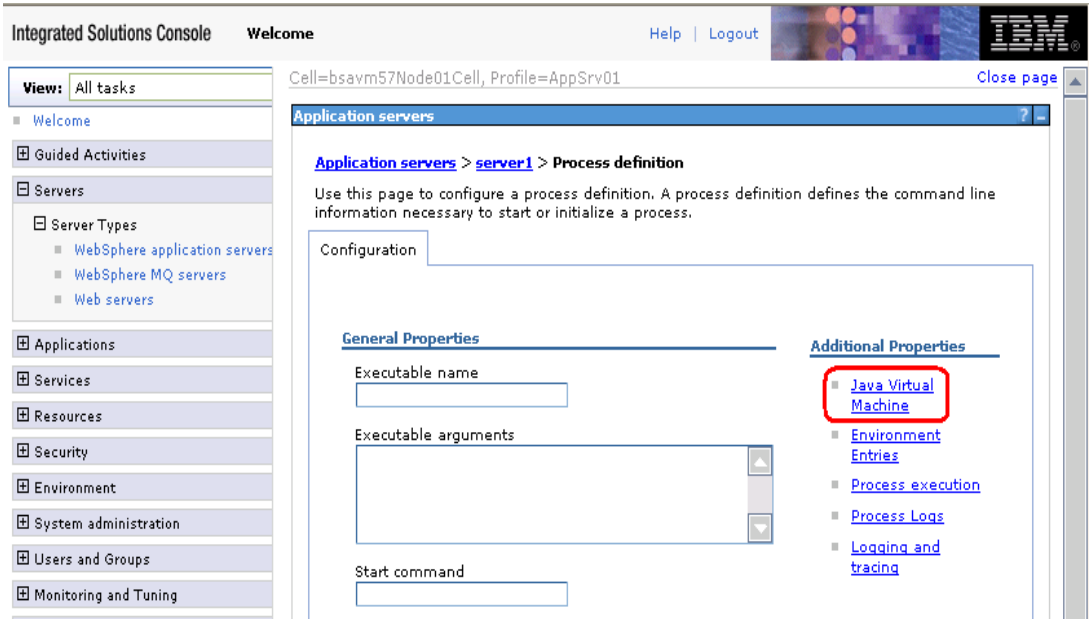
次に，アプリケーション・サーバのインスタンス名（**server1** など）をクリックします。



次に，［**Server Infrastructure**］＞［**Java and Process Management**］で［**Process Definition**］＞［**Java Virtual Machine**］をクリックします。



次に，[**Additional Properties**] で [**Java Virtual Machine**] をクリックします。



8 [Java Virtual Machine] ページの [**Generic JVM Arguments**] ボックスに，JRE instrumenter からの JVM パラメータを入力します。JRE Instrumenter からの JVM パラメータ出力は，前の手順 3 で生成されたものです。

-f MyServer を使用してインストルメントされるクラスを格納するディレクトリの名前を指定する WebSphere 6.1 の例を次に示します。

```
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre;
C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥boot -Xshareclasses:none
```

起動スクリプトを変更せずに，JRE Instrumenter を手動で使用して JRE をインストルメントする WebSphere 6.1 の例を次に示します。

```
-Xbootclasspath/
p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥IBM¥1.5.0¥instr.jre;C:¥MercuryDiagnostics¥Ja
vaAgent¥DiagnosticsAgent¥classes¥boot -Xshareclasses:none
```

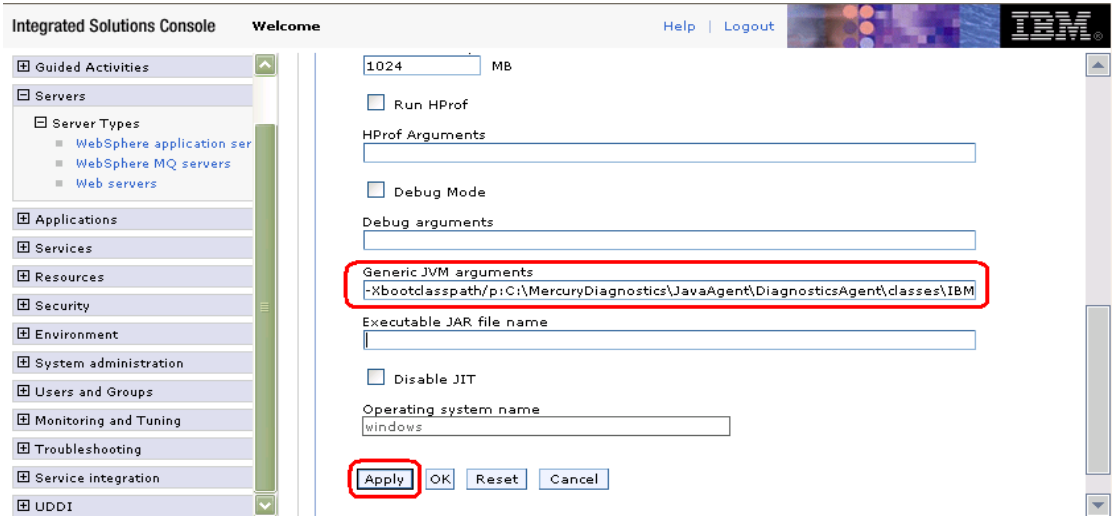
起動スクリプトで JRE Instrumenter へのコマンド・ライン・オプションとして **-f MyServer** を使用する WebSphere 7（または 8）の例を次に示します。

```
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar -Xshareclasses:none
```

起動スクリプトを変更しないか，JRE Instrumenter を手動で実行（自動暗黙モードで JRE Instrumenter を使用）する WebSphere 7（または 8）の例を次に示します。

```
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥auto¥server1¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar -Xshareclasses:none
```

9 変更を適用して保存します。



10 WebSphere アプリケーション・サーバを再起動します。

11 プローブが正しく設定されていることを確認するには，**＜Java Agent のインストール・ディレクトリ＞/DiagnosticsAgent/log/＜Probe ID＞/probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，Java Agent を正しく起動していないか，JRE Instrumenter を実行していないか，Xbootclasspath などの Java パラメータを正しく入力しなかったかのいずれかです。詳細については，217 ページ「JRE Instrumenter とさまざまな起動オプションについて」を参照してください。

WebSphere IDE での JRE Instrumenter の実行

WebSphere IDE を使用している場合，WSAD IDE の正しい Java 実行可能ファイルがインストルメントされたことを確かめるために，JRE Instrumenter を手動で実行する必要があります。

WSAD IDE には，異なる java.exe 実行可能ファイルがあり選択できます。WebSphere の実行に使用する実行可能ファイルをインストルメントする必要があります。

正しい java.exe をインストルメントするには，次の手順を実行します。

1 使用している WebSphere のバージョンを確認します。

2 適切な java.exe の場所を特定します。

3 JRE Instrumenter を実行して正しい JVM を追加します。詳細については，217 ページ「JRE Instrumenter とさまざまな起動オプションについて」を参照してください。

JMX メトリックの収集のための WebSphere の設定

JMX メトリックスの受信を開始するために，場合によっては，WebSphere サーバに Performance Monitoring Infrastructure（PMI）サービスを設定する必要があります。

重要 : Diagnostics がお使いのアプリケーション・サーバを WebSphere サーバとして識別できない場合は，PMI を有効にし，server.policy ファイルに Jar ファイルを追加する必要があります。

WebSphere 6.1/7.0 サーバに JMX メトリックスの収集を設定するには，次の手順を実行します。

1 WebSphere 管理コンソールを開きます。

2 コンソールのナビゲーション・ツリーで，［**Servers**］＞［**Application Servers**］を選択します。コンソールに，アプリケーション・サーバの表が表示されます。

3［Application Servers Table］で，設定するアプリケーション・サーバの名前をクリックします。コンソールに，選択したアプリケーション・サーバの［**Runtime**］タブと［**Configuration**］タブが表示されます。

4［**Configuration**］タブをクリックします。

5［**Configuration**］で次の操作を行います。

➤［Performance］ヘッダで，［**Performance Monitoring Infrastructure (PMI)**］をクリックします。

➤［General Properties］ヘッダで，［**Enable Performance Monitoring Infrastructure (PMI)**］チェック・ボックスを選択します。

➤［Currently monitored statistic set］ヘッダで，［**Extended**］を選択します。

6［**Apply**］または［**OK**］をクリックします。

7 アプリケーション・サーバで Java 2 Security が有効になっている場合は，サーバ・ポリシー・ファイル（**＜ WebSphere 6.x のインストール・ディレクトリ＞ /work/tools/ibm-6.0/websphere/appserver/profiles/default/properties/server.policy** または **＜ WebSphere 7.0 のインストール・ディレクトリ＞ / AppServer/profiles/ ＜ご使用のプロファイル名＞ /properties/server.policy**）を開き，次のセキュリティ権限を追加して，JMX の収集を有効にします。

```
grant codeBase "file:/<probe_install_dir>/lib/probe-jmx.jar"
{ permission java.security.AllPermission; }

grant codeBase "file:/<probe_install_dir>/lib/probe-jmx-was6.jar" {
  permission java.security.AllPermission;
};
```

8 アプリケーション・サーバを再起動します。

例 9 : webMethods の設定

この例で説明する webMethods サーバには 2 つのタイプがあります。

➤ webMethods Integration Server

➤ My webMethods Server

webMethods が提供する起動スクリプトは，サイト管理者によって頻繁にカスタマイズされるため，すべての状況に適用される詳細な設定手順を説明するのは不可能です。代わりに，次の項では，webMethods Integration Server と My webMethods Server の具体的な例を含む一般的な手順を紹介します。サイト管理者は，これらの手順を使って，カスタマイズされた環境で変更を行う方法を示すことができます。

webMethods Integration Server を設定するには，次の手順を実行します。

webMethods Integration Server は，シェルまたはコマンド・スクリプトで起動します。したがって，サーバのインストルメントを実行するように起動スクリプトを変更することをお勧めします。

1 webMethods Integration Server を起動するのに使う起動スクリプトを特定します。サーバの起動方法に基づいて 2 つのオプションがあります。

...¥IntegrationServer¥bin¥server.bat

...¥profiles¥IS¥bin¥runtime.bat

2 起動スクリプトのバックアップ・コピーを作成し，エディタで起動スクリプトを開きます。

3 次に説明するようにファイルを更新します。

a **server.bat** ファイルで，サーバを起動した次の部分を見つけます。

```
if "1%1"=="1-service" (
        if exist LOCKFILE del LOCKFILE
        "%JAVA_EXEC%" -classpath %IS_PROXY_JAR%
com.wm.app.server.CustomServiceUpdater -isdir "%IS_DIR%" -wrapperdir
"%IS_DIR%¥..¥profiles¥IS¥configuration" -binpath "%PATH%" -jvmargs
"%SERVER_VM_OPT% %JAVA2_MEMSET% %JAVA_OPTS%" -progargs
%3#%4#%5#%6#%7#%8#%9
        goto :EOF
)

call "%PROFILES_DIR%¥bin¥start_runtime.bat" %1 %2 %3 %4 %5 %6 %7 %8 %9
```

さらに，この部分のすぐ上に次を追加します。

```
%JAVA_HOME%¥bin¥java -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f
MyServer

set JAVA_OPTIONS=%JAVA_OPTIONS%
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
```

1 行目は JRE Instrumenter を実行する Java コマンドを呼び出します。
2 行目は必要な JVM パラメータを追加します。2 行目に使用する JVM パラメータが不明な場合は，後の手順 5 で追加できます。この 2 つの行で名前を入力し，自動的にインストルメントされる JRE クラスを格納するためのディレクトリを指定します。上記の例では，**MyServer** を自分で選択した名前に置き換えてください。

b **runtime.bat** ファイルで，サーバを起動した次の部分を見つけます。

```
%JAVA_RUN% -Xbootclasspath/a:"%OSGI_CLASSPATH%" %JAVA_OPTS%
%JAVA_SYSPROPS% -cp "%OSGI_FRAMEWORK_JAR%"
org.eclipse.equinox.launcher.Main -configuration %OSGI_CONFIGURATION_AREA%
%CMD_ARGS%
goto end_start_cmd
```

さらに，この部分のすぐ上に例で示している 2 つの行を追加します。

```
%JAVA_HOME%¥bin¥java -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f
MyServer

set JAVA_OPTIONS=%JAVA_OPTIONS%
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
```

1 行目は JRE Instrumenter を実行する Java コマンドを呼び出します。
2 行目は必要な JVM パラメータを追加します。2 行目に使用する JVM パラメータが不明な場合は，後の手順 5 で追加できます。この 2 つの行で名前を入力し，自動的にインストルメントされる JRE クラスを格納するためのディレクトリを指定します。上記の例では，**MyServer** を自分で選択した名前に置き換えてください。

4 変更を起動スクリプトに保存し，変更されたスクリプトを使用してアプリケーション・サーバを再起動します。

5 起動スクリプトの実行結果で，JRE Instrumenter からの出力を検索します（Xbootclasspath を検索）。手順 3 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加した場合，それを JRE Instrumenter の出力にある JVM パラメータと比較します。これらが同じでない場合，JRE Instrumenter で出力された正しい JVM パラメータを使用して起動スクリプトを更新し，アプリケーション・サーバを再起動します。手順 3 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加していない場合，ここで追加してアプリケーション・サーバを再起動します。

注 : JRE Instrumenter の出力が見つからない場合，またはエラー・メッセージがある場合，JRE は適切にインストルメントされない可能性があります。この場合は起動スクリプトを確認し，問題を解決します。

---

6 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞¥DiagnosticsAgent¥log¥＜Probe ID＞¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，JRE のインストルメンテーションに失敗したか，JVM パラメータを正しく設定しなかったかのいずれかです。詳細については，219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照してください。

My webMethods Server 起動スクリプトを設定するには，次の手順を実行します。

My webMethods Server は，スクリプトまたはラッパー設定で起動します。したがって，この例のように起動スクリプトを変更するか，次の手順で説明しているようにラッパー設定ファイルを編集してサーバをインストルメントすることになります。

1 My webMethods Server を起動するのに使う起動スクリプトを特定します。起動スクリプトは ...¥MWS¥server¥bin¥mws.bat です。

2 このファイルのバックアップ・コピーを作成し，エディタでファイルを開きます。

3 次に説明するようにファイルを更新します。

mws.bat ファイルで，次の例で強調表示している RUN_CMD の定義を見つけます。

```
set JAVA_OPTIONS=%JAVA_OPTIONS% -Dserver.name=%SERVER_NAME%
-Djava.awt.headless=true
set PARAMS=
set MAIN_CLASS=com.webmethods.portal.system.PortalSystem
set RUN_CMD=%JAVA%  -cp %CLASSPATH% %JAVA_ARGS% %JAVA_OPTIONS%
%ACTION_PARAMS% -Dmain.class=%MAIN_CLASS%
7 %8 %9
```

さらに，この部分の上に次の例で示している 2 つの行を追加します。

```
%JAVA_HOME%¥bin¥java -jar C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥jreinstrumenter.jar -f
MyServer

set JAVA_OPTIONS=%JAVA_OPTIONS%
-Xbootclasspath/p:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥MyServer¥instr.jre
-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
```

1 行目は JRE Instrumenter を実行する Java コマンドを呼び出します。2 行目は必要な JVM パラメータを追加します。2 行目に使用する JVM パラメータが不明な場合は，後の手順 5 で追加できます。この 2 つの行で名前を入力し，自動的にインストルメントされる JRE クラスを格納するためのディレクトリを指定します。上記の例では，**MyServer** を自分で選択した名前に置き換えてください。

4 変更を起動スクリプトに保存し，変更されたスクリプトを使用してアプリケーション・サーバを再起動します。

5 起動スクリプトの実行結果で，JRE Instrumenter からの出力を検索します（Xbootclasspath を検索）。手順 3 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加した場合，それを JRE Instrumenter の出力にある JVM パラメータと比較します。これらが同じでない場合，JRE Instrumenter で出力された正しい JVM パラメータを使用して起動スクリプトを更新し，アプリケーション・サーバを再起動します。手順 3 で 2 行目（JVM パラメータを設定）を起動スクリプトに追加していない場合，ここで追加してアプリケーション・サーバを再起動します。

注 : JRE Instrumenter の出力が見つからない場合，またはエラー・メッセージがある場合，JRE は適切にインストルメントされない可能性があります。この場合は起動スクリプトを確認し，問題を解決します。

---

6 プローブが正しく設定されていることを確認するには，**＜ Java Agent のインストール・ディレクトリ＞¥DiagnosticsAgent¥log¥＜ Probe ID＞¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，JRE のインストルメンテーションに失敗したか，JVM パラメータを正しく設定しなかったかのいずれかです。詳細については，219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照してください。

My webMethods Server 設定ラッパーを設定するには, 次の手順を実行します。

My webMethods Server は，スクリプトまたはラッパー設定で起動します。したがって，起動スクリプトを変更するか，この例のようにラッパー設定ファイルを編集してサーバをインストルメントすることになります。

1 My webMethods Server を起動するのに使う設定ラッパーを特定します。設定ファイルは ...MWS¥server¥ ＜サーバ名＞ ¥config¥wrapper.conf です。

2 このファイルのバックアップ・コピーを作成し，エディタでファイルを開きます。

3 次に説明するようにファイルを更新します。

wrapper.conf ファイルで，次を追加します（270 と 280 の数字は設定ファイルによって異なります）。

```
wrapper.java.additional.270=-Xbootclasspath/p:
C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥classes¥auto¥MyServer¥instr.jre

wrapper.java.additional.280=-javaagent:C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥lib¥probeagent.jar
```

1 行目は JRE Instrumenter を実行する Java コマンドを呼び出します。2 行目は必要な JVM パラメータを追加します。最初の呼び出しで，2 番目のパラメータ（-Xbootclasspath）では，アプリケーション・サーバの JRE をインストルメントします。-Xbootclasspath パラメータで名前を入力し，インストルメントされるクラスを格納するディレクトリの名前を指定します。上記の例では，**MyServer** を自分で選択した名前に置き換えてください。

4 設定ラッパーへの変更内容を保存し，変更されたラッパーを使用してアプリケーション・サーバを再起動します。

Java Agent が起動し, JRE Instrumenter が暗黙的に実行されて JRE がインストルメントされます。

5 プローブが正しく設定されていることを確認するには，**＜Java Agent のインストール・ディレクトリ＞/DiagnosticsAgent/log/＜Probe ID＞/probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合, JVM パラメータを正しく設定していない可能性があります。

6 必要に応じて, インストルメントされた JRE が使用されるように, アプリケーション・サーバを再起動します。

重要 : 今後設定ラッパーで起動したときに My webMethods Server で使用される JRE を更新する場合，再び My webMethods Server を起動する前に，新しい JRE がインストルメントされるように**＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/classes/auto/MyServer** ディレクトリ（MyServer は自分のディレクトリ名に置き換えます）を削除する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。使用するインストルメンテーション・モードに関する一般的な情報については，222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照してください。

JRE Instrumenter とさまざまな起動オプションについて

JRE Instrumenter は，特許出願中のコレクション・リークの指摘（CLP）などの高度な機能を Java Agent で使用できるようにするため, JRE をインストルメントするユーティリティです。インストールされている JRE が JRE Instrumenter によって変更されることはありません。インストルメントされたクラスのコピーが**＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/classes** ディレクトリの任意の場所に配置されます。複数の JRE がシステムにインストールされている場合は，JRE Instrumenter を使ってそれら JRE をインストルメントできます。

JRE Instrumenter は，アプリケーション・サーバの JVM およびそこで実行中のアプリケーションによって使用される，標準の Java クラスをインストルメントします。また，アプリケーション・サーバがインストルメントされたクラスを使用できるように，アプリケーション・サーバが起動したときに使う必要のある JVM パラメータを提供します。

JRE Instrumenter は異なるコマンド・ライン・オプションを使用して起動可能で，それぞれ独自の利点と制限のある 3 つの異なる方法で使用できます。アプリケーション・サーバの特性に応じて，これらの方法のいずれかを選択します（例については，163 ページ「アプリケーション・サーバの設定例」を参照）。

- **自動明示モード :** アプリケーション・サーバがスクリプトによって起動されているか起動可能な場合，このモードを使用することをお勧めします。このモードを使用するには，アプリケーション・サーバの起動スクリプトに JRE Instrumenter を明示的に非対話的に実行して JRE をインストルメントする行を追加します。スクリプトは, 継続して新しくインストルメントされた JRE を使用して（追加パラメータで）アプリケーション・サーバの JVM を起動します。詳細については，219 ページ「自動明示モードでの JRE Instrumenter の使用」を参照してください。

- **自動暗黙モード：**このモードでは，JRE Instrumenter を明示的に実行する必要はありません。Java Agent を起動するようにアプリケーション・サーバの JVM パラメータを変更し，必要に応じて JRE Instrumenter の実行を指示するだけです。Java Agent が初めて使用される場合，JRE Instrumenter が暗黙的に実行されて JRE をインストルメントします。ただし，初回はこのインストルメントされた JRE は使用されず，アプリケーション・サーバはインストルメントされていない JRE を使用します。次回のアプリケーション・サーバの起動時に，インストルメントされた JRE が使用されます。したがって，Java Agent の完全な監視機能を使用する場合は，Java Agent を有効にした後にアプリケーション・サーバを 2 回再起動する必要があります。詳細については，222 ページ「自動暗黙モードでの JRE Instrumenter の使用」を参照してください。
- **マニュアル・モード：**このモードでは，Java Agent のインストールの最後またはインストール後に JRE Instrumenter を手動で対話的に実行し，JRE をインストルメントする必要があります。次に，JRE Instrumenter で出力されたパラメータに従って，アプリケーション・サーバの JVM パラメータを変更します。HP Diagnostics の以前のバージョンの JRE Instrumenter では，この方法が使用されます。詳細については，224 ページ「マニュアル・モードでの JRE Instrumenter の使用」を参照してください。

JRE が更新された場合（アプリケーション・サーバのパッチを適用する場合など），または Java Agent を更新した場合，再度 JRE のインストルメンテーションが必要な可能性があります。この問題については，各モードで説明します。

次の表に, 4 つの異なるインストルメンテーション方法のそれぞれの要件を要約しています。

		推奨インストルメンテーション（JRE Instrumenter を使用）		
	基本インストルメンテーション	**自動明示モード**	**自動暗黙モード**	**マニュアル・モード**
最低限必要な JRE バージョン	1.5	1.4	1.5	1.4
スクリプトによるアプリケーション・サーバの起動が必要	いいえ	はい	いいえ	いいえ

		推奨インストルメンテーション（JRE Instrumenter を使用）		
	基本インストルメンテーション	自動明示モード	自動暗黙モード	マニュアル・モード
JRE のインストール先の特定が必要	いいえ	いいえ	いいえ	はい
JRE Instrumenter の手動実行が必要	いいえ	いいえ	いいえ	はい
JVM パラメータを設定できる場所の特定が必要	はい *	はい *	はい *	はい *
Java Agent を有効にした後にアプリケーション・サーバの再起動が必要	はい（1 回）	はい（1 回または 2 回）	はい（2 回）	はい（1 回）
JRE のアップグレード / パッチの適用後に保守が必要	いいえ	いいえ	はい	はい

***** JRE 起動パラメータを定義する場所が見つからない場合は, **JAVA_OPTIONS** 環境変数を使用して実行することもできます。

自動明示モードでの JRE Instrumenter の使用

アプリケーション・サーバをスクリプトで起動する場合（WebLogic および JBoss アプリケーション・サーバなど），自動明示モードで JRE Instrumenter を使用することをお勧めします。また，WebSphere アプリケーション・サーバをスクリプトで起動しているまたは起動可能な場合（z/OS 以外のほとんどのプラットフォームが該当）もこのモードをお勧めします。Windows サービスとしてインストールされていない Tomcat の場合もこのモードをお勧めします。

自動明示モードを使用するには，次の 2 つのタスクを行う必要があります。

- アプリケーション・サーバで使用されるのと同じ JRE を使用して JRE Instrumenter を実行するように，アプリケーション・サーバの起動スクリプトを変更します。JRE Instrumenter からの出力を見れば，次のタスクに必要な JVM パラメータがわかります。
- JRE Instrumenter からの出力で見つかるアプリケーション・サーバの JVM パラメータを設定します。

注 : 設定を変更する前に，起動スクリプトの構造，プロパティ値の設定方法，環境変数の使い方を理解しておいてください。変更する前に，変更するファイルのバックアップ・コピーを必ず作成してください。

アプリケーション・サーバの起動スクリプトを変更する場合，アプリケーション・サーバの JVM を起動するために JRE が実行される行を最初に特定する必要があります。次に，この行の真上に次のような行を追加し，アプリケーション・サーバで使用されるのと同じ JRE を使用して JRE Instrumenter を実行します。

```
< Java コマンド> -jar < Java Agent のインストール・ディレクトリ> /
DiagnosticsAgent/lib/jreinstrumenter.jar -f <パス名>
```

< Java コマンド>は JRE Instrumenter でインストルメントする JRE であるため，アプリケーション・サーバの JVM の起動に使用される Java コマンドと**完全に**同じである必要があります。通常，この Java コマンドは，アプリケーション・サーバの JVM を起動する行の先頭部分をコピーすることで取得できます。

次の表に，いくつかの一般的に使用されるアプリケーション・サーバの元の起動スクリプトで使用される Java コマンドを示します（この表はヒントのみを目的としており，お使いのアプリケーション・サーバの起動スクリプトでは異なる Java コマンドを使用する可能性があります)。

Application Server	シェル・スクリプト（.sh）	Windows コマンド・スクリプト（.bat または .cmd）
JBoss	"$JAVA"	"%JAVA%"
Tomcat	${_RUNJAVA}	%_RUNJAVA%
WebLogic	${JAVA_HOME}/bin/java	%JAVA_HOME%¥bin¥java
WebSphere	${JAVA_EXE}	%JAVA_EXE%

＜Java Agent のインストール・ディレクトリ＞は，Java Agent がインストールされているディレクトリです。

＜パス名＞は，相対パスで指定する必要があります。JRE Instrumenter によって，インストルメントされたクラスが**＜Java Agent のインストール・ディレクトリ＞/DiagnosticsAgent/classes/＜パス名＞/instr.jre** ディレクトリに作成されます。Diagnostics で複数のアプリケーション・サーバを実行する場合，JRE Instrumenter の複数のインスタンスが互いに干渉しないように，各アプリケーション・サーバに一意の＜パス名＞（プローブ名など）を指定する必要があります。詳細については，231 ページ「アプリケーション・サーバの複数の Java プロセスの監視設定」も参照してください。

上記の説明に従って起動スクリプトに行を追加すると，起動スクリプトを使用してアプリケーション・サーバを起動するたびに，JRE Instrumenter が実行されて現在の JRE をインストルメントします。また，次のタスクで使用する JVM パラメータも出力されます。通常，JRE Instrumenter の出力は，起動スクリプトの実行結果内にあります。

標準の JRE バージョン 5.0 以降をインストルメントする JRE Instrumenter からの出力例を次に示します。

```
-Xbootclasspath/p: ＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/classes/ ＜パス名＞ /instr.jre
-javaagent: ＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/lib/probeagent.jar
```

標準の JRE バージョン 1.4.x をインストルメントする JRE Instrumenter からの出力例を次に示します。

```
-Xbootclasspath/p: < Java Agent のインストール・ディレクトリ> /DiagnosticsAgent/classes/ <パス名> /instr.jre;
< Java Agent のインストール・ディレクトリ> /DiagnosticsAgent/classes/boot
```

自動明示 JRE インストルメンテーションを使用するための 2 つ目の作業は，JRE Instrumenter の出力に従って，アプリケーション・サーバの JVM パラメータを変更することです。多くの場合，JRE Instrumenter で出力される JVM パラメータを含めるために Java コマンドライン・オプションを変更するだけで済みます。ただし状況によっては（WebSphere アプリケーション・サーバの場合など），設定ファイルを変更するか管理コンソールを使用してこれらの JVM パラメータを追加する必要がある場合もあります。

注 : JRE Instrumenter からの出力を取得するためには，最初のタスクで説明しているように起動スクリプトを変更してアプリケーション・サーバを再起動する必要があります。次に，アプリケーション・サーバの JVM パラメータを変更した後で，アプリケーション・サーバを再び再起動する必要があります（アプリケーション・サーバを 2 回再起動することになります）。ただしほとんどの JRE では，JRE Instrumenter によって指定される実際の JVM パラメータが上の例で指定しているものと同じになるか，上の例で指定しているものを含みます。したがって，こうした「標準設定」の JVM パラメータは変更したスクリプトを実行する前でも安全に追加できます。この手法は，特定のアプリケーション・サーバの手順で使用されます。お使いのアプリケーション・サーバ（JBoss，WebLogic，WebSphere，Tomcat）の例を参照して，自動明示モードを使用した設定方法の詳細な手順を確認してください。

または，JRE Instrumenter から Java コマンドライン・オプションへの出力をリダイレクト（またはパイプ処理）するか，JVM パラメータを別のソースから取得して，2 回の再起動を回避することもできます。

自動暗黙モードでの JRE Instrumenter の使用

GlassFish，NetWeaver，Windows サービスとしてインストールされた Tomcat（スクリプトなし），z/OS にインストールされた WebSphere, TIBCO ActiveMatrix および BusinessWorks などのスクリプトで起動できないアプリケーション・サーバでは，自動暗黙モードで JRE Instrumenter を使用することをお勧めします。

このモードを使用するには, JRE Instrumenter を明示的に実行する必要はなく, JRE Instrumenter は Java Agent によって暗黙的に実行されます。Java Agent を起動するように単にアプリケーション・サーバの JVM パラメータを設定することで，次のパターンに一致する場所を示すパスを含む JVM ブート・クラス・パスが Java Agent で検出されると，自動インストルメンテーション・モードになり，インストルメンテーションされたクラスが作成されて，指定したディレクトリにインストルメンテーションされたクラスのコピーが保存されます。

```
< Java Agent のインストール・ディレクトリ> /DiagnosticsAgent /classes/auto/
<名前> /instr.jre
```

たとえば，次の JVM パラメータを追加したとします。

```
-Xbootclasspath/p: < Java Agent のインストール・ディレクトリ> /DiagnosticsAgent/classes/auto/
ServerOne/instr.jre
-javaagent: < Java Agent のインストール・ディレクトリ> /DiagnosticsAgent/lib/probeagent.jar
```

アプリケーション・サーバの最初の実行時に，**< Java Agent のインストール・ディレクトリ> /DiagnosticsAgent/classes/auto/ServerOne/instr.jre** ディレクトリは存在しない可能性があります。Java Agent は，指定されたディレクトリを作成し，その中にインストルメントされたクラスを入れます。Java Agent はその実行元の（インストルメントされていない）JRE を使用します。

アプリケーション・サーバの最初の実行では，インストルメントされた JRE による利点を得られませんが，その後のすべての実行では最初の実行で用意されたインストルメンテーション済みのクラスが使用されます。

重要 : アプリケーション・サーバで使用する JRE を更新する場合（アプリケーション・サーバのパッチを適用する場合など），または Java Agent を更新する場合，アプリケーション・サーバを再起動する前に，新しい JRE がインストルメントされるように**< Java Agent のインストール・ディレクトリ> / DiagnosticsAgent/classes/auto/ServerOne** ディレクトリ（ServerOne は自分のディレクトリ名に置き換えます）を削除する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。Java Agent で JRE を再度インストルメントするときに，このディレクトリを手動で削除することもできます。

マニュアル・モードでの JRE Instrumenter の使用

手動で JRE Instrumenter を実行して，出力された JVM パラメータをアプリケーション・サーバの起動設定にコピーできます。Oracle アプリケーション・サーバの場合は，マニュアル・モードで JRE Instrumenter を使用することをお勧めします。

JRE Instrumenter は，次の機能を実行します。

- インストルメント可能な JRE を特定する。
- 指定したディレクトリで，追加 JRE を検索する。
- 指定した JRE をインストルメントし，インストルメントされたクラスの場所を示すために JRE の起動スクリプトに追加する必要のあるパラメータを追加する。
- Instrumenter を Windows または UNIX 環境のグラフィカル・インタフェースまたはコンソール・モードで実行した場合，Instrumenter は，インストルメントしたクラスを**＜ Java Agent のインストール・ディレクトリ＞ /DiagnosticsAgent/classes/ ＜ JRE のベンダ＞ / ＜ JRE のバージョン＞** ディレクトリの配下のフォルダに置きます。

重要 : アプリケーション・サーバで使用する JRE を更新する場合（アプリケーション・サーバのパッチを適用する場合など），または Java Agent を更新する場合，JRE Instrumenter を再度実行して新しい JRE をインストルメントし，JVM パラメータを適宜変更する必要があります。このディレクトリを削除しないと，アプリケーション・サーバが起動しなくなる可能性があります。

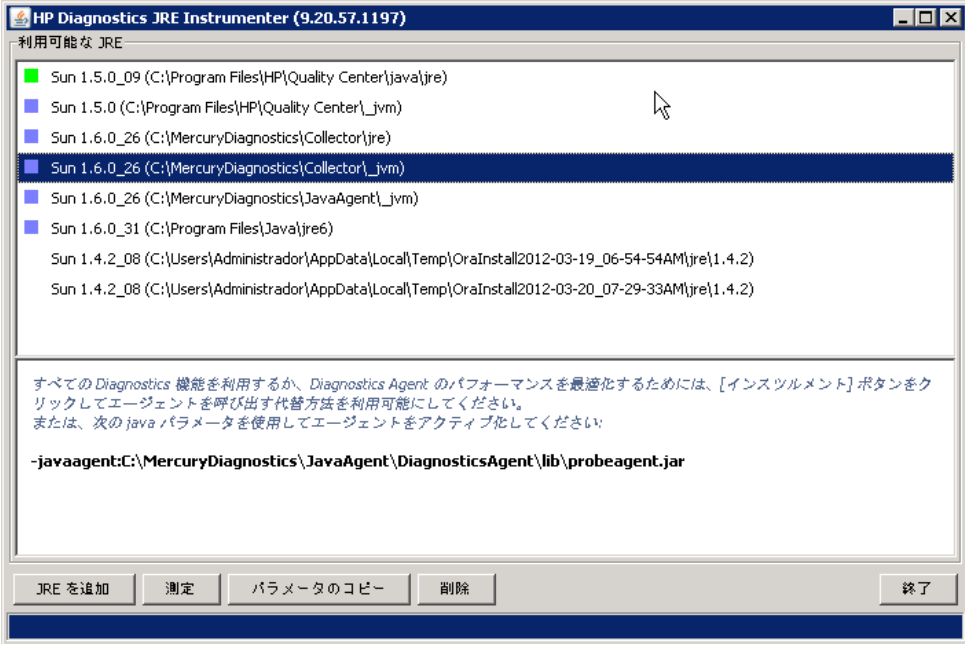
UI モードでの JRE Instrumenter ユーティリティの実行

オプションを使用せずに JRE Instrumenter を実行すると，Instrumenter には，グラフィカル・ユーザ・インタフェースのダイアログが表示されます。

Windows システムで JRE Instrumenter ユーティリティを起動するには，**＜プローブのインストール・ディレクトリ＞ ¥bin¥jreinstrumenter.cmd** コマンドを実行します。

UNIX または Linux で JRE Instrumenter ユーティリティを起動するには，**＜プローブのインストール・ディレクトリ＞ ¥bin¥jreinstrumenter.sh** コマンドを実行します。

Instrumenter には，Instrumenter が検出した JVM がリストされ，インストルメンテーションに使用できます。された JVM には，JVM の名前の前に緑色の四角形が付きます。

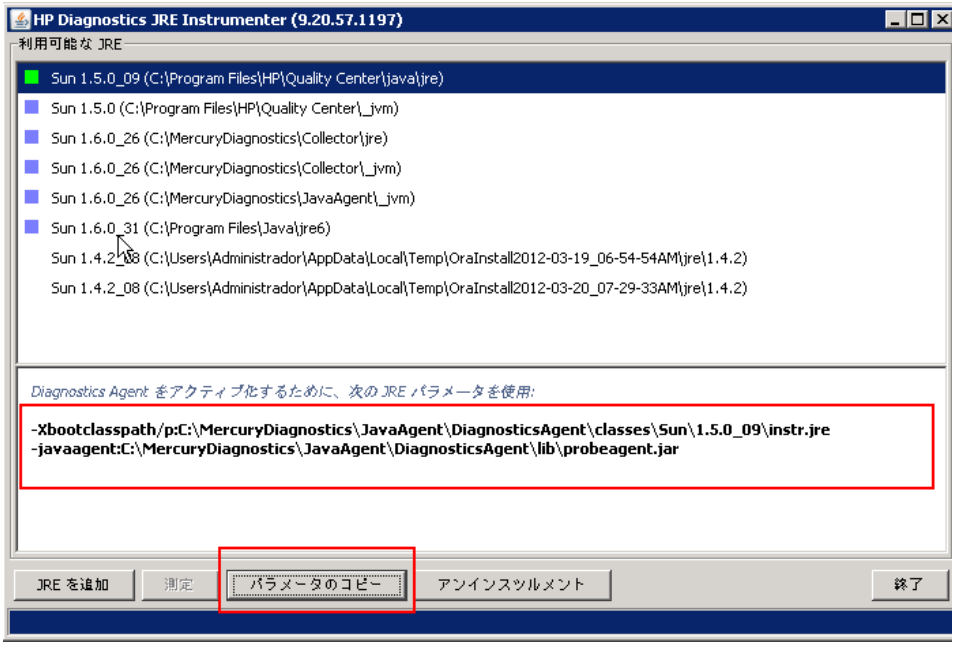


ダイアログに **JRE ディレクトリ**がリストされていない場合は，［**JRE を追加**］ボタンをクリックして JRE を参照します。JVM の検索を開始するディレクトリの場所に移動し，検索を開始するファイルを選択して［ここから検索］をクリックします。Instrumenter は，JVM を検索し，見つかった JVM を［利用可能な JRE］リストに示します。

インストルメントする JRE を選択して［**測定**］をクリックします。

JRE Instrumenter は，選択した JVM のクラスをインストルメントし，**＜プローブのインストール・ディレクトリ＞ /classes** ディレクトリの下のフォルダにインストルメントされたクラスを置きます。また，［利用可能な JRE］リストの下のボックスに，アプリケーション・サーバの起動時に使用する必要のある JVM パラメータが表示されます。

JRE Instrumenter が JRE をインストルメントする際，インストルメントしたクラスをアプリケーションが使うために，アプリケーション・サーバの起動スクリプトに含めなければならない JVM パラメータも作成されます。［利用可能な JRE］リストからインストルメントされた JRE を選択すると，リストの下のボックスに JVM パラメータが表示されます。



［**パラメータのコピー**］をクリックして，対応するパラメータをクリップボードにコピーします。JVM パラメータがクリップボードにコピーされ，この JVM パラメータをアプリケーション・サーバの設定で使用し，Java Agent による監視をアクティブ化できます。

重要：設定の後の手順でクリップボードの内容を使用するため，クリップボードの内容を上書きしないように注意してください。

［**終了**］をクリックして［JRE Instrumenter］ウィンドウを閉じ，アプリケーション・サーバの JVM パラメータの設定を続行します。

JVM パラメータをアプリケーション・サーバの起動スクリプトに挿入する一般的な手順については，228 ページ「アプリケーション・サーバの起動スクリプトに JVM パラメータを含める」を参照してください。JVM パラメータを JBoss, WebLogic，Tomcat などのさまざまなアプリケーション・サーバの起動スクリプトに挿入する具体的な例については，163 ページ「アプリケーション・サーバの設定例」を参照してください。

コンソール・モードでの JRE Instrumenter の実行

＜プローブのインストール・ディレクトリ＞ ¥bin を開いて，JRE Instrumenter の実行可能ファイルを見つけます。次のコマンドを実行します。

```
./jreinstrumenter.sh -console
```

Instrumenter を実行すると，利用可能な処理オプションのリストが表示されます。次の表は，各処理オプションの参照先を示します。

Instrumenter 機能	説明
jreinstrumenter -l	JRE Instrumenter にとって既知の JVM のリストを表示します。JVM ベンダ，JRE バージョン，JRE が置かれている場所を表示します。
jreinstrumenter -i ＜ JRE のディレクトリ＞	特定のディレクトリの JRE をインストルメンテーションに選択します。＜ JRE ディレクトリ＞を，［利用可能な JVM］リストから選択した JRE が存在するフォルダへのパスに置き換えます。 このコマンドは，JRE Instrumenter が，選択した JVM のクラスをインストルメントするように指示し，**＜プローブのインストール・ディレクトリ＞ / classes/ ＜ JVM ベンダ＞ / ＜ JRE のバージョン＞** ディレクトリの下のフォルダにインストルメントされたクラスを置きます。

Instrumenter 機能	説明
jreinstrumenter -a ＜ディレクトリ＞	特定のディレクトリ内で JVM を検索し，見つかった JVM を JRE Instrumenter に既知の JVM のリストに追加します。＜ディレクトリ＞には，Instrumenter で検索を開始する場所へのパスを指定します。 Instrumenter は, ディレクトリとサブディレクトリを含め，指定の場所からディレクトリを検索します。検索が完了すると，使用可能な JVM のリストが更新されて表示されます。

JVM パラメータを JRE Instrumenter の出力からコピーし，Java Agent による監視をアクティブ化するために，アプリケーション・サーバの起動時に取得できる場所に貼り付けることができます。

JRE Instrumenter を終了し，アプリケーション・サーバの JVM パラメータの設定を続行します。

JVM パラメータをアプリケーション・サーバの起動スクリプトに挿入する一般的な手順については，228 ページ「アプリケーション・サーバの起動スクリプトに JVM パラメータを含める」を参照してください。JVM パラメータを JBoss, WebLogic，Tomcat などのさまざまなアプリケーション・サーバの起動スクリプトに挿入する具体的な例については，163 ページ「アプリケーション・サーバの設定例」を参照してください。

アプリケーション・サーバの起動スクリプトに JVM パラメータを含める

JRE Instrumenter が JVM をインストルメントする際, インストルメントしたクラスをアプリケーションが使うために，アプリケーション・サーバの起動スクリプトに含める必要がある JVM パラメータも作成されます。Instrumenter が JVM のインストルメントを完了すると，JVM パラメータが表示されます。

JVM パラメータをクリップボードにコピーし，アプリケーション・サーバの起動時に取得できる場所に貼り付けます。次に，一般的な手順を示します。

WebLogic，WebSphere，JBoss などのアプリケーション・サーバで JVM パラメータを挿入する具体的な例については，163 ページ「アプリケーション・サーバの設定例」を参照してください。

アプリケーション・サーバの設定を更新するには，次の手順を実行します。

1 アプリケーション・サーバの起動スクリプトまたは，JVM パラメータが設定されているファイルを見つけます。

2 アプリケーション・サーバの起動スクリプトに変更を加える前に，起動スクリプトのバックアップ・コピーを作成します。

3 エディタまたはアプリケーション・サーバのコンソールを使用して起動スクリプトを開きます。

4 アプリケーション・サーバを起動する Java コマンド・ラインに，JRE Instrumenter からの Java パラメータを追加します。次に例を示します。

```
-Xbootclasspath/p: ＜プローブのインストール・ディレクトリ＞
¥classes¥Sun¥1.4.2_04\instr.jre;
    <probe_install_dir>¥classes\boot
```

このインスタンスでは，**＜プローブのインストール・ディレクトリ＞**は，Java Agent がインストールされているディレクトリへのパスです。

これによりプローブがアプリケーションに接続します。

次は，Java パラメータを追加する前の起動スクリプトの WebLogic Java コマンド・ラインの例です。

```
"%JAVA_HOME%¥bin¥java" -hotspot -ms64m -mx64m -classpath "%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="C:¥¥bea"
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy="C:¥bea¥wlserver6.1/lib/weblogic.policy" weblogic.Server
```

次は，Java パラメータを追加した後の起動スクリプトの WebLogic Java コマンド・ラインの例です。

```
"%JAVA_HOME%¥bin¥java" -hotspot -ms64m -mx64m
-Xbootclasspath/p: <プローブのインストール・ディレクトリ>
¥classes¥Sun¥1.5.0_17¥instr.jre;
-javaagent: <プローブのインストール・ディレクトリ> ¥lib¥probeagent.jar
-classpath "%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="C:¥¥bea" -Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy="C:¥bea¥wlserver6.1/lib/weblogic.policy" weblogic.Server
```

5 起動スクリプトに変更を保存します。

6 テスト時にアプリケーション・サーバを再起動します。

7 プローブが正しく設定されていることを確認するには，**<プローブのインストール・ディレクトリ> ¥log¥ < Probe ID > ¥probe.log** ファイルのエントリをチェックします。ファイルにエントリがない場合は，アプリケーション・サーバで使用する JRE をインストルメントしていないか，Java Agent を起動するようにアプリケーション・サーバの JVM パラメータを設定しなかったかのいずれかです（本章のアプリケーション・サーバに関する手順を参照してください)。

その他の設定オプション

以下の項では，その他の設定オプションについて説明します。

- 231 ページ「アプリケーション・サーバの複数の Java プロセスの監視設定」
- 235 ページ「アプリケーション・サーバ内の Java Agent のヒープ・サイズの調整」
- 235 ページ「SOAP メッセージ・ハンドラの設定」
- 239 ページ「CI 作成用に新しい J2EE サーバのディスカバリを設定」
- 240 ページ「OSGi フレームワークに基づいたアプリケーションの場合に特に考慮する必要のある事項」
- 241 ページ「Azul ユーザが特に考慮する必要のある事項」

アプリケーション・サーバの複数の Java プロセスの監視設定

お使いのアプリケーション・サーバで複数の Java プロセスを使用している場合，または複数の Java プロセスのパフォーマンス・データを収集する場合は，Agent の追加設定手順を行う必要があります。次の 2 つのオプションがあります。ホストの JVM ごとに個別の Java Agent インストールを設定することも，すべての JVM で共有する Java Agent インストールを 1 つ設定することもできます。

本項の内容

➤ 231 ページ「複数の JVM で共有する 1 つの Java Agent インストールの設定」

➤ 234 ページ「各 JVM の個別の Java Agent インストールの設定」

複数の JVM で共有する 1 つの Java Agent インストールの設定

複数の JVM で 1 つの Java Agent インストールを共有するには，各 JVM にプローブ・インスタンスをそれぞれ設定する必要があります。この設定により，次が有効になります。

➤ Diagnostics サーバとプローブ間で通信を確立する。

➤ プローブを Diagnostics サーバごとに特定する。

➤ JVM によって使用される JRE をインストルメントする。

複数の JVM で共有する 1 つの Java Agent インストールを設定するには，次の手順を実行します。

1 つの Java Agent インストールを使用して複数の JVM を監視する場合，必要に応じて Java Agent を起動するようにアプリケーション・サーバの JVM パラメータを設定する必要があります。各 JVM で異なる JRE インストルメンテーション・モードを使用できます（JRE インストルメンテーション・モードの詳細については，第 6 章，「Java Agent で監視するためのアプリケー ション・サーバの準備」を参照）。

1 使用する各バージョンの JRE をインストルメントしていない場合は今すぐインストルメントしてください。第 6 章，「Java Agent で監視するためのアプリケー ション・サーバの準備」を参照してください。

2 プローブが自動的に選択できるポートの範囲を指定します。Java Agent は，ミニ Web サーバを使用して通信します。プローブが監視している各 JVM に対し，通信のために別々のポートが割り当てられます。標準設定で，ポート番号の範囲（最小 / 最大）は **35000** ～ **35100** です。プローブが 100 個以上の JVM と連動している場合，ポート番号の範囲を増やすことができます。

注 : ファイアウォールによって，ほかの Diagnostics コンポーネントからプローブが分離される場合，指定した範囲内のポートを使用して通信できるようにファイアウォールを設定します。詳細については, 付録 ,「ファイアウォール環境で動作するための Diagnostics の設定」を参照してください。

標準設定と異なる範囲でプローブと通信するようにファイアウォールを設定している場合は，次の説明に従って，ポート範囲の値を適宜更新します。

---

a **＜プローブのインストール・ディレクトリ＞ /etc** フォルダで **webserver.properties** ファイルを見つけます。

b 次のプロパティを設定して，プローブとの通信で使用可能なポートの範囲を調整します。

- ポート番号範囲の最小ポートは，次のプロパティを使用します。

  jetty.port=35000

- ポート番号範囲の最大ポートは，次のプロパティを使用します。

  jetty.max.port=35100

3 次のメソッドの 1 つを使用して一意のプローブ名を割り当てます。

コマンド・ラインのプロパティは，改行を入れずに 1 行で入力してください。この Java コマンド・ラインで定義されたプローブ ID によって，プローブの **id** プロパティを使用して **probe.properties** ファイルに定義されたプローブ名が無効になります。

a Java コマンド・ラインまたは起動スクリプトを使用して, 各 JVM のプローブにカスタムプローブ ID を割り当てます。

-Dprobe.id= ＜一意のプローブ名＞

次の例は，**probe.id** パラメータを追加する前の WebLogic 起動スクリプトです。

```
"%JAVA_HOME%¥bin¥java" -hotspot -ms64m -mx64m -classpath "%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer -Dbea.home="C:¥¥bea"
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:¥bea¥wlserver6.1/lib/weblogic.policy" weblogic.Server
```

次の例は，**probe.id** パラメータを追加した後の WebLogic 起動スクリプトです。

```
"%JAVA_HOME%¥bin¥java" -hotspot -ms64m -mx64m"
-Xbootclasspath/
p:C:¥MercuryDiagnostics¥JAVAProbe¥classes¥Sun¥1.4.1_03;C:¥MercuryDiagnostics
¥JAVAProbe¥classes¥boot"
-classpath "%CLASSPATH%"
-Dprobe.id=<Unique_Probe_Name> -Dweblogic.Domain=petstore
-Dweblogic.Name=petstoreServer
-Dbea.home="C:¥¥bea" -Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:¥bea¥wlserver6.1/lib/weblogic.policy" weblogic.Server
```

b Java パラメータを 1 つ指定したのに同じスクリプトを使用して複数のプローブを開始したときは，%0 文字列を使用して各プローブにカスタムプローブ識別子を作成します。たとえば，1 つの起動スクリプトを使用して複数のプローブ対象のアプリケーション・サーバ・インスタンスを開始するクラスタ化された環境です。

-Dprobe.id=<probeName>%0

Windows では，%%0 を使用します。1 つ目の % を使用して 2 つ目の % をエスケープします。

各プローブに一意のプローブ名を作成するには，**%0** が 1 つの数字と動的に置き換えられます。たとえば，＜プローブ名＞ 0，＜プローブ名＞ 1，のようになります。

4 各プローブで使うポイント・ファイルを指定します。標準設定では，ポイント・ファイルの名前は auto_detect.points です。複数のカスタム・インストルメンテーション・プランを使う必要がある場合，または 1 つの Agent インストールを使用して同じマシンで複数のバージョンの JRE を持っている場合，および 1 つ以上の JRE でレイヤをサポートするカスタム・インストルメンテーションに特定のメソッドやクラスが必要なときに，使用するカスタム・ポイント・ファイルを指定することがあります。

```
-Dprobe.points.file.name=" ＜カスタム AutoDetect ポイント・ファイル＞ "
```

各 JVM の個別の Java Agent インストールの設定

1 つのホストに複数の JVM がある場合，各 JVM インスタンスに Java Agent インストールをそれぞれ設定できます。Agent を複数回インストールし，各 Agent のインストール・ディレクトリにある **probe.properties** ファイルでプローブの **id** プロパティを設定し，プローブのインスタンスを定義します。

JVM ごとにそれぞれインストールされる Agent を設定するには，次の手順を実行します。

1 JRE をインストルメントしていない場合は今すぐインストルメントします。第 6 章，「Java Agent で監視するためのアプリケー ション・サーバの準備」を参照してください。

2 **＜プローブのインストール・ディレクトリ＞ /etc** ディレクトリで **probe.properties** ファイルを見つけます。

次に例を示します。

```
C:¥¥MercuryDiagnostics¥JAVAProbe¥etc¥probe.properties
```

3 次のように，**id** プロパティに，サーバと Diagnostics サーバに一意の名前を割り当てます。

```
id=<uniqueProbeName>
```

プローブが起動すると，プローブのログ・メッセージが保存される **<プローブのインストール・ディレクトリ> /log** ディレクトリにログ・ファイルが作成されます。

アプリケーション・サーバ内の Java Agent のヒープ・サイズの調整

ヒープのサイズは Java Agent およびアプリケーション・サーバのパフォーマンスに影響することがあります。ヒープ・サイズの標準設定値は 64 MB ですが，多くの場合，アプリケーション・サーバでもっと多い量に増加します。Java Agent をアプリケーション・サーバに追加する場合，ヒープ・サイズを増加させて Java Agent で使用するメモリに対応する必要がある場合があります。詳細については，34 ページ「Diagnostics Java Agent ホストの要件」を参照してください。

ヒープ・サイズは，次の JVM 引数を使用してアプリケーション・サーバの JVM 設定で設定します。

```
-Xmx <サイズ>
```

-Xmx <サイズ> オプションで指定する値を更新して，ヒープ・サイズを増やすことができます。このパラメータの設定に関するヘルプについては，JVM のドキュメントを参照してください。

SOAP メッセージ・ハンドラの設定

Diagnostics SOAP メッセージ・ハンドラは，Java プローブが次の機能をサポートするために必要です。

- SOAP 失敗のペイロードを収集する。
- SOAP ヘッダ，ボディまたはエンベロープから SOA コンシューマ ID を確認する。

ほとんどのアプリケーション・サーバでは，インストルメンテーション・ポイントとコード・スニペットが書き込まれて自動的に監視対象の Web サービスの Diagnostics ハンドラを設定します。

重要 : 一部のアプリケーション・サーバについては, Diagnostics SOAP メッセージ・ハンドラを自動的に読み込むための特別なインストルメンテーションが Diagnostics に用意されています。

ただし，WebSphere 5.1 JAX-RPC と Oracle 10g JAX-RPC については，手動設定が必要です。詳細については，237 ページ「Diagnostics SOAP メッセージ・ハンドラの読み込み」を参照してください。

また，一部のアプリケーション・サーバでは Diagnostics SOAP メッセージ・ハンドラが使用できず，カスタム・インストルメンテーションを使用して SOAP ペイロードから SOAP エラーやコンシューマ ID をキャプチャすることもできません。したがって，この機能はアプリケーション・サーバやそのバージョンによっては使用できない場合があります。Diagnostics SOAP メッセージ・ハンドラのサポートの最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

本項の内容

SOAP メッセージ・ハンドラの無効化

標準設定では，SOAP メッセージ・ハンドラは有効になっています。次のようにして SOAP メッセージ・ハンドラを無効にできます。

<probe_install_dir>/etc/inst.properties ファイルで，
details.conditional.properties プロパティを編集し，
mercury.enable.autoLoadSOAPHandler = false を組み込みます。

SOAP メッセージ・ハンドラが無効になっている場合は，チェーン内でハンドラがインストールされている場所を手動で設定する必要があります。

Diagnostics SOAP メッセージ・ハンドラの読み込み

ほとんどのアプリケーション・サーバ上では SOAP メッセージ・ハンドラは自動的に読み込まれますが，次のアプリケーション・サーバでは手動で設定する必要があります。

WebSphere 5.1 JAX-RPC

WebSphere 5.1 JAX-RPC で SOAP メッセージ・ハンドラを設定するには，次の手順を実行します。

注 : WebSphere 6.1 JAX-WS Web サービスの場合，Diagnostics ハンドラはサポートされていません。Diagnostics SOAP ハンドラ・クラスでアプリケーションを再コンパイルする必要があります。

---

1 アプリケーションの Web サービス・デプロイメント記述子（**webservices.xml**）を見つけます。ディレクトリ・パスは次のようになります。

＜インストール先のルート＞ ¥config¥cell¥ ＜サーバ＞ ¥applications¥
＜ Web サービス EAR ＞ ¥deployments¥ ＜ Web サービス名＞ ¥
＜ Web サービス JAR|WAR 名＞ ¥WEB-INF

次に例を示します。

```
C:¥Program Files¥WebSphere¥AppServer¥config¥
cells¥MyServer1¥application¥WebServicesSamples.ear¥
deployments¥WebServicesSamplea¥AddressBookJ2WB.war¥ WEB-INF
```

2 webservices.xml を編集し，<port-component> ごとに Diagnostics ハンドラを追加します。

```
<port-component>
 ......
 <handler>
  <handler-name>Diagnostics RPC Handler</handler-name>
   <handler-class>
    com.mercury.opal.javaprobe.handler.soap.ProbeRPCHandler
   </handler-class>
 </handler>
 ......
</port-component>
```

3 Diagnostics ハンドラの jar（**<probe_install_dir>¥lib¥probeSOAPHandler.jar**）を WebSphere の **lib** ディレクトリにコピーします。

次に例を示します。

```
cp C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥ lib¥probeSOAPHandler.jar
C:¥Program Files¥WebSphere¥AppServer¥lib
```

これらの手順は Windows 上で IBM WebSphere 5.1.0 アプリケーション・サーバを使って開発されました。

Oracle 10g JAX-RPC

Oracle 10g JAX-RPC で SOAP メッセージ・ハンドラを設定するには，次の手順を実行します。

1 アプリケーションの Web サービス・デプロイメント記述子（**webservices.xml**）を見つけます。ディレクトリ・パスは次のようになります。

＜ OC4J のインストール・ルート＞ ¥j2ee¥home¥applications¥ ＜アプリケーション名＞ ¥ ＜デプロイメント名＞ ¥WEB-INF¥webservices.xml

2 webservices.xml を編集し，<port-component> ごとに Diagnostics ハンドラを追加します。

```
<port-component>
 ......
 <handler>
  <handler-name>Diagnostics RPC Handler</handler-name>
   <handler-class>
    com.mercury.opal.javaprobe.handler.soap.ProbeRPCHandler
   </handler-class>
 </handler>
 ......
</port-component>
```

3 Diagnostics ハンドラの jar（**＜プローブのインストール・ディレクトリ＞ ¥lib¥probeSOAPHandler.jar**）を**＜ OC4J のインストール・ルート＞ ¥j2ee¥home¥applib** ディレクトリにコピーします。

これらの手順は，Windows 上で Oracle Containers for J2EE（OC4J）10g Release 3（10.1.3.3）を使って開発されました。

CI 作成用に新しい J2EE サーバのディスカバリを設定

Agent は，Business Service Management の J2EE アプリケーション・サーバと J2EE アプリケーション・ドメイン CI を作成するデータを提供します。

プローブは，JBoss や WebLogic のような有名な J2EE サーバについては CI を自動的に作成します。

アプリケーション・サーバ・ディスカバリがほかの J2EE サーバについて CI を作成するよう設定することもできます。アプリケーション・サーバ名は，JMX によって検出されるように，またはポイント / コード・スニペットによって検出されるように直接に指定または設定できます。

アプリケーション・サーバ・ディスカバリは，次に説明するように，プローブ **etc/metrics.config** ファイルに設定します。

クラス AppServerDiscoveryCollector は **＜プローブのインストール・ディレクトリ＞ /lib/probe-jmx.jar** ファイルにあります。アプリケーション・サーバ・ディスカバリとメトリックス収集の両方を実行する独自の Collector クラスを作成できます。

次は，一般的なアプリケーション・サーバ用のアプリケーション・サーバ・ディスカバリの設定です。Collector の名前では大文字と小文字が区別され，**metrics.config** ファイルのほかの Collector 名と異なっているものである必要があります。

```
<user-defined-collector-name>.class.name =
com.mercury.diagnostics.capture.metrics.jmx.AppServerDiscoveryCollector
<user-defined-collector-name>.class.path = probe-jmx.jar
<user-defined-collector-name>.app_server.configure.discovery = true
<user-defined-collector-name>.app_server.type = <user-defined-type>
<user-defined-collector-name>.app_server.server_name =
<user-defined-server-name>
<user-defined-collector-name>.app_server.domain_name =
<user-defined-domain-name>
```

次に，app-server/javaprobe 起動スクリプトまたは Java コマンド・ラインに，次の Java システム・プロパティ定義を追加する必要があります。

```
-Dapp_server.discovery.collector=<user-defined-collector-name>
```

プローブは，15 分ごとに，Collector（AppServerDiscoveryCollector を含む）を更新し，新しい設定に基づいて検出を行います。

JMX を使って，新しいアプリケーション・サーバ名と J2EE ドメイン名を検出する方法を知っている上級ユーザには，プローブの **etc/metrics.config** ファイルに次の設定を追加できます。

```
<user-defined-jmx-collector-name>.class.name =
com.mercury.diagnostics.capture.metrics.jmx.JMXCollector
<user-defined-jmx-collector-name>.class.path = probe-jmx.jar
<user-defined-jmx-collector-name>.depends.on.class =
javax.management.MBeanServer
<user-defined-jmx-collector-name>.app_server.configure.discovery = true
<user-defined-jmx-collector-name>.app_server.type = <user-defined-type>
<user-defined-jmx-collector-name>.app_server.server_name =
<user-defined-server-name>
<user-defined-jmx-collector-name>.app_server.server_name.discovery.by.jmx =
<jmx-ObjectName>.<jmx-AttributeName>
<user-defined-jmx-collector-name>.app_server.domain_name =
<user-defined-domain-name>
<user-defined-jmx-collector-name>.app_server.domain_name.discovery.by.jmx =
<jmx-ObjectName-1>.<jmx-AttributeName-1>@<jmx-ObjectName-2>.<jmx-AttributeNa
me-2>
```

OSGi フレームワークに基づいたアプリケーションの場合に特に考慮する必要のある事項

お使いのアプリケーションが OSGi フレームワークに基づいている場合，追加のプロパティをいくつか設定する必要がある場合があります。まだ標準設定値でない場合，**osgi.java.profile.bootdelegation** プロパティを標準設定値「ignore」に設定します。次に，**osgi.java.profile** の **org.osgi.framework.bootdelegation** の最後に com.mercury.* を追加します。次に例を示します。

```
org.osgi.framework.bootdelegation= ＜既存のパッケージ＞ ,com.mercury.*
```

Azul ユーザが特に考慮する必要のある事項

Azul では，エンタープライズ Java ユーザに 2 つの拡張性およびパフォーマンスの高いソリューションである，Vega と Zing が提供されます。Vega は，ユーザのローカル・ネットワークに接続する特殊なハードウェア・アプライアンスです。Zing は Vega を仮想化したもので，VMware または KVM のゲスト・イメージ形式で提供されます。Azul アプライアンスの主な利点である革新的な無停止ガベージ・コレクタは，継続的に実行され，最大で数十ギガバイトのヒープを処理できます。両方のアプライアンスは Diagnostics によって同等にサポートされますが，実際にテストが行われたのは Zing のみです。

Azul に含まれる Java SDK または JRE は，Linux や Solaris などの従来のシステムにインストールされますが，起動時に Java コードの実行をアプライアンスに委任します。したがって，Java アプリケーションは起動された場所で実行されているように見えても，実際には異なるシステムで実行されます。これはシームレスに行われるため，アプリケーションはローカル・システムで実行されているかのようにその環境と相互作用します。アプリケーションが JNI 呼び出しを行う場合，ネットワークを通じて元のホストで実行されます。

この実行モデルは，Diagnostics ユーザにとって多数の問題があります。プローブによる JNI の呼び出しはコストが高い上に，ユーザが期待する結果を得られません。

- CPU タイムスタンプは正常に機能しません。元のサーバで使用された CPU 時間がインストルメントされるため，役に立ちません。
- プロセス・メトリックスはフロント・エンドのプロセスをインストルメントするため，これも役に立ちません。
- ほとんどの場合，すべてのシステム・メトリックスも役に立ちません。元のシステムがインストルメントされるため，アプライアンスで実行されるアプリケーションとは無関係です。
- ガベージ・コレクション・メトリックスはわかりにくくなっています。Azul は連続的なガベージ・コレクタを使用するため，ガベージ・コレクションの割合が 100% を超えることは普通です。
- Heap Breakdown および Heap Walker は機能しません。
- （VMware 上の仮想アプライアンスを使用した場合でも）VMware の特殊タイマは機能しません。

Azul VM 用の Diagnostics の設定

Azul の Java コマンドを呼び出すためには, アプリケーションの実行に使用されるアプライアンスを適切に識別するパラメータを追加する必要があります。これが JRE Instrumenter では問題となります（自動暗黙モードで実行している場合を除く）。JRE Instrumenter はバージョンとベンダを特定するためにインストルメントされる JRE を実行する必要がありますが，必要なパラメータは追加できません。

この解決策として，Azul JRE インストールの **azul.properties** ファイルを編集し，必要なパラメータを定義します。この設定は JRE Instrumenter の実行中に行う必要があり，Diagnostics を使用するアプリケーションを実行する場合は削除できます。

考えられる混乱と無意味なオーバーヘッドを排除するため，Diagnostics Agent の使用中は次の設定を使用することをお勧めします。

- **metrics.config** で，「system」および「ProcessMetrics」Collector のすべてのメトリックス，および「Java Platform」Collector のガベージ・コレクション・メトリックスをコメントアウトします。
- **capture.properties** で，use.cpu.timestamps を false に設定します。

7

Java Agent でクライアント監視のためのアプリケーション・サーバの準備

本項では, Java Agent でクライアント監視をするためにアプリケーション・サーバを準備する方法について説明します。

本章の内容

クライアント監視について

クライアント監視では，ユーザのブラウザに表示される Web ページのパフォーマンスをインストルメントし，これらの測定値をバックエンド・サーバ要求に関連付けます。

バックエンド時間，フロントエンド時間，合計時間の 3 つの重要なメトリックスがインストルメントされます。

バックエンド時間とは，Web ページ要求が送信されてから応答の最初のバイトが受信されるまでに経過した時間です。

フロントエンド時間とは，応答の最初のバイトが受信されてからページがロードされるまでに経過した時間です。

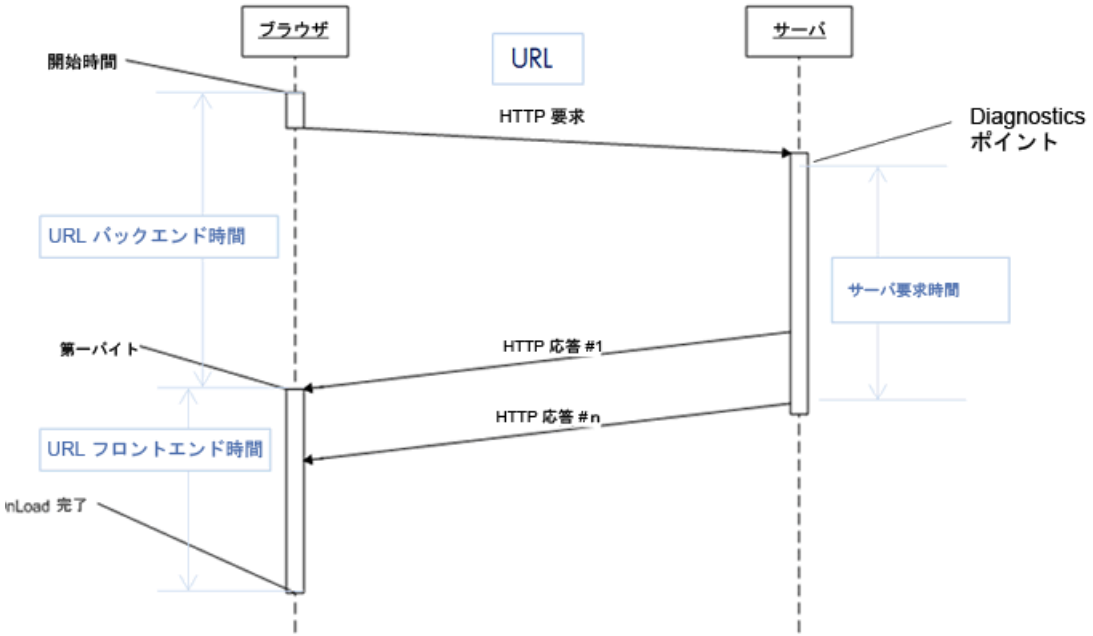
合計時間とは，フロントエンド時間とバックエンド時間を合計した時間です。

クライアント監視では，こうした測定値を集計して，URL，場所，ブラウザと OS の組み合わせごとに表示します。

Web ページのパフォーマンスを監視することで, アプリケーションのオーナーはパフォーマンスの問題をすばやく識別し，層（フロントエンドまたはバックエンド)，場所，ブラウザごとに特徴付けることができます。

問題がバックエンドで起こっている場合，クライアント監視は URL を関連付けられたサーバ要求およびその呼び出しプロファイルに関連付けます。

次に，クライアント監視の例を示します。



クライアント監視の有効化

クライアント監視を有効化するには，.war ファイルをアプリケーション・サーバにデプロイする必要があります。場合によっては Web サーバを設定する必要があります。クライアント監視ビューは, Diagnostics Enterprise UI で利用可能です。

クライアント監視を有効にするには，次の手順を実行します。

クライアント監視が有効な場合，JBoss，Tomcat，WebSphere，WebLogic を使用して利用可能なほとんどの JSP ページは自動的に変更され，<head> タグの近辺に追加の Java Script 呼び出しが含まれます。ブラウザでページを開いてビュー・ソースを選択することにより，どのページがインストルメントされているかを確認できます。

その他のアプリケーション・サーバでは，クライアント監視のために手動でページをインストルメントする必要がある場合があります。詳細については，247 ページ「クライアント監視のための HTML/JSP ページの手動でのインストルメント」を参照してください。

自動の JSP インストルメンテーションを含め，クライアント監視は，この .war ファイルがデプロイされるまで無効のままです。

1 ClientMonitoring.war ファイルをデプロイします。

アプリケーション・サーバの管理コンソールを使用して，**＜プローブのインストール・ディレクトリ＞ ¥contrib¥ClientMonitoring.war** をアプリケーションとしてデプロイします。

クライアント監視は，この .war ファイルがデプロイされるまで無効のままです。

2 Web サーバをアプリケーションのフロントエンドとして設定している場合，次のコンテキスト・ルートもご使用の Web サーバの設定に追加する必要があります：**/ClientMonitoring/***

ヒント：ブラウザでリンク http://hostname:port/ClientMonitoring/B/ にアクセスできる場合，Web サーバが正しく設定されていることを確認できます（空白ページが返されます）。

たとえば，（JBoss および Apache Web サーバに基づいて）クライアント側の監視サポートを BSM に追加する場合は次のようになります。

-- **＜プローブのインストール・ディレクトリ＞ ¥contrib¥ClientMonitoring.war** を BSM ゲートウェイ・サーバの **C:¥HPBSM¥AppServer¥deploy** にコピーします。

-- BSM ゲートウケイ・サーバの Apache Web サーバ設定（**C:¥HPBSM¥Web Server¥conf¥uriworkermap.properties**）に次の行を追加します。

/ClientMonitoring/*=localAjp

-- BSM ゲートウェイ・サーバおよび BSM Apache Web サーバを再起動します。

クライアント監視の設定と無効化

必要であれば，**＜プローブのインストール・ディレクトリ＞ ¥etc¥dynamic.properties** のいくつかのプロパティを更新してクライアント監視を動的に制御できます。

client.monitoring.enable プロパティは，クライアント監視機能を動的に有効または無効にするマスタ・スイッチです。これを false に設定すると，受信したすべてのクライアント監視データ・イベントはドロップされ，JSP ページの自動インストルメンテーションは無効になり，**client.monitoring.sampling.percent** は 0.0 に設定されます（手動でインストルメントされる JSP ページのクライアント監視 Java Script コードを無効にします）。

dynamic.propertes の **client.monitoring.sampling.percent** プロパティを調整することで，サーバにおけるクライアント監視の負荷を軽減できます。

client.monitoring.strict.referrer を **true** に設定することで，参照側を厳密にチェックするように指定することもできます。これにより，クライアント監視でインストルメントされる Web ページに起因するイベントのみを使用するようにできます。標準設定値は false ですが，この設定が環境内で機能する場合には推奨値は true となります。

クライアント監視のための HTML/JSP ページの手動でのインストルメント

次のコードを HTML / JSP ページに追加します。

```
<script>
if (window.t_firstbyte === undefined) {
    var t_firstbyte = Number(new Date());
}
</script>
<script type='text/javascript' src='/ClientMonitoring/boomerang-min.js'>
</script>
<script type='text/javascript' src='/ClientMonitoring/hp_diag-min.js'>
</script>
<script>
BOOMR.init({
    beacon_url: "/ClientMonitoring/B",
    user_ip: '10.0.0.1',
    RT: {
        cookie: "X-HP-CM-RT",
        cookie_exp: 600,
        expandFrames: true,
        hashURLs: true
    },
    HP: {
        cookie: "X-HP-CM-GUID"
    }
});
</script>
```

HTML / JSP ページを手動でインストルメントする場合，inst.properties の次のプロパティを false に設定して自動インストルメンテーションを永続的に無効にできます。これらを変更するには，アプリケーション・サーバを再起動する必要があります。

＜プローブのインストール・ディレクトリ＞ ¥etc¥inst.properties:

details.conditional.properties= ¥
mercury.enable.clientmonitoring.JspWriterImpl.autoinstrumentation=false,¥
mercury.enable.clientmonitoring.CoyoteWriter.autoinstrumentation=false,¥
mercury.enable.clientmonitoring.BodyContentImpl.autoinstrumentation=false,¥

8

.NET Agent のインストール

本項では, .NET Agent のインストール方法について説明し, .NET Agent のセットアップや設定の情報を示します。

本章の内容

➤ .NET Agent のアンインストール（293 ページ）

.NET Agent インストールの概要

.NET Agent ソフトウェアは，監視するアプリケーションのホスト・マシンにインストールされます。.NET Agent で，監視用にアプリケーション・ドメインをインストルメントします。

.NET Agent の要件については，第 1 章，「HP Diagnostics のインストールの準備」を参照してください。

.NET Agent（バージョン 9.x）には .NET Framework 2.0 以降が必要です。.NET Framework をマシンにインストールしてから，.NET Agent のインストールを実行する必要があります。

重要：.NET Framework 1.1 をサポートする必要がある場合は，古いバージョンの .NET Agent（8.x）を使用する必要があります。

WCF の要件と制限事項：.NET Windows Communication Foundation (WCF) サービスを監視するには，.NET Framework 3.0 SP1 以降が必要です。次のトランスポートを使用した WCF バインディングがサポートされています。

- Http
- TCP

サポートされていないトランスポートがアプリケーションで使用されている場合，.NET プローブ・は各 WCF メソッドに対する一般的なサーバ要求のみを作成します。この要求は Web サービスではなく，VM 間の相関は発生しません。

HP Diagnostics / TransactionVision .NET Agent インストーラを実行すると，Diagnostics または TransactionVision，あるいはその両方のデータを収集する .NET Agent がインストールされます。

.NET Agent インストーラは，Agent がインストールされているシステム上の ASP.NET アプリケーションを自動的に検出します。281 ページ「検出および標準インストルメンテーション」を参照してください。

インストーラは，検出された各 ASP.NET アプリケーションの基本作業負荷およびイベントをキャプチャするように Agent を設定します。この Agent 設定は，**probe_config.xml** ファイルを使用して制御します。詳細については，282 ページ「検出した ASP.NET アプリケーション用の自動インストルメンテーションおよび設定」を参照してください。

.NET Agent は，**ポイント・ファイル**を使用して標準インストルメンテーションを有効にし，アプリケーションの監視を可能にします。このポイント・ファイルによって，Agent がアプリケーションについてキャプチャする作業負荷が制御されます 詳細については，第 10 章,「.NET アプリーションのカスタム・インストルメンテーション」を参照してください。詳細については，289 ページ「アプリケーションの標準インストルメンテーションの有効化と無効化」を参照してください。

ASP.NET アプリケーションを監視するインストルメンテーションを可能にするため，次のポイント・ファイルがインストールされて有効になります。

- ASP.NET.points
- ADO.points
- WCF.points
- 次のポイント・ファイルは，ほかの Microsoft テクノロジーを使用するアプリケーションをインストルメントするために使用できます。
- Remoting.points（.NET リモート環境用）
- msmq.points（MSMQ 環境用）
- LWMD.points（アプリケーションのコレクションで使用されるメモリの分析用）

IIS がインストールされた ASP.NET アプリケーション・ドメインが検出されると，そのドメインごとに個別のインストルメンテーション・ポイント・ファイル（**<アプリケーション・ドメイン> .points**）が作成されます。**probe_config.xml** ファイルには，検出された各 ASP.NET アプリケーションの appdomain 参照が含まれます。各 appdomain セクションには，インストルメンテーション・ポイント・ファイルの参照が含まれます。.NET Agent では，このランタイム・インストルメンテーションを使って，特定のアプリケーションからメソッドのレイテンシ情報をキャプチャします。

HP Software-as-a-Service（SaaS）: HP Diagnostics は HP Software-as-a-Service（SaaS）環境にデプロイできます。SaaS デプロイメントでは，Diagnostics .NET Agent が企業の IT 環境にインストールされ，Diagnostics コマンダ・サーバとメディエータ・サーバが HP によって社内の SaaS システムにインストールされます。SaaS でホストされる Mediator が HP の施設にインストールされる場合，.NET Agent のセットアップ中に Diagnostics モードの Agent を設定するために次のオプションを選択します。

最初に .NET Agent インストーラへのアクセスを参照してください。

.NET Agent インストーラへのアクセス

.NET Agent インストーラはさまざまな方法で起動できます。.NET Agent は，Diagnostics インストール・ディスク，BSM インストール・ディスク，または Business Service Management の［ダウンロード］ページからインストールできます。ソフトウェアは SSO ポータルからインストールできます。試用版の HP Diagnostics Profiler for .NET をインストールする場合は，HP ソフトウェア Web サイトのダウンロード・センターからインストーラを起動できます。

Diagnostics インストール元からインストーラにアクセスするには，次の手順を実行します。

➤ Diagnostics インストール DVD（Autorun.exe）からインストール・メニュー・ページが表示されます。メニューから［**Diagnostics Agent for .NET 32-bit**］を選択し，32 ビット Windows バージョンの .NET Agent のインストールを起動します。または，［Diagnostics Agent for .NET 64-bit］を選択し，64 ビット・バージョンの .NET Agent のインストールを起動します。

➤ インストール元で実行可能ファイル **HPDiagTV.NETAgt_ <リリース番号> _ win32.msi**（32 ビット）または **HPDiagTV.NETAgt_ <リリース番号> _win64.msi**（64 ビット）を見つけ，そのファイルを新しいインストール元にコピーしてダブルクリックすることでインストーラを実行することもできます。

254 ページ「.NET Agent のインストール」に進みます。

HP ソフトウェアのダウンロード・センターからインストーラをダウンロードするには，次の手順を実行します。

1 HP Passport のログイン情報を使用して SSO ポータル（http://support.openview.hp.com/selfsolve）にアクセスします。

2 Diagnostics（または TransactionVision）ダウンロードを見つけて，Diagnostics .NET Agent ソフトウェアのダウンロードに適切なリンクを選択します。ダウンロード・センターを使用して，Diagnostics .NET Profiler の試用版 / 評価版を入手することもできます。

3 Web サイトのダウンロード手順を行います。

254 ページ「.NET Agent のインストール」に進みます。

Business Service Management の Diagnostics［ダウンロード］ページからインストーラをダウンロードするには，次の手順を実行します。

1 **Business Service Management** のメイン・メニューから，［**管理**］>［**Diagnostics**］を選択し，［**ダウンロード**］タブをクリックします。または，メイン・メニューから［**管理**］>［**プラットフォーム**］を選択し，［**セットアップと保守**］タブをクリックします。

2 ［ダウンロード］ページで，適切なリンクをクリックして 32 ビット Windows または 64 ビット Windows 用の .NET Agent インストーラをダウンロードします。

注：.NET Agent インストーラを Business Service Management で使用するには，Business Service Management からアクセスするために必要なディレクトリに配置する必要があります。Diagnostic サーバのインストール時にこれを行うか，または .NET Agent インストーラを Diagnostics インストール・ディスクから必要な場所に手動でコピーできます。

---

254 ページ「.NET Agent のインストール」に進みます。

HP ソフトウェア評価版ダウンロード Web サイトから HP Diagnostics Profiler for .NET の評価版インストーラを起動するには，次の手順を実行します。

1 HP ソフトウェア Web サイトのダウンロード・センターにアクセスします。

2 ［**Quick Search**］セクションの［**Products**］リストで，［**Diagnostics**］をクリックし，［**Search**］をクリックします。

3［**Trial Software**］で，該当するリンクを選択します。

4 Web サイトのダウンロード手順を行います。

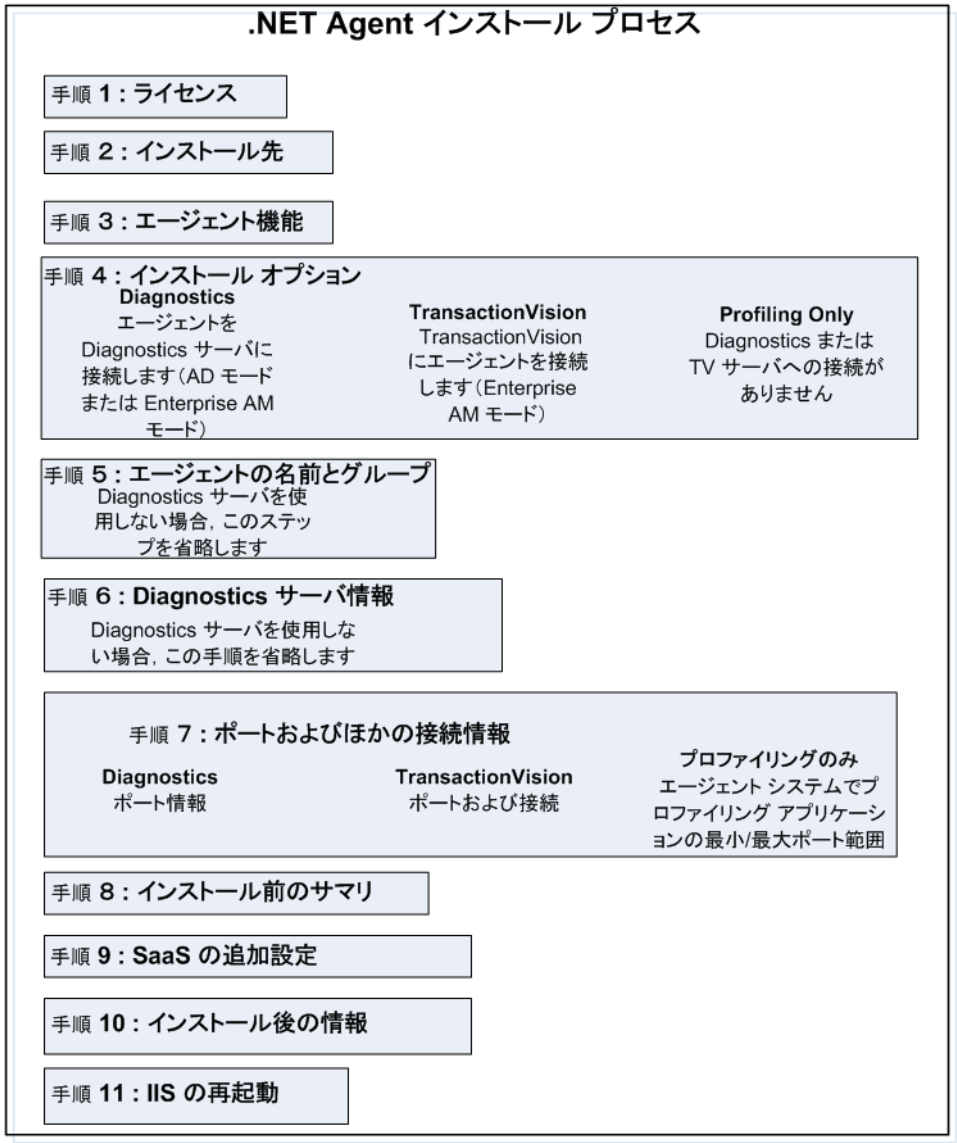
254 ページ「.NET Agent のインストール」に進みます。

.NET Agent のインストール

本項では，.NET Agent を初めてインストールする場合の詳細な手順について説明します。

重要 : ホスト・マシンにすでに .NET Agent がインストールされている場合，次のインストール手順ではなく Agent システムのアップグレード手順に従う必要があります。詳細については，851 ページ「アップグレードとパッチ・インストールの手順」を参照してください。

下の図に .NET Agent のインストール手順の概要を示します。各手順の詳細については，本項のその後の内容を参照してください。



.NET Agent のインストール処理には次の手順が含まれます。手順 1. エンド・ユーザ使用許諾契約書 から開始してください。

- 256 ページ「手順 1. エンド・ユーザ使用許諾契約書」
- 256 ページ「手順 2. インストール先の指定」
- 259 ページ「手順 4. インストールする Agent の機能の選択」
- 257 ページ「手順 3. インストール・オプションの選択」
- 260 ページ「手順 5. Agent の名前とグループ」
- 262 ページ「手順 6. Diagnostics サーバ情報」
- 264 ページ「手順 7. ポートおよび接続情報」
- 270 ページ「手順 8. プレインストール・サマリ」
- 271 ページ「手順 9. Agent を HP SaaS 環境で動作するための追加のセットアップ」
- 273 ページ「手順 10. インストール後の情報」
- 274 ページ「手順 11. IIS の再起動」

手順 1. エンド・ユーザ使用許諾契約書

エンド・ユーザ使用許諾契約書に同意します。

使用許諾契約書を読み，［**使用条件の条項に同意します**］を選択します。

［**次へ**］をクリックして次の手順に進みます。

手順 2. インストール先の指定

Agent のインストール先を指定します。

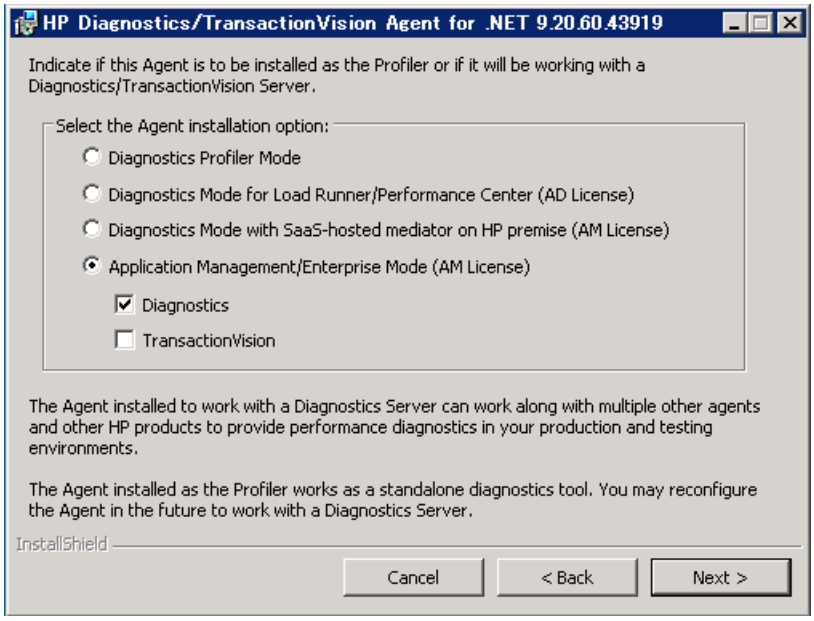
デフォルトでは，Agent は **C:¥MercuryDiagnostics¥.NET Probe** にインストールされます。この場所が ＜プローブのインストール・ディレクトリ＞になります。

デフォルトのディレクトリを受け入れるか，ほかの場所を選択します。ほかの場所を選択するには，ほかのパスを入力するか，または［**Browse**］をクリックしてインストール先のディレクトリを指定します。

［**次へ**］をクリックして次の手順に進みます。

手順 3. インストール・オプションの選択

.NET Agent をサーバに接続しないスタンドアロンの Profiler としてインストール（Diagnostics .NET Profiler の試用版をインストールする場合など）するか，LoadRunner/Performance または Diagnostics や TransactionVision Server で使用するためにインストールするかを指定します。



Agent を使用する環境に適した項目を選択します。

➤ **Diagnostics Profiler Mode :** このオプションは，Diagnostics サーバに接続しない Diagnostics .NET Profiler として Agent をインストールする場合に選択します。通常，HP Diagnostics 製品を購入する前に Diagnostics .NET Profiler の試用版をインストールする場合に選択します。

［Diagnostics Profiler Mode］オプションを選択した場合，**probe_config.xml** の <modes> 要素は .NET Agent のインストール時に **pro** モードに設定されます（548 ページ「<modes> 要素」を参照）。

➤ **Diagnostics Mode for LoadRunner/Performance Center (AD License) :** このオプションは，負荷テスト（または実稼動前）環境の Diagnostics サーバで使用するために Agent をインストールする場合に選択します。この場合，プローブが LoadRunner または Performance Center の実行でのみ使用されます。

AD モードでプローブを実行する利点として，AD モードのプローブは LoadRunner または Performance Center で実行されている場合のみに HP Diagnostics AD ライセンス・キャパシティに対してカウントされます。たとえば，LoadRunner/Performanace Center AD モードで 20 個のプローブがインストールされていて，5 個のみが実行されている場合，その 5 個のみが AD ライセンス・キャパシティに対してカウントされます。AD ライセンス・キャパシティの詳細については，83 ページ「現在接続されているプローブに基づくライセンス情報」を参照してください。

AD モードの場合，Agent は LoadRunner または Performance Center で実行しているときのみデータをキャプチャします。結果は，Default Client:21 など，この実行のための専用 Diagnostics データベースに格納されます。AD モードでは，プローブが LoadRunner または Performance Center の実行の一部である場合を除いて，Agent はリソースを使用したり，サーバにデータを送信しません。

AD ライセンス・オプションを選択した場合，**probe_config.xml** の <modes> 要素は .NET Agent のインストール時に **ad** モードに設定されます（548 ページ「<modes> 要素」を参照）。

➤ **Diagnostics Mode with SaaS-hosted mediator on HP premise (AM License) :** このオプションを選択すると，Agent を SaaS 環境で動作するようにインストールします。この場合，.NET Agent は HP 社内の HP SaaS サーバに接続されます。HP SaaS によってホストされる Diagnostics メディエータ・サーバへの .NET Agent の接続に関する情報は，HP SaaS 管理者によって提供されます。

➤ **Application Management/Enterprise Mode (AM License) :** このオプションは，エンタープライズ（または実運用）環境の Diagnostics サーバや TransactionVision Server で使用するために Agent をインストールする場合に選択します。

Agent を設定する対象を指定します。

- ➤（ローカルにインストールされている）Diagnostics サーバ
- ➤ TransactionVision Server
- ➤ ローカルにインストールされている Diagnostics サーバと TransactionVision Server の両方

TransactionVision を選択した場合，TransactionVision 固有のセットアップ・オプションの詳細については，Business Service Management 文書ライブラリの『HP TransactionVision Deployment Guid』を参照してください。

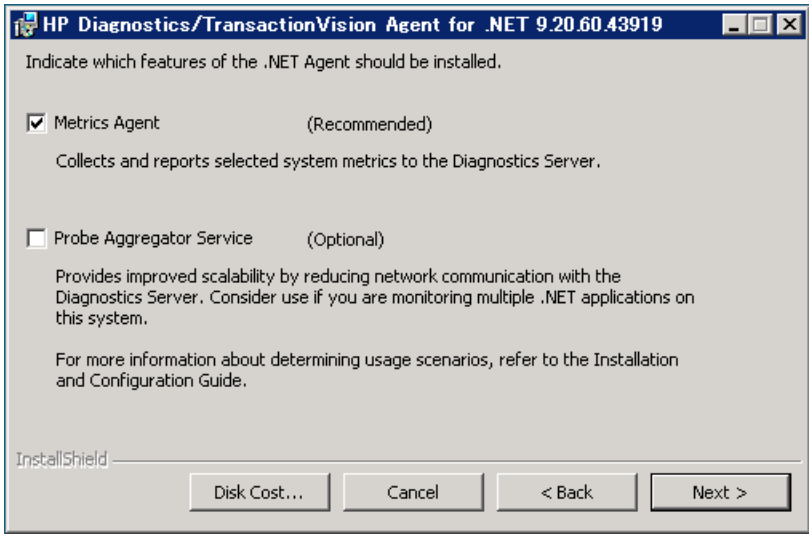
このオプションを選択した場合，**probe_config.xml** の <modes> 要素は，.NET Agent のインストール時に Diagnostics サーバを選択した場合は **enterprise** モード，TransactionVision Server を選択した場合は **tv** モードに設定されます（548 ページ「<modes> 要素」を参照）。

Enterprise モードが設定された Agent は，HP Diagnostics AM ライセンス・キャパシティに対してカウントされます。

［**Next**］をクリックして次の手順に進みます。

手順 4. インストールする Agent の機能の選択

インストールする .NET Agent の機能を選択します。



Metrics Agent: 標準設定でオンになっている Metrics Agent 機能をインストールすることをお勧めします。詳細については，第 17 章，「.NET System Metrics Agent - システム・メトリックスのキャプチャ」を参照してください。ただし，ホスト・マシン上のシステム測定値をキャプチャしない場合は，［**Metrics Agent**］ボックスをオフにできます。

Probe Aggregator: Probe Aggregator サービスをインストールするように選択することもできます。Agent を HP SaaS 環境で動作するようにインストールしている場合，Probe Aggregator ボックスがチェックされます。これは，このオプションが SaaS で必要であり，変更できないためです。

この Probe Aggregator サービスは，Diagnostics メディエータ・サーバにパフォーマンス・データを送信する前に，5 秒間隔で .NET Agent データを集計します。これにより，ネットワークとサーバの通信量が削減されてスケーラビリティが向上しますが，プローブ・システムのオーバーヘッドが増大することがあります。Probe Aggregator をインストールした場合のパフォーマンス低下の詳細については，285 ページ「Probe Aggregator サービス」を参照してください。

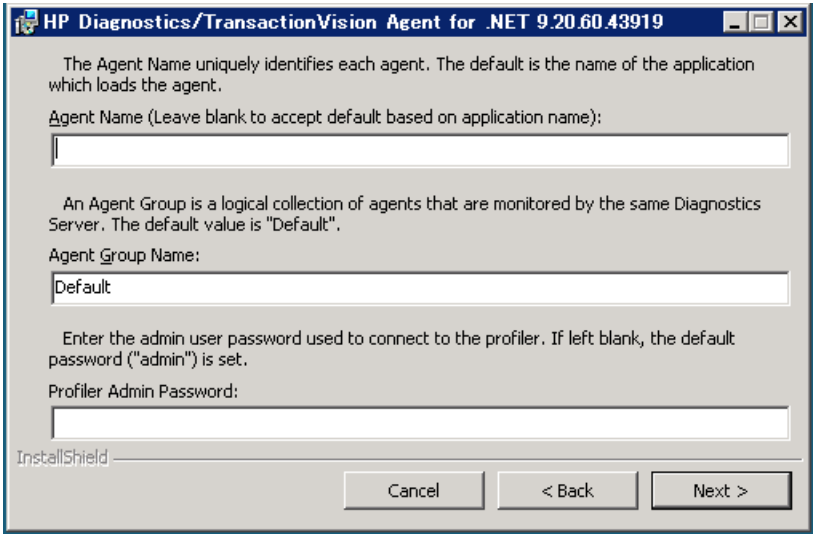
Disk Cost : ホストのドライブ上の空きディスク容量を確認するには，[**Disk Cost**] ボタンをクリックします。この機能を使って，Agent のインストールに十分な空き容量があることを確認します。

[**Next**] をクリックして次の手順に進みます。

手順 5. Agent の名前とグループ

Agent から Diagnostics サーバにレポートしない場合，この手順はスキップします。

Agent 名と Agent グループ名を入力します。



- **Agent Name :** HP Diagnostics 内で Agent を識別する名前です。このフィールドを空欄にした場合，.NET Agent は，監視対象アプリケーションのアプリケーション・ドメイン名に基づいて Agent 名を自動生成します。Agent 名はプローブ・エンティティ名として割り当てられます。

注 :［**Agent Name**］を空欄にして，Agent 名を自動生成させることをお勧めします。Agent 名を自分で入力する場合は，次の情報をよくお読みください。

Agent 名を入力する際の注意事項

- エージェント名に使用できる有効な文字は文字，数字，ダッシュ，アンダースコア，およびピリオドです。
- 監視対象のアプリケーションとインストルメンテーションのタイプを識別しやすい Agent 名を付けてください。

  たとえば，PetWorld というアプリケーションを監視するためにインストールしている .NET Agent の Agent には，次のように名前を付けます。

  PetWorld_Dotnet_Agent

- Agent 名を指定する際，ホストのすべての Agent は，強制的に同じ Agent 名を使用します。

  ［Agent Name］フィールドを空欄にしたときに，Agent によって自動生成される標準設定の Agent 名は，$(APPDOMAIN).NET の指定に相当します。

  標準設定名を変更する場合は，次の代替マクロを使って実行時に名前を変更します。

  - $(MACHINENAME): マシンのホスト名
  - $(APPDOMAIN): アプリケーションのドメイン名
  - $(PID): アプリケーションのプロセス ID
  - $(WEBSITENAME): アプリケーションがホストされる IIS Web サイト
  - $(COMMANDLINE:n) : n はコマンド・ラインのパラメータ番号

    次に例を示します。

    <id probeid=”ILTEST_$(COMMANDLINE:3)_rest” probegroup=”Default” />

    コマンド・ライン iltest “heart and lung” -abc server で ILTEST_server_rest の probeid が返されます。

n が 0 の場合は，実行可能ファイル / コマンド名を示します。

注 :

- アプリケーションが IIS にホストされていない場合，Agent 名は標準設定の $(APPDOMAIN).NET に戻ります。この例は，コンソール・アプリケーションなどです。
- 新規にインストールされた IIS アプリケーションの場合, Windows の［スタート］メニューの HP Diagnostics .NET Agent プログラム・グループから，［**Rescan ASP.NET Applications**］を実行する必要があることがあります。

- **Agent Group Name :** 既存のグループの名前か新規作成するグループの名前を入力します。Agent グループ名のデフォルト値は Default. です。Agent グループ名は大文字と小文字を区別します。Diagnostics では，この名前がプローブ・グループ名として使用されます。

  プローブ・グループは, 同じ Diagnostics サーバに報告するプローブの論理グループです。プローブ・グループのパフォーマンス測定値は追跡され，さまざまな Diagnostics ビューに表示できます。

  たとえば，グループレベルのパフォーマンスやプローブ・エンティティ単位のパフォーマンスを監視するために，1 つのプローブ・グループに特定のエンタープライズ・アプリケーションのすべてのプローブを割り当てることができます。

- **Profiler Admin Password :** .NET Diagnostics Profiler への接続に使用する管理ユーザ・パスワードを入力します。空白のままにした場合，標準設定のパスワード（admin）が設定されます。

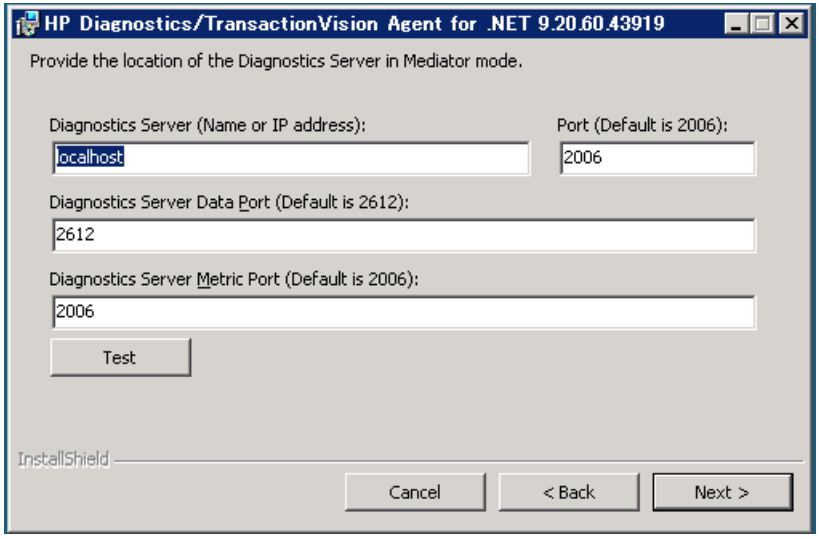
  ［**Next**］をクリックして次の手順に進みます。

手順 6. Diagnostics サーバ情報

Agent から Diagnostics サーバにレポートしない場合，または Agent を HP SaaS 環境で動作するようにインストールしている場合，この手順はスキップします。Agent と SaaS によってホストされる Diagnostics サーバ間の通信の設定に関する詳細は，HP SaaS 管理者によって提供されます。

.NET Agent が Diagnostic メディエータ・サーバと通信するために必要な情報を入力します。

Probe Aggregator サービスをインストールするように選択した場合は，［Diagnostics Server Data Port］でなく［Probe Aggregator Data Port］，［Diagnostics Server Metric Port］でなく［Probe Aggregator Metric Port］が表示されます。



- ［**Diagnostics Server (Name or IP address)** ］ボックスに，Diagnostic メディエータ・サーバのホストのホスト名と IP アドレスを入力します。
- 単純なホスト名ではなく，完全修飾ホスト名を指定してください。OS が混在し，その 1 つが UNIX の環境で，これは正確なネットワーク・ルーティングのために不可欠です。
- ［**Diagnostics Server Data Port**］ボックスに，Diagnostics サーバ が Agent の通信のためにリスンしているポート番号を入力します。標準設定のポート番号は 2612 です。Diagnostics サーバをインストールした後にポートを変更した場合は，標準設定のポートを使用しないで，変更後のポート番号を指定します。
- Probe Aggregator サービスをインストールするように指定した場合は，［Diagnostics Server Data Port］でなく［**Probe Aggregator Data Port**］ボックスが表示されます。プローブ集計がインストールされている場合は，Diagnostics メディエータ・サーバが Agent 通信をリッスンするポート番号を入力します。標準設定のポート番号は 2626 です。Diagnostics サーバをインストールした後にポートを変更した場合は，標準設定のポート番号を使用しないで，変更後のポート番号を指定します。

- ［**Diagnostics Server Metric Port**］ボックスに，Diagnostics サーバが System Metrics Agent からの通信のためにリッスンしているポート番号を入力します。デフォルトのポート番号は **2006** です。Diagnostics サーバをインストールした後にポートを変更した場合は，標準設定のポートではなく，変更後のポート番号を指定します。
- Probe Aggregator サービスをインストールするように指定した場合は，［Diagnostics Server Metric Port］でなく［**Probe Aggregator Metric Port**］ボックスが表示されます。プローブ集計がインストールされている場合は，Diagnostics メディエータ・サーバが Agent 通信をリッスンするポート番号を入力します。標準設定のポート番号は 45000 です。Diagnostics サーバをインストールした後にポートを変更した場合は，標準設定のポート番号を使用しないで，変更後のポート番号を指定します。
- 接続チェックを実行して，Diagnostics サーバが実行中でインストール・ホストからアクセス可能なことを確認するには，［**Test**］をクリックします。
- 接続チェックでは，入力した Diagnostic メディエータ・サーバの情報に間違いがないかどうか，Diagnostics サーバのホストと Agent のホストの間に接続エラーがないかどうかをすぐに確認できます。Diagnostic メディエータ・サーバホストへの接続が解決できない場合，エラー・メッセージが表示されます。
- ［**Next**］をクリックして次の手順に進みます。

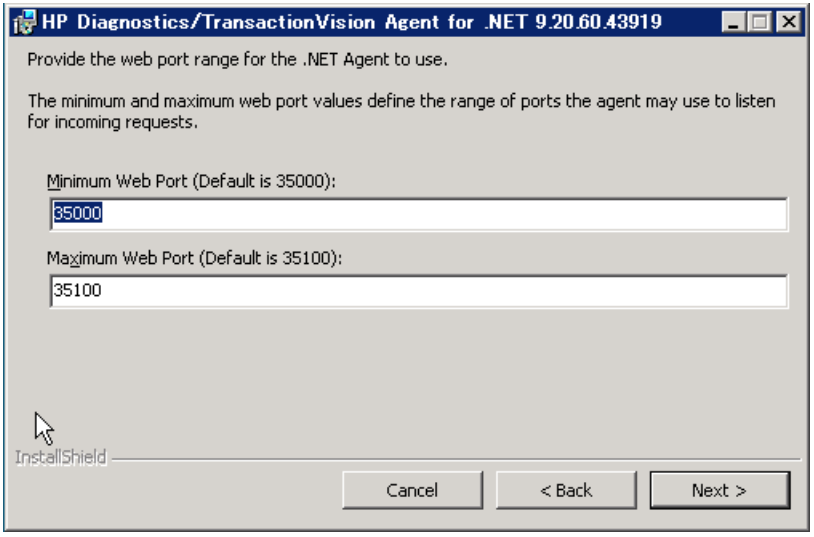
手順 7. ポートおよび接続情報

選択したインストール・オプションに応じて，表示されるポートおよび接続設定ダイアログは異なります。次から選択して設定に進みます。

- Diagnostics サーバのポート接続情報
- TransactionVision Server のポートおよび接続情報
- Diagnostics および TransactionVision Server に接続しない Profiler モード

Agent が Diagnostics サーバ で機能するようにインストールする場合は，次のダイアログ・ボックスが表示されます。

使用する .NET Agent の Web ポート範囲を指定します。



- **Minimum Web Port :** Agent に割り当てる Agent ホストのポート範囲の最小ポート番号を入力します。
- **Maximum Web Port :** Agent に割り当てる Agent ホストのポート範囲の最大ポート番号を入力します。

注 : デフォルトの範囲は，35000 以上 35100 以内です。

Web ポート範囲の上限と下限は，［**Minimum Web Port**］および［**Maximum Web Port**］フィールドで定義されます。Web ポート範囲には，Agent が使用可能なポートが含まれます。

Agent は，起動時に，この範囲の中から，最小ポート番号から最大ポート番号に向かって空きポートを特定しようとします。すでにほかの Agent またはアプリケーションが要求した場合は，範囲内のポートが使用中の可能性があります。

ポート範囲の最小サイズは，Agent のホストで同時に動作する Agent の最大数と等しくなります。

Web ポート範囲を設定する際の検討事項

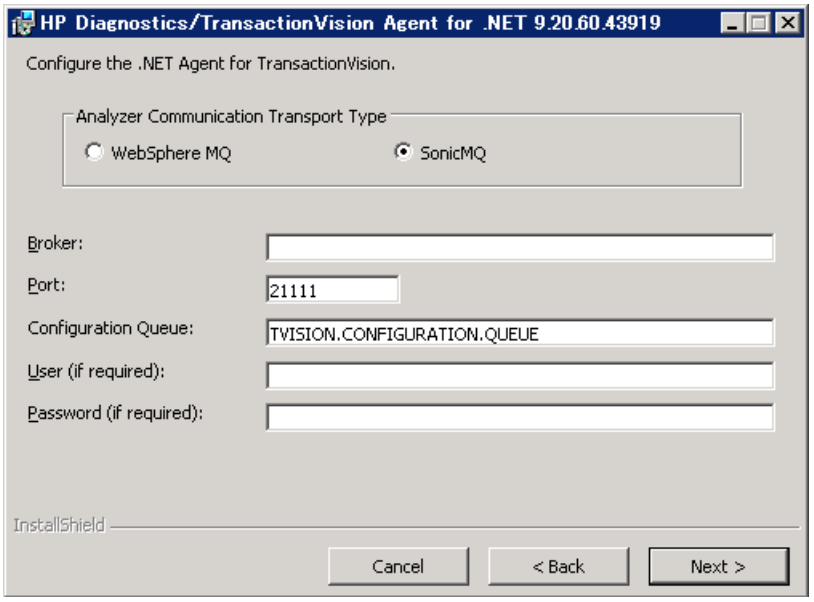
- Agent が ASP.NET アプリケーションと連携して動作している場合，ASP.NET の appdomain の再利用のために，ポートの数を 2 倍にしてください。
- Agent と，Agent と通信するコンポーネントの間にファイアウォールを置いている場合は，範囲内のポートに対してファイアウォールを開きます。必要な大きさになるように範囲を調整します。

［**Next**］をクリックして次の手順に進みます。Agent を TransactionVision 環境で動作するオプションも選択している場合は，次の項の追加設定を参照してください。

Agent が TransactionVision 環境で機能するようにインストールする場合は，次のダイアログ・ボックスが表示されます。

TransactionVision Server と連携する Agent をインストールするように選択した場合は，インストール中に追加画面が表示されます。TransactionVision 環境での Agent の使用の詳細については，『TransactionVision Deployment Guid』を参照してください。

［Configure the .NET Agent for TransactionVision］ダイアログ・ボックスが表示されます。



Messaging Middleware Provider を選択します。WebSphere MQ または SonicMQ を選択できます。

SonicMQ は，.NET Agent に含まれます。このオプションを指定すると，Sonic MQ .NET クライアント（Sonic.Client.dll - Progress SonicMQ .NET Client, バージョン 7.6.0.112）が Agent インストールの一部としてインストールされます。

あるいは，サードパーティの WebSphere MQ インストールを使用することもできます。この場合，監視するホストに MQ シリーズ .NET クライアント（amqmdnet.dll - WebSphere MQ Classes for .NET，バージョン 1.0.0.3）をインストールする必要があります。

標準設定では，SonicMQ が選択されます。

➤ SonicMQ については，次のように入力します。

Broker : Sonic ブローカが実行されるホスト名。普通は Analyzer ホスト名です。

Port : ブローカが通信を行うポート。標準設定では 21111 です。

Configuration Queue : 構成キューの名前。標準設定では TVISION.CONFIGURATION.QUEUE です。

User : SonicMQ インストールで接続に必要な場合は，ユーザ ID。標準設定では，ユーザ名は不要です。

Password : SonicMQ インストールで必要とされる場合は，パスワード。これは，**PassGen** ユーティリティを使用して作成される暗号化形式になります。標準設定では，パスワードは不要です。**PassGen** の詳細については，『Using Transaction Management』の「Command-Line Utilities」を参照してください。

➤ WebSphere MQ については，次のように入力します。

Host : WebSphere MQ キュー・マネージャが常駐するホスト。

Port : WebSphere MQ キュー・マネージャのポート番号。

Configuration Queue : 構成キューの名前。

User : WebSphere インストールで接続に必要な場合は，ユーザ ID。

Password : WebShere MQ インストールで接続に必要な場合は，パスワード。これは，PassGen ユーティリティを使用して作成される難読化形式になります。**PassGen** の詳細については，『Using Transaction Management』の「Command-Line Utilities」を参照してください。

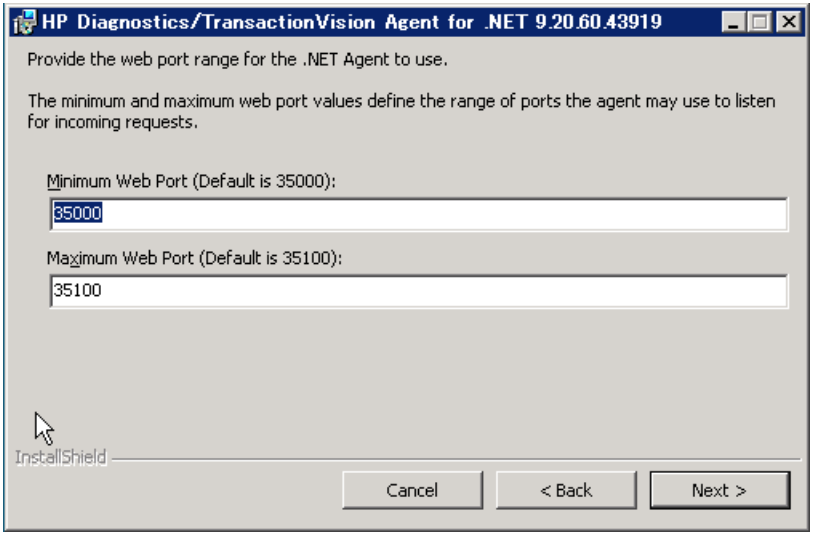
Websphere MQ channel : WebSphere MQ キュー・マネージャのチャネル名。

Websphere MQ Q Manager : WebSphere の CCSID。

［**Next**］をクリックして次の手順に進みます。

Profiler モードで Agent をインストールする場合は，次のダイアログ・ボックスが表示されます。

使用する .NET Agent の Web ポート範囲を指定します。



- **Minimum Web Port :** Agent に割り当てる Agent ホストのポート範囲の最小ポート番号を入力します。
- **Maximum Web Port :** Agent に割り当てる Agent ホストのポート範囲の最大ポート番号を入力します。

注 : デフォルトの範囲は，35000 以上 35100 以内です。

Web ポート範囲の上限と下限は，［**Minimum Web Port**］および［**Maximum Web Port**］フィールドで定義されます。Web ポート範囲には，Agent が使用可能なポートが含まれます。

Agent は，起動時に，この範囲の中から，最小ポート番号から最大ポート番号に向かって空きポートを特定しようとします。すでにほかの Agent またはアプリケーションが要求した場合は，範囲内のポートが使用中の可能性があります。

ポート範囲の最小サイズは，Agent のホストで同時に動作する Agent の最大数と等しくなります。

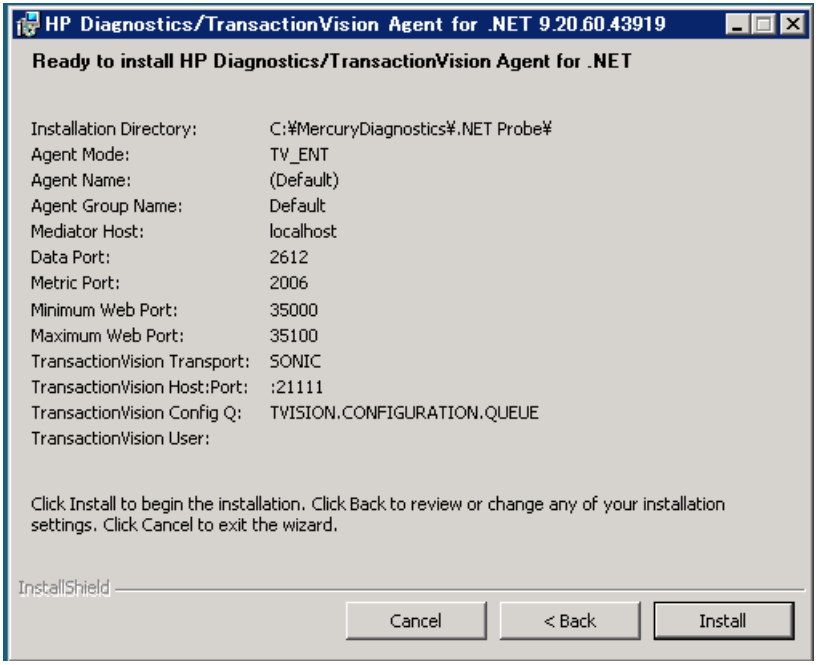
Web ポート範囲を設定する際の検討事項

- Agent が ASP.NET アプリケーションと連携して動作している場合，ASP.NET の appdomain の再利用のために，ポートの数を 2 倍にすることをお勧めします。
- Agent と，Agent と通信するコンポーネントの間にファイアウォールを置いている場合，範囲内のポートに対してファイアウォールを開く必要があります。そのために，場合によっては範囲を広げる必要があります。

［**Next**］をクリックして次の手順に進みます。

手順 8. プレインストール・サマリ

プレインストール・サマリ画面が開きます。変更を加える場合は［**Back**］をクリックします。.NET Agent のインストールを始める場合は，［**Install**］をクリックします。



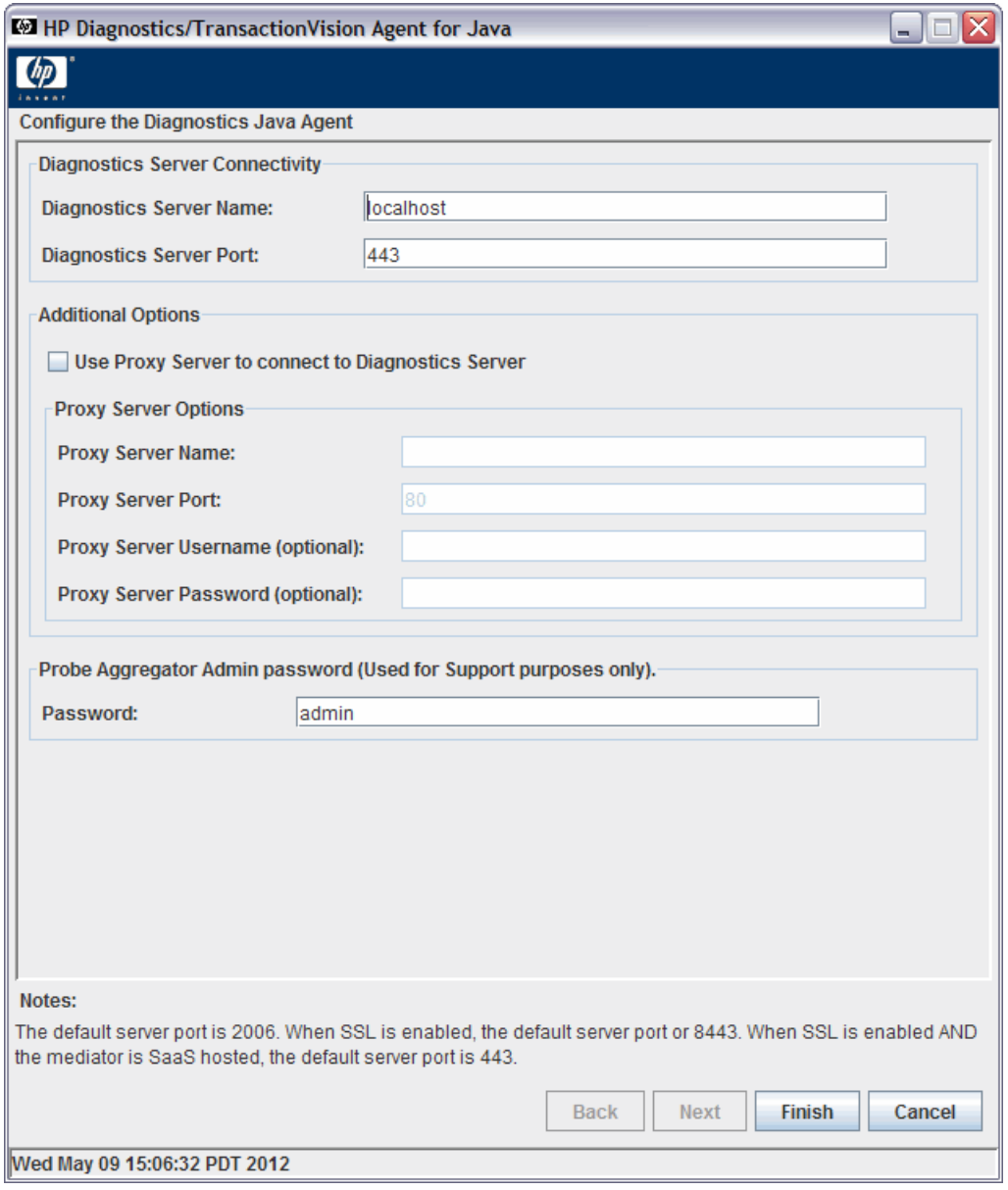
注 : Profiler として使用するためだけに Agent をインストールする場合, 測定値ポートの接続性テストはありません。

Agent を HP SaaS 環境で動作するようにインストールしている場合は手順 9 に進み，それ以外の場合は次の手順をスキップして手順 10 に進んでください。

手順 9. Agent を HP SaaS 環境で動作するための追加のセットアップ

Agent を HP SaaS 環境で動作するようにインストールしている場合，SaaS セットアップ・モジュールが自動的に開始され，[**スタート**] > [**すべてのプログラム**] > [**HP Diagnostics .NET プローブ**] > [**SaaS Setup**] を選択することでいつでも SaaS セットアップ・モジュールを実行できます。

SaaS セットアップ・モジュールで次のダイアログが表示されます。HP SaaS 環境用の Agent をセットアップしていない場合，このダイアログは表示されません。


HP Diagnostics/TransactionVision Agent for Java
Configure the Diagnostics Java Agent
Diagnostics Server Connectivity
Diagnostics Server Name:
localhost
Diagnostics Server Port:
443
Additional Options
Use Proxy Server to connect to Diagnostics Server
Proxy Server Options
Proxy Server Name:
Proxy Server Port:
80
Proxy Server Username (optional):
Proxy Server Password (optional):
Probe Aggregator Admin password (Used for Support purposes only).
Password:
admin
Notes:
The default server port is 2006. When SSL is enabled, the default server port or 8443. When SSL is enabled AND the mediator is SaaS hosted, the default server port is 443.
Back
Next
Finish
Cancel
Wed May 09 15:06:32 PDT 2012

- **Diagnostics サーバの接続**: HP SaaS 環境では, Diagnostics サーバは HP によって HP の施設にあるシステムでセットアップされます。SaaS 環境の標準設定のポートは **443** です。使用する Diagnostics サーバのホスト名およびポートの情報は，HP SaaS 管理者によって提供されます。
- プロキシ・サーバを使用して Diagnostics メディエータ・サーバと通信する場合，［**プロキシ サーバを使用して Diagnostic サーバに接続する**］チェックボックスをオンにして適切なオプションを入力します。HP SaaS 環境では, 会社で外部のサーバへの通信にプロキシが必要になる場合にこのオプションを選択します。

  プロキシ・サーバのオプション：

  - **プロキシ・サーバ名**：プロキシ・サーバのホスト名。
  - **プロキシ・サーバ・ポート**：プロキシ・サーバのポート。
  - **プロキシ・サーバ・ユーザ名（オプション）**：プロキシ・サーバの認証に使用するユーザ。
  - **プロキシ・サーバ・パスワード（オプション）**：プロキシ・サーバの認証に使用するパスワード。
- **Probe Aggregator 管理者のパスワード**：パスワードは自動的に .NET Profiler 管理パスワード（手順 5 で入力）と同じパスワードに設定されるため，SaaS 用に最初に Agent をセットアップするときには，このフィールドは表示されません。後から Probe Aggregator 管理パスワードを変更する場合は，SaaS セットアップ・モジュールを再び実行すればこのフィールドが表示されます。

次の手順に進んでインストールを完了します。

手順 10. インストール後の情報

最後のインストール画面で，［Show the Windows Installer Log］チェック・ボックスを選択することで，ログ・ファイルを表示してエラーを確認できます。

インストーラを終了するには，［**Finish**］をクリックしてください。

SaaS 環境で Agent を使用している場合，HP SaaS によってホストされるサーバで証明書が生成され, インストールおよびセットアップ中にダウンロードされます。.NET Agent 用のホストに証明書をコピーしてインポートする必要があります。詳細については, 276 ページ「SaaS 環境の場合 - 証明書のインポート」を参照してください。

インストール後の作業の詳細については，275 ページ「インストール後の作業」を参照してください。

準備が完了したら IIS を再起動する必要がありますが，次の手順を確認してください。

手順 11. IIS の再起動

ASP.NET アプリケーションで .NET Agent を使用するには，Agent のインストールおよびセットアップの終了後，IIS または Web パブリッシング・サービスを再起動する必要があります。

コマンド・ラインまたは［**スタート**］>［**ファイル名を指定して実行**］メニューから IIS を再起動するには，iisreset と入力して **Enter** キーを押します。

Diagnostics の場合，このコマンドによって Web パブリッシング・サービスが再起動しますが，.NET Agent はすぐに起動しません。次回，アプリケーションで Web ページが要求され，Agent が起動し，アプリケーションがインストルメントされ，Agent が Diagnostics コマンド・サーバに登録を行います。

TransactionVision の場合，このコマンドによって Web パブリッシング・サービスが再起動しますが，.NET Agent はすぐに起動しません。次回，アプリケーションで Web ページが要求されると，エージェントが起動し，アプリケーションがインストルメントされ，エージェントが Analyzer から構成キュー・メッセージを読み取ります。

注 : ASP.NET は，アプリケーションが再配備されたことを検出した場合や，設定されているリソースしきい値をアプリケーションが超えている場合など，さまざまな状況でアプリケーションを自動的に再起動します。

ASP.NET が .NET Agent によって監視されているアプリケーションを再起動すると，Agent は非アクティブになり，新しい Agent が起動します。このような状況では，複数の Agent が同時に Diagnostics コマンド・サーバに登録し，Diagnostic メディエータ・サーバに接続しているという重複期間が生じている可能性があります。この場合，LoadRunner / Performance Center および Business Service Management は，アプリケーションの再起動シーケンス中にエラーを報告することがあります。

次の項に進んで，インストール後の作業の詳細を確認します。

インストールの確認の詳細については，276 ページ「.NET Agent インストールの確認」を参照してください。

インストール後の作業

.NET Agent の追加設定の詳細については，次のトピックを参照してください。

- .NET Agent が自動的にアプリケーションを検出し, 標準インストルメンテーションを設定して監視を有効にする方法の詳細については，281 ページ「検出および標準インストルメンテーション」を参照してください。
- Diagnostics 用の .NET Agent 設定の詳細，およびより詳しいトピックへのリンクについては，278 ページ「Diagnostics 用の .NET Agent 設定について」を参照してください。
- TransactionVision 用の .NET Agent 設定の詳細，および TransactionVision で .NET Agent を使用してトレースできるイベントのタイプについては, 279 ページ「TransactionVision 用の .NET Agent 設定について」を参照してください。
- 289 ページ「アプリケーションの標準インストルメンテーションの有効化と無効化」
- プロキシを使用した環境での設定については 627 ページ「HTTP プロキシ用の Diagnostics サーバおよび Agent の設定」を，ファイアウォールについては 631 ページ「ファイアウォール環境で動作するための Diagnostics の設定」を, HTTPS の有効化については 797 ページ「コンポーネント間での HTTPS 有効化」を参照してください。

.NET Agent インストールの確認

最後のインストール画面で，［**Show the Windows Installer Log**］チェック・ボックスを選択することで，ログ・ファイルを表示してエラーを確認できます。

プローブが起動するまで，.NET Agent は Diagnostics サーバに登録されません。インストルメントされたアプリケーションを実行すると，プローブが起動して Diagnostics サーバに登録されます。ASP.NET アプリケーションでは，インストルメントされるアプリケーションのページが要求されたときに初めて登録されます。

.NET のプローブ・インスタンスが開始されたら，Diagnostics Enterprise UI を起動してプローブが動作していることを確認できます。**http:// ＜ Diagnostics コマンダ・サーバ＞ :2006/** に移動します。標準設定のユーザ / パスワード（**admin/admin**）を使用するか，異なるログイン情報が設定されている場合はそのログイン情報を使用できます。

または，［システムの状況］ビューで，.NET Agent デプロイメントおよび .NET Agent をホストするマシンに関する情報を確認することもできます。

システム・ビューにアクセスするには，次の手順を実行します。

1. http://＜Diagnostics コマンド・サーバ名＞:2006/query/ から Mercury System カスタマとして Diagnostics UI を開きます。
2. クエリ・ページで，リスト内から Mercury System カスタマを見つけて，Diagnostics を開くリンクを選択します。
3. Diagnostics にログインして，［アプリケーション］ウィンドウで［全組織］を選択し，Diagnostics ビューを開くためのリンクを選択します。
4. ［ビュー］表示枠にシステム・ビューのビュー・グループが表示されます。ビュー・グループを開き，システムの状況ビューまたはシステム・キャパシティ・ビューを選択します。

SaaS 環境の場合 - 証明書のインポート

SaaS 環境で Agent を使用している場合，HP SaaS によってホストされるサーバで証明書が生成され，.NET Agent ホスト・システムにダウンロードされます。証明書は次に説明するようにインポートする必要があります。非 SaaS 環境で .NET Agent 用に HTTPS 通信を有効化する場合は，付録 C,「コンポーネント間での HTTPS 有効化」を参照してください。

HP SaaS によってホストされるサーバから証明書をインポートするには，次の手順を実行します。

1 HP SaaS によってホストされるサーバの証明書は，次のファイルとして .NET Agent ホスト・システムにダウンロードされます。

C:¥MercuryDiagnostics¥.NET Probe¥ProbeAggregator¥etc¥jasm_server0.cer
C:¥MercuryDiagnostics¥.NET Probe¥ProbeAggregator¥etc¥jasm_server1.cer

2 .NET Agent ホスト・システムの Windows タスクバーで，［**スタート**］＞［**ファイル名を指定して実行**］を選択します。

3 mmc と入力し，［**OK**］をクリックして Microsoft 管理コンソールを実行します。

4 Micorosoft 管理コンソールのメニューで，［**ファイル**］＞［**スナップインの追加と削除**］をクリックして［スナップインの追加と削除］ダイアログを表示します。

5 ［スナップインの追加と削除］ダイアログで［**追加**］をクリックします。

6 ［利用できるスタンドアロン　スナップイン］リストから［**証明書**］を選択して，［**追加**］をクリックします。

7 ［証明書スナップイン］ダイアログ・ボックスで［**コンピュータ　アカウント**］を選択して，［**次へ**］をクリックします。

8 ［コンピュータの選択］ダイアログ・ボックスで，［**ローカル コンピュータ：（このコンソールを実行しているコンピュータ）**］を選択します。

9 リストから［**Worldwide Web 発行サービス**］を選択し，［**完了**］をクリックします。

10 ［スタンドアロン スナップインの追加］ダイアログで［**閉じる**］をクリックします。

11 ［スナップインの追加と削除］ダイアログで［**OK**］をクリックします。

12 Microsoft 管理コンソールでは，［コンソール ルート］ダイアログの左側のペインに，証明書（ローカル・コンピュータ）のリストが展開されます。

13 ［証明書（ローカル コンピュータ）］で，［信頼されたルート証明機関］を展開します。

14 ［信頼されたルート証明機関］で［証明書］を右クリックし，［**すべてのタスク**］＞［**インポート**］を選択して［証明書のインポート ウィザード］を開きます。

15 ［**次へ**］をクリックして，［証明書のインポート ウィザード］の次の画面に進みます。

16 ［**参照**］をクリックして，証明書がダウンロードされた C:¥MercuryDiagnostics¥.NET Probe¥ProbeAggregator¥etc¥ ディレクトリに移動します。それぞれの証明書を選択してインポートします。

a［**次へ**］をクリックしてファイルをインポートします。

b［**次へ**］をクリックして，「信頼されたルート証明機関」のデフォルトの証明書の保存場所を受け入れます。

17［証明書のインポート ウィザードの完了］で［**完了**］をクリックします。

18［証明書のインポート ウィザード］の確認ダイアログで［**OK**］をクリックします。［信頼されたルート証明機関］で［**証明書**］を選択し，追加した証明書を見つけます。

19 IIS を再起動します。

Diagnostics 用の .NET Agent 設定について

.NET Agent 設定のカスタマイズや，カスタム・インストルメンテーションの追加は，使用環境や診断するパフォーマンス問題に合わせて実行できます。

インストーラは，一緒に機能してアプリケーションの基本負荷をキャプチャするように，ASP.NET アプリケーションと .NET Agent を設定します。1 つ以上の ASP.NET アプリケーションが，インストーラが検出できない方法で配備されていることがあります。また，アプリケーションのカスタム・クラスのパフォーマンス・メトリックスをキャプチャするように，標準インストルメンテーションを拡張する必要があることがあります。

Diagnostics では，**probe_config.xml** ファイルを使用して追加設定を行うことができます。このファイルの詳細については，第 13 章，「.NET Agent 設定ファイルについて」を参照してください。.NET Agent の詳細設定については，第 14 章，「.NET Agent の詳細設定」を参照してください。

Diagnostics では，アプリケーション環境に固有の状況を処理するためのカスタム・インストルメンテーション・ポイントも作成できます。カスタム・インストルメンテーションに関する一般的な情報については，第 10 章，「.NET アプリーションのカスタム・インストルメンテーション」を参照してください。

TransactionVision 用の .NET Agent 設定について

TransactionVision で使用する .NET Agent では，.NET アプリケーションからイベントをキャプチャして，そのイベントを TransactionVision Analyzer に送信します。TransactionVision の詳細については，Business Service Management 文書ライブラリを参照してください。

TransactionVision 用の .NET Agent 設定

.NET Agent の標準設定を使用して，監視されるアプリケーションの特定の .NET イベントのトレースを開始できます。.NET Agent 設定をカスタマイズして，.NET イベントのトレースと TransactionVision Analyzer への送信を制御できます。

標準設定を上書きするには，**＜ Agent のインストール・ディレクトリ＞ /etc/probe_config.xml** ファイルにアクセスします。Diagnostics と TransactionVision の両方に設定できる要素の詳細については，507 ページ「.NET Agent 設定ファイルについて」を参照してください。

probe_config.xml ファイルの <modes> 要素は，Diagnostics と TransactionVision の両方のインストール時に設定されます（548 ページ「<modes> 要素」を参照）。

TransactionVision 環境で機能する .NET Agent のインストールを選択した場合，**probe_config.xml** ファイルの <modes> 要素は tv に設定されます。これが選択された唯一のモードである場合，Agent は TV のみのモードで機能します。つまり，Profiler および Diagnostics プローブが無効になり，TV イベントのみが生成されます。Diagnostics などのほかのモードで機能する .NET Agent のインストールを選択した場合，TV イベントと Diagnostics データ収集の両方が有効になります。

TransactionVision 固有の設定および TransactionVision トランスポート固有の設定を指定するために，**probe_config.xml** ファイルの次の要素が TransactionVision 専用に使用されます。

- <tv> 要素（詳細については，575 ページ「<tv> 要素」を参照）
- <timeskew> 要素（詳細については，570 ページ「<timeskew> 要素」を参照）
- <transport> 要素（詳細については，572 ページ「<transport> 要素」を参照）

➤ <gentvhttpeventforwcf> 要素（詳細については，526 ページ「<gentvhttpeventforwcf> 要素」を参照）

.NET Agent で TransactionVision Analyzer との通信に SonicMQ トランスポートが使用されている場合，SSL を有効化できます。詳細については，『HP Business Service Management Hardening Guide』(PDF) を参照してください。

標準設定では，.NET Events は相関されていません。相関を有効にするには，HP TransactionVision ドキュメントを参照してください。

TransactionVision で .NET Agent を使用してトレースできるイベントのタイプ

TransactionVision で使用する .NET Agent では，次のタイプの .NET イベントをトレースします。

1 Web サービス

a ASP.NET（*.asmx）- クライアントおよびサーバ

イベントを生成するには，**ASP.NET.points** ファイルを使用します。

b WCF（*.svc）- クライアントおよびサーバ

イベントを生成するには，**wcf.points** ファイルを使用します。

c REST スタイル・サービス - サーバ

イベントを生成するには，**wcf.points** ファイルを使用して，404 ページ「監視のための WCF REST サービスの設定」の説明に従ってアプリケーションのインストルメンテーションを設定します。

2 ADO.NET を使用して実行されるデータベース呼び出し

イベントを生成するには，**ADO.points** ファイルを使用します。

3 .NET リモート処理 - クライアントおよびサーバ

.NET リモート処理イベントを生成するには，**Remoting.points** ファイルを使用して，411 ページ「.NET Remoting のインストルメンテーションを設定する方法」の説明に従ってインストルメンテーション用のアプリケーションを設定します。

4 MSMQ - 送信および受信（非同期）

イベントを生成するには，**Msmq.points** ファイルを使用します。

5 HTTP

a クライアント発信 - REST サービスへの呼び出しを含む

イベントを生成するには，**ASP.NET.points** ファイルを使用します。

b ASP.NET 受信 / サーバ（POST，GET，PUT）（***.aspx**）

HTTP のイベントを生成するには，**ASP.NET.points** ファイルを使用します。

6 ユーザ定義イベント

detail 引数 **tv:user_event** を使用します（392 ページ「ポイント・エントリ（省略可能）」を参照）。

イベントの生成を無効にするには, 関連するポイント・ファイルを範囲から削除します。

検出および標準インストルメンテーション

.NET Agent インストーラは，インストルメント可能な ASP.NET アプリケーションを自動的に検出します。.NET Agent のインストール後，Agent で IIS 設定を再スキャンして追加または変更を特定するように要求できます。

インストール時の ASP.NET アプリケーションの検出

.NET Agent インストーラは，Agent のインストール時に，マシン上の ASP.NET アプリケーションを検出します。.NET Agent インストーラは，IIS 設定を調べ，ASP.NET アプリケーションを参照することがある仮想ディレクトリ・エントリを探して，アプリケーションを検出します。

ASP.NET アプリケーションは，検出されない方法でインストールされていることがあります。例として，ASP.NET アプリケーションが仮想ディレクトリとしてでなく，Web ディレクトリとしてインストールされている場合があります。

インストール後の ASP.NET アプリケーションの検出

既存の ASP.NET アプリケーション・デプロイメントを変更した場合，または新しい ASP.NET アプリケーションをインストールした場合は，IIS 設定の再スキャンを要求できます。

IIS 設定を再スキャンして **probe_config.xml** ファイルを更新するように Agent に要求するには，［**スタート**］＞［**HP Diagnostics .NET プローブ**］＞［**Rescan ASP.NET Applications**］を選択します。

検出した ASP.NET アプリケーション用の自動インストルメンテーションおよび設定

.NET Agent インストーラは，検出された各 ASP.NET アプリケーションの ASP.NET/ADO/WCF 基本負荷をキャプチャするように Agent を設定します。Agent は，次の設定手順を実行します。

➤ アプリケーション特有のキャプチャ・ポイント・ファイル・テンプレートを作成する。

キャプチャ・ポイント・ファイルでは，各アプリケーションに対して Agent がキャプチャする負荷を制御するインストルメンテーションを定義します。キャプチャ・ポイント・ファイルのインストルメンテーションを変更して，Agent がアプリケーション固有のカスタム・メソッドのパフォーマンス・データをキャプチャできるようにする命令を指定できます。

➤ **＜プローブのインストール・ディレクトリ＞ /etc** ディレクトリにある **probe_config.xml** ファイルに **appdomain** タグを作成します。**appdomain** タグの属性は，.NET Agent の動作（ポイントと有効な属性）を決定します。詳細については，第 13 章，「.NET Agent 設定ファイルについて」を参照してください。

注 : Diagnostics は，**process** タグの **enablealldomains** 属性を **true** に設定して appdomain タグの有効な属性を上書きすることにより，検出されたすべてのアプリケーションのインストルメンテーションを有効にします。アプリケーションのインストルメンテーションの有効化および無効化については，288 ページ「記録の無効化」を参照してください。

IIS によって配備された ASP.NET アプリケーションの CI 作成用 IIS メタデータの検出

Diagnostics 9.0x 以降の場合，Diagnostics はアプリケーション・インフラストラクチャの要素とビジネス・トランザクションに対して，Business Service Management 9.0 以降の Run-time Service Model（RTSM）内に CI とモデルの関係を作成します。

.NET Agent インストーラは CI 作成のために，IIS バージョン 6.x 以降でデプロイされた ASP.NET アプリケーション用の IIS 設定メタデータを自動的に検出します。検出された IIS 設定メタデータは，**＜プローブのインストール・ディレクトリ＞ ¥etc** ディレクトリ内の **iis_discovery_data.xml** ファイルに記述されます。.NET Agent のインストール後，Agent が IIS 設定を再スキャンし，追加または変更に合わせて更新するように要求できます。

注 : この情報は，Business Service Management 9.0 以降との統合に関するものです。

➤ IIS によって配備された ASP.NET アプリケーション用 CI の実行時作成

実行時に，.NET Agent は **iis_discovery_data.xml** ファイル内で，インストルメントされた appdomain に関連する IIS 設定メタデータを検索します。関連付けられたメタデータが見つかると，Agent はデータを Diagnostic サーバに転送し，Diagnostic サーバは .NET アプリケーション用の Business Service Management Run-time Service Model CI を作成します。.NET アプリケーションに関する Business Service Management 9.0 Run-time Service Model モデルとの統合については，第 21 章，「Business Service Management と Diagnostics 間の統合のセットアップ」を参照してください。

➤ インストール時に IIS によって配備された ASP.NET アプリケーションの IIS メタデータの検出

.NET Agent インストーラは，Agent のインストール時に，IIS によって配備された ASP.NET アプリケーションをマシン上で検出します。.NET Agent インストーラは WMI（WMEB）プロバイダ内で IIS 設定メタデータを検索して，アプリケーションを検出します。関連するメタデータは，**iis_discovery_data.xml** ファイルに書き込まれます。

➤ インストール後に IIS によって配備された ASP.NET アプリケーションの IIS メタデータの検出

既存の ASP.NET アプリケーション・デプロイメントを変更したとき，または新しい ASP.NET アプリケーションをインストールしたときに，IIS 設定メタデータの再スキャンを要求する必要があります。IIS 設定を再スキャンして **iis_discovery_data.xml** ファイルに記述するように Agent に要求するには，［**スタート**］＞［**HP Diagnostics .NET プローブ**］＞［**Rescan ASP.NET Applications**］のショートカットを実行します。新しい **iis_discovery_data.xml** ファイルは，ユーザが編集するためのものではないことに注意してください。このショートカットを実行すると，このようなユーザ編集が上書きされます。

- IIS によって配備された ASP.NET アプリケーションの権限の要件

  ユーザ名には .NET Agent がインストールされたマシンに関する Administrator 権限が必要です。この権限がないと，WMI クエリに失敗し，**iis_discovery_data.xml** ファイルが作成されません。

- IIS によって配備された ASP.NET アプリケーションの検出のデバッグ

  iis_discovery_data.xml ファイルが作成されない場合，または一部のメタデータが不正確であるとみなされる理由がある場合は，詳細なデバッグ・ファイルを作成して，WMI クエリの結果を調べることができます。詳細なデバッグ・ファイルを作成できるようにするには，［**スタート**］＞［**HP Diagnostics .NET プローブ**］＞［**Rescan ASP.NET Applications**］ショートカットを選択して，［Target Property］の最後のパラメータを「false」から「true」に変更します。［Rescan ASP.NET Applications］ショートカットを実行すると，**＜プローブのインストール・ディレクトリ＞ /log/AutoDetect.log** が作成されます。このショートカットを実行する場合は，Administrator 権限が必要なことに注意してください。HP サポートに **AutoDetect.log** を送信して，分析できます。

非 ASP.NET アプリケーション

.NET Agent をインストールすると，自動的に ASP.NET アプリケーションが検出され，**probe_config.xml** に各アプリケーションの設定が作成され，各アプリケーション用のテンプレート・ポイント・ファイルが作成されます。NT サービス，コンソール・アプリケーション，UI クライアントなどの非 ASP.NET アプリケーションについては，**probe_config.xml** に適切な設定を作成して，各アプリケーションを監視するように .NET Agent を設定するとともに，アプリケーション内のどのポイントを監視するかを示すポイント・ファイルを作成する必要があります。

次に，**SimpleConsoleHost.exe** というアプリケーションに関する **probe_config.xml** の設定例を示します。

```
<process name="SimpleConsoleHost">
      <points file="SimpleConsoleHost.points"/>
      <logging level=" "/>
</process>
```

次に，SimpleConsoleHost.exe というアプリケーションに関するポイント・ファイルの設定例を示します。

```
[SimpleConsoleHost]
class = MyNamespace.SimpleConsoleHost
method = !.*
ignoreMethod = Main
layer = SimpleConsoleHost
```

詳細については，第 10 章，「.NET アプリーションのカスタム・インストルメンテーション」を参照してください。

Probe Aggregator サービス

Probe Aggregator サービスは，.NET Agent インストールの一環としてインストールできます。このサービスは Windows サービスである **HP Probe Aggregator** として実行されます。

Probe Aggregator サービスは，Diagnostic メディエータ・サーバにパフォーマンス・データを送信する前に，5 秒おきにプローブ・データを集計します。このサービスは，複数のアプリケーションのインストルメンテーションに基づいて収集されたデータのボリュームが大きく，集計しないとネットワーク・トラフィックが極端に増大する場合に便利です。データ処理の技術的な図については，850 ページ「.NET Probe Aggregator のデータ・フロー」を参照してください。

Probe Aggregator サービスを使用しない基本的な .NET Agent インストールでは，メソッドの開始や停止が発生すると，パフォーマンス・データが Diagnostic メディエータ・サーバに送信されます。

Probe Aggregator サービスを使用すると，パフォーマンスが犠牲になります。したがって，使用環境の要件を評価する必要があります。たとえば，複数の .NET プローブ・インスタンスが同じシステムで実行されている場合は，Probe Aggregator の使用を検討してください。ネットワークの実際のオーバーヘッドは，監視対象のアプリケーションによって異なるため，ネットワーク帯域幅を節約し，Mediator の負荷を軽減した場合，アプリケーション・システムのメモリ使用量が増大するかどうかを判別する必要があります。

.NET Agent と一緒に Probe Aggregator サービスをインストールした場合，Probe Aggregator サービスは自動的に実行され，.NET プローブからの接続を待機します。Probe Aggregator の標準設定は，.NET Agent インストール中に実行されます。Probe Aggregator の設定パラメータ（SQL トレンドしきい値など）を設定する場合は，**＜プローブのインストール・ディレクトリ＞¥ProbeAggregator¥etc¥probeaggregator.properties** ファイルが使用されます。

インストール後に Probe Aggregator サービスをインストールする場合は，.NET Agent インストールを再実行して，［**変更**］ボタンを選択します。その後，**Probe Aggregator サービス**をインストールするためのチェック・ボックスを選択します。

.NET Agent を削除またはアンインストールすると，Probe Aggregator サービスも削除されます。詳細については，287 ページ「Diagnostics Agent for .NET の有効化と無効化」を参照してください。

Probe Aggregator サービスを無効化または有効化する方法については，287 ページ「Diagnostics Agent for .NET の有効化と無効化」を参照してください。

Azure クラウドでデプロイされた .NET アプリケーションの監視

Microsoft では，Azure アプリケーションを作成して Microsoft Windows Azure クラウド・インフラストラクチャにデプロイする Windows Azure SDK を開発者に提供しています。Diagnostics .NET Agent は，Azure SDK を活用して Azure インフラストラクチャへの .NET Agent のシームレスなデプロイメントを可能にしています。デプロイされた .NET Agent は，Azure クラウドで実行中のアプリケーションを監視してパフォーマンス・データを収集し，分析して問題を検出するために HP Diagnostics サーバにレポートします。Windows Azure クラウドのアプリケーションを監視する .NET Agent のインストールおよび設定の詳細については，.NET Agent AzurePack zip ファイル内の **AzurePackReadMe.pdf** を参照してください。

.NET Agent のバージョンの確認

サポートを依頼する際，インストールしている Diagnostics コンポーネントのバージョンを把握しておくと便利です。

.NET Agent のバージョンを確認するには，次の手順を実行します。

➤ **＜ Agent のインストール・ディレクトリ＞ ¥bin¥HP.Profiler.dll** を右クリックし，メニューから［**プロパティ**］を選択します。［プロパティ］ダイアログで，［バージョン］タブを選択してコンポーネントのバージョン情報を表示します。

Diagnostics UI のシステムの状況ビューを使用することもできます（付録 D，「管理者用のシステム・ビューの使用」）。

Diagnostics Agent for .NET の有効化と無効化

.NET Agent はインストール時に有効になります。Web サーバを再起動してアプリケーションの URL にアクセスすると，.NET Agent はパフォーマンス情報の収集を開始します。

起動してパフォーマンス・メトリックスを収集しないように .NET Agent を無効にできます。

.NET Agent を無効にするには，次の手順を実行します。

➤［**スタート**］>［**すべてのプログラム**］>［**HP Diagnostics .NET Probe**］>［**Disable HP .NET Probe**］を選択します。

無効にした .NET Agent を有効にするには，次の手順を実行します。

➤［**スタート**］>［**すべてのプログラム**］>［**HP Diagnostics .NET Probe** ］>［**Enable HP .NET Probe**］を選択します。

注 : .NET Agent を無効にすると, プローブ測定値コレクタとアクティブなプローブだけが無効になります。システム測定値コレクタは無効になりません。システム・メトリックスの収集を有効または無効にするプロセスは，標準の Windows サービス・マネージャを通じて制御されます。プローブを有効または無効にした効果は，調査されたアプリケーションを次に再起動したときのみ現れます。現在実行中のアプリケーションには影響がありません。

Probe Aggregator サービスがインストールされていて，実行されている場合は，[スタート]メニューからこのサービスを有効または無効にできます。[**スタート**]>[**すべてのプログラム**]>[**HP Diagnostics .NET Probe**]>[**Disable HP .NET Probe**]または［**Enable HP .NET Probe**］を選択します。［Disable HP .NET Probe］を選択すると，.NET プローブが無効化されて Probe Aggregator サービスに無効のマークが付けられますが, サービスは停止しません (実行中のプローブが残っている場合)。［Enable HP .NET Probe］を選択すると，.NET プローブが有効にになって，Probe Aggregator サービスが **automatic** タイプに戻り，必要に応じて起動されます。

記録の無効化

次の例のように, **probe_config.xml** ファイルの ASP.NET プロセス・セクションの **logging level** タグを変更して，プローブ・アプリケーションのログ記録機能を無効にできます。

```
<process name="ASP.NET">
      <logging level="off"/>
</process>
```

次の例のように，インストルメンテーション・セクションの **logging level** タグを変更して，プローブ・のインストルメンテーションの記録機能を無効にできます。

```
<instrumentation>
      <logging level="off" />
</instrumentation>
```

アプリケーションの標準インストルメンテーションの有効化と無効化

.NET Agent を初めてインストールしたときに，検出されたすべてのアプリケーションの標準の ASP.NET / ADO インストルメンテーションは有効になりますが，アプリケーション特有のインストルメンテーションは有効になりません。インストルメンテーションが有効または無効になっているアプリケーションを制御するには，.NET Agent の **probe_config.xml** ファイル内にある <process> 要素の enablealldoamins 属性や，<appdomain> 要素の属性を使用します。

アプリケーションのインストルメンテーションを無効にすると，プロセスのオーバーヘッドや，監視するパフォーマンスがある環境とは無関係のアプリケーションの Diagnostics ビューの情報を混乱させるのを回避できます。

すべてのアプリケーションのインストルメンテーションを有効にすると，.NET Agent は検出されたすべてのアプリケーションのパフォーマンスを監視できるようになるため，Diagnostics および Profiler ユーザ・インタフェースのビューで，すべてのアプリケーションのパフォーマンス測定値を確認できるようになります。

次に，< process >要素の enablealldomains attribute 属性に関するルールを示します。

- enablealldomains = false : < appdomain >のリスト内にドメインがない場合は，どのドメインも有効になりません。
- enablealldomains = false : < appdomain >のリスト内にドメインがある場合は，「enable」属性が true に設定されているか，< appdomain >の enable 属性で定義されていないときに，ドメインが有効になります。
- enablealldomains = true : < appdomain >のリスト内にドメインがある場合は，「enable」属性に関係なく，リスト内のドメインのみが有効になります。
- enablealldomains = true : < appdomain >のリスト内にドメインがない場合は，すべてのドメインが有効になります。
- enablealldomains 属性が定義されていない : enablealldomains = true の場合と同様です。

アプリケーションのインストルメンテーションを有効または無効にするには，次の手順を実行します。

1 **<process>** 要素の **enablealldomains** 属性を **false** に設定します。これにより，各 appdomain タグの属性は，それぞれのアプリケーションのインストルメンテーションの状態を制御できるようになります。appdomain エントリがなければ，有効なアプリケーションはありません。

2 インストルメンテーションを有効にする各アプリケーションで，**<appdomain>** 要素の **enabled** 属性を **true** に設定します。

3 インストルメンテーションを無効にする各アプリケーションで，**<appdomain>** 要素の **enabled** 属性を **true** に設定します。

次に，あるアプリケーションに対してインストルメンテーションを有効にし，別のアプリケーションに対して無効にした場合の例を示します。

```
<process name="ASP.NET" enablealldomains="false">
    <points file="ASP.NET.points" />
    <points file="ADO.points" />
    <appdomain name="1/ROOT/myApplication" website=”Default Web Site”
enabled="true">
        <points file="DefaultWebsite-myApplication.points" />
    </appdomain>
    <appdomain name="1/ROOT/myApplicationTwo" website=”Default Web Site”
enabled="false">
        <points file="DefaultWebsite-myApplicationTwo.points"/>
    </appdomain>
</process>
```

すべてのアプリケーションのインストルメンテーションを有効にするには，次の手順を実行します。

➤ **<process>** 要素の **enablealldomains** 属性を **true** に設定します。これにより，各 **<appdomain>** 要素の属性の設定が変更されるため，さまざまな属性を設定しなくても，インストルメンテーションを有効にできます。

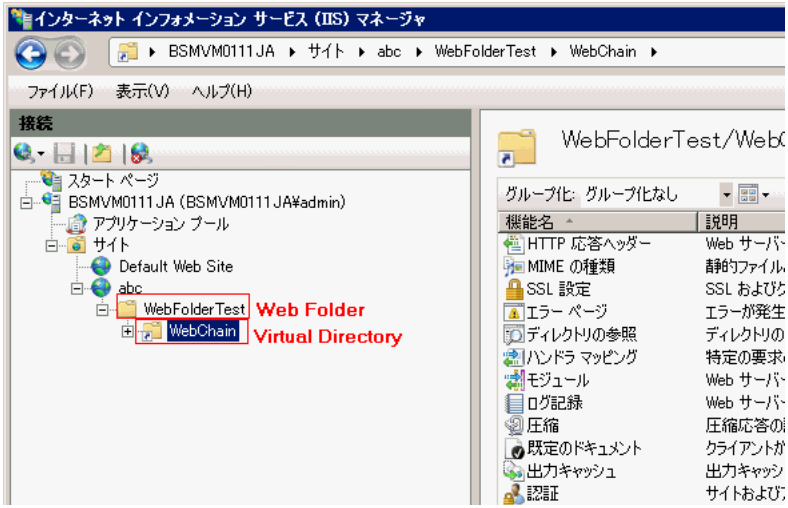
次に，すべてのアプリケーションに対してインストルメンテーションを有効にした場合の例を示します。

```
<process name="ASP.NET" enablealldomains="true">
    <logging level=""/>
    <points file="ASP.NET.points"/>
    <appdomain name="1/ROOT/myApplication" website=”Default Web Site”
enabled="false">
        <points file="DefaultWebsite-myApplication.points"/>
    </appdomain>
    <appdomain name="1/ROOT/myApplicationTwo" website=”Default Web Site”
enabled="false">
        <points file="DefaultWebsite-myApplicationTwo.points"/>
    </appdomain>
</process>
```

.NET Web アプリケーションが検出されない場合のトラブルシューティング

Microsoft Windows 2003 Server および IIS 6 環境では，Web サイトの Web フォルダ内に仮想ディレクトリがある場合，.NET Agent で仮想ディレクトリの検出に失敗する可能性があります。これは，Diagnostics で使用される Microsoft WMI プロバイダには Web サイト・ツリーの展開に問題があるためです。WMI プロバイダは Web フォルダを IIS Web ディレクトリとして正しく認識できないため，Diagnostics はそのフォルダ内の仮想ディレクトリを検出できません。次の例を参照してください。

この例では，Web サイト abc 内に WebFolderTest という Web フォルダがあります。この Web フォルダ内には WebChain という仮想ディレクトリがあります。



WMI プロバイダでの問題が原因で，この Web サイトの WMI によるリストには WebFolderTest/WebChain 仮想ディレクトリは表示されません。.NET Agent は, Web アプリケーションの検出に WMI プロバイダによるリストを使用します。そのため，このような状況では .NET Agent が Web フォルダ内の仮想ディレクトリを検出できない可能性があります。

Microsoft では，メタベースを直接修正するか，次のような簡単なスクリプトで ADSI を使用してフォルダ・スタイルを設定することを推奨しています。

```
Set objRoot = GetObject("IIS://localhost/W3SVC/1/Root/WebFolderTest")
objRoot.KeyType = "IIsWebDirectory"
objRoot.SetInfo()
```

スクリプトを使用する代わりに，IIS で Web フォルダをアプリケーションとして手動で設定できます。この設定が完了したら, そのフォルダを非アプリケーションに戻すことができますが，プロパティはすでに設定されているため，Diagnostics で Web アプリケーションとして検出できます。

別の方法として，**probe_config.xml** ファイルの ASP.NET appdomain リストで，除外された APPDOMAIN を手動で追加します。

その他の .NET Agent のトラブルシューティングのヒント

Agent を正常に起動できない場合は，次の内容を確認します。

- Web サーバを再起動してアプリケーションの URL にアクセスすることで, Agent がデータの収集を開始することを確認します。
- probe_config.xml ファイルが作成されていて，書式が正しいこと（欠落した閉じタグがない，など）を確認します。これを実行するには，Web ブラウザでファイルを開きます。
- Windows イベント・ログで,「HP Diagnostics」という名前のメッセージを確認します。このログは，.NET Agent のみによって使用されます。アプリケーションのインストルメンテーションを試行するたびにメッセージが記録されます。

.NET Agent のアンインストール

.NET Agent をアンインストールするには，次の手順を実行します。

1. SOAP を使用している Web アプリケーションをすべて停止します。
2. Windows の［コントロール パネル］から［**プログラムの追加と削除**］を選択し，［**HP Diagnostics/TransactionVision Agent for .NET**］を選択してアンインストールします。
3. Web アプリケーションを再起動します。

Probe Aggregator サービスを削除する場合は, .NET Agent をアンインストールして，Probe Aggregator サービスも削除できます。または，.NET Agent のインストールを再実行し，［**変更**］ボタンを選択して，［**Probe Aggregator Service**］をインストールするためのチェック・ボックスをクリアできます。

第 IV 部

Java および .NET アプリケーションを監視するためのカスタム・インストルメンテーション

本項の内容

- Java アプリーションのカスタム・インストルメンテーション
- .NET アプリーションのカスタム・インストルメンテーション

9

Java アプリーションのカスタム・インストルメンテーション

このセクションでは, .NET Agent によるパフォーマンス測定値の収集を可能にするために Diagnostics がアプリケーションのクラスとメソッドに適用するインストルメンテーションの制御方法について説明します。

本章の内容

インストルメンテーションとキャプチャ・ポイント・ファイルについて

インストルメンテーションとは，アプリケーションのクラス・ファイルが仮想マシンのクラス・ローダによって読み込まれるときにプローブがそのクラス・ファイルに挿入するバイトコードを参照することです。インストルメンテーションによって，プローブは実行時間のインストルメンテーション，呼び出しのカウント，引数の取り出し，例外の取得，およびメソッド呼び出しとスレッドの関連付けが可能になります。各プローブ・スタンスのインストルメンテーション・ポイントは，キャプチャ・ポイント・ファイルで指定されます。

Java Agent をインストールする際，あらかじめ定義されたキャプチャ・ポイント・ファイルが，使用しているプラットフォームに適したポイントのセットと一緒にインストールされます。定義済みの Java ポイントが含まれる標準設定のキャプチャ・ポイント・ファイルは，**＜プローブのインストール・ディレクトリ＞ ¥etc¥ auto_detect.points** にあります。

キャプチャ・ポイント・ファイルは，インストルメンテーションの範囲を制御できるようにします。これにより，アプリケーションのパフォーマンスを把握するのに必要なすべての情報が Diagnostics から提供され，ユーザは情報収集に余計な費用をかけたり無関係な情報に混乱することがなくなります。キャプチャ・ポイント・ファイル内のインストルメンテーション定義は，*ポイント*と呼ばれます。ポイントは，インストルメントするメソッド，インストルメント方法，およびインストールするインストルメンテーションを定義します。

ポイントには，命令を複数のメソッド，クラス，およびパッケージまたは名前空間の仕様に適用するために，命令を「ワイルドカード（未知数）にする」正規表現を含めることができます。正規表現の使用の詳細については，882 ページ「正規表現の使用」を参照してください。

カスタム・インストルメンテーションを追加するには，標準設定の **auto_detect.points** ファイルのコピーを作成し，別の名前を付け，それを使ってすべてのインストルメンテーション・カスタマイズを行います。この予防策により，Java Agent の新しいバージョンにアップグレードしたり，インストーラが auto_detect.points ファイルをオーバーレイしたりするときに，カスタム・インストルメンテーションが失われなくなります。

キャプチャ・ポイント・ファイルでポイントをカスタマイズして，標準設定ポイントの範囲に含まれないアプリケーションのエリアにメソッド，クラス，パッケージおよび名前空間を含めることができます。カスタム・ポイントが必要になる一般的なケースは，J2EE アプリケーションに javax.ejb.SessionBean インタフェースからの派生でないビジネス・ロジックが含まれる場合です。また，デフォルト・ポイントを上書きしてレイヤを変更したり，呼び出し側の特定のメソッドから追跡する場合も必要になります。

キャプチャ・ポイント・ファイルのポイントは，レイヤにグループ化されます。レイヤは，優先順位付けプロセスの一部として比較可能な，重要な情報の層にパフォーマンス測定値を体系付けます。また，インストルメンテーションの収集動作を制御します。

Java Agent にインストールされているキャプチャ・ポイント・ファイルのポイントは，標準設定のレイヤにグループ化されます。デフォルトのレイヤをカスタマイズして，新しいレイヤを作成できます。標準設定レイヤの詳細については，383 ページ「標準の Java クラスおよびメソッド用に定義された標準設定のレイヤ」を参照してください。

注 :

- 標準設定のキャプチャ・ポイント・ファイル名は，**＜プローブのインストール・ディレクトリ＞ ¥etc¥probe.properties** で指定します。
- 代わりにカスタム・ポイントを含むコピーが使用されるように，標準設定のファイル名を変更するには, -Dprobe.points.file.name=<newPointsFileName_NoExtension> JVM プロパティを使用します。

キャプチャ・ポイント・ファイルのポイントのコーディング

次の引数は，キャプチャ・ポイント・ファイルのポイントを定義する場合に使用できます。

```
[Point-Name]              =< ポイントの一意の名前 >
;-------------------------------------------------------------------------
class      = < クラス名または正規表現 >
method                    = < メソッド名または正規表現 >
signature                 = < メソッド・シグネチャまたは正規表現 >
ignore_cl                 = < クラス名または正規表現のリスト >
ignore_method             = < メソッド名または正規表現のリスト >
ignore_tree               = < クラス名または正規表現のリスト >
method_access_filter      = < クラス名または正規表現のリスト >
deep_mode                 = < ソフト・モードまたはハード・モード >
scope                     = < メソッドまたは正規表現のリスト >
ignoreScope               = < メソッドまたは正規表現のリスト >
detail                    = <list of specifiers>
layer                     = < レイヤ名 >
layerType                 = < レイヤ・タイプ >
rootRenameTo              = < 文字列 >
keyword                   = < キーワード >
priority                  = < 整数値 >
active                    = <True，False>
```

引数については，次のセクションで説明します。

- 301 ページ「必須ポイント引数」
- 302 ページ「ポイント・エントリ（省略可能）」

必須ポイント引数

CLP, LWMD, RMI および SAP RFC, HttpCorrelation および JDBC SQL を除くすべてのポイントには，次の引数が必ず含まれる必要があります。

引数	説明
Point-Name	ポイントの一意の名前。
class	インストルメントするクラスまたはインタフェースの名前を指定する。名前には，完全なパッケージ / 名前空間名が必要で，パッケージ・レベルの間にピリオドを使用します。任意の有効な正規表現を使用できる。
method	インストルメントするメソッドの名前を指定する。そのために，メソッド名は，class 引数によって指定されるクラスまたはインタフェースで定義されたメソッドと一致する必要があります。任意の有効な正規表現を使用できる。
signature	メソッド・シグネチャ（< JDK インストール> /bin.javap -s）のための javap シンボル・エンコーディングを使って，メソッドのシグネチャ（パラメータと結果の型）を指定する。
layer	レイヤ，サブレイヤ，またはこのポイントのデータがグループ化される層を指定する。レイヤは，インストルメンテーションの収集制御の一部です。 ポイントのレイヤは，/（スラッシュ）でレイヤ名を区切ることで，ネスト・レイヤまたはサブレイヤで指定できる。スラッシュの後に指定されたレイヤは，スラッシュの前に指定されているレイヤのサブレイヤです。サブレイヤ名の後にほかのスラッシュとレイヤ名をコーディングして，サブレイヤに独自のサブレイヤを設定できる。 UI で，レイヤのサブレイヤは，親レイヤの配下に表示される。たとえば，JSP および Struts というサブレイヤは，Web レイヤの下に表示され，Web から JSP および Struts へのドリルダウンが可能。

以下は，必須引数を含むカスタム・ポイントの例です。

```
[MyCustomEntry_1]
; comments here….
class = myPackage.myClass.MyFoo
method = myMethod
signature = !.*
layer = myCustomStuff
```

注：ポイントのほとんどの引数に正規表現を使用できます。それらの正規表現は，前に感嘆符を置く必要があります。正規表現の使用の詳細については，882 ページ「正規表現の使用」を参照してください。

ポイント・エントリ（省略可能）

ポイント定義には，次の引数を 1 つ以上含めることができます。

引数	説明
keyword	**keyword** は，特別なインストルメンテーション・クラスによって処理される・ポイントを示す。このキーワードの値は **inst.properties** のプロパティとして検索され，見つかったプロパティの値がインストルメンテーション・クラス名になる。特別なインストルメンテーション・ポイントでは，ここで説明していない実装固有の引数を使用できる（inst.properties ファイルのコメントを参照）。
ignore_cl	無視するクラス名または正規表現をカンマ区切りリストで指定する。**ignore_cl** で指定したクラスのいずれかに一致するクラスはインストルメントされません。
ignore_method	無視するメソッドをカンマ区切りリストで指定する。**ignore_method** で指定したメソッドのいずれかに一致するメソッドはインストルメントされません。

引数	説明
ignore_tree	クラスまたは正規表現のリスト。リストのいずれかの項目と一致するクラスのサブクラスはインストルメンテーションから除外される。
method_access_filter	メソッド指定子のカンマ区切りリスト。使用できる指定子は，**static**，**private**，**protected**，**package**，および **public** です。このポイントを照合するメソッドのアクセス指定子がこのリストのいずれかの値と一致すると，そのメソッドはインストルメントされない。
deep_mode	サブクラスの処理方法を指定する。この引数は，3 つの値を受け入れる。 ➤ **none** – 値「none」は，**deep_mode** 引数を指定しないものに類似する。サブクラスの処理方法に影響しません。 ➤ **soft** – 値「soft」は，クラス，メソッド，およびシグネチャのエントリに一致するすべてのクラスまたはインタフェースに対して，一致するメソッドとシグネチャを実装するサブクラスまたはサブインタフェースもインストルメントするように要求する。 ➤ **hard** – 値「hard」は，クラス，メソッド，およびシグネチャのエントリに一致するすべてのクラスまたはインタフェースに対して，任意の深さのサブクラスまたはサブインタフェースのすべてのメソッドをインストルメントするように要求する。通常, hard モードは, インタフェースのポイントで使われます。**注意 :** hard モードでは広範なインストルメンテーションが実行されるため，プローブのオーバーヘッドが非常に高くなる恐れがある。
scope	インストルメンテーションが実行されるコンテキストを制約する。この引数が指定されている場合, 挿入されたバイトコードは呼び出し側になる。この引数の値には，任意の有効な正規表現を使用できる。scope 値は，標準の Java 表記を使ったパッケージ，クラスおよびメソッド名のカンマ区切りのリストです。
ignoreScope	メソッド名または正規表現を一覧表示し, scope 引数に指定されたスコープに含まれる特定のパッケージ，クラスおよびメソッドを除外する。

引数	説明
detail	より具体的なキャプチャ命令を指定する。これは，以下のカンマ区切りリストです。 ➤ **caller** – 呼び出し側でインストルメンテーションを実行する。キーワードが指定されていない場合，デフォルトのインストルメンテーションである呼び出される側のインストルメンテーションが実行されます。 ➤ **args:n** – **n** 番目の引数の **toString()** メソッドを呼び出します。返された文字列は，Diagnostics コンソールのメソッドの引数フィールドに表示される。キャプチャされた文字列は，layer 引数の集計パラメータとして使用できます。**n** の値は **1** ～ **256** で指定可能です。 ➤ **args:0** – 現在のクラス・インスタンスまたは呼び出される側のオブジェクトで **toString()** を呼び出す。静的メソッドは，呼び出される側のオブジェクトのクラス名を返す。 ➤ **before:code: ＜コード・キー＞** – キーで指定されたコード・スニペットを，ポイントに適合するメソッドのバイトコードの先頭に挿入する。コード・スニペットが実行されたときのスタックの最後の文字列値は，Diagnostics コンソールのメソッドの引数フィールドに表示され，layer 引数の集計パラメータとしても使用できる。 code-key 引数には，このポイントのために作成したコード・スニペットに対して生成したセキュア・コード・キーを指定する。コード・スニペットついては 308 ページ「コード・スニペットを使用したポイントの定義」を，コード・キーについては 322 ページ「コード・スニペットの安全の確保」をそれぞれ参照。 ➤ **after:code:＜コードキー＞**- キーで指定されたコード・スニペットを，ポイントに適合するメソッドのバイトコードのすべての出口に挿入する。after コード・スニペットでは，実行後にスタック上に値を残さないようにする必要がある。

引数	説明
detail（続き）	➤ **disabled -** バイトコードに挿入されたインストルメンテーションがデータを報告するのを禁止する。インストルメンテーション制御 Web ページを使って無効になったポイントを動的に有効にして，データの報告を開始できる。この Web ページには，次の Profiler URL を使ってアクセスできる。 http:// ＜プローブのインストール・ディレクトリ＞: ＜プローブのポート＞ /inst/layer ➤ **outbound -**［送信呼び出し］画面に表示されるようにメソッドにフラグを設定する。また，このインストルメンテーション・エントリの Diagnostics 引数を解析して，送信要求に関する追加情報を Diagnostics ダッシュボードに表示可能かどうかを特定する。 ➤ **no-correlation -** 相関サポート・テクノロジを使用しない「outbound」ポイントで使用する。 ➤ **method-no-trim -** このポイントによってインストルメントされるメソッドが実行されたときに，レイテンシ・ベースのトリミングを行わないことを示す。 ➤ **method-trim -** このポイントによってインストルメントされるメソッドのすべての呼び出しが「トリミング」される（つまり，報告されない）ことを示す。ただし，対応するコード・スニペットがある場合は，副作用が通常どおり発生する。 ➤ **lifecycle -** オブジェクト・ライフサイクルの監視に関係するものとしてインストルメンテーション・ポイントを識別する。 ➤ **no-layer-recurse -** 呼び出される側が別のレイヤに属する場合を除き，このポイントによってインストルメントされるメソッドから呼び出されたメソッドの記録を禁止する。 ➤ **is-statement -** java.sql.Statement クラスの呼び出しをマークする。 ➤ **is-prepare-statement -** キャプチャする java.sql.Statement オブジェクトを返す呼び出しをマークする。 ➤ **method-cpu-time -** CPU の収集が完全にオフである場合（cpu.timestamp.collection.method = 0）を除き，このメソッドについて，レイテンシに加えて CPU 包含時間を収集する。

引数	説明
detail（続き）	➤ **condition** - 指定された条件を満たさないかぎり，このポイントによるインストルメンテーションを禁止する。条件は静的なものであり，**inst.properties** の details.conditional.properties プロパティ（またはコマンド・ライン）で定義される。 ➤ **when-root-rename** - このポイントによってインストルメントされるメソッドが最初に実行されるメソッドである場合に，常にサーバ要求の名前を変更するようにプローブに指示する。 ➤ **diag** - HP Diagnostics に関係するものとしてポイントをマークする（標準設定）。 ➤ **tv: <キー>** - HP Transaction Vision に関係するものとしてポイントをマークする。 ➤ **no-tv** - HP Transaction Vision と競合するものとしてポイントをマークする。Transaction Vision がアクティブになるように設定されていると，このようなポイントによる Java コード・インストルメンテーションが完全に禁止する。 ➤ **add-field:<アクセス>:<タイプ>:<名前>** - 指定したフィールドをインストルメンテーション対象のクラスに追加する。 ➤ **gen-instrument-trace** - このポイントがインストルメンテーションに使用されるたびに，スレッドのスタック・トレースが stdout に出力される。 ➤ **gen-runtime-trace** - このポイントによってインストルメントされるメソッドが実行されるたびに，スレッドのスタック・トレースが stdout に出力される。 ➤ **trace** - このポイントによってインストルメントされる各メソッドの各入口または出口で，インストルメンテーションに関する詳細情報が probe.log に出力される。 ➤ **sub-point: <キー>** - インストルメンテーション時の追加処理を指定する。キーは，**inst.properties** に存在し，処理に使用されるクラス名を識別するものである必要がある。 ➤ **store-thread** - 対応するコード・スニペットで使用されるすべての特殊フィールドが，スレッド固有のデータ構造に格納される。 ➤ **store-fragment** - 対応するコード・スニペットで使用されるすべての特殊フィールドが，現在のサーバ要求の属性として格納される。

引数	説明
detail（続き）	➤ **store-method** - 対応するコード・スニペットで使用されるすべての特殊フィールドが, このポイントによってインストルメントされるメソッドの呼び出しの属性として格納される。 ➤ **ws-operation** - インストルメンテーション・エントリが受信 Web サービス呼び出しのものであることを指定する。また，このインストルメンテーション・エントリの Diagnostics 引数を解析して，Web サービス要求に関する追加情報を Diagnostics ダッシュボードに表示可能かどうかを特定する。
rootRenameTo	**when-root-rename** detail の情報が有効な場合に, サーバ要求を識別する。
layerType	一部のインストルメンテーション対象メソッドに対する特殊な処理方法を指定する。次の 3 つの値が有効。 ➤ **method** - 特殊な処理を行わない（標準設定）。 ➤ **trended_method** - トレンド・メソッド・ビューに表示するメソッドを特定する。 ➤ **Portlet** - ポータル・コンポーネント・ビューで使用するポートレット・ライフサイクル・メソッドを特定する。これらは HP Diagnostics によって設定され，変更できません。 ➤ **sql** - SQL ビューで SQL のキャプチャに使用するメソッドを特定する。これらは HP Diagnostics によって設定され，変更できません。
priority	特定のメソッドに適用できるインストルメンテーション・ポイントが複数存在するときや，Diagnostics Agent が自動的に競合を解決できないときは, ポイントの **priority** によって使用するポイントが決定される。priority の高いポイントが優先する。標準設定の値はゼロ。
active	ポイントをアクティブ化または非アクティブ化します。true に設定すると, ポイントはアクティブになる。false に設定した場合，ポイントは非アクティブになり，プローブに無視される。

コード・スニペットを使用したポイントの定義

カスタム・コード引数は，ポイントのバイトコードに挿入するコードのスニペットを指定します。ポイントのコード・スニペットは，**args:n** 引数で指定したとおりにオブジェクトの **toString()** メソッドを呼び出して返された値が Diagnostics コンソールに有益な情報を提供しないとき，またはインストルメントしたメソッドの複数の引数を表示する必要があるときに使われます。

ポイントのコード・スニペットは，ポイントの detail 引数でキーワード **before:code: ＜コードキー＞**または **after:code: ＜コードキー＞**を使って宣言されます。before と after は，インストルメントされるメソッドの前または後でコード・スニペットを実行するために使用されます。通常，コード・スニペットは，code-key 引数を使ってコード・スニペットに未承認の変更が加えられないようにすることで安全が保たれます。code-key 引数の値は，実行中のプローブの code-key ジェネレータ・ページを使って生成することができ，任意の Java Agent インストールで有効になります。code-key の詳細については，322 ページ「コード・スニペットの安全の確保」を参照してください。

ポイントの実際のコード・スニペットは，**＜プローブのインストール・ディレクトリ＞ /etc/code/custom_code.properties** ファイルに入力されます。これらのスニペットは, code-key の値を使ってキャプチャ・ポイント・ファイルのポイントに関連付けられます。コード・スニペットは，OGNL に似た構文を使用する pseudo Java コードを使って作成されます。コード・スニペットを使うと，インストルメントしたメソッドでアクセス可能なメソッドを，インストルメントしたバイトコードから呼び出すことができます。コード・スニペットに返されるオブジェクトはキャストでき，それらのメソッドも実行できます。コード・スニペットの末尾には文字列またはオブジェクトを配置する必要があり，バイトコードに解析するステートメントのスタックに **toString()** を残すことができます。コード・スニペットの最後の文字列は，返される引数値で使用され，Diagnostics コンソールに表示されます。

コード・スニペットを使って，特定のフラグメントの値を直接格納したり，その後のコード・スニペットで使用できる値を格納したりすることもできます。これらの機能は，特殊フィールドと **store-fragment** や **store-thread** などのキーワード detail を介して使用できます。

注 : コード・スニペットは非常に強力なツールであり，プローブで生じるオーバーヘッドに影響を与える可能性があるため，使用には注意が必要です。そのため，Diagnostics では，インストルメンテーション時にプローブでコード・スニペットを使用する前に，code-key をコード・スニペットと一緒に指定する必要があります。

本項の内容

コード・スニペットの使用

<プローブのインストール・ディレクトリ> /etc/auto_detect.points のポイントを指定するときに，コード・スニペットを使用する場合は，次の情報を使用します。

```
class      = javax.jms.TopicPublisher
method     = publish
signature  = !¥(Ljavax/jms/Topic.*
deep_mode  = soft
layer      = Messaging/JMS/Producer
detail     = outbound,no-correlation,before:code:6d0f3088
```

detail 引数の before:code エントリは，コード・スニペットがポイントに入力されたことを示します。code-key 値は，コード・スニペットのコードを保護し，ポイントと実際のコード・スニペットを関連付けます。

ポイントに関連付けられたコード・スニペットは，次の例のように，**<プローブのインストール・ディレクトリ> /etc/code/custom_code.properties** に入力する必要があります。

```
# Used by [JMS-TopicPublisher2]
6d0f3088 = #topic =
@ProbeCodeSnippetHelper@.checkForTempName(#arg1.getTopicName()); ¥
&drq;DIAG_ARG:type=jms&name=topic:&drq;+ #topic + &drq;&target=topic://&drq; +
#topic;
```

コード・スニペットは，code-key の値を使ってキャプチャ・ポイント・ファイルのポイントに関連付けられます。

コード・スニペットの構文

ここでは，コード・スニペットの作成に使用する必要がある構文について説明します。

➤ リテラル

コード・スニペットでは，次のリテラル・タイプだけがサポートされています。

リテラル・タイプ	構文例
文字列	" 文字列 "
ブール値	true，false
整数	42
null 定数	null
型，値のない定数	void

➤ 文字列の連結

コード・スニペットでは，基本的な文字列の連結がサポートされています。

連結タイプ	構文例
2 つの文字列	" 文字列 " + " もう 1 つの文字列 "
文字列とリテラル	" 文字列 " + 42

➤ ローカル・メンバ

デフォルトのローカル・メンバは，インストルメントされたメソッドに渡された現在のインスタンスまたはオブジェクトをコード・スニペットから参照する方法を提供します。これらのローカル・メンバは，メソッドを呼び出したり，それらの参照から値を取得したりします。

変数	用途
#callee	インスタンス・メソッドの呼び出される側のオブジェクトを参照します。Java の「this」参照に相当します。静的メソッドを参照する場合は使用しないでください。
#arg1, #arg2, ..., #argN	呼び出される側のメソッド呼び出しの引数を参照します。
#return	後のコード・スニペットのためにメソッドの最後の戻り値を参照します。
#classloader	HP ソフトウェア内部使用のために予約されています。

注：一部のインストルメンテーション・ポイントでは，特別な変数参照をサポートしています。たとえば，**CLApplicationDiscoveryPoint** では #classloader 変数をサポートしています。

➤ DIAG_ARG 文字列

コード・スニペットでは，一連の値を連結して 1 つの DIAG_ARG 値を作成できます。この値を使って，1 つの DIAG_ ARG 形式の文字列で特定のタイプのデータをすべて返すことにより，Web サービスや JMS など，いくつかの一般的なタイプのサポート・データをインストルメンテーションできます。

タイプ	フィールド（必須）	定義
ws	&ws_name	Web サービスの名前
	&ws_op	Web サービスの操作名
	&ws_ns	Web サービスの名前空間
	&ws_port（受信のみ）	Web サービスのポート名
	&target（発信のみ）	発信 Web サービスのターゲット
jms	&name	キューまたはトピックの名前
	&target	ターゲットのキューまたはトピックの名前

DIAG_ARG 文字列の形式では，タイプ・フィールドと値（ローカル変数）を次のように 1 つの文字列に連結します。

```
&drq;DIAG_ARG:type=ws&ws_name=&drq;+ #servicename +&drq;&ws_op=&drq;+
#operation +¥ &drq;&ws_ns=&drq;+ #ns +&drq;&ws_port=&drq;+ #port;
```

DIAG_ARG 文字列は，Web サービスの受信データ用の store-fragment 特殊フィールド（##WS_inbound_* で始まる特殊フィールド）と組み合わせて使用しないでください。Web サービス受信データを収集する場合は，単独で使用します。

➤ 特殊フィールド（store-fragment）

標準設定の特殊フィールドを使用すると，共通イベントのフラグメント関連データをコード・スニペットから簡単に渡すことができます。このメカニズムは既存のイベントを補うものであり，それに置き換わるものではありません。フラグメントのローカル・ストレージには，カスタム・イベントより多くのオーバーヘッド・コストがかかります。これらの変数は，**store-fragment** detail の設定とともに使用する必要があります。

変数	用途
##WS_consumer_id	特定のフラグメントのコンシューマ ID を格納する。
##WS_SOAP_fault_code	SOAP 障害コードを格納する。
##WS_SOAP_fault_reason	SOAP 障害の理由を格納する。
##WS_SOAP_fault_detail	SOAP 障害の詳細を格納する。
##WS_inbound_service_name	受信 Web サービスの名前を格納する。
##WS_inbound_operation_name	受信 Web サービスの操作名を格納する。
##WS_inbound_target_namespace	受信 Web サービスのターゲット名前空間を格納する。
##WS_inbound_port_name	受信 Web サービスのポート名を格納する。

➤ 特殊フィールド（store-thread）

追加の特殊フィールドを使用すると，スレッドの存続期間中に関連データをコード・スニペットで簡単に格納できます。これらのスレッド・ローカル・ストレージ変数には関連するオーバーヘッドがあるため，使用時には注意が必要です。これらの変数は，必ず store-thread detail の設定とともに使用してください。

これらの変数は，後のコード・スニペットで取得できます。そのためには，getThreadContextValue（文字列値）メソッドまたは getAndRemoveThread ContextValue（文字列値）メソッドを使ってプローブの ThreadContextProxy クラス参照を呼び出します。文字列値には，変数名から先頭の ## 記号を除いたものを指定します。値を最後に取得するときは，値をメモリから消去したり，次のスレッドのために値を削除したりするために，必ず getAndRemoveThread ContextValue（文字列値）を呼び出してください。

変数	用途
##SOAPHandler_wsname	後で SOAP ハンドラが使用する Web サービスの名前を格納する。
## ＜任意の文字列＞	後続のコード・スニペットで取得できるように，値を格納する。

➤ クラス参照とスタティック・メンバ

次の例のように，クラスの前に「スタティック」であることを示す @ 記号を付けたり，アクセスされているメソッドに @ 記号をマークしたりすることによって，スタティック・メンバ / メソッドにアクセスできます。

```
@java.lang.System@.out (&drq;Hello World&drq;);
```

```
@com.mercury.diagnostics.capture.metrics.countingCollector@.incrementCounter();
```

コード・スニペットの引数は，Java クラスがパーサが理解できるマーカで囲まれているときに Java クラス構文をサポートします。次の例は，@ 記号をマーカとして使用する方法を示します。

```
@java.lang.System@
```

```
@java.lang.System@out (Static field)
```

コード・スニペット・ヘルパー

一部の機能は，コード・スニペットで使用できる限定的な構文を使ってコーディングすることが非常に難しく，場合によっては不可能です。

このため，コード・スニペット環境には，ProbeCodeSnippetHelper と ProbeCodeSnippetHelperV5 という 2 つのヘルパー・クラスが用意されています。CodeSnippetHelperV5 では, Java 5 以降でのみ使用可能な API がいくつか使用されます。

次に，ProbeCodeSnippetHelper の機能を示します。

```
/*
 * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
 */

package com.mercury.opal.capture.proxy;

/**
 * Used to help out Code Snippets
 */
public class ProbeCodeSnippetHelper {

  /**
   * When a Special Field holds a reference to the string below,
   * it will not be stored in the Fragment Local Storage,
   * or Invocation Local Storage
   */
  public static final String DO_NOT_STORE = ...

  /**
   * Helper to convert an int to an Integer
   * @param i
   * @return a new Integer object having the value of i
   */
  public static Object intToInteger(int i) {
    ...
  }

  /*
   * Mark the current thread, if not marked yet
   * @return true, if and only if the thread had been already marked
   */
  public static boolean testAndSetRecursiveFlag() {
    ...
  }

  /*
   * Unmark the current thread
   */
  public static void clearRecursiveFlag() {
    ...
  }

  /**
   * Helper method to call ResourceBundle.getString() and catch any exceptions that
   *  might be thrown
   * @param theBundle the ResourceBundle on which to call getString
   * @param key the key to pass getString
   * @return the value returned from getString, or null if an exception occurred
   */
  public static String getStringFromResourceBundle(ResourceBundle theBundle, String key) {
    ...
  }
```

```
    /*
   * Helper methods to allow our cross-vm coloring to piggyback ride across
   * the custom outbound calls in which the application passes [only] a String.
   * The actual transport technology is irrelevant.
   * Instead of sending the original message, a composite message ("envelope")
   * will be passed. The composite message includes both the original message
   * and Diagnostics Probe ENCODED cross-vm coloring.
   * On the receiving end, the composite message will be received, but only
   * the original message will be passed to the application, and the coloring
   * will be retained by the probe.
   */

  /**
   * Create a composite message, given the coloring and the original message.
   * @param coloring - the correlation String obtained via the ENCODED coloring,
   * may be null
   * @param originalMessage - the original messsage sent by the application
   * @return - the composite message, null if and only if the originalMessage is null
   */
  public static String createDiagEnvelope(String coloring, String originalMessage) {
     ...
  }


  /**
   * Extract the coloring from the composite message (envelope).
   * @param envelope - the composite message or the original message
   * @return the coloring as created on the sender side, or null if not present
   */
  public static String extractColoringFromDiagEnvelope(String envelope) {
     ...
  }


  /**
   * Extract the original message from the composite message (envelope).
   * Works properly, even if the sender side has not been instrumented, and
   * there's no envelope.
   * @param envelope - the composite message or the original message
   * @return the original message (before coloring)
   */
  public static String extractOriginalMessageFromDiagEnvelope(String envelope) {
      ...
  }


}
```

次に，ProbeCodeSnippetHelperV5 の機能を示します。

```
/*
 * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
 */

package com.mercury.opal.capture.jdk15.agent;

/**
 * Used to help out Code Snippets using Java 5 or later
 */
public class ProbeCodeSnippetHelperV5 {

  /**
   * Get the annotation of the specified type from the class or its superclass,
   * or its implemented interfaces
   * @param theClass  The class to get the annotation for
   * @param annClass  The annotation class to look for
   * @return
   */
  public static Object getEndpointClassAnnotation(Class theClass, Class annClass) {
    ...
  }

  /**
   * Get the method annotation of the specified type from the class
   * or its superclass, or its implemented interfaces
   * @param theClass  the class
   * @param methodName  the method name
   * @param argCount  the argument count
   * @param annClass  the class annotation type
   * @param methodAnnClass  the method annotation type
   * @return
   */
  public static Object getEndpointMethodAnnotation(Class theClass, String methodName,
      String argCount, Class annClass, Class methodAnnClass) {
    ...
  }

  /**
   * Helper method to get an annotation element value. If the annotation
   * does not have the element, return null.
   * @param annClass   The class of the annotation
   * @param instance   The annotation instance object
   * @param elementName  The element name
   * @return  The element value for the annotation instance, or null
   */
  public static String getAnnotationElementValue(Class annClass, Object instance, String elementName) {
    ...
  }

  /**
   * This helper method is used to serialize a DOM document.
   * This method uses APIs available in DOM Level 3 or newer, which are
   * available with a 1.5 or later JVM.
   * @param document
   * @return The serialized form (XML) of the input DOM document
   */
  public static String serializeDOMToString(Document document) {
    ...
  }

}
```

➤ メソッド呼び出しのスタックを使って複数の行をつなぐ

コード・スニペットのメソッド呼び出しのスタックは，複数の行をつなぐことができます。バイトコードを構築するパーサは，ステートメントのスタックの解析を続ける必要がある場合，各キャリッジ・リターンの前に「¥」（円マーク）を必要とします。ステートメントのコード・スニペット・スタックの最後の行は，「¥」（円マーク）を含めずに，キャリッジ・リターンだけで終わる必要があります。

```
@java.lang.System@.out (&drq;Hello World&drq;);¥
&drq;Callee Name=&drq;+#callee.getName().toString();
```

➤ キャスト

オブジェクトを返すメソッドを呼び出す際，通常，返されるオブジェクトでメンバを呼び出すのにキャストが必要です。キャストは，オブジェクト参照でサポートされています。オブジェクトをほかの型にキャストするには，そのオブジェクトの参照に続けて，「<」記号と「>」記号の間にキャスト参照を置きます。以下は，キャストの例です。

```
#arg1<com.myCompany.myFoo>.myMethod();

This is equivalent to the Java statement:

((com.myCompany.myFoo)arg1).myMethod();
```

```
@some.class.Foo@foo<com.myCompany.myFoo>.myMethod();

Would be equivalent to the java statement:

((com.MyCompany.myFoo)some.class.Foo.foo).doSomething();
```

```
#foo = #arg1<bar>.b(); #foo.toString();

Creates the following java equivalent:

String foo = ((Bar)arg1).b(); ((Object)foo).toString();
```

注 : キャストは，#classloader などの特殊な型でサポートされていません。

➤ メソッド呼び出し

メソッド呼び出しは，スニペットの引数に含まれます。メソッド呼び出しのサポートには，引数およびメソッドのチェーンを使用する呼び出しと，使用しない呼び出しが含まれます。次に，コード・スニペットの引数に含まれているメソッド呼び出しの例を示します。

```
#arg1.toString()
```

```
#arg2.getSomething().getSomethingElse()
```

```
#callee.getSomething(&drq;foo&drq;, #arg1).somethingElse()
```

```
@some.Class@.staticMethod()
```

正しく解析するために，メソッド呼び出しのスタティック参照の後にドットが必要です。

```
@java.lang.System@out.println(&drq;Here I am!&drq;)
```

ランタイム時にバイトコードの生成を高速化するために（リフレクションを回避），次の例のように，メソッドから返される型を指定できます。

```
#arg1.getSomething()<some.class.Here>
```

これは，メソッドが引数を取る場合，またはスタティック・フィールドが使われる場合は有効でありません。

➤ 複数のステートメント

コード・スニペットでは，1 つのコード・スニペットに複数のステートメントを含めることができます。これは，複数のオブジェクトがスタックに残ることが予想される **CLApplicationDiscoveryPoint** などのインストルメンテーションで必要です。ほかの状況でも便利です。

```
@java.lang.System@out.println(&drq;Look out!&drq;);
#arg2.getSomething();
```

➤ ローカル・メンバの割り当て

サポートされている標準設定の「ローカル」変数のほかに，独自のローカル・メンバを作成して，呼び出したメソッドによって返されるオブジェクト参照を保持できます。

新しいローカル・メンバを作成するには，ローカル・メンバ名の前に「#」記号を入力します。パーサによって，ユーザのローカル・メンバが作成されます。

```
#myBar = #arg2.getName();¥
#myUpperBar = #myBar.toUpper();¥
&drq;Target Name=http://&drq;+myUpperBar+&drq;/services&drq;;
```

➤ 特殊フィールドの割り当て（store-fragment）

事前定義された特殊フィールドを使って，呼び出されたメソッドから返されたオブジェクト参照を格納できます。インストルメンテーション・ポイントに **store-fragment** detail キーワードを入力するとともに，特殊フィールドの名前の前に「##」記号を入力します。

```
##WS_SOAP_fault_code = #arg2;¥
##WS_SOAP_fault_reason = #arg3;¥
##WS_SOAP_fault_detail = (#arg4 == null ? null : #arg4.toString());"";
```

➤ 特殊フィールドの割り当て（store-thread）

特殊フィールドを使って，呼び出されたメソッドから返されたオブジェクト参照を格納できます。インストルメンテーション・ポイントに **store-thread** detail キーワードを入力するとともに，特殊フィールドの名前の前に「##」記号を入力します。

```
# Used by [SOA_Broker_Payload_Handler]
##SOA_Manager_Inbound_Payload=#callee.getRequestDocument();&drq;&drq;;
```

格納された値を後のコード・スニペットで取得するには，特殊フィールドの値から先頭の ## 記号を除いたものを指定して getThreadContextValue を呼び出します。

```
#temp_soam_payload=@com.mercury.opal.capture.proxy.ThreadContextProxy@.get
ThreadContextValue(&drq;SOA_Manager_Inbound_Payload&drq;);
```

格納された値を後のコード・スニペットで取得してから削除するには，同じ値から先頭の ## 記号を除いたものを指定して getAndRemoveThreadContextValue メソッドを呼び出します。getAndRemoveThreadContextValue を呼び出してメモリを解放し，次の出現に備えることが非常に重要です。

```
#temp_soam_payload=@com.mercury.opal.capture.proxy.ThreadContextProxy@.
getAndRemoveThreadContextValue( (&drq;SOA_Manager_Inbound_Payload&drq;);
```

- **条件ロジック**

コード・スニペット構文では，Java の if-else ステートメントに相当する制限付きの条件ロジックを使用できます。この構文を使うと，== 演算子と != 演算子の両方を使って，同じ型のオブジェクト参照，または整数やブールのプリミティブを比較できます。この構文を使って，リテラル値とほかのプリミティブの比較はできません。

以下は，参照の比較方法の例です。

```
(value1 == value2 ? <if_True_codeSnippet>:<if_False_codeSnippet>)
```

以下は，メソッドを呼び出す前にオブジェクトが null でないことを確認する方法の例です。

```
(#arg1 == null ? "Unknown" : #arg1.getSomething())
```

これは，次の Java ステートメントに相当します。

```
if (arg1==null) return "Unknown" else return arg1.getSomething();
```

- 例外処理

例外を処理する場合は，次の構文のような限定的な形式が使用されます。

```
!{ <code-snippet-code>}!
```

指定したコードが実行され，上記の式の値が例外としてスローされます。コードの実行中に例外がスローされなかった場合は，null になります。

コード・スニペットの安全の確保

標準設定では，インストルメント中にプローブがコード・スニペットを使う前に，有効なコードキーとコード・スニペットを指定する必要があります。コードキーを要求することで，誤ってインストルメンテーションを実行し，プローブのオーバーヘッドが著しく高くなるのを防ぎます。

コードキーを生成する際，Diagnostics では，キーを生成する前にコード・スニペットの構文が有効なことを確認します。Diagnostics は，アプリケーションをインストルメントする際，code-key 引数に入力された値がポイントのコード・スニペットに対して計算されたコードキーと一致することを確認します。コードキーが一致しない場合，Diagnostics は，コード・スニペットを無視して，インストルメンテーション・ポイントを作成しません。

コード・スニペットのコードキーの生成

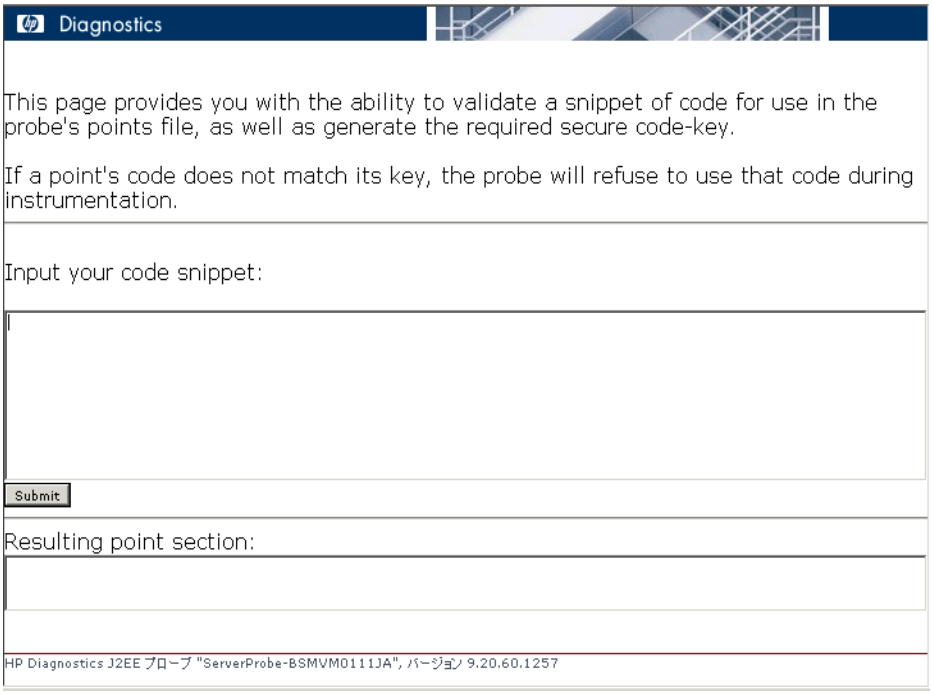
Java Agent には，ユーザが入力したコード・スニペットからコードキーを生成するツールがインストールされています。

コードキーを生成するには，次の手順を実行します。

1 ご使用のブラウザで，次の URL にあるページを開きます。

http:// ＜プローブのホスト＞ : ＜プローブのポート＞ /inst/code-key

Diagnostics に，次の例のようなコード・スニペット構文を検証してコードキーを生成できるページが表示されます。



2 ［**Input your code snippet**］テキスト・ボックスに，**auto_detect.points** ファイルの code 引数に指定したコード・スニペットを入力して，［**Submit**］をクリックします。

注 : コード・スニペットには，code = 引数名の後のテキストがすべて含まれます。

---

3 Diagnostics の［**Resulting point section**］テキスト・ボックスに，コード・スニペットの検証結果とコードキーの生成が表示されます。

コード・スニペットが有効な場合，Diagnostics には，コードキーとコード引数の両方の値が表示されます。キャプチャ・ポイント・ファイルにこれらの値を入力します。

コード・スニペットが有効でない場合，Diagnostics には，検出された問題を伝えるエラー・メッセージが表示されます。問題を訂正し，もう一度［**Submit**］をクリックして訂正したコードを検証します。

コードキーのセキュリティ・チェックを無効にする

標準設定では，Diagnostics は，code-key 引数の値が，アプリケーション・インストルメント時に生成した値と一致するかどうかを検証します。**<プローブのインストール・ディレクトリ> /etc/inst.properties** ファイルに **require.code.security.key** プロパティを挿入し，値 **false** を設定することにより，このセキュリティ・チェックを無効にできます。

注 : このプロパティを使うときは十分に注意してください。このセキュリティ・チェックを無効にすると，予期しないプロセス・オーバーヘッドとパフォーマンス監視結果が生じることがあります。

クラス・マップ・キャプチャの制御

クラス・マップによって，Diagnostics は，サーバ要求に呼び出されたクラスとメソッドに関する詳細情報を提供できます。この情報は，パフォーマンス問題のソースの検索対象を絞りこんで，アプリケーションを適切にインストルメントする場合に役立ちます。クラス・マップを使用すると，マップがエージェントのホスト・システム上に作成されるために余計なオーバーヘッドが生じます。

標準設定では，**probe.properties** ファイルに **use.class.map=false** プロパティが設定されます。この設定を **true** に変更すると，クラス・マップが有効になります。

インストルメンテーションの例

このセクションの例では，キャプチャ・ポイント・ファイル内にポイントを作成したり，変更したりして，アプリケーションのインストルメンテーションをカスタマイズする方法を示します。

本項の内容

- カスタム・レイヤおよびサブレイヤ
- ワイルドカード・メソッド
- 特定のメソッドの無視
- トレンド・メソッド・ビューのキャプチャ・メソッド
- クラスの特定のメソッドだけのキャプチャ
- 文字列を返す特定のメソッドのキャプチャ
- 制御された範囲でのキャプチャ
- hard および soft deep_mode
- 引数のキャプチャ
- 受信および発信 Web サービス
- ルート・メソッド名の変更
- クラスへのフィールドの追加
- インスタンス・ツリーへの属性の引き渡し
- アクセス・フラグによるメソッドのトリミング
- 直接再帰の記録禁止
- 呼び出し側のインストルメンテーションの実行
- 割り当て分析の設定
- ライトウェイトなメモリ診断（LWMD）の設定
- コレクション・リークの指摘の設定
- JDBC 結果セットのオブジェクト・ライフサイクルの監視の有効化
- 無効化されたポイントの追加と実行時の有効化
- メソッドがトリミングされないようにする指定

- メソッドが常にトリミングされるようにする指定
- メソッドの CPU 時間の収集の有効化
- SAP JCO ライブラリのバージョンに基づいた SAP RFC のインストルメンテーションの変更
- ポイントのインストルメンテーションおよび実行時情報の出力（デバッグ専用）

カスタム・レイヤおよびサブレイヤ

- 次のポイントは，myCompany.myFoo クラスのメソッド myMethod に対して，「FOO」というレイヤの中に「BAR」というカスタム・サブレイヤを作成します。

```
[myCompany.myFoo_customLayer]
class = myCompany.myFoo
method = myMethod
signature = !.*
layer = FOO/BAR
```

ワイルドカード・メソッド

- 次のポイントは，MyCompany.MyFoo クラスのすべてのメソッドをキャプチャします。

```
[myCompany.myFoo_AllMethods]
class = myCompany.myFoo
method = !.*
signature = !.*
layer = FOO/BAR
```

特定のメソッドの無視

- 次のポイントは，setHomeInterface メソッドと getHomeInterface メソッドを除いて，MyCompany.MyFoo クラスのすべてのメソッドをキャプチャします。

```
[myCompany.myFoo_AllMethodsExcept]
class = myCompany.myFoo
method = !.*
ignoreMethod = !setHomeInterface.*, !getHomeInterface.*
signature = !.*
layer = FOO/BAR
```

➤ 次のポイントは，MyCompany.logging クラスに含まれるメソッドを除いて，MyCompany パッケージ / 名前空間のメソッドをすべてキャプチャします。

```
[myCompany_All_Methods_except_from_MyCompany_Logging]
class = !myCompany¥..*
method = !.*
ignore_cl = MyCompany.logging
signature = !.*
layer = FOO/BAR
```

トレンド・メソッド・ビューのキャプチャ・メソッド

➤ 次のポイントは，必要なデータをキャプチャして myMethod メソッドのトレンド・メソッド・ビューに入力します。

```
[myCompany.myFoo_customLayer]
class = myCompany.myFoo
method = myMethod
signature = !.*
layer = FOO/BAR
layertype = trended_method
```

クラスの特定のメソッドだけのキャプチャ

➤ 次のポイントは，MyCompany.MyFoo クラスのコンストラクタのすべてのメソッドをキャプチャします。

```
[myCompany.myFoo_Constructor]
class = myCompany.myFoo
method = <init>
signature = !.*
layer = FOO/BAR
```

➤ 次のポイントは，MyCompany.MyFoo クラスの 1 つのコンストラクタのすべてのメソッドをキャプチャします。

```
[myCompany.myFoo_Singleton]
class = myCompany.myFoo
method = <clinit>
signature = !.*
layer = FOO/BAR
```

➤ 次のポイントは，MyCompany.MyFoo クラスの setFoo メソッドをキャプチャします。

```
[myCompany.myFoo_setFoo]
class = myCompany.myFoo
method = setFoo
signature = !.*
layer = FOO/BAR
```

➤ 次のポイントは，MyCompany.MyFoo クラスのすべての「set」メソッドをキャプチャします。

```
[myCompany.myFoo_AllSets]
class = myCompany.myFoo
method = !set.*
signature = !.*
layer = FOO/BAR
```

➤ 次のポイントは, MyCompany パッケージ/名前空間のすべてのメソッドをキャプチャします。

```
[myCompany_All_Methods]
class = !myCompany¥..*
method = !.*
signature = !.*
layer = FOO/BAR
```

文字列を返す特定のメソッドのキャプチャ

➤ 次のポイントは，MyCompany.MyFoo クラスの java.lang.String を返す，引数を持たない getFoo メソッドをキャプチャします。

```
[myCompany.myFoo_GetFoo_String]
class = myCompany.myFoo
method = getFoo
signature = ()Ljava¥lang¥String
layer = FOO/BAR
```

制御された範囲でのキャプチャ

- 次のポイントは，MyCompany.logging クラスから呼び出される MyCompany パッケージ / 名前空間のメソッドをすべてキャプチャします。詳細については，343 ページ「呼び出し側のインストルメンテーションの使用」を参照してください。

```
[myCompany_All_Methods_from_MyCompany_Logging]
class = !myCompany¥..*
method = !.*
signature = !.*
scope = MyCompany.logging
layer = FOO/BAR
```

- ignoreScope 引数を使って，scope 引数に指定されたスコープに含まれる特定のパッケージ，クラスおよびメソッドを除外します。次のポイントは，myMethod メソッドから呼び出されるものを除いて，MyCompany.logging クラスから呼び出される MyCompany パッケージ/名前空間のメソッドをすべてキャプチャします。詳細については，343 ページ「呼び出し側のインストルメンテーションの使用」を参照してください。

```
[myCompany_All_Methods_except_from_MyCompany_Logging]
class = !myCompany¥..*
method = !.*
signature = !.*
scope = MyCompany.logging
ignoreScope = MyCompany.logging¥myMethod
layer = FOO/BAR
```

hard および soft deep_mode

- 次のインタフェース定義は，soft および hard 両方の deep_mode の例で使われます。

```
public interface Interface1 {

    public void callerMethod();

}
```

➤ 次のクラスは，soft および hard 両方の deep_mode の例で使われます。

```
public class Class1 implements Interface1 {
   public void callerMethod(){
      calleeMethod();
      calleeMethod2();
   }

   public void calleeMethod(){
      System.out.println("hello world");
  //more code lines here…
   }

   public void calleeMethod2(){
      System.out.println("hello world 2");
   }
}
```

➤ 次のポイントは，Class1 クラス内の「callerMethod」をキャプチャします。

```
[Training-1]
class      = Interface1
method = !.*
signature = !.*
deep_mode  = soft
layer      = Training
```

➤ 次のポイントは，Class 1 のすべてのメソッド（「callerMethod」，「calleeMethod1」，「calleeMethod2」など）をキャプチャします。

```
[Training-1]
class      = Interface1
method = !.*
signature = !.*
deep_mode  = hard
layer      = Training
```

引数のキャプチャ

Diagnostics に表示される引数は，コード・スニペットがスタック上に残した最後の文字列です。コード・スニペットは，バイトコードに解析されるステートメントのスタック上に toString() を残すことができる文字列またはオブジェクトで終了する必要があります。

重要 : 引数をキャプチャする場合は，十分に注意する必要があります。キャプチャした引数のすべての有効値からなるセットが有限でない場合，エージェントは Java ヒープ領域不足になります。

➤ メソッド myCompany.myFoo.myMethod() をインストルメントしようとしており，myFoo には文字列を返す getComponentName() という別のメソッドがあるとします。次の例では，Diagnostics の引数として getComponentName() の結果が表示されます (#callee は，ここではインスタンス・メソッドの呼び出される側のオブジェクトを参照します)。

```
[myCompany_componentName_as_argument]
class = myCompany.myFoo
method = myMethod
signature = !.*
detail = before:code: 8d2509eb
layer = FOO/BAR
```

custom_code.properties ファイルのコード・スニペットは次のように入力されます。

```
8d2509eb = #callee.getComponentName()
```

➤ 次のポイントは，myMethod の最初の引数をキャプチャし，それを Diagnostics でキャプチャされた引数として表示します。この引数は，サブレレイヤ名としても使用されます。これは，レイヤに ${ARG} を含めることによって行われます。この例では，キャプチャされた引数（ここでは myMethod の最初の引数）に myArg という値がある場合，レイヤは FOO/myArg になります。

```
[myCompany_capture_firstArg_and_also_show_as_layer]
class = myCompany.myFoo
method = myMethod
signature = !.*
detail = before:code: 358f05d6
layer = FOO/${ARG}
```

custom_code.properties ファイルのコード・スニペットは次のように入力されます。#arg2 を使用すると，2 番目の引数が代わりにキャプチャされます。

```
358f05d6 = #arg1.toString()
```

受信および発信 Web サービス

ポイントの detail 引数に「outbound」または「ws-operation」キーワードが含まれる場合，Diagnostics は，メソッド呼び出しについて表示する追加情報のコード・スニペット・スタックの最後の文字列を解析しようとします。

➤ 受信 Web サービス（「ws-operation」detail を使用する必要がある）の場合，文字列は次のようになります。

```
"DIAG_ARG:type=ws&ws_name=” + < Web サービス名> +"&ws_op=” +
<操作名> +”&ws_ns=” +<ターゲット名前空間> +”&wsOport=” + < WS ポート>
```

➤ 発信 Web サービス（「outbound」detail を使用する必要がある）の場合，文字列は次のようになります。

```
"DIAG_ARG:type=ws&ws_name=” + < Web サービス名> +"&ws_op=” +
<操作名> +”&target=” + <ターゲット名>
```

次に例を示します。

```
class      = weblogic.wsee.ws.WsStub
method = invoke
signature  = (Ljava/lang/String;Ljava/lang/String;Ljava/util/Map;Ljava/util/Map;)Ljava/
lang/Object;
layer = Web Services
detail     = outbound,before:code:edd75e36
```

custom_code.properties ファイルのコード・スニペットは次のように入力されます。

```
edd75e36 = #service = #callee.getService().getWsdlService();¥
#qname = #service.getName();¥
"DIAG_ARG:type=ws&ws_name="+ #qname.getLocalPart() +"&ws_op="+ ¥
#callee.getMethod(#arg1).getOperationName().getLocalPart() +"&target="+ ¥
#callee.getProperty("javax.xml.rpc.service.endpoint.address");
```

ルート・メソッド名の変更

➤ 次のポイントについて検討します。

```
class      = Statement
method   = execute
layer      = Database/JDBC/Execute
detail     = when-root-rename
rootRenameTo   = mySuffix
```

このようなメソッドが最終的にルート・メソッドだった場合，そのサーバ要求の名前は Background-mySuffix になり，サーバ要求のタイプは RootRename になります。

➤ 代わりに，次のポイントについて検討します。

```
class      = Statement
method   = execute
layer      = Database/JDBC/Execute
detail     = when-root-rename
```

rootRenameTo プロパティが省略されることに注意してください。このようなサーバ要求の名前は（Database が最初のサブレイヤであるため）Background - Database になり，サーバ要求のタイプは再び RootRename になります。

クラスへのフィールドの追加

➤ 次のポイントについて検討します。

```
class      = com.corp.Foo
method   = bar
detail      = add-field:protected:Object:serviceName
```

上記の detail によって，クラス com.corp.Foo に次の 1 つのフィールドと 2 つの public な設定 / 取得メソッドが追加されます。

```
protected transient Object serviceName
public void _diag_set_serviceName(Object arg)
public Object _diag_get_serviceName()
```

インスタンス・ツリーへの属性の引き渡し

➤ 次の例では，my_attribute の名前を com.corp.Foo.bar() のキャプチャされたすべてのインスタンスに付加します。

呼び出しプロファイルには，display_ というプレフィックスが付いた名前と，対応する値が表示されます。

```
class      = com.corp.Foo
method   = bar
detail      = store-method,code:f59f0c5c
```

コード・スニペット：

```
f59f0c5c = ##my_attribute="value-of-my-attribute";"";
```

アクセス・フラグによるメソッドのトリミング

➤ 次の例では，クラス com.corp.Foo の（静的メソッドを除く）すべてのメソッドをインストルメントします。

```
class      = com.corp.Foo
method = !.*
signature = !.*
method_access_filter = static
```

直接再帰の記録禁止

➤ 次の例では，メソッド com.corp.Foo.bar がそれ自体（または同じレイヤ内の何か）を呼び出した場合，2 回目の呼び出しは記録されません。これは，**detail = no-layer-recurse** によって行われます。

ただし，これは直接再帰のみが対象です。com.corp.Foo.bar が別のレイヤにあるインストルメンテーション対象メソッドを呼び出した後で, このレイヤがこのメソッドを再び呼び出した場合は，すべてのメソッドが記録されます。

```
class     = com.corp.Foo
method   = bar
layer      = Example/MyBar
detail    = no-layer-recurse
```

呼び出し側のインストルメンテーションの実行

➤ 次のポイントでは,（標準設定の呼び出される側のインストルメンテーションではなく）呼び出し側のインストルメンテーションが実行されます。これは，**detail = caller** によって行われます。

呼び出し側でインストルメンテーションを実行するもう 1 つの方法は, 343 ページ「呼び出し側のインストルメンテーションの使用」に記載されているように,「scope」プロパティを使用することです。

```
class     = com.corp.Foo
method   = bar
detail    = caller
```

割り当て分析の設定

次の 2 つの例では，パッケージ com.mycompany.mycomponent 内の java.lang.Integer の割り当てを追跡します。ただし，これらは 2 つの点で異なります。

➤ 最初の例（**detail = leak**）では，追跡が管理されます。Profiler 内でユーザが［**開始**］をクリックすると追跡が開始し，［**停止**］をクリックすると停止します。2 つ目の例（**detail = deallocation**）では, アプリケーションの起動時に追跡を開始します。

➤ 1 つ目の例では, 通常のインストルメンテーションに関してポイントが無効になります。つまり,「新しい Integer」がインスタンス・ツリーに表示されません。2 つ目の例では，表示されます。

例 1 - 管理される場合 : Profiler 内でユーザが［**開始**］をクリックすると追跡が開始し，［**停止**］をクリックすると停止します。

```
[Leak]
scope    = !com¥.mycompany¥.mycomponent¥..*
class     = java.lang.Integer
keyword = allocation
detail   = leak
active    = true
```

例 2 - 管理されない場合 : アプリケーションの起動時に追跡を開始します。

```
[Leak]
scope    = !com¥.mycompany¥.mycomponent¥..*
class     = java.lang.Integer
keyword = allocation
detail   = deallocation
active    = true
```

これらのポイントは，いずれも反映された割り当てをキャプチャしません。反映された割り当てのキャプチャを有効にするには，「reflection」detail をポイントに追加します（つまり，**detail = leak,reflection** とします）。

ライトウェイトなメモリ診断（LWMD）の設定

➤ 次の例では，com.mercury.mycomponent パッケージの内部で発生したコレクションのコレクション診断を有効にします。この例は，**auto_detect.points** ファイルに含まれています。デフォルトでは，active = false に設定されています。

```
[Light-Weight Memory Diagnostics]
scope    = !com¥.mycompany¥.mycomponent¥..*
class     = java.lang.Integer
keyword = lwmd
active    = true
```

dynamic.properties ファイルのプロパティ **lwm.diagnostics.capture=true** も設定する必要があります。詳細については，『HP Diagnostics ユーザー・ガイド』のコレクションとリソースの表示に関するセクションを参照してください。

コレクション・リークの指摘の設定

JRE のバージョンに関係なく，Java Agent でコレクション・リークの指摘（CLP）機能を使用する場合は，アプリケーション・サーバの適切なモードを使用して JRE Instrumenter を実行する必要があります。JRE のインストルメントの詳細については，第 6 章，「Java Agent で監視するためのアプリケー ション・サーバの準備」を参照してください。

標準設定では，**auto_detect.points** ファイルでコレクション・リークの指摘が有効になっています。

```
[Collection Leak Pinpointing]
keyword = clp
```

dynamic.properties ファイル内の次のプロパティを設定すると，コレクション・リークを報告するようにを設定できます。これらと同じ値は，Java Profiler の［構成］タブの UI でも設定できます（502 ページ「コレクション・リーク・レポートの有効化と構成」を参照）。

clp.diagnostics.reporting=true

Diagnostics UI でレポートを有効にします。UI でこの機能のレポートを無効にするには，チェックボックスをオフにします。

clp.diagnostics.growth.time.threshold.flag = 60m

コレクションのサイズが増大する継続時間のしきい値。コレクションのサイズが増大する継続期間がこのしきい値を超える場合，プローブによってメモリ・リークとしてフラグを設定されます。誤検知を回避するには，この値を，アプリケーションがすべてのキャッシュを完全に初期化するための所要期間よりも長くする必要があります。

clp.diagnostics.nongrowth.time.threshold.unflag = 60m

すでにフラグが設定されているリーク中のコレクションで，このしきい値期間について継続的にサイズの増大が停止した場合，プローブはリークとしてのフラグを設定解除します。

JDBC 結果セットのオブジェクト・ライフサイクルの監視の有効化

事前に設定された一部のインストルメンテーション・ポイントでは，オブジェクト・ライフサイクルを監視できますが，標準設定では無効になっています。次に，上記の 2 つの例を示します。

この例は，JDBC 結果セットのオブジェクト・ライフサイクルの監視を有効にする方法を示しています。オブジェクト・ライフサイクルの監視の詳細については，『HP Diagnostics ユーザー・ガイド』のメモリおよびオブジェクト・ライフサイクルのアナリシスに関するセクションの［アロケーション / ライフサイクル分析］タブを参照してください。

この例では，次の 2 つのアクションが必要です。

1 **inst.properties** を参照して details.conditional.properties を検索します。mercury.enable.resourcemonitor.jdbcResultSet=true を設定します。

2 対応する open インストルメンテーション・ポイントの範囲を指定します（次の例を参照）。

次の例では，プローブがパッケージ com.mycompany.mycomponent の内部で JDBC 結果セットのオブジェクト・ライフサイクルを監視します。

```
[Lifecycle-JDBC-ResultSet-Open]
scope    = !com¥.mycompany¥.mycomponent¥..*
class      = java.sql.Statement
method  = !(getResultSet.*)|(executeQuery.*)
signature  = !.*¥)Ljava/sql/.*ResultSet;
detail = condition:mercury.enable.resourcemonitor.jdbcResultSet,lifecycle,caller

[Lifecycle-JDBC-ResultSet-Close]
class      =
!(java¥.sql¥.ResultSet)|(weblogic¥.jdbc¥.wrapper¥.ResultSet)|(com¥.ibm¥.ws¥.rsadapte
r¥.jdbc¥.WSJdbcResultSet)
method  = !(close)|(closeWrapper)
signature = !.*
deep_mode  = soft
detail =
condition:mercury.enable.resourcemonitor.jdbcResultSet,before:code:513a2b36,metho
d-trim
```

無効化されたポイントの追加と実行時の有効化

➤ 次の例では，ポイントが無効になっています。これは，インストルメンテーションが行われないということではありません。インストルメンテーションは発生しますが，データは収集されません。これによって，このようなポイントのランタイム・オーバーヘッドが大幅に減少します。

アプリケーションの実行中にデータ収集を有効にするには，［レイヤ］ページ（http:// ＜プローブのホスト＞ : ＜プローブのポート＞ /inst/layer から，または Profiler から［構成］タブを選択して［View instrumentation］を選択する）にアクセスし，レイヤ **myLayer** を探して［**有効化**］をクリックします。

```
[My Example]
class      = Example
method = !.*
layer   = myLayer
detail    = disabled
```

インストルメンテーションがまったく発生しないように設定する場合は，**active=false** を使用します。ただし，このようなポイントを実行時に有効にすることはできません。

メソッドがトリミングされないようにする指定

➤ 次の例では，レイテンシのトリミングが Example.myMethod() に適用されません。

```
[My Example]
class      = Example
method = myMethod
detail    = method-no-trim
```

メソッドが常にトリミングされるようにする指定

➤ 次の例では，メソッド Example.myMethod() が報告されません。ただし，ポイントに関連付けられたコード・スニペットは常に実行されます。

```
[My Example]
class      = Example
method = myMethod
detail    = method-trim, before:code:...
```

メソッドの CPU 時間の収集の有効化

➤ 次の例では,「method-cpu-time」detail によってメソッド Example.myMethod () の CPU 時間の収集が行われます。

```
[My Example]
class      = Example
method = myMethod
detail    = method-cpu-time
```

SAP JCO ライブラリのバージョンに基づいた SAP RFC のインストルメンテーションの変更

＜プローブのインストール・ディレクトリ＞ /etc/inst.properties ファイルでは，使用する SAP JCO のバージョンに応じて 2 つのポイントが定義されています。使用しないバージョンをコメントアウトします。2.1.10 以降のバージョンでは，com.mercury.opal.capture.inst.SapRfcinstrumentationPoint2_1_10 を使用します。それ以外の場合，標準設定では 2.1.9 以前のバージョンで動作します。

ポイントのインストルメンテーションおよび実行時情報の出力（デバッグ専用）

次の例では，標準出力と probe.log にいくつかのデバッグ情報を出力します。

➤ **gen-instrument-trace** detail を指定すると，このポイントがメソッドのインストルメンテーションに使用されるたびに，スレッドのスタック・トレースが stdout に出力されます。

➤ **gen-runtime-trace** を指定すると，Example.myMethod() が実行されるたびに，スレッドのスタック・トレースが stdout に出力されます。

➤ **trace** detail を指定すると，Example.myMethod() が実行されるたびに，インストルメンテーションに関する詳細情報が probe.log に出力されます。

```
[My Example]
class      = Example
method = myMethod
detail    = gen-instrument-trace, gen-runtime-trace, trace
```

カスタム・インストルメンテーションのオーバーヘッドについて

カスタム・インストルメンテーションを作成する際，調べるアプリケーションのレイテンシが極めて長くなる可能性があるため，アプリケーションの過インストルメンテーションに注意する必要があります。クラスでインストルメンテーションを実行するほど，クラスローダのレイテンシが長くなるため，過度のレイテンシが発生します。バイトコード量が常にほとんど同じであるため，インストルメンテーションのオーバーヘッドはすべてのメソッドでほぼ修正され，カスタム・インストルメンテーションはメソッドのレイテンシや CPU のオーバーヘッドに同様の影響を与えません。したがって，CPU およびレイテンシ
のオーバーヘッドにおける物理的な割合は，メソッドの実行時間に正比例して変化します。

たとえば，メソッドで 100 ms かかり，インストルメンテーションを 101 ms で実行する場合，オーバーヘッドは 1% になります。メソッドで 10 ms かかり，インストルメンテーションの応答が 11 ms に変わると，オーバーヘッドは 10% になります。このメソッドが頻繁に呼び出されない場合，アプリケーションに対する全体のレイテンシは最小限になります。ただし，インストルメンテーション対象のメソッドが頻繁に呼び出されると，そのオーバーヘッドの割合が非常に小さくても，全体のレイテンシがアプリケーションの応答のレイテンシに影響を与えることがあります。

従来のプロファイラとは異なり，HP Diagnostics はバイトコード・インストルメンテーションを採用しています。このため，標準設定のインストルメンテーションを選択することによって，インストルメンテーションによるオーバーヘッドを平均 3 ～ 5% に最小化できます。インストルメンテーションによるレイテンシ・オーバーヘッドが高いメソッドは，アプリケーションのほかのコンポーネントよりも頻繁に呼び出される場合や，優先順位付けに必要な具体的な情報がインストルメンテーションによって提供される場合（JNDI ルックアップなど）のみ，インストルメントされます。

アプリケーションのインストルメンテーションをカスタマイズする際は，Diagnostics データのオーバーヘッドも考慮する必要があります。インストルメントするメソッドが多いほど，プローブがシリアル化し，ネットワークを通じて Diagnostics サーバに渡すデータの量が増えます。監視中のシステムのパフォーマンスに不要な影響を与えないように，Java プローブの標準設定を変更してプローブに Diagnostics のデータ量を調整できます。プローブの設定を不適切にチューニングすると，Java エージェントがインストールされた物理マシンで CPU，メモリ，およびネットワークのオーバーヘッドが生じる可能性があります。レイテンシ，CPU，メモリおよびネットワークのオーバーヘッドについては，第 12 章，「Java Agent およびアプリケーション・サーバの詳細構成」を参照してください。

レイヤ単位のインストルメンテーション制御

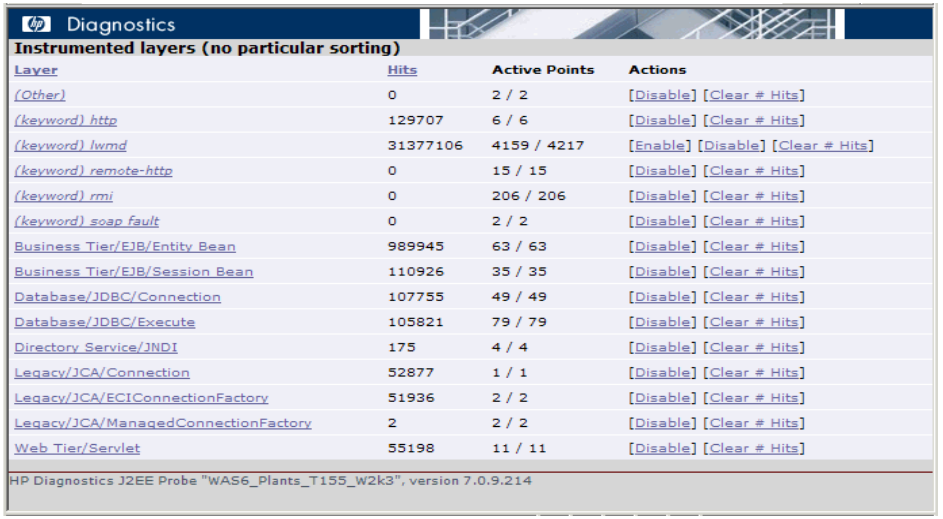
デフォルトで，キャプチャ・ポイント・ファイルで定義されているレイヤは有効になっています。ポイントに details=disabled 引数を含めると，プローブが起動したときにレイヤが無効になります。

JDK 1.5 のクラスマップを使うと，JVM インスタンスを再起動することなく，JVMTI インタフェースを使ってメソッドとクラスを動的にインストルメントできます。ほかのすべての仮想マシンは，キャプチャ・ポイント・ファイルに加えた変更を適用するために，JVM インスタンスの再起動を要求します。

インストルメンテーションをメソッド内に置くと，データ収集と実行中の CPU およびメソッドのレイテンシ・オーバーヘッドをレイヤごとに制御できます（［Instrumented Layers］ページを参照）。

［Instrumented Layers］ページには，次の URL からアクセスできます。

http:// ＜プローブのホスト＞ : ＜プローブのポート＞ /inst/layer



［Instrumented Layers］ページからレイヤを無効にするには，ページのレイヤに関連付けられている［**Disable**］リンクをクリックします。レイヤが無効になり，リンクは［**Enable**］に切り替わります。必要に応じて，レイヤを再度有効にすることができます。

高度なインストルメンテーションの例

本項の内容

呼び出し側のインストルメンテーションの使用

標準設定では, Diagnostics のすべてのインストルメンテーションは呼び出される側のインストルメンテーションであり，バイトコードがメソッド呼び出し内に置かれています。呼び出し側のインストルメンテーションは，中ではなくインストルメントするメソッドへの呼び出しの近くにインストルメントのバイトコードを置くプロセスを参照します。

呼び出し側のインストルメンテーションでは，インストルメンテーションの配置を微調整できますが，スコープで指定された各クラス内で，ポイントで指定されているクラスまたはメソッドの参照があるかどうかをチェックする必要があるため，アプリケーション・クラスローダの所要時間が長くなる可能性があります。

呼び出し側のインストルメンテーションの一般的な使い方では，**rt.jar** のメソッドの呼び出しをインストルメントします。従来のクラスローダからではなく，ブートクラスローダを使って仮想マシンにロードしたクラスは，直接インストルメントできません。これらのメソッドの呼び出しをインストルメンテーションするには，呼び出し側のインストルメンテーションを使用する必要があります。

次の例では, **javax.xml.parsers.SAXParser** および **javax.xml.parsers.DocumentBuilder** の parse メソッドの呼び出しの近くに, 任意のクラスの任意の (**!.***) メソッドを解析するバイトコードを置いて, 2 つのメソッドの呼び出しをインストルメントします。**javax.xml.parsers.SAXParser** および **javax.xml.parsers.DocumentBuilder** クラスの両方が rt.jar に含まれ, ブートクラスローダによって仮想マシンにロードされるため, 呼び出し側のインストルメンテーションが必要になります。

```
[XML-DOM-JDK14]
;---------- Interface --------------
Class = !javax¥.xml¥.parsers¥.(SAXParser|DocumentBuilder)
method     = parse
signature = !.*
scope = !.*
layer   = XML
```

次の例では，com.myCompany.myFoo クラスから呼び出される javax.naming.Context の「lookup」メソッドの呼び出しをインストルメントし，FOO レイヤの JNDI サブレイヤに置きます。

```
[JNDI-lookup-FOO]
;------------- Server side JNDI hook --------------
class      = javax.naming.Context
method     = lookup
signature  = (Ljava/lang/String;)Ljava/lang/Object;
scope = !com¥.myCompany¥.myFoo¥..*
deep_mode  = soft
layer   = FOO/JNDI
```

注 :

- ポイントによってバイトコードが正しく置かれていることを確認するには，**＜プローブのインストール・ディレクトリ＞/log/＜プローブ名＞/detailReport.txt** ファイルで Unique Header Name エントリ（つまり，[JNDI-lookup-FOO]）をチェックします。
- パフォーマンス問題を解決するための最終的な優先順位付け手順で，疑いのあるアプリケーション・エリアから呼び出されるメソッドごとにクラスマップを使用し，ポイントを個別に作成するのは，現実的でないことがあります。呼び出し側のインストルメンテーションの 1 つ以上のレベルを使って，ボトルネットの疑いのある 1 つ以上のメソッド内で費やした時間を特定するのが一般的です。これは，Diagnostics に次のレベルのプロファイルの呼び出しを入力する際に便利な方法です。

次の例では，「myMethod」メソッドによって **com.myCompany.myFoo** クラス内で実行されるメソッドの呼び出しをインストルメントします。

```
[MethodsCalledByFoo.myMethod]
class = !.*
method = !.*
scope = !com¥.myCompany¥.myFoo¥.myMethod.*
layer = FOO/other
```

次に，**com.myCompany.myFoo** クラスで「myMethod」メソッドによって呼び出される「set」メソッドに，引数をキャプチャする例を示します。

```
[SetMethodsCalledByFoo.myMethod]
class = !.*
method = !set.*
scope = !com¥.myCompany¥.myFoo¥.myMethod.*
detail = args:1
layer = FOO/other
```

URI 集計のインストルメンテーション

通常，アプリケーションは同じ URL を使ってさまざまなワークフローにアクセスします。アプリケーションで URI（例：
http:// ＜ myserver ＞ /myApplication?page=home）引数を使ってワークフローを区別する場合，さまざまな URI を異なるサーバ要求として解析および処理するように Diagnostics を設定できます。

URI 集計は，[HttpCorrelation] ポイントから制御されます。args_by_class の有効な正規表現エントリは，URI パターンごとに作成する必要があります。

次の例では，Diagnostics コンソールに ServerRequest を一意に表示できます。

```
http:// ＜ myserver ＞ /myApplication?page=home
http:// ＜ myserver ＞ /myApplication?page=openReport
```

```
[HttpCorrelation]
args_by_class=!.*&page
```

次に，複数の URI パラメータを使用して URI を解析する例を示します。

```
args_by_class=!.*&page&role
```

注 : オーバーヘッドおよびデータ保存に影響するため，セッション・パラメータまたは一意の URI 値は使用しないでください。

WebLogic 環境では，URI 集計を使うときに **＜プローブのインストール・ディレクトリ＞ /etc/inst.properties** に **use.weblogic.get.parameter=true** を設定して，URI 集計が ServletRequest の入力ストリームを消費しないようにしてください。

CORBA VM 間インストルメンテーション

Common Object Requesting Broker Architecture (CORBA) 標準を使用することで, さまざまなコンピュータ言語で記述され, さまざまなシステムで実行されるコンポーネントを連携させることができます。

CORBA VM 間インスタンス・ツリーを相関付ける場合のインストルメンテーションは，**＜プローブのインストール・ディレクトリ＞ ¥etc¥auto_detect.points** ファイルで指定します。

次の手順に従って，CORBA の VM 間インスタンス・ツリーを有効にします。

1 **auto_detect.points** ファイルで, CORBA VM 間ポイントのコメントアウトを解除します。

2 アプリケーション・サーバの起動時に，次の JVM 引数を指定します。

-Dorg.omg.PortableInterceptor.ORBInitializerClass.com.mercury.opal.javaprobe.handler.corba.CorbaORBInitializer

3 次の jar ファイルをクラスパスに配置します。

＜ Java エージェントのインストール・ディレクトリ＞ /lib/probeCorbaInterceptors.jar

RMI インストルメンテーションの使用

キャプチャ・ポイント・ファイルの RMI（VM 間）ポイントは，標準設定で非アクティブになっています。アプリケーションのクロス VM プロセスをキャプチャするには，このポイントをアクティブにする必要があります。RMI 呼び出しの両側でこのポイントがアクティブになっている Java プローブがある場合，Diagnostics は両方の仮想マシンの呼び出しツリー・データを関連付けます。

```
[RMI]
keyword = rmi
layer   = CrossVM
active  = false
```

クラスタ化された環境での RMI インストルメンテーション

＜プローブのインストール・ディレクトリ＞ /etc/inst.properties ファイルの **weblogic.t3.rmi** プロパティは，RMI インストルメンテーションで VM 間 RMI パフォーマンス測定値をキャプチャする方法を制御します。標準設定で，**weblogic.t3.rmi** は false に設定されているため，一般的な RMI インストルメンテーションを使ってパフォーマンス測定値を収集します。クラスタ化された環境において，**weblogic.t3.rmi** が false に設定されているときにアプリケーション・エラーが発生するのを防ぐため，クラスタのすべてのサーバで RMI インストルメンテーションをオンにする必要があります。

weblogic.t3.rmi が true に設定されている場合，一般的な RMI インストルメンテーションが無効になり，WebLogic の T3 プロトコルだけを使って RMI VM 間測定値をキャプチャします。これにより，Java プローブは，RMI インストルメンテーションが有効な調査対象クラスタにある一部のサーバだけで機能するようになります。

スレッドのローカル・ストレージを使った SOAP ペイロードの格納

次の例は，スレッドのローカル・ストレージの使い方を示します。特に，スレッドのローカル・ストレージから SOAP ペイロードを格納（および削除）する方法を示します。標準設定では，特定のアプリケーション・サーバでしか SOAP ペイロードはキャプチャされません。サポート・マトリクスの詳細については，491 ページ「SOAP の障害ペイロード・データの構成」を参照してください。

次の例が適用されるのは，追加設定をしないと Diagnostics でペイロードがキャプチャされないアプリケーション・サーバだけです。

まず，どこからペイロードにアクセスするかを特定する必要があります。ペイロードは DispatchController.dispatch() というメソッドの 2 番目の引数であるとします。

キーワード store-thread は，対応するコード・スニペットの特殊フィールド（ここでは My_Inbound_Payload）をスレッドのローカル・ストレージに格納するように Java プローブに指示します。このキーワードは, 両方のポイントが同じスレッドでヒットするかぎり，別のコード・スニペットから参照できます。次の例は，ペイロードの検索例を示します（下の「フラグメントのローカル・ストレージを使った Web サービスのフィールドの格納」）。

```
[MyAppServer-SoapPayload-Capture]
class      = com.myCompany.DispatchController
method = dispatch
signature = !¥(Ljava/lang/Object;Ljava/lang/Object;¥).*
layer = Web Services
detail    = before:code: ae7f0a58,store-thread


# Used by [MyAppServer-SoapPayload-Capture]
ae7f0a58 = ##My_Inbound_Payload=#arg2;"";
```

複数のスレッド間の相関

非同期サーバ要求は，サーバ要求の開始イベントと終了イベントの間にスレッドを切り替えるサーバ要求のことです。最も単純な場合では，1 つのスレッドが要求を受信し，一部を処理した後，要求を別のスレッドに渡します。渡されたスレッドは処理を完了し，応答を要求元に戻します。

Java エージェントが複数のスレッド間で相関を実行できるように，Diagnostics にはコード・スニペットを通して使用できる 2 つの操作が用意されています。

- parkFragment(Object anchor)

この操作を実行すると，現在のスレッドが現在のサーバ要求を実行しなくなります。Java Agent で記録されるすべてのメソッド呼び出しは，この時点で強制的に終了します。つまり，これらのメソッドの一部は実行を継続しますが，その処理は現在のサーバ要求に無関係となります。さらに，parkFragment の呼び出し後に現在のスレッドが一部のインストルメンテーション対象メソッドを呼び出した場合も，これらの呼び出しは報告されません。サーバ要求は実行しているとはみなされず，指定されたオブジェクト（アンカー）は後で（通常は別のスレッドによって）再開されるサーバ要求の一意の ID としてアプリケーションで使用されます。

- resumeFragment(Object anchor)

  この操作を実行すると，以前に放置したサーバ要求が，現在のスレッドで再開されます。引数（anchor）は，サーバ要求を識別するためのものです。アクティブなすべてのメソッド呼び出しでは，開始時刻が強制的に現在時刻にリセットされます。つまり，これらのメソッドの実行中に有効期限を過ぎた場合も，再開中のサーバ要求には関係がありません。現在のスレッドでサーバ要求が実行されていた場合，サーバ要求は無視（削除）されます。

  これらの操作を使用するときは，正しいアンカー・オブジェクト，および正しいスレッド切り替えポイントの識別をアプリケーション専門家が行うことが重要です。

  競合が発生することに注意してください。フラグメントの「放置」期間が長すぎた場合は，対応する再開操作が実行された後，そのフラグメントは失われます（probe.log に警告が書き込まれます）。この競合状況を回避する便利な方法が 2 つあります。1 つ目は，現在のスレッドがサーバ要求を実際に中止する少し前に parkFragment をサイレントに呼び出す方法です。2 つ目は，スレッド間でオブジェクトを渡すために一般的に使用される，アプリケーション内蔵の同期機能を利用する方法です。

  「放置した」フラグメントを確認するには，pending-fragment サーブレットを使用します。現在実行中のメソッドの代わりに，「PARKED SERVER REQUEST」と表示されます。

この機能を使用するには通常，監視対象アプリケーション内のスレッド切り替えポイントを識別し，対応するインストルメンテーション・ポイントにコード・スニペットを挿入する必要があります。BEA AquaLogic のサポートは，設定しなくても利用できます。

次に，コード・スニペットが対応している 2 つのインストルメンテーション・ポイントの例を示します。これらは，AquaLogic でサポートされています。

この例の最初のポイントは，AquaLogic が別のサーバにサブ要求を送信すると，必ず実行されます。インストルメントされたメソッド PipelineContextImpl.dispatch(...) は，サブ要求が正常に送信された場合，true を戻します。サブ要求を送信するスレッドは応答を待機しないで，引き続きパイプラインから次のサーバ要求を取得します。

したがって，コード・スニペットは戻り値を調べ，true である場合は，現在のサーバ要求が停止されることをプローブに通知します。サーバ要求は，サーバ要求が受信するたびに AquaLogic が作成する MessageContext オブジェクトによって識別されます。

```
[BEA_ALSB_AsyncDispatch]
; instrumentation point for AquaLogic Service Bus asynchronous dispatch
class      = com.bea.wli.sb.pipeline.PipelineContextImpl
method = dispatch
signature  = !¥(Lcom/bea/wli/sb/context/MessageContext;.*
detail     = after:code:549b1b59
layer      = Service Bus/AquaLogic

# Used by [BEA_ALSB_AsyncDispatch]
# Asynchronously dispatches a subrequest for a service, the response will be
# processed on another thread
549b1b59 = (#return == true ?
@ThreadContextProxy@.parkFragment(#location,#arg1) : void);
```

サブ要求から応答を受信すると，AquaLogic は outerCallback.onReceive Response(...) を実行します（通常は別のスレッドで実行）。元のサーバ要求の処理が再開され，コード・スニペットによってプローブに通知されます。

この場合，サーバ要求を表す MessageContext オブジェクトは，インストルメンテーション対象メソッドの引数として使用できません。RouterCallback オブジェクトから抽出する必要があります。

```
[BEA_ALSB_ProxyService_Callback_Response]
; instrumentation point for AquaLogic Service Bus callback function
class      = com.bea.wli.sb.pipeline.RouterCallback
method     = !(onError)|(onReceiveResponse)
signature = !.*
layer      = Service Bus/AquaLogic
detail     = before:code:dba72078

# Used by [BEA_ALSB_ProxyService_Callback_Response]
# Resume processing of a server request when the reply for a subservice comes back
# (or when the server request was moved to the response pipeline internally)
dba72078 =
@ThreadContextProxy@.resumeFragment(#location,#callee._context.getMessageCon
text());"";
```

フラグメントのローカル・ストレージを使った Web サービスのフィールドの格納

次に，ポイントとコード・スニペットの機能の例を示します。

- フラグメントのローカル・ストレージを使って Web サービスの特定のフィールド（ws_name や ws_op など）を格納する方法。この方法は，「DIAG_ARG」文字列を指定する代わりに使用できます。
- （前の例で）格納されたペイロードをスレッドのローカル・ストレージから取得（および削除）する方法
- SOAP ペイロードからコンシューマ ID を抽出する方法
- フラグメントのローカル・ストレージを使ってコンシューマ ID を格納する方法

Web サービスは特殊な方法で処理されるため，一部のフィールドをキャプチャする必要があります。これらのフィールドについては，310 ページ「コード・スニペットの構文」を参照してください。

最初に，必要なフィールド（Web サービスの名前，操作，名前空間，ポート番号）にアクセスできるインストルメンテーション・ポイントを見つけます。インストルメンテーション対象のアプリケーションには，必要なすべてのフィールドにアクセスできる特定のクラスがあり，com.myCompany.MyWSContext という名前が付いているとします。上記のすべてのフィールドを設定する場合は，このクラスのインスタンスにアクセスする必要があります。さまざまなケースが考えられます。その 1 つは，メソッド MyWSFactory.create() の第 1 引数として MyWSContext が渡されたときであるとします。これがインストルメントするメソッドになります。

このインストルメンテーション・ポイントを次に示します（各行については後で説明します）。

```
class      = com.myCompany.MyWSFactory
method   = create
signature  = !¥(Lcom/myCompany/MyWSContext;.*
layer = Web Services
detail    = ws-operation, before:code: f334f0b4,store-fragment
```

上記ポイントの最初の 3 行は，com.myCompany.MyWSFactory.create (MyWSContext, *) に一致するすべてをインストルメントするようにプローブに示します。

4 行目は，このポイントのレイヤを指定します。

5 行目は，このポイントに関する追加情報（detail）をプローブに提供します。

- 1 つ目の detail（ws-operation）は，このポイントを受信 Web サービスとして扱うようにプローブに指示する重要な情報です。
- 2 番目の detail（before:code: f334f0b4）は，このポイントに適合するメソッドの開始時に，対応するコード・スニペットを挿入するようにプローブに指示します。実際のコード・スニペットを以下に示します。f334f0b4 という番号は，http:// ＜プローブのホスト＞ : ＜プローブのポート＞ /inst/code-key にアクセスし，テキスト・ボックスにコード・スニペットを貼り付けることによって生成されたものです。
- 3 つ目の detail キーワード（store-fragment）は，対応するコード・スニペットに含まれるすべての特殊フィールド（##）をサーバ要求の属性として格納するようにプローブに指示します。

対応するコード・スニペットを次に示します（このコード・スニペットの各行については，後で説明します)。

```
f334f0b4 = #wsContext=#arg1;¥
##WS_inbound_service_name=#wsContext.getServiceName();¥
##WS_inbound_operation_name=#wsContext.getOperationName();¥
##WS_inbound_target_namespace=#wsContext.getNamespaceURI();¥
##WS_inbound_port_name=#wsContext.getEndpoint();¥
#soap_payload =
@com.mercury.opal.capture.proxy.ThreadContextProxy@.getThreadContextValue("My
_Inbound_Payload");¥
#consumer_id = (#soap_payload == null ? null :
@com.mercury.opal.capture.proxy.ProbeCodeSnippetHelper@.getConsumerIdFromDo
cument(##WS_inbound_service_name<java.lang.String>,#soap_payload<org.w3c.do
m.Document>));¥
##WS_consumer_id = (#consumer_id == null ?
@ProbeCodeSnippetHelper@DO_NOT_STORE : #consumer_id);"";
```

1 行目 : f334f0b4 = #wsContext=#arg1;¥

前述のように, f334f0b4 という番号は, http:// ＜プローブのホスト＞ : ＜プローブのポート＞/inst/code-key にアクセスし, テキスト･ボックスにコード･スニペットを貼り付けることによって生成されたものです。実際のコード・スニペットは，f334f0b4 = の後から始まります。最初の式は #wsContext=#arg1 です。この式は，メソッドの最初の引数（この場合は MyWSContext オブジェクト）をローカル変数（wsContext）に割り当てます。

2 行目 : ##WS_inbound_service_name=#wsContext.getServiceName();¥

この式は，フラグメントのローカル・ストレージを使ってサービス名を格納します。正確な変数名（WS_inbound_service_name）を使用することが重要です。これらの変数名は，310 ページ「コード・スニペットの構文」の「特殊フィールド」に記載されています。

3 行目 : ##WS_inbound_operation_name=#wsContext.getOperationName();/

この式は，フラグメントのローカル・ストレージを使って WS 操作を格納します。正確な変数名（WS_inbound_operation_name）を使用することが重要です。これらの変数名は，310 ページ「コード・スニペットの構文」の「特殊フィールド」に記載されています。

4 行目 : ##WS_inbound_target_namespace=#wsContext.getNamespaceURI();¥

この式は，フラグメントのローカル・ストレージを使って WS 名前空間を格納します。正確な変数名（WS_inbound_target_namespace）を使用することが重要です。これらの変数名は，310 ページ「コード・スニペットの構文」の「特殊フィールド」に記載されています。

5 行目 : ##WS_inbound_port_name=#wsContext.getEndpoint();¥

この式は，フラグメントのローカル・ストレージを使って WS ポート名を格納します。正確な変数名（WS_inbound_port_name）を使用することが重要です。これらの変数名は，310 ページ「コード・スニペットの構文」の「特殊フィールド」に記載されています。

上記の 5 行があれば，受信 Web サービスを正常にキャプチャできます。コード・スニペットの残りの部分では，SOAP ペイロードからコンシューマ ID をキャプチャする処理を行います。この処理は省略可能であり，インストルメンテーション対象のアプリケーション・サーバが，追加設定なしで SOAP ペイロードのキャプチャがサポートされるアプリケーション・サーバではない場合にのみ必要です。詳細については，上記の例を参照してください。次の例では，前の例でキャプチャした SOAP ペイロードを参照します。

6 行目 : #soap_payload = @com.mercury.opal.capture.proxy.ThreadContextProxy@.getAndRemoveThreadContextValue("My_Inbound_Payload");¥

この式は，スレッドのローカル・ストレージに格納されたペイロードを取得して削除し（ペイロードの格納方法については前の例を参照），それをローカル変数（soap_payload）に格納します。

7 行目 : #consumer_id = (#soap_payload == null ? null : @com.mercury.opal.capture.proxy.ProbeCodeSnippetHelper@.getConsumerIdFromDocument(##WS_inbound_service_name<java.lang.String>,#soap_payload<org.w3c.dom.Document>));¥

この式は，consumer_id ローカル変数を設定します。ペイロードが null の場合は，consumer_id も null に設定されます。null でない場合は，サービス名を使って，consumer.properties のエントリに基づいてコンシューマ ID の照合を行います。コンシューマ ID の照合の詳細については，480 ページ「コンシューマ ID の構成」を参照してください。

8 行目 : ##WS_consumer_id = (#consumer_id == null ? @ProbeCodeSnippetHelper@DO_NOT_STORE : #consumer_id);"";

この最後の行では，このコンシューマ ID ローカル変数がこのサーバ要求のコンシューマ ID になります。正確な変数名（WS_consumer_id）を使用することが重要です。これらの変数名は，310 ページ「コード・スニペットの構文」の「特殊フィールド」に記載されています。

カスタム・インストルメンテーションの注釈の使用

バージョン 1.5 以降の JVM を使用するアプリケーションは，カスタム注釈（InstrumentationPoint）を使ってメソッドのインストルメンテーションを「強制」できます。InstrumentationPoint は，Diagnostics Java Agent の lib ディレクトリにある **annotation.jar** ファイルに含まれています。InstrumentationPoint 注釈を使用するクラスをコンパイルするときは，このファイルのコピーをクラスパスに置きます。この注釈は，次のように定義されています（InstrumentationPoint.java）。

```
/*
 * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
 * ------------------------------------------------------------------------
*/
@Retention(value = RetentionPolicy.RUNTIME)
@Target(value = ElementType.METHOD)
public @interface InstrumentationPoint {
  String layer();
  String keyword() default “”;
  String layerType() default “method”;
  String detail() default “”;
  String code() default “”;
  Boolean active() default true;
}
```

この機能を使用するには，**inst.properties** の **look.for.annotations** プロパティを true（標準設定）に設定する必要があります。

開発

1. Diagnostics Java Agent の lib ディレクトリにある **annotation.jar** ファイルのパスをアプリケーション・ビルドのクラスパスに追加します（またはこの jar をアプリケーションにコピーします）。
2. 監視する必要があるメソッドのクラスをインポートします。

   ```
   import com.mercury.diagnostics.common.api.InstrumentationPoint;
   ```

3. 監視するメソッドを特定し，注釈を追加します。

```
@InstrumentationPoint(ARGS)
public void launchTest4()
```

このインスタンスでは，ARGS に次の項目が含まれます（これらの引数の詳しい意味については，ポイント・ファイルのドキュメントを参照してください）。

- layer=" レイヤ名 "
- keyword= " キーワード "
- layerType=" タイプ "
- detail="detail"
- active="true/false"

例

次の例では，InstrumentationPoint 注釈を使ったコードを示します。

```
/*
 * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
 ------------------------------------------------------------------------
 */

 import com.mercury.diagnostics.common.api.InstrumentationPoint;

 ...

 @InstrumentationPoint(layer=”my_app”,detail=”
 diag,method-no-trim,method-cpu-time”)
 public void myMethod1(Object x, String y) {
    ...
 }
```

上記の例では，myMethod1 がインストルメントされ，すべてのインスタンス・ツリーにノードとして表示されます。このメソッドは，そのレイテンシがメソッドのレイテンシの最小しきい値（デフォルトで 51 ms）より短くてもトリミングされません。さらに，このメソッドによる包括的な（子を含む）CPU 消費量がインストルメントされ，報告されます。

新しいテクノロジまたはカスタム・テクノロジの VM 間相関処理の構成

Diagnostics では，複数の Java 仮想マシン（JVM）を対象とした呼び出しプロファイルを表示できます。このような VM 間呼び出しプロファイルおよびテクノロジは，パフォーマンスの問題にクライアントとサーバが含まれる場合に非常に便利です。どのアプリケーションが問題の原因であるかを特定する場合，クライアントまたはサーバの呼び出しプロファイルを個別に確認しても，これらは相関処理されていないため断続的な問題を解決できない可能性があります。VM 間呼び出しプロファイルには，相関処理されたクライアントとサーバが 1 つの呼び出しツリーで表示されます。

すぐに使用できる Java Agent では，JMS，HTTP/S（Web サービスのみ），RMI，SAP，TIBCO，Corba などのさまざまなテクノロジでこの機能がサポートされています。Diagnostics の最新バージョンでは，新しいテクノロジまたはカスタム・テクノロジの VM 間相関処理を設定するためのサポートが追加されています。

VM 間相関処理の手法はコード・スニペットで公開されているため，インストルメンテーション・ポイントやコード・スニペットを準備して，ほぼすべてのプロセス間通信（社内開発した通信手法や従来の通信手法など）を相関処理できます。通信手法の唯一の要件は，色分けと呼ばれる追加の文字列をメッセージで送信できることです。

色分け文字列は，Java Agent によってクライアント側で作成され，ユーザが作成したコード・スニペットで送信メッセージにアタッチされます。メッセージの受信後に，サーバ側のユーザが作成したコード・スニペットによってメッセージから色分けが抽出され，サーバ側のエージェントに渡されて解析および処理されます。

そのため，VM 間通信手法に関連するユーザの主な役割は，色分けを送信メッセージに埋め込み，受信したメッセージから色分けを抽出することに限定されます。これには，クライアント側（送信ポイント）やサーバ側（受信ポイント）のコードの場所（インストルメンテーション・ポイント）を特定することも含まれます。詳細な例については，363 ページ「カスタム・テクノロジの VM 間相関処理を設定するための チュートリアル」を参照してください。また，カスタム VM 間相関処理を設定するのに役立つ 3 つの API については，361 ページ「カスタム転送 VM 間相関処理を容易にするための API」を参照してください。

クライアント側

送信呼び出し（クライアント側）では，新しい **outbound: ＜色分けタイプ＞** detail を使用します。

次の色分けタイプを利用できます。

- default
- sap
- none
- snippet

none を除くすべての色分けタイプでは，テクノロジ・タイプ，呼び出しターゲットの名前，ID を含む特殊な引数を提供するコード・スニペットが関連付けられている必要があります。

引数の形式は次のとおりです。

DIAG_ARG:type= ＜タイプ＞ &name= ＜名前＞ &target= ＜ターゲット＞

＜タイプ＞は，リモート呼び出しに使用されるテクノロジ・タイプで，＜名前＞と＜ターゲット＞はテクノロジに依存する値です。テクノロジ・タイプは，受信インストルメンテーション・ポイントに使用されるものと同じにする必要があります（361 ページ「サーバ側」を参照）。

snippet を除くすべての色分けタイプでは，プローブで適切な色分けが生成され，後で相関処理できるように Diagnostics サーバにレポートされます。ただし，この時点では送信メッセージのマークは解除されたままです。

none を除くすべての色分けタイプでは，別のインストルメンテーション・ポイント（送信ポイントの後（理想的には送信メソッドの実行時）にヒット）のコード・スニペットで，生成された色分けが送信メッセージにアタッチされる必要があります。

ICorrelationColor RemoteCaptureProxy.getCurrentColor(#location)を呼び出すと，直近に生成された色分けを取得できます。

独自の VM 間通信に対応するために **snippet** を使用できます。これは，コード・スニペットからの直接呼び出しによって色分けが明示的に作成されることを意味します。snippet の色分けの場合，上記の順序が逆になります。つまり，送信ポイントのヒット前（多くの場合，メッセージにアタッチする直前）に色分けが生成されます。コード・スニペットはエージェントを呼び出す前に実行されるため，これには，送信ポイントの **before** コード・スニペットで色分けを作成する場合が含まれています。

コード・スニペットから色分けを作成するには，次の手順を実行します。

1 次の呼び出しを作成します。
ICorrelationColor RemoteCaptureProxy.createColoring(#location, ＜タイプ＞ , ＜ Diagnostics の引数＞)

タイプとして次の値を使用します。

RemoteCaptureProxy.ENCODED_COLORING（**default** の場合）

RemoteCaptureProxy.SAP_R3_COLORING（**sap** の場合）

使用するタイプがわからない場合は，default を使用してください。

2 手順 1 で返されたオブジェクトに対して **grabCorrelationString()** の呼び出しを作成し，戻り文字列を送信メッセージに挿入します（テクノロジに依存する手法を使用）。ここはユーザが工夫できる部分です。

ヒント：文字列メッセージを使用する場合，次のヘルパー API を使用してこの手順を実行します。

ProbeCodeSnippetHelper.createDiagEnvelope(String coloring, String originalMessage)

3 **outbound:snippet** detail を使用してインストルメンテーション・ポイントをヒットします。これにより，新しい色分けは作成されずに，直近に作成された色分けが自動的に使用されます。送信ポイントを実行すると，色分けが実際に使用されたことがプローブに通知され, VM 間呼び出しプロファイルの接続ポイントで考慮されるメソッドが特定されます。同時に発生する VM 間通信では，送信呼び出しのレイテンシを適切にキャプチャできるように，メッセージの送信と確認の受信の両方に使用されるメソッドに **outbound** detail を使用することをお勧めします。

サーバ側

受信呼び出し（サーバ側）では，**inbound: <テクノロジ・タイプ>** detail を使用します。新しい VM 間テクノロジに対応するには，独自のテクノロジ・タイプ名を使用します。既存のテクノロジ名（サーバ要求タイプ）と競合しないように確認します。サーバ要求タイプには，ADO，CICS，Corba，HTTP，JDBC，JMS，MSMQ，RMI，Remoting（.NET），SAP ABAP タイプ，Web サービスなどがあります。また，Pseudo や RootRename という名前のサーバ要求タイプもあります。

before コード・スニペットで次の手順を実行する必要があります。

1 テクノロジに依存する手法（送信呼び出しに使用する手法に対応）を使用して，受信メッセージから相関処理文字列を抽出します。

ヒント：ProbeCodeSnippetHelper.createDiagEnvelope() が以前に使用されている場合，ProbeCodeSnippetHelper.extractColoringFromDiagEnvelope (String envelope) を使用して相関処理文字列を取得します。

また，ProbeCodeSnippetHelper.extractOriginalMessageFromDiagEnvelope (String envelope) を使用して元のメッセージを取得します。

2 キャプチャ引数（**before** コード・スニペット）と抽出した相関処理文字列の 2 つの文字列をスタックに残します。

カスタム転送 VM 間相関処理を容易にするための API

カスタム転送 VM 間相関処理を容易にするために 3 つのヘルパー API が追加されました（これらの使用については，上の項のヒントや，314 ページ「コード・スニペット・ヘルパー」を参照してください）。363 ページ「カスタム・テクノロジの VM 間相関処理を設定するための チュートリアル」にも，例が記載されています。

➤ ProbeCodeSnippetHelper.createDiagEnvelope(String coloring, String originalMessage)

- ProbeCodeSnippetHelper.extractColoringFromDiagEnvelope(String envelope)
- ProbeCodeSnippetHelper.extractOriginalMessageFromDiagEnvelope(String envelope)

HTTP/S のサポート

サーバ側の HTTP/S のサポートは組み込まれており，標準設定で有効になっています。HttpServlet の標準の J2EE 実装，Jetty 実装，Apache Catalina 実装は Java Agent によって自動的に認識されます。これらのテクノロジのいずれかが使用されている場合，サーバ側のユーザ・アクションは不要です。

クライアント側では，java.net.URL クラスの openConnection メソッドが Java Agent によって自動的にインストルメントされ，直近に生成された色分け（存在する場合）が送信 HTTP 要求に埋め込まれます。いずれかの HTTP 要求ヘッダが色分けの送信に使用されます。ヘッダはサーバ側のエージェントによって識別されます。

そのため，クライアント側の HTTP のサポートは，上記のルールから除外されます。送信ポイントと対応する DIAG_ARG を提供する必要はありますが，色分けを送信メッセージに埋め込むことを心配する必要はありません。

たとえば，Diagnostics Mediator では次のポイントが使用されます。

```
[RemoteHttpComponent-Outbound-1]
class      = com.mercury.diagnostics.common.net.registrar.RemoteHttpComponent
method     = getConnection
signature  = (Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/
String;Ljava/lang/String;)Ljava/net/HttpURLConnection;
priority   = 1
detail     = method-no-trim,outbound:default,before:code:7b1125e2
layer      = Network.RemoteHttpComponent
```

getConnection メソッドの最初の引数は String で，接続 URL を表します。参照されるコード・スニペットにより，接続 URL からホスト名とポートが抽出され，ターゲット ID として使用されます。HTTP/S サポートの組み込み部分に調和する方法でこの変換を容易にする特殊なユーティリティ・メソッドが RemoteCaptureProxy によって提供されます。

```
7b1125e2 = #target=@RemoteCaptureProxy@.getTargetFromUri(#arg1); ¥
"DIAG_ARG:type=http&name="+#target+"&target="+#target;
```

カスタム・テクノロジの VM 間相関処理を設定するためのチュートリアル

このチュートリアルでは，カスタム転送ソリューションとして共有ブロック・キューを使用する簡素化されたクライアント - サーバ・アプリケーションを扱います。クライアントはキューに「String」メッセージを追加してこのメッセージを送信します。サーバは「String」メッセージをキューから取り出してこのメッセージを受け取ります。

この例では，（簡潔にするために）1 つの JVM で実行されますが，本来は 2 つのスレッドを使用して，2 つの JVM で実行されるアプリケーションをシミュレートします （1 つの JVM で実行されるスレッドを相関処理する場合，より簡単なソリューションがあります。詳細については，349 ページ「複数のスレッド間の相関」を参照してください）。

サンプル・コードを以下に示します。

```
public class SimulatedCrossVM {


 private static int INTERVAL = 5 * 1000; // 5 seconds
 private static BlockingQueue<String> queue = new LinkedBlockingQueue<String>();

 private static class ReceiverSide extends Thread {

  public ReceiverSide() {
   super("Receiver");
  }

  public void execute(String receivedString) throws InterruptedException {
   System.out.println("Executing message: " + receivedString);
   sleep(2 * INTERVAL);
```

```
 }    private void receiveAndHandleMessage() throws InterruptedException {
   String message = null;
   message = queue.take();
   // Handle it
   execute(message);
  }

  public void run() {
   try {
    while (true) {
     receiveAndHandleMessage();
    }
   }
   catch (Throwable t) {
    // oops
    t.printStackTrace();
   }
  }
}

private static class SenderSide extends Thread {

  // For simulated TCP connection
  private String destHost;
  private int destPort;

  public SenderSide(String host, int port) {
   super(host + ":" + port);
   destHost = host;
   destPort = port;
  }

  public void sendMessage(String origMessage) throws InterruptedException {
   queue.put(origMessage);
  }

  private String generateMessage() {
   String message = "T" + System.currentTimeMillis();
   return message;
  }

  private void generateAndSendMessage() throws InterruptedException {
   sleep(2 * INTERVAL);
   // generate message
   String message = generateMessage();
```

```
    System.out.println("Sender's original message: " + message);
    // And send it (outbound call)
    sendMessage(message);
    sleep(INTERVAL);
   }

   public void run() {
    try {
     while (true) {
      generateAndSendMessage();
     }
    }
    catch (Throwable t) {
     // oops
     t.printStackTrace();
    }
   }
  }

  public static void main(String[] args) {
   SenderSide sender = new SenderSide("fake-host", 12345);
   ReceiverSide receiver = new ReceiverSide();

   sender.start();
   receiver.start();
  }
}
```

このコードを実行すると，次の結果が得られます。

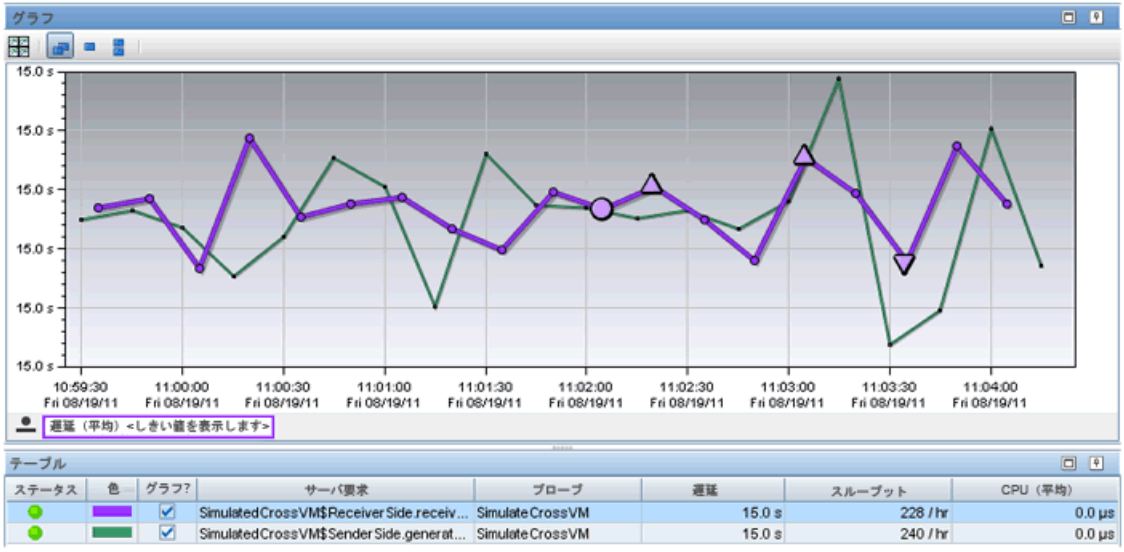
```
Sender's original message: T1313785958127

Executing message: T1313785958127
```

手順 1：メソッドをインストルメントする

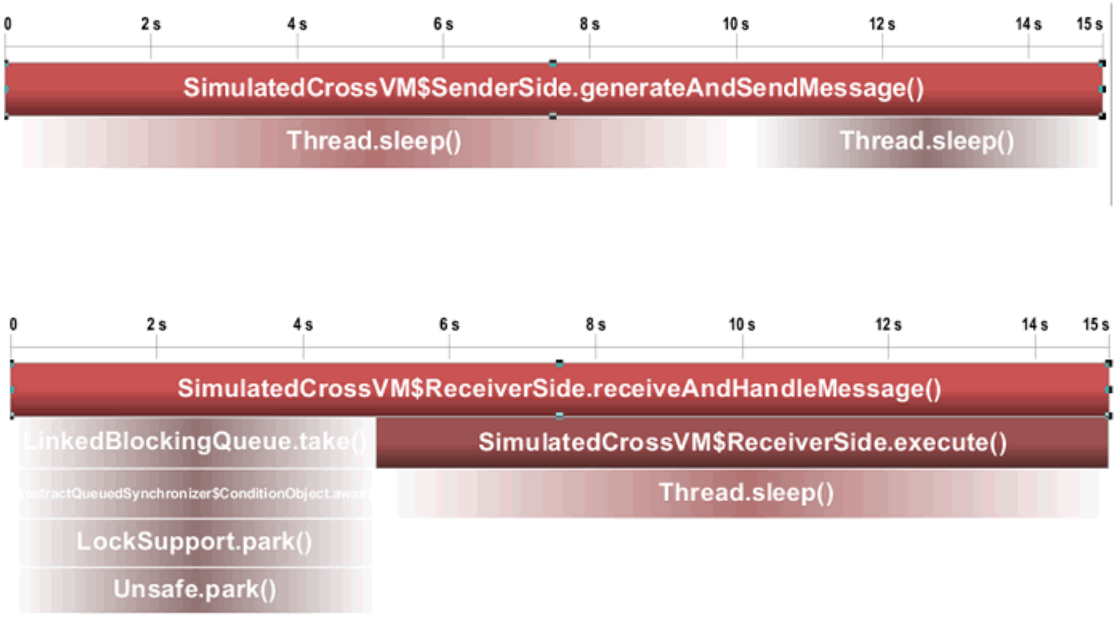
メソッドをインストルメントすることで, Diagnostics はどのメソッドが重要なのかを認識できます。これらはカスタム・メソッドであるため，すぐに使用できるインストルメンテーション・ポイントでは何も実行されません。次のインストルメンテーション・ポイントを追加して etc/autodetect.points ファイルを編集します。インストルメンテーション・ポイントを定義するためのガイダンスについては，372 ページ「Java Profiler UI からのインストルメンテーションの保守」を参照してください。

```
[SimCrossVM-Sender]
class     = SimulatedCrossVM$SenderSide
method    = generateAndSendMessage
signature = !.*
layer     = Sending


[SimCrossVM-Outbound]
class     = SimulatedCrossVM$SenderSide
method    = sendMessage
signature = !.*
layer     = Sending

[SimCrossVM-Receiver]
class     = SimulatedCrossVM$ReceiverSide
method    = receiveAndHandleMessage
signature = !.*
layer     = Receiving


[SimCrossVM-Inbound]
class     = SimulatedCrossVM$ReceiverSide
method   = execute
signature = !.*
layer     = Receiving
```

結果：インストルメントするこのテスト・プログラムを実行すると，次のサーバ要求が表示されます。



次に，送信側と受信側の呼び出しプロファイルを示します。



手順 2 : 送信側のロジックに「色分け」を追加する

この手順では，クライアントから送信されるメッセージに「色分け」を追加します。インストルメントするサーバでこの色分けされたメッセージを受信すると，HP Diagnostics によって相関処理が行われます。この部分は扱いが難しいため，コード・スニペット構文に精通していない場合は，308 ページ「コード・スニペットを使用したポイントの定義」を参照してください。

まず，コード・スニペット（outbound:snippet）を使用する送信ポイントとしてメソッドをマークし，メソッドを呼び出す前に実行するコード・スニペット（before:code:5ea4753f）を特定します。最初の引数を使用するため，より具体的なシグネチャ（!¥(Ljava/lang/String;.*）を提供することをお勧めします。

```
[SimCrossVM-Outbound]
class      = SimulatedCrossVM$SenderSide
method     = sendMessage
signature  = !¥(Ljava/lang/String;.*
layer      = Sending
detail     = outbound:snippet,before:code:5ea4753f
```

対応するコード・スニペットを以下に示します。1 行目では，サーバのホスト名および宛先ポートを含む文字列（#target）が作成されます。2 行目では，新しい文字列（#diagArg）が定義されます。この後に特殊な構文（DIAG_ARG:type= ＜タイプ＞ &name= ＜名前＞ &target= ＜ターゲット＞）が続きます。＜タイプ＞はテクノロジ・タイプで，任意の名前を選択できます。これは次の手順で使用されます。＜名前＞と＜ターゲット＞はテクノロジに依存する値で，UI に表示されます。これらの値も任意に選択できます。3 行目では，3 番目の文字列（#color）が定義されます。これは，ほかからのメソッドの特定の呼び出しを識別するために使用されます。4 行目では，色分けされた文字列でメソッドの最初の引数が更新されます。これにより，変更された文字列が sendMessage から送信されます。最後の 5 行目では，HP Diagnostics で使用できるように色分けをスタックに配置します。

1 5ea4753f = #target=#callee.destHost+":"+#callee.destPort; ¥

2 #diagArg = "DIAG_ARG:type=CB-TCP&name=Senders.sendMessage&target="+#target; ¥

3 #color = (null == #arg1 ? null :
@RemoteCaptureProxy@.createAndGrabColor(#location,
@RemoteCaptureProxy@ENCODED_COLORING, #diagArg.toString())); ¥

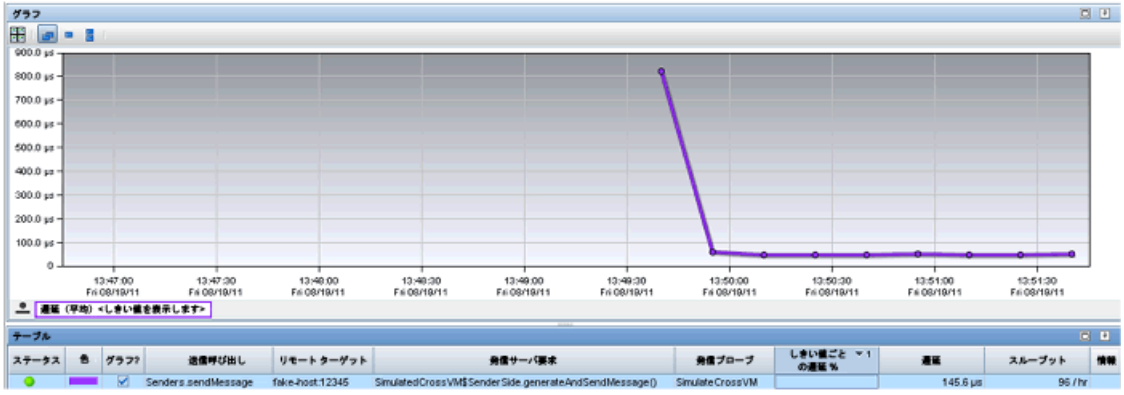
4 #arg1 = @ProbeCodeSnippetHelper@.createDiagEnvelope(#color, #arg1); ¥

5 #diagArg;

この例を実行すると，次のように出力が更新されます。受信側で取得した文字列メッセージが，送信された文字列メッセージと異なっています。これは，4 行目のコード・スニペットの結果です。多くの場合，受信側でこれをうまく処理できません。これは，クライアントとサーバで同じインストルメンテーションを使用していない場合（特に両方ともインストルメントされていない場合）に「偶然」発生する可能性があるため，受信側の動作を記録しておくことをお勧めします。

送信側の元のメッセージ : T1313786970403
実行メッセージ : HP_DIAG1_!Dhf/ABAABKrh3Qf0cy7yaLsAAAAAAA9mYWtlLWhvc3Q6MTIzNDUAYTEzMTM3ODY5NjAzODgmU2ltdWxhdGVDcm9zc1ZNJlNpbXVsYXRlZENyb3NzVk0kU2VuZGVyU2lkZS52b2lkIGdlbmVyYXRlQW5kU2VuZE1lc3NhZ2UoKSZcMCZcMCZcMCY=:T1313786970403

この時点では，UI の一部の［送信呼び出し］のみが変更されています。［送信呼び出し］および［リモート ターゲット］カラムの値は，コード・スニペットの＜名前＞と＜ターゲット＞に入力した値です。



手順 3：受信側の色分けを削除する

最後の手順では，送信側の元の「色分けされていない」メッセージを受信側で取得できるように受信側で色分けを削除します。ます，手順 2 で定義したコード・スニペットで使用されるテクノロジ・タイプを使用して，このポイントを受信ポイントとしてマークし，このメソッドを呼び出す前に実行するコード・スニペットを割り当てます。また，この引数はコード・スニペットで使用されるため，より具体的なシグネチャを指定します。

```
[SimCrossVM-Inbound]
class      = SimulatedCrossVM$ReceiverSide
method   = execute
signature  = !¥(Ljava/lang/String;.*
detail     = before:code:d2c83d3c,inbound:CB-TCP
layer      = Receiving
```

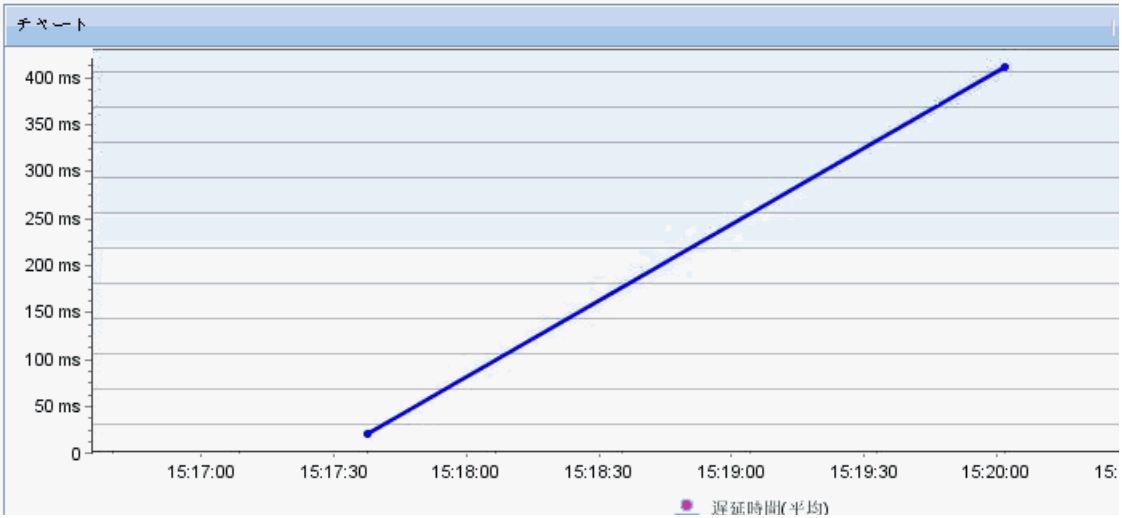
対応するコード・スニペットを以下に示します。1 行目では，受信メッセージから色分けが抽出されます。2 行目では，メソッドの最初の引数が更新され，クライアントから送信された元のメッセージに復元されます。3 行目では，HP Diagnostics で使用できるように色分け（および空の文字列）がスタックに配置されます。

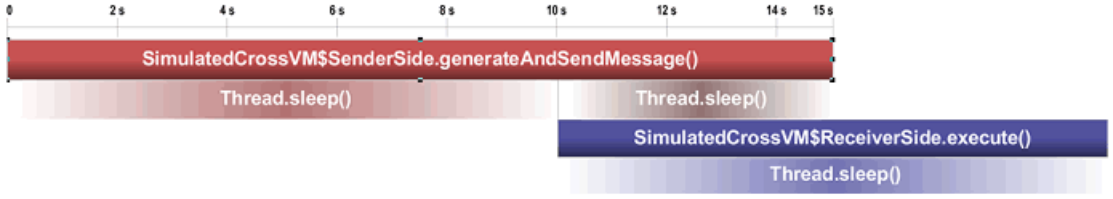
1. d2c83d3c = #coloring=@ProbeCodeSnippetHelper@.extractColoringFrom DiagEnvelope(#arg1); ¥
2. #arg1=@ProbeCodeSnippetHelper@.extractOriginalMessageFromDiagEnvelope (#arg1); ¥
3. "";#coloring;

これで，プログラムの出力が元に戻りました。

```
Sender's original message: T1313789287234
Executing message: T1313789287234
```

［サーバ要求］ビューには，送信側の「generateAndSendMessage」で VM 間呼び出しプロファイルが利用できることが示されています。この呼び出しプロファイルを開いて確認すると，クライアントとサーバの呼び出しプロファイルが結合されていることがわかります。このサンプル・アプリケーションでは不十分ですが，実際のアプリケーションでは，パフォーマンスの問題が発生している場所（クライアント，サーバ，またはその両方）がわかります。





この呼び出しプロファイルは，少し変わっていますが非同期アプリケーションの典型例です。クライアントは応答を待機せずに，一部の処理を続行します（エラーの場合は 5 秒間スリープ）この間にサーバは要求を処理し，数秒後に完了させます。メソッドの時間間隔が次のようにツリー形式で表示されます。中に 2 が記載されているひし形は JVM の深さを表しています。サーバがさらに別の送信呼び出しを作成すると，3 以上になります。VM 間相関処理は，そのような場合に特に便利です。多数の JVM でパフォーマンスの問題の原因を見つけることを想像してみてください。

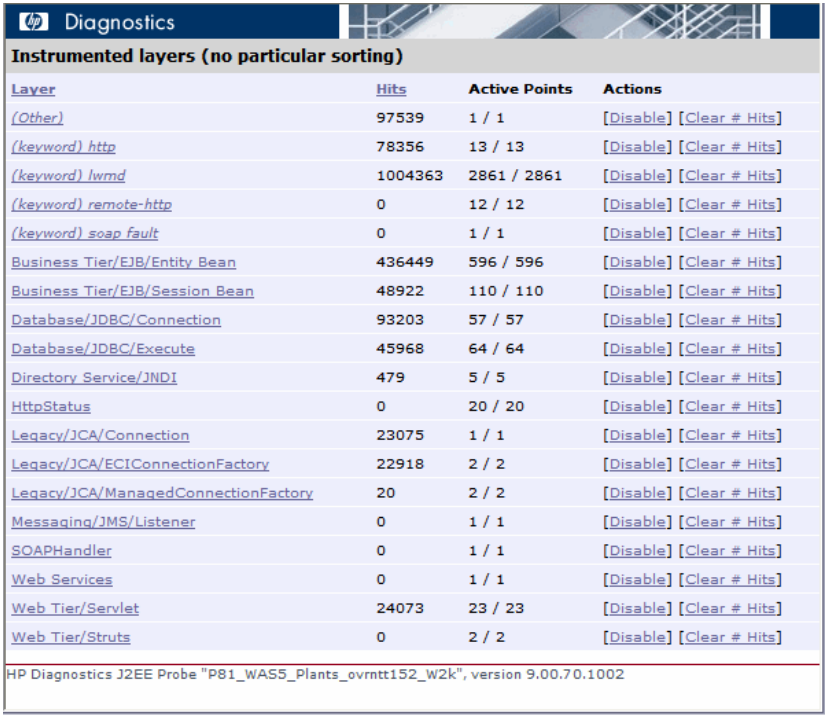
Java Profiler UI からのインストルメンテーションの保守

Java Diagnostics Profiler の［構成］タブを使用して，インストルメンテーション・ポイントを保守し，Java Agent キャプチャ・ポイント・ファイルまたはプロパティ・ファイルを手動で編集することなくプローブ設定を編集できます。プロファイル処理を開始しているかどうかに応じて，Java Diagnostics Profiler から［構成］タブにアクセスできます。

Java Diagnostics Profiler の［測定］セクションでは，プローブが監視しているアプリケーションのインストルメンテーションを表示および更新できます。［編集］ダイアログでは，Diagnostics でアプリケーションのインストルメンテーションに使うキャプチャ・ポイント・ファイルに定義したように，インストルメンテーション・ポイントを表示および編集できます。

現在のインストルメンテーションの確認

現在のキャプチャ・ポイント・ファイルのポイントの結果としてインストルメントされたレイヤ，クラスおよびメソッドを確認するには，［構成］タブの［測定］セクションにある［**表示 ...**］をクリックします。［Instrumented Layers］ページが表示されます。


Diagnostics

Instrumented layers (no particular sorting)

Layer	Hits	Active Points	Actions
(Other)	97539	1 / 1	[Disable] [Clear # Hits]
(keyword) http	78356	13 / 13	[Disable] [Clear # Hits]
(keyword) lwmd	1004363	2861 / 2861	[Disable] [Clear # Hits]
(keyword) remote-http	0	12 / 12	[Disable] [Clear # Hits]
(keyword) soap fault	0	1 / 1	[Disable] [Clear # Hits]
Business Tier/EJB/Entity Bean	436449	596 / 596	[Disable] [Clear # Hits]
Business Tier/EJB/Session Bean	48922	110 / 110	[Disable] [Clear # Hits]
Database/JDBC/Connection	93203	57 / 57	[Disable] [Clear # Hits]
Database/JDBC/Execute	45968	64 / 64	[Disable] [Clear # Hits]
Directory Service/JNDI	479	5 / 5	[Disable] [Clear # Hits]
HttpStatus	0	20 / 20	[Disable] [Clear # Hits]
Legacy/JCA/Connection	23075	1 / 1	[Disable] [Clear # Hits]
Legacy/JCA/ECIConnectionFactory	22918	2 / 2	[Disable] [Clear # Hits]
Legacy/JCA/ManagedConnectionFactory	20	2 / 2	[Disable] [Clear # Hits]
Messaging/JMS/Listener	0	1 / 1	[Disable] [Clear # Hits]
SOAPHandler	0	1 / 1	[Disable] [Clear # Hits]
Web Services	0	1 / 1	[Disable] [Clear # Hits]
Web Tier/Servlet	24073	23 / 23	[Disable] [Clear # Hits]
Web Tier/Struts	0	2 / 2	[Disable] [Clear # Hits]

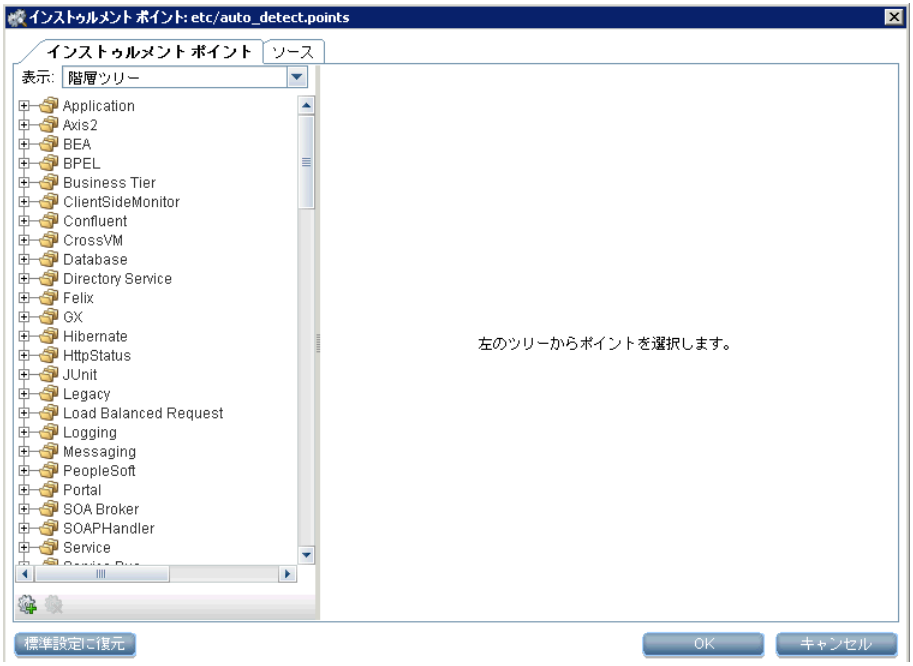
HP Diagnostics J2EE Probe "P81_WAS5_Plants_ovrntt152_W2k", version 9.00.70.1002

［Instrumented layers］ページには，インストルメントされたレイヤ，レイヤのインストルメンテーション・ポイントがトリガされた回数，および現在レイヤでアクティブなポイントの数が一覧表示されます。次のカラムが表示されます。

列	説明
Layer	インストルメントされたレイヤを表示する。このカラムのレイヤ名は，プローブによって監視されたレイヤの処理に関する詳細情報を提供するページにリンクしている。**注 :** 実際にインストルメントされたポイントに定義されているレイヤだけが表示される。
Hits	一覧のレイヤのポイントによって監視されているクラスとメソッドが呼び出された回数が表示される。［**Actions**］列の［**Clear # Hits**］リンクを使って回数をリセットできます。
Active Points	現在アクティブなポイントの数，および特定のレイヤに定義されているポイントの総数を表示する。
Actions	表示されたレイヤの情報を操作するためのリンクが表示される。使用できるアクションは次のとおりです。 ➤ **Disable** : 選択したレイヤのすべてのポイントを無効にしてデータをキャプチャしないようにします。インストルメンテーションはそのままになり，再び有効にすることができます。ここでポイントを有効または無効にすると，次のアプリケーションを再起動するまで有効または無効のままになります。ポイントの標準設定の有効ステータスを変更する場合は，300 ページ「キャプチャ・ポイント・ファイルのポイントのコーディング」を参照。 ➤ **Clear # Hits** : 選択したレイヤの［# Hits］列に表示されている数をリセットします。

インストルメンテーション・ポイントの保守

アプリケーション内の監視対象をプローブに伝えるインストルメンテーション命令を提供するポイントを保守するには，Java Diagnostics Profiler の［構成］タブに移動して，［共有しているインストルメンテーション］または［インスタンスのインストルメンテーション］で［**編集 ...**］をクリックします。［インストゥルメント ポイント］ダイアログが開きます。

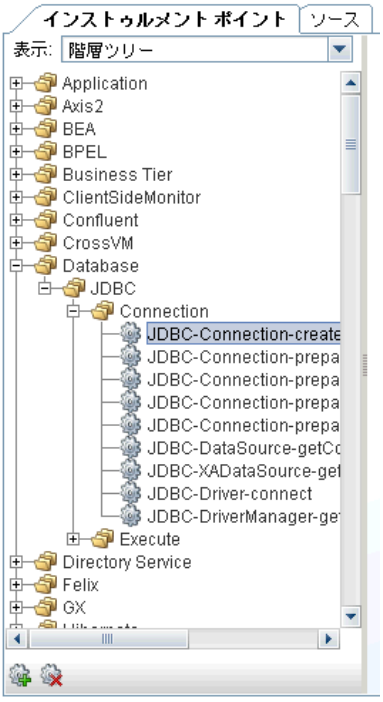


インストルメンテーションは次の 2 つの方法で編集できます。［**インストゥルメント ポイント**］タブでポイントのリストまたはツリーを視覚的に使うか，または［**ソース**］タブのキャプチャ・ポイント・ファイルのソースを通じて編集します。

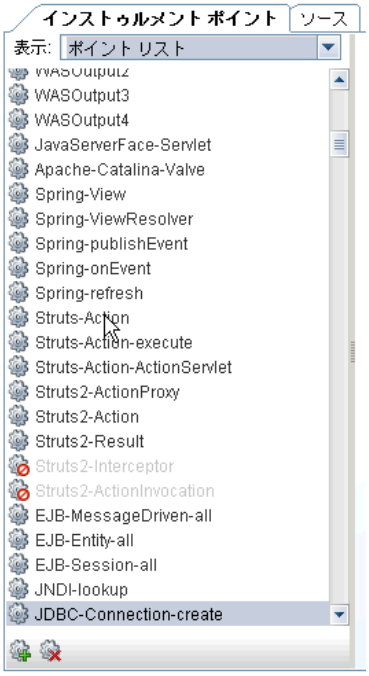
既存のポイントの選択および表示

[インストゥルメント ポイント] ダイアログのナビゲーション・バーを使って，保守するキャプチャ・ポイント・ファイルのポイントを簡単に見つけることができます。[**表示**] ドロップダウンから選択することで，ポイントを一覧表示する形式を指定できます。

ドロップダウンから［**階層ツリー**］を選択すると，ポイントに割り当てたレイヤとサブレイヤに従ってキャプチャ・ポイント・ファイルのポイントをツリー形式で表示されます。



ドロップダウンから［**ポイント リスト**］を選択すると，キャプチャ・ポイント・ファイル内のポイントが昇順で一覧表示されます。

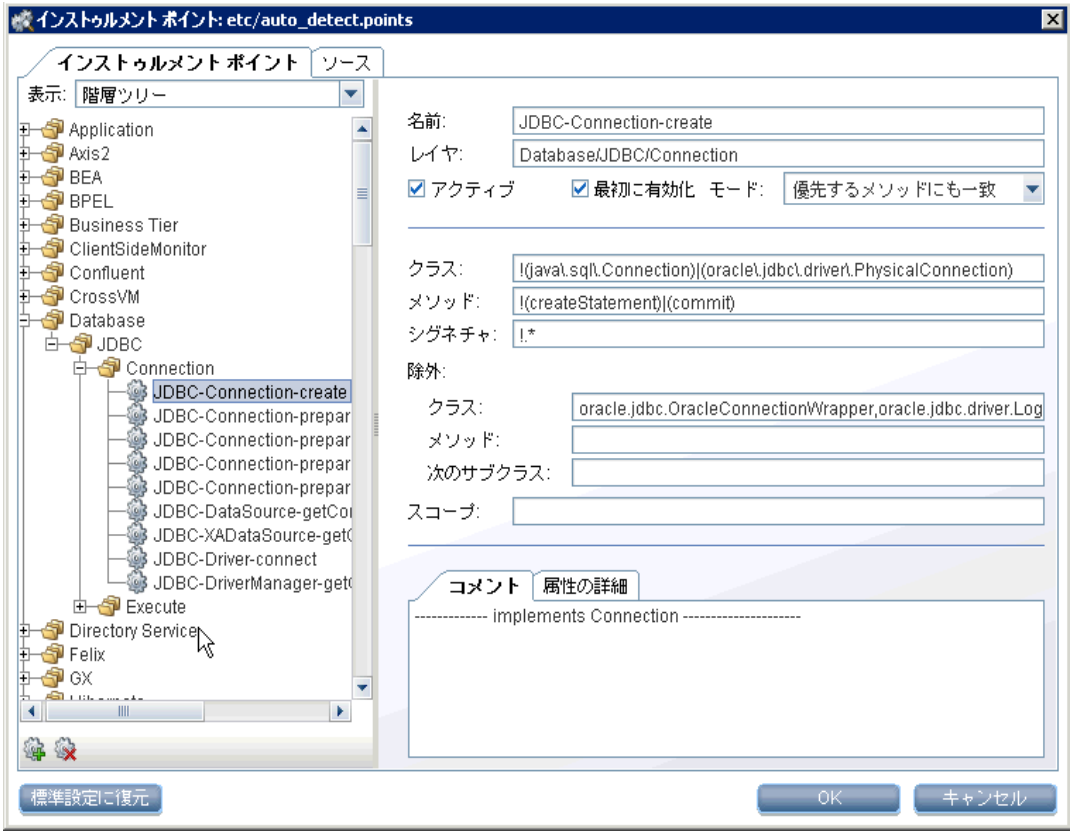


表示または保守するポイントが見つかったら，ナビゲーション・バーでポイントを選択します。［表示 / 編集］パネルに選択したポイントの詳細が表示され，ポイントを操作できます。

既存のポイントの更新

ナビゲーション・バーからレイヤまたはサブレイヤを選択すると，［表示 / 編集］パネルに，ポイントの選択を指示するプロンプトだけが表示されます。

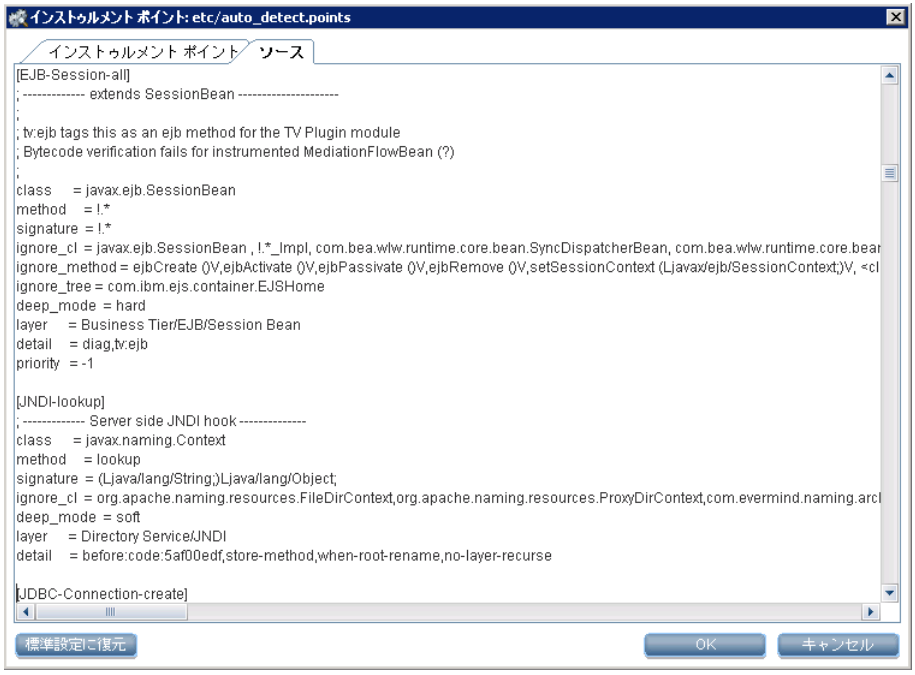
既存のポイントを更新するには，ナビゲーション・バーからポイントを選択します。Profiler には，［表示 / 編集］パネルの［インストゥルメント ポイント］タブにポイントの詳細情報が表示されます。



キャプチャ・ポイント・ファイルのポイントを定義するときに一般的に使われる引数は，別のデータ・フィールドとして表示され，簡単に必要な変更を加えることができます。画面の下部にある［**属性の詳細**］タブに，詳細な引数が表示されます。ポイントのコメントは，［**コメント**］タブに表示されます。変更したら，［**OK**］をクリックします。［変更を適用］をクリックして，［構成］タブを使って加えた変更をすべて適用します。

キャプチャ・ポイント・ファイルのポイントの定義に使用可能なすべての引数については，300 ページ「キャプチャ・ポイント・ファイルのポイントのコーディング」を参照してください。

次に，［ソース］タブの例を示します。



既存のポイントまたはレイヤの削除

ナビゲーション・バーに表示されているポイントまたはレイヤを削除できます。

ポイントまたはレイヤを削除するには，次の手順を実行します。

1 ［インストゥルメント ポイント］タブのナビゲーション・バーからポイントまたはレイヤを選択します。

2 ［ポイントを削除］をクリックします。Profiler は，ナビゲーション・バーのリストから選択したエンティティを削除します。

Profiler の［構成］タブからすべてのインストルメンテーション・ポイントの更新を適用するまで，選択したエンティティは，実際にキャプチャ・ポイント・ファイルから削除されません。

3 [**OK**] をクリックして，[インストゥルメント ポイント] ダイアログを閉じます。

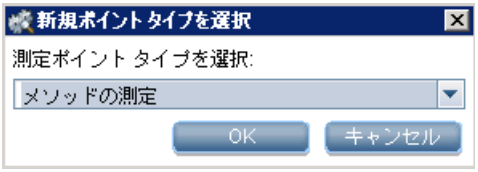
4 [**変更を適用**] をクリックして，[構成] タブを使って加えた変更をすべて適用します。

新しいポイントの追加

インストルメンテーションにポイントを追加できます。

ポイントを追加するには，次の手順を実行します。

1 [新規ポイント] をクリックします。Profiler に，[新規ポイント タイプを選択] ダイアログが表示されます。



2 ドロップダウンから適切なポイント・タイプを選択して [**OK**] をクリックします。

[インストゥルメント ポイント] タブに，選択したポイント・タイプに新しいポイントを作成するために初期化された [表示 / 編集] セクションが表示されます。

3 タブの適切な場所に，新しいポイントの引数とコメントを入力します。

レイヤ情報を入力すると，ナビゲーション・バーの新しいポイントのエントリが更新され，正しい既存のレイヤにポイントを表示します。指定したレイヤが存在しない場合は，新しいレイヤに表示されます。

Profiler の [構成] タブからすべてのインストルメンテーション・ポイントの更新を適用するまで，新しいポイントは，実際にキャプチャ・ポイント・ファイルに追加されません。

4 [**OK**] をクリックして，[インストゥルメント ポイント] ダイアログを閉じます。

5 [**変更を適用**] をクリックして，[構成] タブを使って加えた変更をすべて適用します。

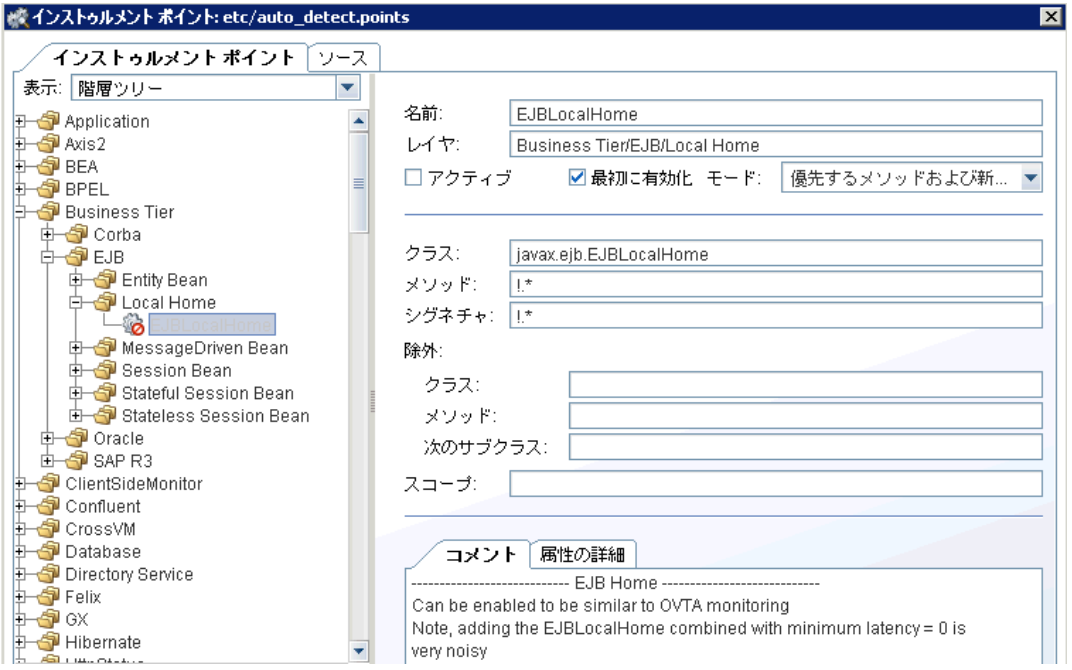
OVTA のようなポイントのアクティブ化

ServletFilter および EJB ローカル・ホーム・メソッドに対する Java プローブのインストルメンテーションにポイントが含まれます。これらのインストルメンテーション・ポイントは，OVTA（OpenView Transaction Analyzer）Java モニタに似た追加機能を提供します。

ServletFilter ポイントでは，サーバ・フィルタを正しく監視するために HttpCorrelation2 ポイントも有効にする必要があります。これは，ServletFilter では，最初に Diagnostics が HTTP サーバ要求を確認するためです。

標準設定で，EJBLocalHome，ServletFilter および関連する HttpCorrelation2 インストルメンテーション・ポイントはアクティブになっていません。非アクティブなポイントは，下の図のようにインストルメンテーション・ポイントの隣のアイコンに赤い記号で示されます。これらのポイントを使うには，UI を使用するか，**auto_detect.points** ファイルを直接編集して，このファイル内で active=true を設定します。

376 ページ「既存のポイントの選択および表示」の説明のように，Profiler UI でこれらのポイントを見つけ，[**Business Tier**]>[**EJB**]>[**LocalHome**]>[**EJBLocalHome**] ポイントか，または［**Web Tier**］>［**Servlet**］>［**ServletFilter**］ポイントおよび［**HttpCorrelation2**］ポイントに移動します。



これらのポイントをアクティブに設定するには，次の手順を実行します。

1 インストルメンテーション・ポイント・ナビゲーション・バーからポイントを選択して，Profiler にポイントの詳細を表示します。［**アクティブ**］チェック・ボックスをオンにします。

2 ［**OK**］をクリックして，［インストゥルメント ポイント］ダイアログを閉じます。

3 ［**変更を適用**］をクリックして，［構成］タブを使って加えた変更をすべて適用します。アプリケーション・サーバを再起動して（プローブを再起動して），新規にアクティブ化したポイントを有効にします。

デフォルトのポイントの復元

Profiler または HP Diagnostics を使って問題の診断を完了したら，場合によっては，標準設定のインストルメンテーションを復元して，より堅牢なインストルメンテーションからオーバーヘッドが生じるのを避ける必要があります。

インストルメンテーションにデフォルトの設定を復元するには，次の手順を実行します。

1 ［**標準設定に復元**］をクリックします。

Profiler の［構成］タブからすべてのインストルメンテーション・ポイントの更新を適用するまで，インストルメンテーション・ポイントは，実際にキャプチャ・ポイント・ファイルに追加されません。

2 ［**OK**］をクリックして，［インストゥルメント ポイント］ダイアログを閉じます。

3 ［**変更を適用**］をクリックして，［構成］タブを使って加えた変更をすべて適用します。

標準の Java クラスおよびメソッド用に定義された標準設定のレイヤ

HP Diagnostics では，キャプチャ・ポイント・ファイルに指定されている命令に基づいて，クラスおよびメソッドのパフォーマンス測定値を*レイヤ*および*サブレイヤ*にグループ化します。類似のシステム・リソースを使用されるアプリケーションで処理するためにパフォーマンス測定値を一緒に報告できるよう，デフォルトのレイヤが定義されています。レイヤは，パフォーマンスの問題が継続する可能性のあるシステムのエリアを隔離および特定しやすくします。

次の表に，標準 Java クラスおよびメソッドで定義された標準設定のレイヤおよびサブレイヤを示します。

キャプチャ・ポイント・ファイルには，プラットフォーム特有のレイヤも定義されています。多くの場合，これらのレイヤは，次の表に定義されている最上位の親レイヤのサブレイヤです。レイヤのパフォーマンス・データは，Diagnostics UI の［ロード ビュー］に表示されます。

Java EE レイヤ

レイヤ	サブレイヤ	親レイヤ
Web Tier	JSP Servlets Struts Session Spring Struts2	
Business Tier	EJB CORBA	
Web Services		
EJB	Entity Bean Session Bean Local Home Stateless Session Bean Stateful Session Bean MessageDriven Bean	Business Tier
Directory Service	JNDI	
Database	JDBC	
JDBC	Execute Connection	Database
Messaging	JMS Spring	
JMS	Producer Listener Consumer	Messaging
Spring	Producer Consumer	Messaging
Hibernate		

ポータル・レイヤ

Diagnostics では, ポータルの処理に関連するクラスおよびメソッド呼び出しのパフォーマンス測定値を Portal Component レイヤにグループ化します。各 Portal Component レイヤは，ポータル・ライフサイクル・メソッドのレイヤに細分化されます。ポータル・レイヤの詳細については,『HP Diagnostics ユーザー・ガイド』を参照してください。

10

.NET アプリーションのカスタム・インストルメンテーション

このセクションでは，.NET Agent によるパフォーマンス・メトリックスの収集を可能にするために HP Diagnostics がアプリケーションのクラスとメソッドに適用するインストルメンテーションの制御方法について説明します。

本章の内容

インストルメンテーションとキャプチャ・ポイント・ファイルについて

インストルメンテーションとは，アプリケーションのクラス・ファイルが CLR によって読み込まれるときにプローブがそのクラス・ファイルに挿入するバイトコードを参照することです。インストルメンテーションを行うと，プローブは実行時間のインストルメンテーション，呼び出しのカウント，例外の取得，およびメソッド呼び出しとスレッドの関連付けが可能になります。各プローブのインストルメンテーション・ポイントは，キャプチャ・ポイント・ファイルで指定されています。

キャプチャ・ポイント・ファイルは，インストルメンテーションの範囲を制御できるようにします。これにより，アプリケーションのパフォーマンスを把握するのに必要なすべての情報が Diagnostics から提供され，ユーザは情報収集に余計な費用をかけたり無関係な情報に混乱することがなくなります。キャプチャ・ポイント・ファイルに含まれるインストルメンテーション定義は，どのメソッドをインストルメントするか，どのようにインストルメントするか，そしてどのインストルメンテーションをインストールするかをプローブに伝える *points*（ポイント）と呼ばれます。

ポイントには，命令を複数のメソッド，クラス，または名前空間仕様に適用するために，命令を「ワイルドカード（未知数）にする」正規表現を含めることができます。正規表現の使用の詳細については，882 ページ「正規表現の使用」を参照してください。

キャプチャ・ポイント・ファイルでポイントをカスタマイズして，標準設定ポイントの範囲に含まれないアプリケーションのエリアにメソッド，クラス，および名前空間を含めることができます。

Microsoft の .NET の仕様には，Web メソッドと WCF メソッドをインストルメントする場合を除き，ビジネス・ロジックが実装する必要がある統合インタフェースまたは推奨インタフェースは含まれません。したがって，.NET プローブではほとんどの場合，.NET アプリケーションのビジネス・ロジック・クラスおよびメソッドのパフォーマンスに関する有益なメトリックスを収集できるように，キャプチャ・ポイント・ファイルにカスタム・ポイントが必要になります。

キャプチャ・ポイント・ファイルのポイントは，レイヤにグループ化されます。レイヤは，優先順位付けプロセスの一部として比較可能な，重要な情報の層にパフォーマンス・メトリックスを体系付けたり，インストルメンテーションの収集動作を制御したりします。

キャプチャ・ポイント・ファイルのポイントは，標準設定レイヤにグループ化されます デフォルトのレイヤをカスタマイズして，新しいレイヤを作成できます（422 ページ「標準 .NET アプリケーションの標準設定レイヤ」を参照してください）。

.NET キャプチャ・ポイント・ファイルの検索

.NET Agent をインストールすると, 事前に定義されたデフォルトのキャプチャ・ポイント・ファイルがインストールされます。

ASP.NET アプリケーション用の標準設定キャプチャ・ポイント・ファイルは, **＜プローブのインストール・ディレクトリ＞ ¥etc¥** にあります。これらのファイルには，次の表に示されたほかのポイント・ファイルと，**Asp.Net.points**, **Ado.points**，および **WCF.points** が含まれています。

また，.NET Agent インストーラは，IIS 配備済み ASP.NET アプリケーション・ドメインを検出すると，検出したドメインごとに個別のキャプチャ・ポイント・ファイルを自動作成します。自動的に検出および作成されたポイント・ファイルを修正して，アプリケーション・ドメインに対してカスタム・インストルメンテーション・ポイントを有効にする必要があります。これらのキャプチャ・ポイント・ファイルは, **＜プローブのインストール・ディレクトリ＞ ¥etc¥ ＜アプリケーション・ドメイン＞ .points** ファイル内にあります。これらのポイント・ファイルおよび標準設定のポイント・ファイルは，.NET Agent によって読み取られます。

インストール時には，標準設定のポイント・ファイルである **Asp.Net.points**, **Ado.points**，**WCF.points** だけが有効になります。次の標準設定の .NET ポイント・ファイルは, **＜プローブのインストール・ディレクトリ＞ /etc** ディレクトリにインストールされますが，有効にされません。

デフォルトのポイント・ファイル（最初は無効）	インストルメンテーション・ターゲット
Asp.Net.IExecutionStep.points	IIS5，IIS6 および IIS7。このファイルは，IIS ポイントよりに替わるものです。
IIS.points	IIS5 および IIS6
Lwmd.points	ライトウェイトなメモリ診断
Msmq.points	Microsoft Message Queuing（MSMQ インストルメンテーション）
Remoting.points	.NET Remoting
WebServices.points	ASP.NET Web サービス

上記のポイント・ファイルを有効にするには，**probe_config.xml** ファイルの appdomain 範囲内の**< points >**要素に，各ファイルへの参照を追加します。**probe_config.xml** ファイルの各要素の詳細については，第 13 章,「.NET Agent 設定ファイルについて」を参照してください。

TransactionVision に固有の .NET プローブ・インストルメンテーションについては,『HP TransactionVision Deployment Guid』を参照してください。

キャプチャ・ポイント・ファイルのポイントのコーディング

次の引数は，ポイント・ファイルのポイントを定義する場合に使用できます。

```
[Point-Name]=< ポイントの一意の名前 >
;-------------------------------------------------------------------------
class        = < キャプチャするクラス名 / パッケージ名 >
method     = < キャプチャするメソッド名 >
signature  = < メソッドのシグネチャ >
ignoreClass = < 無視するクラス >
ignoreMethod= < 無視するメソッド・プロトタイプ >
ignoreTree= < 無視するクラス階層 >
deep_mode= < ソフト・モードまたはハード・モード >
scope       = < メソッドのカンマ区切りのリスト >
ignoreScope= < メソッドのカンマ区切りのリスト >
detail                = < 指定子のリスト >
keyword    = < キーワード >
layer        = < レイヤ名 >
layerType = < レイヤ・タイプ >
```

注意 : インストールのアップグレード時に変更内容が失われるため, 標準設定のポイント・ファイルは変更しないでください。アプリケーション固有のインストルメンテーション・ポイントは，カスタム・キャプチャ・ポイント・ファイルに格納してください。

正規表現リストとして指定可能なすべての引数には，260 文字という文字数の上限があり，この上限を超えると値は切り捨てられます。引数については，次のセクションで説明します。

必須ポイント引数

LWMD，HttpCorrelation，WSCorrelation，および WCF を除くすべてのポイントには，次の引数が含まれている必要があります。

引数	説明
Point-Name	ポイントの一意の名前。
class	インストルメントするクラスまたはインタフェースの名前を指定する。名前には完全な名前空間名が必要で，名前空間レベルとクラス・レベルの間にはピリオドを使用します。任意の有効な正規表現を使用できる。
method	インストルメントするメソッドの名前を指定する。そのために，メソッド名は，class 引数によって指定されるクラスまたはインタフェースで定義されたメソッドと一致する必要があります。任意の有効な正規表現を使用できる。
layer	レイヤ，サブレイヤ，またはこのポイントのデータがグループ化される層を指定する。レイヤは，インストルメンテーションの収集制御の一部です。 ポイントのレイヤは，/（スラッシュ）でレイヤ名を区切ることで，ネスト・レイヤまたはサブレイヤで指定できる。スラッシュの後に指定されたレイヤは，スラッシュの前に指定されているレイヤのサブレイヤです。サブレイヤ名の後にほかのスラッシュとレイヤ名をコーディングして，サブレイヤに独自のサブレイヤを設定できる。

以下は，必須引数を含むカスタム・ポイントの例です。

```
[MyCustomEntry_1]
; comments here….
class = myNameSpace.myClass.MyFoo
method = myMethod
layer = myCustomStuff
```

注 : ポイントのほとんどの引数に正規表現を使用できます。それらの正規表現は，前に感嘆符を置く必要があります。正規表現の使用の詳細については，882 ページ「正規表現の使用」を参照してください。

ポイント・エントリ（省略可能）

ポイント定義には，次の引数を 1 つ以上含めることができます。

引数	説明
keyword	特殊なインストルメンテーションを指定する。keyword 引数は特定の機能を有効にする場合に使用できる（たとえば，WCF キーワードを指定すると WCF 機能が有効になる）。また，ポイント定義を特殊機能に関連付けることもできる。この例として RemotingServer キーワードと Remoting.points ファイルが挙げられる。 ➤ **HttpCorrelation** : HTTP 経由のクライアント / サーバ・メソッド呼び出しの相関を有効にする。 ➤ **WsCorrelation** : クライアント側の Web サービス相関ロジックを有効にし，.NET および Java テクノロジに関する，未処理の HTTP クライアント要求呼び出しの相関を有効にする。 ➤ **WCF** : WCF 機能を有効にする。 ➤ **REST** : WCF REST サービス・インストルメンテーションを有効にする。 ➤ **lwmd** : lwmd インストルメンテーションを有効にする。 ➤ **リモート処理** : .NET Remoting フレームワークのインストルメンテーションを有効にする。 ➤ **RemotingServer** : .NET Remoting サーバのポイントを，それらのポイント用の特別な .NET Remoting ロジックに関連付ける。詳細については，411 ページ「.NET Remoting のインストルメンテーションを設定する方法」を参照してください。
ignoreClass	無視するクラスをカンマ区切りリストで指定する。**ignoreClass** で指定したクラスのいずれかに一致するクラスはインストルメントされません。

引数	説明
ignoreMethod	無視するメソッドをカンマ区切りリストで指定する。**ignoreMethod** で指定したメソッドのいずれかに一致するメソッドはインストルメントされません。
ignoreTree	指定されたクラスから継承したクラスに実装されているメソッドのインストルメンテーションを無視する。したがって，メソッドのクラス階層ツリー全体が無視される。
deep_mode	サブクラスの処理方法を指定する。この引数は，3 つの値を受け入れる。 ➤ **none** - 値 none は，**deep_mode** 引数を指定しないものに類似する。サブクラスの処理方法に影響しません。 ➤ **soft** - 値 soft は，クラス，メソッド，およびシグネチャのエントリに一致するすべてのクラスまたはインタフェースに対して，一致するメソッドとシグネチャを実装するサブクラスまたはサブインタフェースもインストルメントするように要求する。 ➤ **hard** - 値 hard は，クラス，メソッド，およびシグネチャのエントリに一致するすべてのクラスまたはインタフェースに対して，任意の深さのサブクラスまたはサブインタフェースのすべてのメソッドをインストルメントするように要求する。通常，hard モードは，インタフェースのポイントで使われます。**注意 :** hard モードでは広範なインストルメンテーションが実行されるため，プローブのオーバーヘッドが非常に高くなる恐れがある。
scope	インストルメンテーションが実行されるコンテキストを制約する。このエントリが指定されている場合，挿入されたバイトコードは呼び出し側になる。この引数の値には，任意の有効な正規表現を使用できる。scope 値は，メソッド名のカンマ区切りのリストで表される。
ignoreScope	scope 引数で指定された範囲に含まれるメソッドから，特定のメソッドを除外する。この引数の値には，任意の有効な正規表現を使用できる。ignoreScope 値は，メソッド名のカンマ区切りのリストで表される。

引数	説明
detail	より具体的なキャプチャ命令を指定する。 次の返された文字列は，［プロファイルの呼び出し］ビューの詳細表示枠にあるメソッドの［引数］フィールドに表示される。これは，以下のカンマ区切りリストです。 **args:n** – 一致するメソッドでサポートされているすべての引数タイプをキャプチャする。値「n」を指定すると，すべての引数がキャプチャされます。1 ～ 256 の値を入力することもできます。 **args:0** – 現在のクラス・インスタンスまたは呼び出される側のオブジェクトで ToString() を呼び出す。静的メソッドでは，この引数は無効である。 ***args:1** – メソッドが最上位要求の場合，引数にサーバ要求のキー引数としてマーク（*****）する。 detail 引数では次の値も取得する。 **tv:user_event** - 一致するメソッドの TransactionVision イベントを生成する。TransactionVision イベントの一部として，メソッドのパラメータは要求ペイロードとして収集され，戻り値は応答ペイロードとして収集される。パラメータまたは戻り値のオブジェクトによって返される ToString() 値が表示される。一部のパラメータおよび戻り値は収集されない可能性がある。 .NET アプリケーションのほぼすべてのメソッドから TransactionVision イベントを生成できるようにすることで，さまざまなトランザクション・トレースに対応できる。TransactionVision イベントを生成するメソッドを指定する。イベント数が多くなり過ぎて TransactionVision のパフォーマンスが低下しないように，一度に 1 つのメソッドのイベント生成を指定することを強く推奨する。(利便性のためにサポートされているが）ワイルド・カードは指定しないようにする。

引数	説明
layerType	一部のインストルメント済みメソッドに対する特殊な処理方法を指定する。次の 3 つの値が有効である。 ➤ **trended_method** - トレンド・メソッド・ビューに表示するメソッドを特定する。 ➤ **sql** - SQL ビューで SQL のキャプチャに使用するメソッドを特定する。これらは HP Diagnostics によって設定され，変更できません。
signature	シグネチャ（戻り値とパラメータの型）を指定する。たとえば，System.String(System.int32, System.String) など。任意の有効な正規表現を使用できる。

インストルメンテーションの例

次に，キャプチャ・ポイント・ファイル内にポイントを作成したり，変更したりして，アプリケーションのインストルメンテーションをカスタマイズする方法を示します。

本項の内容

➤ 407 ページ「非 ASP.NET アプリケーションまたは Windows アプリケーション用のポイントの設定方法とセットアップ方法」

➤ 411 ページ「.NET Remoting のインストルメンテーションを設定する方法」

カスタム・レイヤおよびサブレイヤ

➤ 次のポイントは，myCompany.myFoo クラスのメソッド myMethod に対して，「FOO」というレイヤの中に「BAR」というカスタム・サブレイヤを作成します。

```
[myCompany.myFoo_customLayer]
class = myCompany.myFoo
method = myMethod
layer = FOO/BAR
```

ワイルドカード・メソッド

➤ 次のポイントは，MyCompany.MyFoo クラスのすべてのメソッドをキャプチャします。

```
[myCompany.myFoo_AllMethods]
class = myCompany.myFoo
method = !.*
layer = FOO/BAR
```

特定のメソッドの無視

➤ 次のポイントは，setHomeInterface メソッドと getHomeInterface メソッドを除いて，MyCompany.MyFoo クラスのすべてのメソッドをキャプチャします。

```
[myCompany.myFoo_AllMethodsExcept]
class = myCompany.myFoo
method = !.*
ignoreMethod = setHomeInterface,getHomeInterface
layer = FOO/BAR
```

➤ 次のポイントは, MyCompany.logging クラスに含まれるメソッドを除いて, MyCompany 名前空間のメソッドをすべてキャプチャします。

```
[myCompany_All_Methods_except_from_MyCompany_Logging]
class = !myCompany¥..*
method = !.*
ignoreClass = MyCompany.logging
layer = FOO/BAR
```

トレンド・メソッド・ビューのキャプチャ・メソッド

➤ 次のポイントは, 必要なデータをキャプチャして myMethod メソッドのトレンド・メソッド・ビューに入力します。

```
[myCompany.myFoo_customLayer]
class = myCompany.myFoo
method = myMethod
layer = FOO/BAR
layertype = trended_method
```

クラスの特定のメソッドだけのキャプチャ

➤ 次のポイントは, MyCompany.MyFoo クラスのすべての非静的コンストラクタ・メソッドをキャプチャします。

```
[myCompany.myFoo_Constructor]
class = myCompany.myFoo
method = .ctor
layer = FOO/BAR
```

➤ 次のポイントは, MyCompany.MyFoo クラスのすべての静的コンストラクタ・メソッドをキャプチャします。

```
[myCompany.myFoo_Singleton]
class = myCompany.myFoo
method = .cctor
layer = FOO/BAR
```

➤ 次のポイントは, MyCompany.MyFoo クラスの setFoo メソッドをキャプチャします。

```
[myCompany.myFoo_setFoo]
class = myCompany.myFoo
method = setFoo
layer = FOO/BAR
```

➤ 次のポイントは，MyCompany.MyFoo クラスの，名前に「set」が含まれるすべてのメソッドをキャプチャします。

```
[myCompany.myFoo_AllSets]
class = myCompany.myFoo
method = !.*set.*
layer = FOO/BAR
```

➤ 次のポイントは，MyCompany 名前空間のすべてのメソッドをキャプチャします。

```
[myCompany_All_Methods]
class = !myCompany¥..*
method = !.*
layer = FOO/BAR
```

文字列を返す特定のメソッドのキャプチャ

➤ 次のポイントは, MyCompany.MyFoo クラスの System.String を返す getFoo メソッドをキャプチャします。

```
[myCompany.myFoo_GetFoo_String]
class = myCompany.myFoo
method = getFoo
signature = !System.String¥(.*
layer = FOO/BAR
```

呼び出し側のインストルメンテーション

標準設定では，Diagnostics のすべてのインストルメンテーションは呼び出される側のインストルメンテーションであり，バイトコードがメソッド呼び出し内に置かれています。呼び出し側のインストルメンテーションは，メソッドの中ではなく，インストルメントするメソッドへの呼び出しの近くにインストルメンテーションのバイトコードを配置しているプロセスを意味します。

呼び出し側のインストルメンテーションでは，インストルメンテーションの配置を微調整できますが，スコープで指定された各クラス内で，ポイントで指定されているクラスまたはメソッドの参照があるかどうかをチェックする必要があるため，アプリケーションの初期化にかかる時間が長くなる可能性があります。

インストルメントする呼び出し側を指定するには，scope 引数と ignoreScope 引数を使用します。次の 2 つの例は，呼び出し側のインストルメンテーションを示しています。

- 次のポイントは, MyCompany.logging クラスから呼び出される MyCompany 名前空間のメソッドをすべてキャプチャします。

```
[myCompany_All_Methods_from_MyCompany_Logging]
class = !myCompany¥..*
method = !.*
scope = !MyCompany.logging.*
layer = FOO/BAR
```

- ignoreScope 引数を使って, scope 引数に指定されたスコープに含まれる特定のクラスおよびメソッドを除外します。次のポイントは，myMethod メソッドから呼び出されるものを除いて，MyCompany.logging クラスから呼び出される MyCompany 名前空間のメソッドをすべてキャプチャします。

```
[myCompany_All_Methods_except_from_MyCompany_Logging]
class = !myCompany¥..*
method = !.*
scope = !MyCompany.logging.*
ignoreScope = MyCompany.logging.myMethod
layer = FOO/BAR
```

引数のキャプチャ

キャプチャする引数は，ポート・ファイル・セクションの detail キーで指定します。

次の例では，n 番目の引数の ToString() メソッドを呼び出しています。返された文字列は，［プロファイルの呼び出し］ビューのメソッドの［引数］フィールドに表示されます。detail=args:1,...args:4, *args:3

注意が必要な特殊な値がいくつかあります。

- args:n – 一致するメソッドでサポートされているすべての引数タイプをキャプチャします。値「n」を指定すると，すべての引数がキャプチャされます。1 ～ 256 の値を入力することもできます。
- args:0 – 現在のクラス・インスタンスまたは呼び出される側のオブジェクトで ToString() メソッドを呼び出します。
- args 引数に * を追加すると（*args:1），キー引数としてマークされます。

各メソッド呼び出しの引数を表示する場合は，キー引数を指定しないでください。このようにすると，キャプチャされたインスタンス・ツリーに関する詳細情報が得られるため，このインスタンスが最大のツリーである理由や，例外発生時に渡された値を調べる際に役立つことがあります。

メソッドに関するサーバ要求を引数別にグループ化するには，キー引数を指定します。キー引数は個別の値を使用してサーバ要求を集計します。引数に個別の値が多数含まれる場合は，個別の値それぞれに一意のサーバ要求が生成されるため，キー引数の候補として適切ではありません。

注 : 引数のキャプチャを指定しなかった場合も，呼び出しツリー内のメソッドが例外をスローしたときは，引数がキャプチャされます。これらの引数は，［プロファイルの呼び出し］ビューの例外の詳細ページにある［スタック トレース］セクションに表示されます。詳細については，［プロファイルの呼び出し］ビューのオンライン・ヘルプを参照してください。

次に示す引数のキャプチャの例は，次のコードに関連付けられています。

```
[ILTest]
class = !ILTest_NameSpace.ILTest_Class
method = methodWithParams
detail = args:0, *args:3, args:5, args:7
layer = myFunctionLayer
```

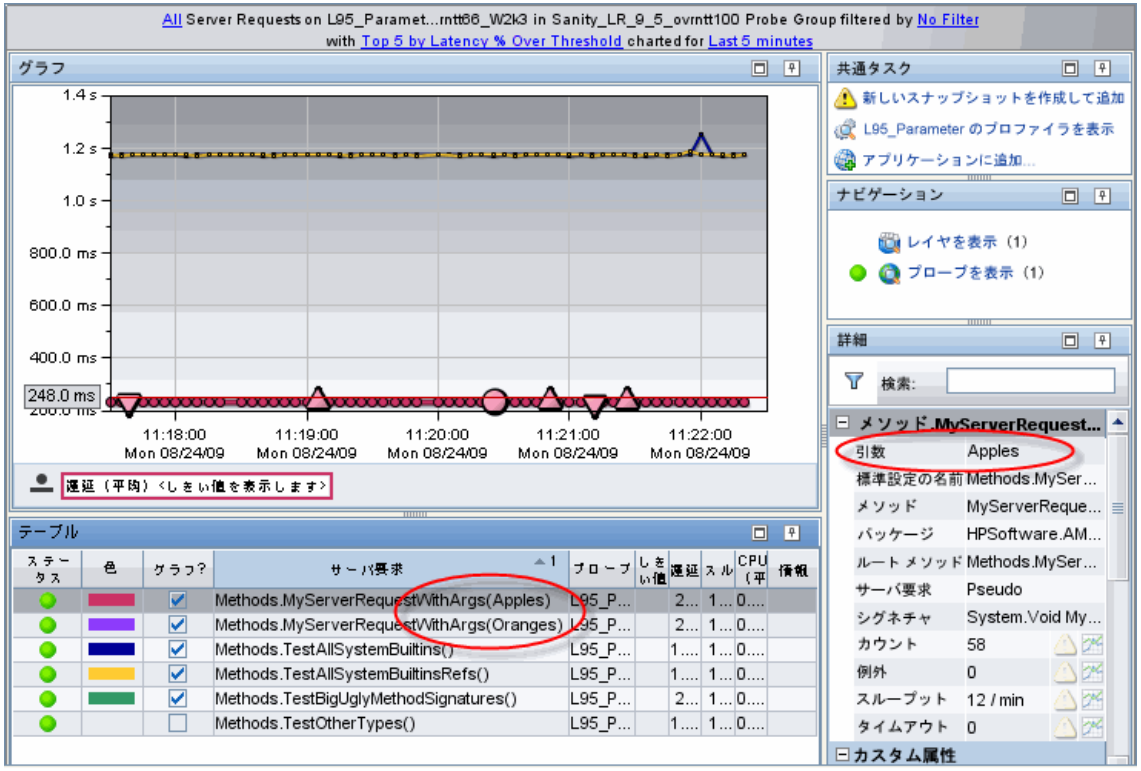
次に，関連するコードの例を示します。

```
class ILTest_Class
{
public bool methodWithParams
(string param1, int param2, string QnameParam3, long param4, object param5, int
param6, double param7)
{

... some implementation

}
}
In this example the defined detail will capture ILTest_Class.ToString(args:0)
param1, QnameParam3, param5 and
param7.
```

最上位メソッドが methodWithParams の場合，QnameParam3 の値はサーバ要求 ID の一部となります。

キャプチャする引数がキー引数（アスタリスク *****）としてマークされていて，メソッドが最上位メソッドの場合，引数値はサーバ要求 ID の一部になります。

たとえば, Shipping Type というメソッドのパラメータを使用してさまざまな出荷を処理している場合に, Shipping Type 引数をキー引数として指定すると, メソッドで処理される出荷ごとに（リンゴやミカンごとに）集計されたビューを表示できます。



キー引数を指定すると，［プロファイルの呼び出し］ビューの［詳細］表示枠にある［引数］フィールドにキー引数が表示されます。［詳細］表示枠の［メソッドの引数］にも引数が表示されます。

キャプチャ対象の引数にキー引数のマーク（アスタリスク *）がない場合は，［プロファイルの呼び出し］ビューの［メソッドの引数］にのみ引数が表示されます。

監視のための WCF REST サービスの設定

.NET プローブの場合，WCF REST サービスは標準設定で **WCF.points** ファイルのすぐに使用できる **keyword=REST** 値に基づいて監視されます。これらの REST サービスは，Web サービスとして監視され，パフォーマンス・データは Diagnostics UI の SOA サービス・ビューに表示されます。

次の項の説明に従って REST サービスを詳細に設定できます。

REST サービスの設定

WCF REST スタイル・サービスでは，操作が url パラメータにエンコードされる場合があります。次に例を示します。

HTTP Method: PUT Url: http://localhost:81/RestNOSvc/AccountsRESTService/{ID}?op={OPERATION} op can be "deposit" or "withdrawal"

これらのタイプのサービスで操作を区別するには，REST サービス・メソッドの操作パラメータを**キー引数**として指定します。これにより，個別の操作として表示できるようになります。引数のキャプチャの概要については，400 ページ「引数のキャプチャ」を参照してください。

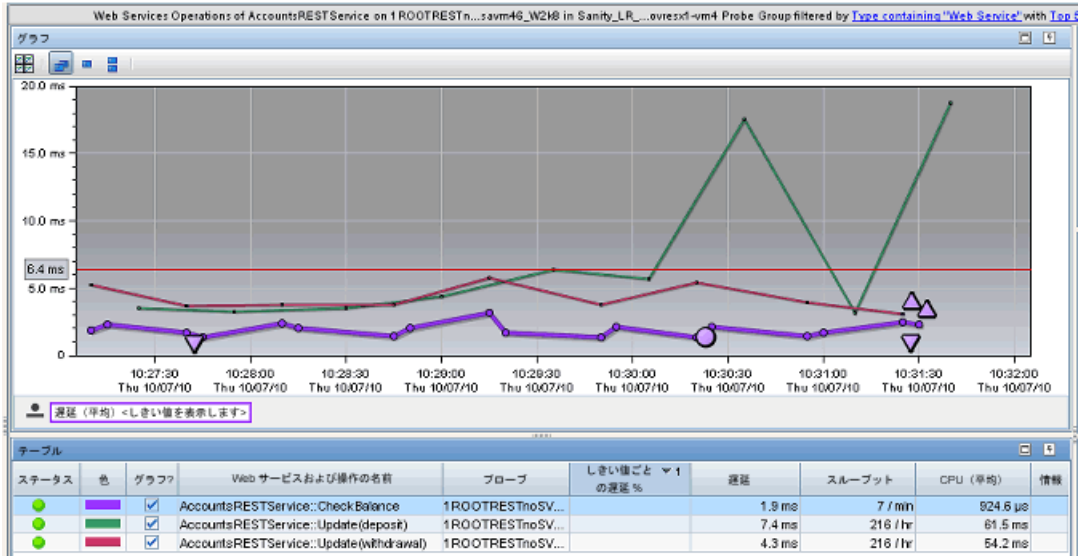
たとえば，メソッドの場合は次のようになります。

```
[WebInvoke(UriTemplate = "{id}?op={operation}", Method = "PUT")]
public TransactionResult Update(string id, string operation, long Amount)
```

操作はキー引数で，ポイント・ファイルで次のように指定できます。

```
[WebSite2-RestNOSvc]
class = !HP.Test.WcfRestService.*
method = Update
detail = *args:2
layer = WebSite2-RestNOSvc
```

次の［SOA Services Operations］ビューの例は，この設定の結果で，個別の操作がテーブルに表示されています。



REST クライアントの設定

REST サービス・クライアントは，HTTP クライアントの呼び出しと同じで区別できません。そのため，REST サービス・クライアントである .NET アプリケーションを監視するには，**probe_config.xml** ファイルで設定オプション **<httpclient showurl="false"/>** を設定し，送信呼び出しの数が多くなることや記号テーブルの急激な増大を避ける必要があります。呼び出しの数は，クライアントがアクセスする一意の URL に起因し，多くの場合，URL で ID がエンコードされます。

次に例を示します。

/RestNOSvc/AccountsRESTService/
8FFD2F34-E334-4E1E-A940-50FCCCACE1D1

Guid は異なるアカウント ID を表します。

deep_mode の例

次のインタフェース定義は，soft および hard 両方の deep_mode の例で使われます。

```
public interface Interface1 {

    public void callerMethod();

}
```

次のクラスは，soft および hard 両方の deep_mode の例で使われます。

```
public class Class1 implements Interface1 {
    public void callerMethod(){
        calleeMethod();
        calleeMethod2();
    }

    public void calleeMethod(){
        Console.WriteLine("hello world");
      //more code lines here…
    }

    public void calleeMethod2(){
        Console.WriteLine("hello world 2");
    }
}
```

➤ 次のポイントは，Class1 クラス内の callerMethod をキャプチャします。

```
[Training-1]
class      = Interface1
method = !.*
deep_mode  = soft
layer      = Training
```

➤ 次のポイントは，Class 1 内のすべてのメソッド（つまり callerMethod，calleeMethod1，および calleeMethod2）をキャプチャします。

```
[Training-1]
class      = Interface1
method = !.*
deep_mode  = hard
layer      = Training
```

非 ASP.NET アプリケーションまたは Windows アプリケーション用のポイントの設定方法とセットアップ方法

本項では，非 ASP.NET アプリケーションまたは Windows アプリケーションのインストルメンテーションを可能にする **probe_config.xml** ファイルとカスタム・ポイント・ファイルの両方の設定方法について説明します。Windows サービス，コンソール・アプリケーション，Windows Forms アプリケーション，および WPF アプリケーションのインストルメンテーションは Windows アプリケーションのインストルメンテーションとみなされ，このような意味で使用されます。

Windows Application の設計

監視対象の Windows アプリケーションの設定方法を考える場合の重要な考慮事項は，.NET プローブが長期間実行されるプロセスを監視するように設計されていることです。したがって，Windows アプリケーションが数秒間実行された後で終了するように設計されている場合は，その実行に関するデータをまったく確認できない可能性があります。Windows アプリケーションがすぐに終了すると，プローブが Diagnostics サーバまたは Diagnostics .NET Profiler との通信を確立して維持する前に，appdomain がシャットダウンされ，プローブがシャットダウンされることがあるためです。

次に，簡単な Windows アプリケーションの例を示します。この例を使って，Windows アプリケーションのインストルメンテーションを設定するときに検討が必要な重要な概念をいくつか説明します。

```
namespace Hello_dotNet_nameSpace
{
      class someclass
      {
            static void Main(string[] args)
            {
// do something

                  // read form commandline then exit
                  clReader myClReader = new clReader();
                  String cl;
                  cl = myClReader.readCl();
            }
      }
      // Command Line Reader
      public class clReader
      {
            public String clread;

            public String readCl()
            {
                  System.Console.WriteLine("Continue?");
                  clread = Console.ReadLine();
                  return clread;
            }
      }
}
```

Hello_dotNet.exe Windows アプリケーションは，1 つのメソッドを呼び出し，ユーザがコマンド・ラインで何かを入力するのを待ってから終了する Main() です。プローブは，このアプリケーションが終了するまでアクティブになります。

Hello_dotNet.points ファイルの作成

＜プローブのインストール・ディレクトリ＞ ¥bin フォルダには **Reflector.exe** コマンド・ライン・ユーティリティがあります。このユーティリティを Hello_dotNet.exe Windows アプリケーションに対して実行することにより，推奨されるポイント・ファイルを取得できます。Reflector ユーティリティの詳細については，590 ページ「アプリケーションのクラスとメソッドの検出」を参照してください。

Reflector.exe と Hello_dotNet.exe アプリケーションの両方が同じフォルダにある場合は，次のコマンドを実行します。

```
Reflector.exe Hello_dotNet.exe
```

出力は標準出力に送信されます。ほかの情報とともに，次の推奨 Hello_dotNet.points が表示されます。

```
-------------------------------------------------------------------------
Sample .points by Namespace
-------------------------------------------------------------------------
[Hello_dotNet_nameSpace]
class = !Hello_dotNet_nameSpace.*
layer = Hello_dotNet_nameSpace
```

推奨ポイントはそのまま使用できますが, Windows アプリケーションに Main() のような，インストルメントされていてもアプリケーションが終了するまで exit を返さないメソッドがある場合は別です。この場合，そのメソッドはアプリケーションが存続する間ずっと実行されるため，アプリケーションが終了するまで何も報告されません。プローブはアプリケーションの終了時にアンロードされるため，インストルメンテーション・ポイントからデータをまったく取得できない可能性があります。

この状況を解消するには，Main() メソッドまたはそれに類するメソッドがインストルメントされないようにポイント・ファイルを作成します。次の Hello_dotNet.points ファイルは，この方法を示しています。Main() は someclass に実装されていると仮定します。

Hello_dotNet.points :

```
[Hello_dotNet_nameSpace]
class = !Hello_dotNet_nameSpace.*
ignoreClass = Hello_dotNet_nameSpace.someclass
layer = Hello_dotNet_nameSpace

[ignore]
class = Hello_dotNet_nameSpace.someclass
ignoreMethod = Main
layer = Hello_dotNet_nameSpace
```

このタイプのポイント・ファイルの重要な点が太字で示されています。[ignore] セクションは，Main() メソッドを無視しますが，Hello_dotNet_nameSpace.someclass 内にほかのメソッドがあれば，これらのメソッドをインストルメントします。

インストルメンテーション用の Windows アプリケーションの設定

Hello_dotNet.exe Windows アプリケーションをインストルメントするように .NET プローブを設定するには，次の XML を **probe_config.xml** ファイルに追加します。これは，ファイルの末尾にある **</probeconfig>** エントリの直前に追加できます。

```
<process name="Hello_dotNet">
      <points file="Hello_dotNet.points" />
      <instrumentation>
        <logging level="" />
      </instrumentation>
      <logging level="" />
</process>
```

注 : probe_config.xml ファイルに上記の変更を加える前に, **Hello_dotNet.points** ファイルを **＜プローブのインストール・ディレクトリ＞ ¥etc** フォルダに置く必要があります。

必要な子要素はこのポイント・ファイルだけです。インストルメンテーション，ログ処理，およびモードは省略可能です。インストルメンテーションの対象または対象外となるメソッドを診断するときは，次のインストルメンテーションの設定を使用できます。

```
    <instrumentation>
       <logging level="points ilasm" />
    </instrumentation>
```

.NET Remoting のインストルメンテーションを設定する方法

.NET プローブを設定して，.NET Remoting のクライアントおよびサーバ・アプリケーションのインストルメンテーションをサポートするカスタム・インストルメンテーションを追加できます。サポートされる設定は次のとおりです。

- HTTP バインドと TCP バインド
- バイナリ形式と SOAP 形式

構成

標準設定では，.NET プローブは Remoting アプリケーションをインストルメントできません。クライアント・アプリケーションとサーバ・アプリケーションの両方のカスタム・インストルメンテーション・ポイントを追加する必要があります。

リモート処理に関しては，次の 2 つのインストルメンテーション・キーワードがあります。

リモート処理 : リモート処理キーワードは，リモート処理フレームワークのさまざまなポイントのインストルメンテーションを可能にします。

RemotingServer : RemotingServer キーワードは，リモート処理メソッドを実装し，そのクラスのメソッドのインストルメンテーションとほかの同じようなメソッドの意図しないインストルメンテーションとを区別するクラスを特定します。

クライアントの例

次に, 非常に簡単な Windows アプリケーションの例を示します。この例を使って, リモート処理クライアント・アプリケーションのインストルメンテーションを設定するときに検討する必要がある重要な概念をいくつか説明します。

```
namespace HPSoftware.AM.Tests.Remoting.SimpleRemoting
{
   class SimpleConsoleClient
   {
      [STAThread]
      static void Main(string[] args)
      {
         const string msg1 = "How are you?";

         String filename =
AppDomain.CurrentDomain.SetupInformation.ConfigurationFile;
         RemotingConfiguration.Configure(filename, false);

         MyRemotableObject remoteObject = new MyRemotableObject();

         doit(remoteObject, myMsg);

         Console.WriteLine();
         Console.WriteLine("(Press any key to exit)");
         Console.ReadKey();
      }

      public static void doit(MyRemotableObject obj, String message)
      {
            Console.WriteLine(obj.GetEnlightenment(message));
      }
}
```

407 ページ「非 ASP.NET アプリケーションまたは Windows アプリケーション用のポイントの設定方法とセットアップ方法」で説明したように, Reflector ユーティリティを使ってリモート処理クライアントのポイント・ファイルの設定方法を簡単に確認できます。

SimpleConsoleClient Remoting Windows アプリケーションをインストルメントするようにプローブを設定するには，次の XML を **probe_config.xml** ファイルに追加します。

```
<process name="SimpleConsoleClient">
        <points file="Remoting.points" />
        <points file="SimpleConsoleClient.points" />
        <instrumentation><logging level="" /></instrumentation>
        <logging level="" />
 </process>
```

<points file="Remoting.points" /> エントリを追加する必要があります。

SimpleConsoleClient.exe が格納されているディレクトリで作業していて，Reflector.exe が PATH に含まれている場合は，コマンド・ラインで Reflector を実行して，SimpleConsoleClient.exe のインストルメンテーションの解読情報と推奨されるポイント・ファイル設定を表示できます。

```
Reflector SimpleConsoleClient.exe
```

このコマンドの出力には次の内容が含まれます。

```
---------------------------------------------------------------------
Sample .points by Namespace
---------------------------------------------------------------------
[HPSoftware.AM.Tests.Remoting.SimpleRemoting]
class = !HPSoftware.AM.Tests.Remoting.SimpleRemoting.*
layer = HPSoftware/AM/Tests/Remoting/SimpleRemoting
---------------------------------------------------------------------
(1 classes)  Namespace: HPSoftware.AM.Tests.Remoting.SimpleRemoting
---------------------------------------------------------------------
HPSoftware.AM.Tests.Remoting.SimpleRemoting.SimpleConsoleClient (8
Methods)
   Equals System.Boolean(System.Object)
   Finalize              System.Void()
   GetHashCode           System.Int32()
   GetType               System.Type()
   doit                  (method signature information unavailable))
   Main                  System.Void(System.String[])
   MemberwiseClone  System.Object()
   ToString              System.String()
```

推奨される SimpleConsoleClient.points は次のとおりです。

```
[HPSoftware.AM.Tests.Remoting.SimpleRemoting]
class = !HPSoftware.AM.Tests.Remoting.SimpleRemoting.*
layer = HPSoftware/AM/Tests/Remoting/SimpleRemoting
```

ただし，これらの設定ではデータを生成するインストルメンテーションが作成されません。その理由は，407 ページ「非 ASP.NET アプリケーションまたは Windows アプリケーション用のポイントの設定方法とセットアップ方法」で説明したように, Main() のようなメソッドを無視する必要があるためです。Main() を無視する必要があることを考慮すると，次のようなポイント・ファイルの設定が考えられます。

```
[HPSoftware.AM.Tests.Remoting.SimpleRemoting]
class = !HPSoftware.AM.Tests.Remoting.SimpleRemoting.*
ignoreMethod = Main
layer = HPSoftware/AM/Tests/Remoting/SimpleRemoting
```

これらの設定は役立つことがあり，データが生成されますが，データの精度を高める必要があります。その理由は，主にプローブのパフォーマンスにあります。インストルメントするメソッドが増えるほど，プローブのパフォーマンスがインストルメンテーション対象アプリケーションに与える影響は大きくなります。たとえば，設定からワイルドカード（!.*）を削除できれば，設定の範囲が明示的になります。

Reflector の出力から，インストルメントされるクラスは実際には次の 1 つだけであることがわかります。

```
HPSoftware.AM.Tests.Remoting.SimpleRemoting.SimpleConsoleClient
```

次のようにして，クラスの設定からワイルドカードを削除できます。

```
class = HPSoftware.AM.Tests.Remoting.SimpleRemoting.SimpleConsoleClient
```

また，Reflector の出力にはメソッドの設定が含まれていません。メソッドの設定がないデフォルトでは，すべてのメソッドがインストルメントされることになります。Equals, Finalize, GetHashCode, GetType, MemberwiseClone, ToString の各メソッドは，そのほとんどが System.Object から継承されたために存在しているだけなので，インストルメントする必要性はほとんどありません。ただし，doit メソッドはリモート処理クライアントの呼び出しをラップするため，インストルメンテーションが必要な場合があります。次の設定は，SimpleConsoleClient.points ファイルに対する推奨設定です。

```
[HPSoftware.AM.Tests.Remoting.SimpleRemoting]
class = HPSoftware.AM.Tests.Remoting.SimpleRemoting.SimpleConsoleClient
method = doit
layer = HPSoftware/AM/Tests/Remoting/SimpleRemoting
```

サーバの例

次に，Windows アプリケーションの例を示します。この例を使って，リモート処理サーバ・アプリケーションのインストルメンテーションを設定するときに検討する必要がある重要な概念をいくつか説明します。

Remoting クライアントとサーバで共有されるリモート処理可能オブジェクトと SimpleConsoleServer.exe Remoting サーバ・アプリケーションの両方の C# コード・スニペットを示します。

次に，リモート処理可能オブジェクトの C# コード・スニペットを示します。

```
HPSoftware.AM.Tests.Remoting.SimpleRemoting
{

    public class MyRemotableObject : MarshalByRefObject
    {
      const string response = "I'm just fine!";

        public MyRemotableObject()
        {
        }

        public String GetEnlightenment(string message)
        {
            return response;
        }
    }
}
```

次に，SimpleConsoleServer.exe の C# コード・スニペットを示します。

```
namespace HPSoftware.AM.Tests.Remoting.SimpleRemoting
{
   class SimpleConsoleServer
   {
      [STAThread]
      static void Main(string[] args)
      {
         String filename =
AppDomain.CurrentDomain.SetupInformation.ConfigurationFile;
         RemotingConfiguration.Configure(filename, false);

         Console.WriteLine("Server is running...    press any key to exit");
         Console.ReadKey();
      }
   }
}
```

SimpleConsoleServer Remoting Windows アプリケーションをインストルメントするようにプローブを設定するには，次の XML を **probe_config.xml** ファイルに追加します。

```
<process name="SimpleConsoleServer">
        <points file="SimpleConsoleServer.points" />
        <instrumentation><logging level="" /></instrumentation>
        <logging level="" />
</process>
```

<points file="Remoting.points" /> エントリを追加する必要はありません。

リモート処理サーバ用のポイント・ファイルには，1 つ以上のセクションを含めることができます。1 つ目のセクションは，リモート処理可能オブジェクトに関するもので，必須のセクションです。2 つ目のセクションは，リモート処理サーバのインストルメンテーションに関するもので，追加できます。その他の省略可能なセクションは，リモート処理メソッドまたはリモート処理サーバによって呼び出される可能性があるほかのメソッドをインストルメントするために，必要に応じて追加できます。まず，リモート処理可能オブジェクトのセクションを作成します。

一部のアセンブリには，リモート処理可能オブジェクトが含まれています。ここでは，RemotableObjects.dll 内にあると想定します。

RemotableObjects.dll に対して Reflector を実行すると, 次の内容を含む出力が表示されます。

```
---------------------------------------------------------------------
Sample .points by Namespace
---------------------------------------------------------------------
[HPSoftware.AM.Tests.Remoting.SimpleRemoting]
class = !HPSoftware.AM.Tests.Remoting.SimpleRemoting.*
layer = HPSoftware/AM/Tests/Remoting/SimpleRemoting

---------------------------------------------------------------------
(1 classes)  Namespace: HPSoftware.AM.Tests.Remoting.SimpleRemoting
---------------------------------------------------------------------
HPSoftware.AM.Tests.Remoting.SimpleRemoting.MyRemotableObject (17
Methods)
   __RaceSetServerIdentitySystem.Runtime.Remoting.ServerIden…)
   __ResetServerIdentity              System.Void()
   CanCastToXmlType                   System.Boolean(System.String,System…)
   CreateObjRef                       System.Runtime.Remoting.ObjRef(Syste…)
   Equals                             System.Boolean(System.Object)
   Finalize                           System.Void()
   GetComIUnknown                     System.IntPtr(System.Boolean)
   GetEnlightenment                   System.String(System.String)
   GetHashCode                        System.Int32()
   GetLifetimeService                 System.Object()
   GetType                            System.Type()
   InitializeLifetimeService          System.Object()
   InvokeMember                       System.Object(System.String,System…)
   IsInstanceOfType                   System.Boolean(System.Type)
   MemberwiseClone                    System.MarshalByRefObject(System…)
   MemberwiseClone                    System.Object()
   ToString                           System.String()
```

Remoting クライアントの例については, 推奨されるポイント設定をそのままでは使用できません。リモート処理可能オブジェクトをインストルメントするクラスを確実に特定する必要があります。そのためには，System.MarshalByRefObject から継承するのにリモート処理可能オブジェクトが必要であり，したがって CreateObjRef，GetLifetimeService，InitializeLifetimeService，MemberwiseClone の各メソッドがオブジェクトに含まれている必要があります。上記の Reflector の出力から，明らかに HPSoftware.AM.Tests.Remoting.SimpleRemoting.MyRemotableObject クラスがリモート処理可能オブジェクトをインストルメントするクラスの候補であることがわかります。

リモート処理可能オブジェクトのセクションには，**keyword = RemotingServer** エントリを含める必要があります。このエントリは，プローブの Instrumenter がこのセクションのポイント設定用の特別な処理を実行することを示しています。この特別な処理によって，2 つのことが行われます。まず，System.MarshalByRefObject から継承したクラスのすべてのメソッドがインストルメントされます。したがって，インストルメントするリモート処理メソッドを指定する必要はありません。すべてのリモート処理メソッドがインストルメントされます。このため，このセクションにメソッドのエントリは必要ありません。次に，このキーワードによって，System.MarshalByRefObject から継承したクラスでインストルメントされるメソッドのインストルメンテーションが，指定したクラスに限定されます。このことが重要なのは，System.MarshalByRefObject から継承する System クラスやユーザ・クラスがほかにも数多くあり，それらを意図せずにインストルメントしたくないためです。

以上の観察結果に基づいて，推奨されるリモート処理可能オブジェクトのセクションを次に示します。

```
[RemotableObject]
keyword = RemotingServer
class = HPSoftware.AM.Tests.Remoting.SimpleRemoting.MyRemotableObject
layer = RemotableObject
```

今度は，省略可能なリモート処理サーバのセクションを作成できます。このセクションを作成する必要があるのは，リモート処理メソッドとは無関係に呼び出されるサーバ・ロジックを監視する場合だけです。

SimpleConsoleServer.exe に対して Reflector を実行すると，次の内容を含む出力が表示されます。

```
---------------------------------------------------------------------
Sample .points by Namespace
---------------------------------------------------------------------
[HPSoftware.AM.Tests.Remoting.SimpleRemoting]
class = !HPSoftware.AM.Tests.Remoting.SimpleRemoting.*
layer = HPSoftware/AM/Tests/Remoting/SimpleRemoting


---------------------------------------------------------------------
(1 classes)  Namespace: HPSoftware.AM.Tests.Remoting.SimpleRemoting
---------------------------------------------------------------------
HPSoftware.AM.Tests.Remoting.SimpleRemoting.SimpleConsoleServer (7
Methods)
   Equals              System.Boolean(System.Object)
   Finalize            System.Void()
   GetHashCode         System.Int32()
   GetType             System.Type()
   Main                System.Void(System.String[])
   MemberwiseClone     System.Object()
   ToString            System.String()
```

407 ページ「非 ASP.NET アプリケーションまたは Windows アプリケーション用のポイントの設定方法とセットアップ方法」で説明したように，推奨されるポイント設定をそのままでは使用できません。Main() メソッドを無視する必要があります。

以上の観察結果に基づいて，SimpleConsoleServer.points ファイルに対する推奨設定を次に示します。

```
[RemotableObject]
keyword = RemotingServer
class = HPSoftware.AM.Tests.Remoting.SimpleRemoting.MyRemotableObject
layer = RemotableObject

[RemotingServer]
class = HPSoftware.AM.Tests.Remoting.SimpleRemoting.SimpleConsoleServer
ignoreMethod = Main
layer = RemotingServer
```

その他のセクションは，リモート処理メソッドまたはリモート処理サーバによって呼び出される可能性があるほかのメソッドをインストルメントするために，必要に応じて追加できます。

カスタム・インストルメンテーションのオーバーヘッドについて

カスタム・インストルメンテーションを作成する際，調べるアプリケーションのレイテンシが極めて長くなる可能性があるため，アプリケーションの過インストルメンテーションに注意する必要があります。バイトコード量が常にほとんど同じであるため，インストルメンテーションのオーバーヘッドはすべてのメソッドでほぼ修正され，カスタム・インストルメンテーションはメソッドのレイテンシや CPU のオーバーヘッドに同様の影響を与えません。CPU およびレイテンシのオーバーヘッドにおける物理的な割合は，実行に要するメソッドの時間に正比例して変化します。

たとえば，メソッドで 100ms かかり，インストルメンテーションを 101ms で実行する場合，オーバーヘッドは 1% になります。メソッドで 10 ms かかり，インストルメンテーションの応答が 11 ms に変わると，オーバーヘッドは 10% になります。このメソッドが頻繁に呼び出されない場合，アプリケーションに対する全体のレイテンシは最小限になります。ただし，インストルメンテーション対象のメソッドが頻繁に呼び出されると，そのオーバーヘッドの割合が非常に小さくても，全体のレイテンシがアプリケーションの応答のレイテンシに影響を与えることがあります。

呼び出されたどのメソッドのプロファイルも作成できる従来のプロファイラと異なり，HP Diagnostics はバイトコード・インストルメンテーションを採用しています。これにより，インストルメンテーションによるオーバーヘッドを平均 3 ～ 5% に最小化するために，デフォルトのインストルメンテーションを選択できます。インストルメンテーションによるレイテンシのオーバーヘッドが高いメソッドは，アプリケーションのほかのコンポーネントに関連してときどき呼び出される場合や，インストルメンテーションが優先順位付けに必要な情報を提供する場合にのみ，インストルメントされます。

アプリケーションのインストルメンテーションをカスタマイズする際は，Diagnostics データのオーバーヘッドも考慮する必要があります。インストルメントするメソッドが多いほど，プローブがシリアル化し，ネットワークを通じて Diagnostics サーバに渡すデータの量が増えます。監視中のシステムのパフォーマンスに不要な影響を与えないように，プローブの標準設定を変更してプローブに Diagnostics のデータ量を調整できます。プローブの設定を不適切に変更すると，プローブがある物理マシンで CPU，メモリ，およびネットワークのオーバーヘッドが生じる可能性があります。レイテンシ，CPU，メモリおよびネットワークのオーバーヘッドについては，第 14 章，「.NET Agent の詳細設定」を参照してください。

標準 .NET アプリケーションの標準設定レイヤ

HP Diagnostics では，ポイント・ファイルに指定されている命令に基づいて，クラスおよびメソッドのパフォーマンス・メトリックスを *レイヤ*および*サブレイヤ*にグループ化します。類似のシステム・リソースを使用されるアプリケーションで処理するためにパフォーマンス測定値を一緒に報告できるよう，デフォルトのレイヤが定義されています。レイヤは，パフォーマンスの問題が継続する可能性のあるシステムのエリアを隔離および特定しやすくします。

次の表に，標準 .NET アプリケーションで定義された標準設定のレイヤおよびサブレイヤを示します。

.NET レイヤ

レイヤ	サブレイヤ	親レイヤ
Web Tier	IIS	
IIS	ExecutionSteps	
Database	ADO	
ADO	Execute Connection Fill Update Cache	Database
Messaging	Sender Receiver	
Web Services	Soap Http WCF	
LWMD		
HTTP Client		
Outbound Calls		

11

Diagnostics サーバの詳細設定

本項では，Diagnostics サーバの詳細設定について説明します。詳細設定は，この製品に関する深い知識を有する熟練ユーザを対象にしています。コンポーネントのプロパティを変更する際は，十分に注意してください。

本章の内容

- Business Service Management のサンプル・キュー・サイズと Web サービス CI の頻度の設定（452 ページ）
- Diagnostics サーバ設定ページを使用した Diagnostics の設定（453 ページ）
- 実運用環境で処理できるプローブを増やすための Diagnostics サーバの最適化（453 ページ）

Diagnostics コンポーネント間の時刻の同期

Diagnostics データを正しく保存および関連付けるために，Diagnostics コンポーネント間で時刻を同期させることは不可欠です。データの同期化を容易にするために，Diagnostics データは，Diagnostics サーバ（Commander モード）の同期された GMT 時刻に調整および保存されます。同期化によって，UI が特定できるローカル時刻でデータを正しく表示できるようになります。

次の項では，時刻の同期の機能方法，および時刻が同期されるようにコンポーネントを正しく設定する方法について説明します。

VMware ホストで実行するプローブ収集には，時間の同期化に関する追加の要件があります。詳細については，468 ページ「VMware 上で実行中のプローブの時刻の同期」を参照してください。

時刻の同期について

Diagnostics の時刻の同期は Diagnostics コマンド・サーバから始め，その時刻と，指定の**時刻ソース**の GMT 時刻の間の誤差を特定します。使用する**時刻ソース**は，**＜Diagnostics サーバのインストール・ディレクトリ＞/etc/server.properties** の **timemanager.time_source** プロパティを使って設定されます。

timemanager.time_source プロパティの有効な値は次のとおりです。

- **NTP :** NTP サーバを GMT 時刻のソースとして使用することを示します。これはデフォルトの値です。

  使用する NTP サーバは，**＜Diagnostics サーバのインストール・ディレクトリ＞/etc/server.properties** の **timemanager.ntp.servers** プロパティの値としてリストされます。

注 : リストのいずれかの NTP サーバに，Diagnostics サーバから接続できることを確認してください。接続できない場合は，リストの最初のサーバとしてローカル NTP サーバを追加してください。

- **BAC :** 登録した Business Service Management ゲートウェイ・サーバを GMT 時刻のソースとして使用することを示します。

注 : Business Service Management が Database 時刻に設定されている場合，この設定を時刻ソースに使うように Diagnostics コマンド・サーバを設定する必要もあります。

- **SERVER :** Diagnostics コマンド・サーバを**時刻ソース**に使用することを示します。

  これは，Diagnostics サーバをスタンドアロン・モードで使用している場合のみ使います。

Mediator モードの Diagnostics サーバは，Diagnostics サーバ（Mediator モード）と Diagnostics サーバ（Commander モード）の間の時間差を確認して時刻を同期します。

Diagnostics サーバ（Commander モード）が**時刻ソース**と同期しなかった場合は，Diagnostics サーバ（Mediator モード）が「非同期」とみなされます。同期していない Diagnostics サーバ（Mediator モード）は，同期に成功するまで 15 秒おきに同期を試行します。

Diagnostics プローブが Diagnostics サーバ（Mediator モード）または Diagnostics サーバ（Commander モード）に接続するときに，Diagnostics サーバとプローブの時間差が特定されます。

プローブが非同期の Diagnostics サーバに接続しようとすると，プローブ接続が許可されず，ドロップされます。

データは GMT に基づいて保存されるため，さまざまなコンポーネントにとってタイムゾーンやサマータイムの差は問題ではありません。たとえば，Diagnostics UI に表示されるデータは，UI を実行しているタイムゾーンで正しく表示されるように調整されます。

注 : データはすべて調整され，Diagnostics サーバ（Commander モード）の同期された GMT 時刻に保存されます。したがって，**時刻ソース**と正しく同期されていないマシンで UI が実行している場合，その UI に表示されるデータは，UI マシンの時刻と同期された GMT 時刻の時間差の分だけ変更されて表示されます。

Diagnostics サーバでの時刻の同期の設定

Diagnostics コマンダ・サーバを同期するには，次の手順を実行します。

注 : Diagnostics サーバ（Mediator モード）の時刻は，Diagnostics サーバ（Commander モード）に自動的に同期されるため同期設定は不要です。

Diagnostics サーバ（Commander モード）の時刻を同期できるようにするには，次の手順を実行します。

1 Diagnostics サーバのデフォルト設定は，**＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/server.properties** の **timemanager.time_source** プロパティの値は **NTP** などに設定されます。

Diagnostics サーバがインターネットに接続していて，**timemanager.ntp.servers** プロパティに指定されている使用可能な NTP サーバ・リストのサーバに接続できる場合は，標準設定が機能するため，変更は不要です。

Business Service Management は標準で時刻の同期に NTP を使用するため，標準設定を推奨します。

2 Diagnostics サーバがインターネットに接続していないため，**timemanager.ntp.servers** プロパティに指定されている使用可能な NTP サーバ・リストに接続できない場合は，次のいずれかを実行する必要があります。

➤ Diagnostics サーバ（Commander モード）が接続可能なローカル NTP サーバをセットアップします。**＜ Diagnostics サーバのインストール・ディレクトリ＞/etc/server.properties** の **timemanager.ntp.servers** で，最初のエントリとしてこの NTP サーバをリストします。

注 : プライマリ NTP サーバが使用できない場合のために，NTP サーバをバックアップしてください。

➤ Business Service Management または HP Software as a Service（SaaS）環境で Diagnostics を使用している場合，**＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/server.properties** の **timemanager.time_source** プロパティを **BAC** に設定して Business Service Management を指定できます。これにより, Diagnostics サーバが登録されている Business Service Management コア・サーバに接続して時刻を確立できるようになります。

注 : Diagnostics を使用するように Business Service Management をセットアップする方法については，第21 章,「Business Service Management と Diagnostics 間の統合のセットアップ」を参照してください。

➤ Diagnostics サーバ（Commander モード）を Business Service Management で使わずにスタンドアロン・モードで使う予定があり，かつインターネット接続がなく NTP サーバと時刻を同期できない場合は，**＜Diagnostics サーバのインストール・ディレクトリ＞/etc/server.properties** の **timemanager.time_source** プロパティを **SERVER** に設定できます。これにより，Diagnostics サーバは，**時刻ソース**として自らの時刻を使用するようになります。

注 : データが指定の**時刻ソース**を使ってキャプチャおよび保持されたら，**＜Diagnostics サーバのインストール・ディレクトリ＞/etc/server.properties** の **timemanager.time_source** プロパティの値を変更しないことをお勧めします。データがキャプチャされた後に**時刻ソース**を変更すると，キャプチャおよび保持されたデータに重大な破損が生じる恐れがあります。これは，保持されているデータが，Diagnostics サーバが GMT と同期していなかった期間にキャプチャされたものであることが原因です。後で Diagnostics サーバと GMT が同期するようになってからキャプチャされたデータは，何度も再集計したり，データが属さない時刻バケットにデータを記録したりできます。

大規模インストールの Diagnostics サーバの設定

20 以上のプローブで Diagnostics サーバ（Mediator モード）を使用している場合，Diagnostics サーバのデフォルト設定に変更を加えることをお勧めします。

注 : Diagnostics サーバ（Commander モード）では，Diagnostics サーバ（Mediator モード）として機能するようにプローブを割り当てていない限り，設定を変更する必要はありません。

ヒープ・サイズの調整

ヒープのサイズは，Diagnostics サーバ（Mediator モード）のパフォーマンスに影響する可能性があります。ヒープが小さすぎると，Diagnostics サーバ（Mediator モード）が一定期間ハングすることがあります。ヒープが大きすぎると，Diagnostics サーバ（Mediator モード）でガベージ・コレクションが長時間遅れることがあります（特に，複数 CPU / コアまたは高速 CPU などの十分な CPU リソースを使用できない場合）。

ヒープ・サイズの標準設定値は 512 MB です。ヒープ・サイズは，次の場所にある **server.nanny** ファイルで設定します。Windows では **<Diagnostics サーバのインストール・ディレクトリ>¥nanny¥windows¥dat¥nanny¥**，Solaris では **<Diagnostics サーバのインストール・ディレクトリ>/nanny/solaris/launch_service/dat/nanny/**

次の VM 引数を使って，サイズを設定します。**???** は MB のサイズを表します。

-Xmx???m

Diagnostics サーバ（Mediator モード）が「ハング」する問題が発生した場合，**-Xmx???m** オプションで指定した値を更新して，指定したヒープ・サイズを大きくできます。

Diagnostics サーバ（Mediator モード）のヒープ・サイズを調整するには，次の手順を実行します。

1 次の表に従って，Diagnostics サーバ（Mediator モード）で必要なヒープの量を決めます。

プローブの数	推奨ヒープ・サイズ
Java プローブ 50 台以内	512 MB
Java プローブ 100 台以内	1400 MB
Java プローブ 200 台以内	3000 MB（64 ビット）
.NET プローブ 10 台以内	350 MB
.NET プローブ 20 台以内	700 MB
.NET プローブ 50 台以内	1400 MB

注 : 推奨ヒープ・サイズは，マシンの物理メモリの 75% 以内です。マシンに 1 GB ある場合，ヒープ・サイズは 768 MB 以内にします。

---

3 つ以上の CPU またはコア（4 つのコアを推奨）が搭載されたシステムで Diagnostics を実行することを強くお勧めします。このような環境では，-XX:+UseConcMarkSweepGC のように Garbage Collector を同時マーク / スイープに変更します。

64 ビットの JVM では，次のオプションが有効であることを確認します。

-XX:+UseCompressedOops

VMware のインストールでは，VMware の『Enterprise Java Applications on VMware Best Practices Guide』に記載されているベスト・プラクティスに従います。本質的には，複数の vCPU と固定メモリ割り当て（バルーン化またはディスクへのスワップなし）を使用し，VMware Tools を確実にインストールします。

2 編集する **server.nanny** ファイルを開きます。ファイルは次の場所にあります。

Windows では**＜Diagnostics サーバのインストール・ディレクトリ＞¥nanny¥windows¥dat¥nanny¥**，Solaris では**＜Diagnostics サーバのインストール・ディレクトリ＞/nanny/solaris/launch_service/dat/nanny/**。

3 適切な start_＜プラットフォーム＞ 行で，**-Xmx???m** オプションで指定したヒープ・サイズを計算した最適なヒープ・サイズに変更します。

-Xmx???m

??? で表されている現在のヒープ・サイズを 768 MB に置き換えます。

-Xmx768m

server.nanny ファイルでこの行を変更する前は，次のように表示されます。

```
start_nt="C:¥MercuryDiagnostics¥Server¥jre¥bin¥javaw.exe" -server -Xmx512m
-Dsun.net.client.defaultReadTimeout=70000
-Dsun.net.client.defaultConnectTimeout=30000
"-javaagent:C:¥MercuryDiagnostics¥Server¥probe¥lib¥probeagent.jar"
-classpath "C:¥MercuryDiagnostics\Server¥lib¥mediator.jar;
C:\MercuryDiagnostics¥Server¥lib¥loading.jar;
C:\MercuryDiagnostics¥Server¥lib¥common.jar;
C:\MercuryDiagnostics¥Server¥lib¥mercury_picocontainer-1.1.jar"
com.mercury.opal.mediator.util.DiagnosticsServer
```

server.nanny ファイルでこの行を変更した後は，次のように表示されます。

```
start_nt="C:¥MercuryDiagnostics¥Server¥jre\bin\javaw.exe" -server -Xmx768m
-Dsun.net.client.defaultReadTimeout=70000
-Dsun.net.client.defaultConnectTimeout=30000
"-javaagent:C:¥MercuryDiagnostics¥Server¥probe¥lib¥probeagent.jar"
-classpath "C:¥MercuryDiagnostics¥Server¥lib¥mediator.jar;
C:\MercuryDiagnostics¥Server¥lib¥loading.jar;
C:\MercuryDiagnostics¥Server¥lib¥common.jar;
C:\MercuryDiagnostics¥Server¥lib¥mercury_picocontainer-1.1.jar"
com.mercury.opal.mediator.util.DiagnosticsServer
```

プローブから取得するデータ量の調整

大規模な呼び出しプロファイルでは，プローブとサーバの間にかなりのネットワーク帯域幅が必要であり，サーバにもかなりの CPU リソースが必要です。

ネットワークがボトルネックになっている場合（Windows のタスク・マネージャに表示される Mediator のネットワーク使用率が 25% を超えている場合や，プローブが起動しても 100% を下回る可用性を報告する場合など）は，トリミングによって生成されるデータを減らしたり，プローブ・システムの CPU が十分に利用されていない場合は圧縮を有効にしたりする必要があります。また，サーバがプローブからデータを取得する頻度を減らすこともできます。

プローブの主なトリミング・パラメータを次に示します。

- **capture.properties** ファイル：
  - maximum.stack.depth = 25
  - maximum.method.calls = 1000（たとえば 25 に設定して，呼び出しプロファイル内のメソッドの総数を制限できます）
  - minimum.method.latency = 51ms
- **dispatcher.properties** ファイル：
  - minimum.fragment.latency = 51ms（たとえば，101 ms まで増やすことができます）。ただし，標準設定ではトリミングにより合成トランザクション（BPM/vugen/LoadRunner/Performance Center）が影響を受けることはないため，これらすべてのサーバ要求が報告される点に注意してください。

トリミングの詳細については，サーバの場合は 441 ページ「サーバ要求名に基づくトリミングの設定」，.NET Agent の場合は 597 ページ「レイテンシのトリミングおよびスロットリングの設定」と602 ページ「深さのトリミングの設定」，Java エージェントの場合は464 ページ「Agent におけるメソッドの自動トリミングの制御」を参照してください。

圧縮を有効にするには，プローブで **webserver.properties: rhttp.gzip.replies = true** を設定します。これにより，サーバ上のネットワーク・トラフィックが大幅に減少します。ただし，プローブ（およびサーバ）には圧縮用の追加の CPU が必要です。

ネットワーク・トラフィックを減少させるには，プローブからデータを取得する頻度を変更する方法もあります。標準設定では，トレンドを 5 秒ごとに取得し，ツリー（呼び出しプロファイル）を 45 秒ごとに取得します。呼び出しツリーの頻度を下げるには，**server.properties** ファイルに含まれる Mediator の **probe.trees.pull.interval** を変更します（たとえば，呼び出しプロファイルに含まれるメソッドの数に応じて 90 秒や 240 秒などにします）。最初に呼び出しツリーの取得頻度を小さくします。それでも十分でない場合は，**probe.trends.pull.interval** を 10 秒などに変更して，トレンド取得頻度を小さくします。

これらのパラメータのいずれかを変更した場合は，プローブまたはサーバを再起動する必要があります。

追加の調整

50 個を超えるプローブが接続されている場合は，プローブからのデータ取得に使用するスレッドの数を増やします。メディエータごとに，**probe.pull.max.threads=30** を設定し，サーバを再起動します。

さらに，**webserver.properties, jetty.threads.max=500** を設定して，Jetty に使用できるスレッドの数を増やすこともできます。

呼び出しツリーとトレンドのファイル（付録 E,「Diagnostics のデータ管理」も参照）が非圧縮状態で非常に大きくなった（4 GB を超えた）場合は，一部のプローブの負荷を新しい Mediator に移動することをお勧めします。移動しない場合，大量のデータのためにファイルの集計と圧縮が遅れ始めるおそれがあります。

1 つのサーバに多くのプローブが接続されている場合は，標準設定のパージ設定（5 GB）では十分でない可能性があります。詳細については，834 ページ「データの保存」を参照してください。

デフォルトの Diagnostics サーバ・ホスト名の変更

ファイアウォールまたは NAT が機能している場合，または Diagnostics サーバ（Mediator モード）のホストがマルチホーム・デバイスに設定されている場合は，Diagnostics サーバ（Mediator モード）をインストールしたときに割り当てたホスト名を使って，Diagnostics サーバ（Commander モード）が Diagnostics サーバ(Mediator モード)と通信できないことがあります。**registered_hostname** プロパティを使って，Diagnostics サーバ（Mediator モード）が Diagnostics サーバ（Commander モード）への登録に使う標準設定のホスト名を変更できます。

Diagnostics サーバ（Mediator モード）の標準設定のホスト名を変更するには，**＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/server.properties** にある **registered_hostname** プロパティを，Diagnostics サーバ（Commander モード）が Diagnostics サーバ（Mediator モード）と通信できるようになる代替マシンの名前または IP アドレスに設定します。

デフォルトの Diagnostics サーバ・ポートの変更

Diagnostics サーバのホスト設定で，標準設定の Diagnostics ポートを使用できない場合は，Diagnostics サーバとプローブおよびほかの Diagnostics サーバとの通信に別のポートを選択します。

重要 : 新しいポート番号が別のアプリケーションに使われていないこと，およびほかの Diagnostics コンポーネントがこのポートと通信できることを確認します。

Diagnostics を配備した後でほかのポート番号の使用を決めた場合は，デプロイメント内の指定コンポーネントごとに，次の表のプロパティを新しいポート番号で更新して，正常に通信できるようにする必要があります。

コンポーネント・タイプ	プロパティ
Diagnostics コマンド・サーバ	**<Diagnostics サーバのインストール・ディレクトリ>/etc** ➤ webserver.properties - jetty.port ➤ server.properties ñ commander.url ➤ probe/etc/dispatcher.properties - registrar.url
Diagnostic メディエータ・サーバ	**<Diagnostics サーバのインストール・ディレクトリ>/etc** ➤ server.properties ñ commander.url **<Diagnostics サーバのインストール・ディレクトリ>/probe/etc** ➤ dispatcher.properties - registrar.url
プローブ	**<プローブのインストール・ディレクトリ>/etc** ➤ dispatcher.properties - registrar.url

あるホストから別のホストへの Diagnostics サーバの移行

次の手順では，あるホストから別のホストに Diagnostics サーバを移行する方法示します。新しいホスト1名は，古いホスト名と異なっていることが前提となります。

あるホストから別のホストに Diagnostics サーバを移行するには，次の手順を実行します。

1. プロセス・リストに java/javaw プロセスがないことを確かめて，既存の Diagnostics サーバがシャットダウンしたことを確認します。この場合，Windows システムではタスク・マネージャを，UNIX システムでは ps をそれぞれ使用します。
2. Diagnostics コマンダ・サーバの登録を Business Service Management から解除します。
3. 新しいホストで新しい Diagnostics サーバをインストールします。
4. Windows では，インストーラが完了すると Diagnostics サーバが自動的に起動するため，Diagnostics サーバをシャットダウンする必要があります。

UNIX では，サーバは自動的に起動しないため，シャットダウンする必要はありません。

プロセス・リストに java/javaw プロセスがないことを確かめて，Diagnostics サーバがシャットダウンしたことを確認します。この場合，Windows システムではタスク・マネージャを，UNIX システムでは ps をそれぞれ使用します。

古い Diagnostics サーバのホスト名を把握しておく必要があります（この名前は /archive ディレクトリで見つけることができます)。

5 新しい Diagnostics サーバで＜Diagnostics サーバのインストール・ディレクトリ＞/archive ディレクトリを削除します。

6 ＜Diagnostics サーバのインストール・ディレクトリ＞/archive フォルダとすべてのサブフォルダを，古いサーバから新しいサーバの＜Diagnostics サーバのインストール・ディレクトリ＞/ にコピーします。

7 新しい Diagnostics サーバのホスト名が古い Diagnostics サーバのホスト名と異なっている場合は，＜Diagnostics サーバのインストール・ディレクトリ＞/archive/mediator- ＜ホスト名＞の名前を変更して，＜ホスト名＞に新しい Diagnostics サーバのホスト名を反映させる必要があります。たとえば，古いホスト名が oldhost で，新しいホスト名が newhost の場合は，次のように変更できます。

＜Diagnostics サーバのインストール・ディレクトリ＞/archive/mediator-＜oldhost＞から＜Diagnostics サーバのインストール・ディレクトリ＞/archive/mediator-＜newhost＞

8 新しい Diagnostics サーバで＜Diagnostics サーバのインストール・ディレクトリ＞/storage/ ディレクトリを削除します。

9 ＜ Diagnostics サーバのインストール・ディレクトリ＞ /storage/ フォルダとすべてのサブフォルダを，古いサーバから新しいサーバの＜Diagnostics サーバのインストール・ディレクトリ＞/ にコピーします。

10 新しいサーバの＜Diagnostics サーバのインストール・ディレクトリ＞/storage/server- ＜ホスト名＞の名前を変更して，＜ホスト名＞に新しい Diagnostics サーバのホスト名を反映させます。たとえば，古いホスト名が oldhost で，新しいホスト名が newhost の場合は，次のように変更できます。

＜Diagnostics サーバのインストール・ディレクトリ＞/storage/server-＜oldhost＞から＜Diagnostics サーバのインストール・ディレクトリ＞/storage/server-＜newhost＞

11 ＜Diagnostics サーバのインストール・ディレクトリ＞/etc フォルダを，古いサーバから新しいサーバの＜ Diagnostics サーバのインストール・ディレクトリ＞/ にコピーし，新しいライセンスを etc フォルダにコピーします。

12 新しい Diagnostics サーバを起動し，Business Service Management で登録します。

13 新しいサーバが Commander の場合は，すべての Mediator で etc フォルダをスキャンし，古いサーバ名を新しいサーバ名に変更する必要があります。dispatcher.properties ファイルを再チェックし，コマンダ・サーバのホスト名が変更されたことを確認します。その後，すべての Mediator を再起動します。

プローブからコマンダ・サーバに直接レポートしているか，プローブが接続されているメディエータ・サーバを移行している場合を除き，プローブ側では変更は必要ありません。変更が必要な場合は，プローブ・システムで etc フォルダをスキャンし，古いサーバ名を新しいサーバ名に変更します（dispatcher.properties ファイルを再チェックし，メディエータ・サーバのホスト名が変更されたことを確認します)。

マルチホーム環境用の Diagnostics サーバの設定

Diagnostics サーバのホスト・マシンに，複数のネットワーク・インタフェース・カード（NIC）を装着できます。Diagnostics サーバ・プロセスは，ホスト上のすべてのインタフェースでリッスンします。カスタマ環境によっては，マシンのすべてのネットワーク・インタフェースでアプリケーションがリスンできないことがあります。お使いの環境にこのような制限がある場合は，次の手順を使って，Diagnostics サーバが特定のネットワーク・インタフェースをリスンするように設定します。

イベント・ホスト名の設定

Diagnostics サーバ・ホストに複数のネットワーク・インタフェースがあり，Diagnostics サーバでリッスンするホスト名を指定する場合，**event.hostname** プロパティを設定する必要があります。

このプロパティは次の場所にあります。

＜Diagnostics サーバのインストール・ディレクトリ＞/etc/server.properties

プロパティ **event.hostname** はコメントアウトせず，ホスト名の値を指定します。

デフォルトで，**event.hostname** プロパティは設定されていません。したがって，Diagnostics サーバはすべてのホスト名をリスンします。

jetty.xml ファイルの変更

jetty.xml ファイルには，Diagnostics サーバのリッスンが許可されるインタフェースを定義する部分があります。標準設定では，Diagnostics サーバ付属の **jetty.xml** にリッスンは定義されていません。Diagnostics サーバはすべてのインタフェースをリッスンします。

マシンの特定のネットワーク・インタフェースでリスンする Diagnostics サーバを設定するには，次の手順を実行します。

1 **＜Diagnostics サーバのインストール・ディレクトリ＞/etc/jetty.xml** を開いて，次の行を見つけます。

<Configure class="org.mortbay.jetty.Server">

2 この行の後に次のコード・ブロックを追加し，＜Set name="Host"＞.....＜/Set＞を変更して NIC の IP アドレスを含めます。

```
<Call name="addListener">
   <Arg>
     <New class="org.mortbay.http.SocketListener">
       <Set name="Host">127.0.0.1</Set>
       <Set name="Port"><SystemProperty name="jetty.port" default="2006"/></Set>
       <Set name="MinThreads">1</Set>
       <Set name="MaxThreads">5</Set>
       <Set name="MaxIdleTimeMs">30000</Set>
       <Set name="LowResourcePersistTimeMs">5000</Set>
       <Set name="ConfidentialPort">8443</Set>
       <Set name="IntegralPort">8443</Set>
    </New>
   </Arg>
 </Call>
```

3 前の手順を繰り返し，コード・ブロックの新しいコピーを追加して，Diagnostics サーバでリスンする各インタフェースの NIC の IP アドレスを設定します。

</Configure> タグが最後の NIC のリスナ・コードの後にくるようにしてください。

注 : Diagnostics サーバにアクセスするコンポーネントが，Diagnostics サーバのホスト名を **jetty.xml** ファイルでホスト値に指定した IP アドレスに解決できるようにしてください。一部のシステムでは，ホスト名を Diagnostics サーバ・ホストのほかの IP アドレスに解決できます。詳細については，433 ページ「デフォルトの Diagnostics サーバ・ホスト名の変更」を参照してください。

---

jetty.xml のサンプル・ファイル

次の例は，Diagnostics サーバがループバックでリスンする Diagnostics サーバの **jetty.xml** ファイルと，システムの IP アドレスを示します。

```
<!-- Configure the Jetty Server                                             -->
<!-- ======================================================= -->
<Configure class="org.mortbay.jetty.Server">
<!--======================================================= -->
<!-- Configure the Request Listeners                                        -->
  <!--======================================================= -->
<Call name="addListener">
   <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Host">127.0.0.1</Set>
      <Set name="Port"><SystemProperty name="jetty.port" default="2006"/></Set>
      <Set name="MinThreads">1</Set>
      <Set name="MaxThreads">5</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
      <Set name="ConfidentialPort">8443</Set>
      <Set name="IntegralPort">8443</Set>
    </New>
   </Arg>
 </Call>

<-Listen on specific IP Address on this machine for incoming Commander calls->
<Call name="addListener">
  <Arg>
   <New class="org.mortbay.http.SocketListener">
     <Set name="Host">10.241.3.109</Set>
     <Set name="Port"><SystemProperty name="jetty.port" default="2006"/></Set>
      <Set name="MinThreads">1</Set>
      <Set name="MaxThreads">5</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
      <Set name="ConfidentialPort">8443</Set>
      <Set name="IntegralPort">8443</Set>
    </New>
  </Arg>
 </Call>
</Configure>
```

Diagnostics サーバのメモリ使用量の削減

トランザクション・タイムアウト時間は，Diagnostics サーバが長期間古いデータを保持することでメモリを使いすぎないようにするための安全機能です。Diagnostics サーバは，Diagnostics サーバにトランザクションが完了したことを伝えるトランザクション完了通知（ELT : End of Transaction Notification）を受け取るまで，受信したトランザクションの情報をすべて保持します。トランザクションのタイムアウト時間は，Diagnostics サーバがトランザクションのデータを受信するたびにリセットされます。

Diagnostics サーバ（Commander モード）が実行されているマシンが過負荷になっている場合（CPU に負荷がかかりすぎているか，または 1 秒あたりに処理するトランザクションが多すぎる）や, Load Generator または Business Service Management と Diagnostics コマンド・サーバの間，または Business Process Monitor と Business Service Management の間にネットワーク接続の問題がある場合は，トランザクションが完了したときに通知する ELT を Diagnostics サーバが受信できないことがあります。トランザクション・タイムアウト時間が切れるまでに ELT を受信しないと，Diagnostics サーバは，ELT が送信されていないと判断し，トランザクションのデータ処理に進み，トランザクション・データが使用しているメモリを解放します。

correlation.txn.timeout プロパティでは，トランザクション・タイムアウト時間の期間を設定します。Diagnostics サーバでメモリ不足が生じる場合は，トランザクションの完了までに Diagnostics サーバが待機する時間を短縮するために，トランザクション・タイムアウト時間を短くすることがあります。このプロパティの値を調整する場合は，注意してください。複数のプローブが Diagnostics サーバにデータを送信していて，アクティブなトランザクションが特定の Diagnostics サーバでアイドルになっていることがあります。このプロパティの設定値が小さすぎると，トランザクションが正しく報告されないことがあります。このプロパティの値を小さくする必要がある場合は，テストで一番長いトランザクションよりも 90 秒長い値を設定してください。

サーバ要求名に基づくトリミングの設定

サーバ要求名に基づくトリミングを使うと，プローブの設定やインストルメンテーションを変更せずに，Diagnostics サーバのパフォーマンス問題の原因とみられるサーバ要求をフィルタリングするように Diagnostics を設定できます。

注 : サーバ要求名に基づくトリミングは，プローブで設定する遅延や深さのトリミングの代わりには使用できません。

＜ Diagnostics サーバのインストール・ディレクトリ＞¥etc¥trimming.properties ファイルの **trim.fragment** プロパティを使って，Diagnostics でトリミングするサーバ要求フラグメントの名前を指定できます。Diagnostics は，リアル・ユーザおよび仮想ユーザ両方のサーバ要求のフラグメントをトリミングします。

標準設定では，**trim.fragment.1** および **trim.fragment.2** は，**trimming.properites** でコメント・アウトされています。トリミングするフラグメントを指定するには，プロパティの 1 つのコメント・アウトを解除して，Diagnostics ビューに一覧表示されたときにトリミングするフラグメントの名前を入力します。3 つ以上のフラグメントをトリミングする必要がある場合は，追加の **trim.fragment** プロパティを作成します。最後に番号をインクリメントして，各プロパティ名を一意にしてください。たとえば，次の **trim.fragment** プロパティの名前は **trim.fragment.3** になります。

このようなプロパティ設定の結果トリミングされるイベントとフラグメントは，ドロップされたイベントおよびドロップされたフラグメントの数で計算されます。

HP Diagnostics での複合アプリケーション検出の自動化

複合アプリケーション検出（CAM）は，アプリケーション・サーバ（プローブ）をグループ化し，プローブからほかのコンポーネントに対して行う呼び出しを追跡することによってこれらのアプリケーション・サーバに接続された新しいコンポーネントを継続的に検出できる便利な方法を提供します。

Diagnostics には，UI でのアプリケーションの設定に加え，CAM 用にスクリプトを作成するためのサポートが備えられています。これにより，UI 外で新しく追加したプローブに基づいて新しいアプリケーションの動的な作成が可能になります。

アプリケーションのスクリプト作成

新しいアプリケーションを作成するために使用するスクリプトは，各 Mediator の**etc/appDiscoveryRules.properties** に保存されます。Diagnostics に付属するスクリプトには，いくつかの例が含まれています。

アプリケーションは通常，プローブ名，プローブ・グループ名，サーバ要求名などのエンティティ・プロパティで利用可能な特定のパターンに基づいています。Diagnostics データ・モデルのクエリおよびインスタンスの選択には，**/groupby** パスが使用されます。このクエリ・パスは，次に示すようなアプリケーション検出用のスクリプトに使用され，しきい値や警告の設定の自動化にも使用されます。詳細については，『Diagnostics Data Model and Query API Guide』を参照してください。

次の例では，プローブ名の一部に基づいて新しいアプリケーションを作成する簡単な方法を示します。

```
# Example:
#
#   Put all probes with a particular naming pattern into
#   an application.
#
#   If you have a very consistent naming convention for probes, you
#   can auto-insert new probes into the appropriate application.
#
#   In the below example, we put any probe that has a name starting
#   with "cs_" into the "Sales" application.

/groupby[name¥='Default¥ Client']/probegroup/probe=¥
    String probeName = probe.getName();\
    if( probeName.startsWith("cs_") ) {\
        uid=name="Sales";\
    }
```

/groupby 定義（**青色の太字**）は，この Mediator ですべてのプローブを定期的にクエリし，返されたプローブ名に対してスクリプト（*赤色の斜体* の JavaScript）を実行します。上のコード例では，"**cs_**" で開始するすべてのプローブに対して "**Sales**" という名前のアプリケーションが作成されます。

名前に加え，アプリケーション権限も指定できます。スクリプトには，アプリケーション権限を指定するためのほかの例が含まれています。

さらに，サーバ要求や SQL ステートメントなどの関連するすべてのプローブエンティティを自動的に含めることができます。そのためには，変数 **discoveryPolicies** に値 "**applyAppFilterToProbeContents**" を設定します。

```
/groupby[name¥='Default¥ Client']/probegroup/probe=¥
    String probeName = probe.getName();¥
    if( probeName.startsWith("cs_") ) {\
        uid=name="Sales";¥
        discoveryPolicies="applyAppFilterToProbeContents"; ¥
    }
```

環境間での複合アプリケーションの移動

スクリプトを作成すると，QA 環境から実運用環境へといった環境間でのアプリケーションの移動を簡単に行うことができます。ただし，すべてのアプリケーション定義は，スクリプトを使用して作成する必要があります。プローブが Mediator に対してレポートしない場合でも，すべての Mediator で 1 つのマスタ・スクリプトを使用できます。

実運用環境と実運用前の環境間で動作する命名スキームを使用することが重要です。これを実現するには，次のようにします。

- 実運用環境と実運用前の環境間で不変の特定なプローブ・グループにプローブを配置します。
- スクリプトで上記の例に示すようなアプリケーション名を作成できるプローブの命名規則を使用します。

  プローブが同じプローブ・グループに属し，この名前が環境間で不変（ただしプローブ名は変わる）の場合は，スクリプトで **probegroup.getName()** を使用してプローブ・グループ名にアクセスします。

```
/groupby[name¥='Default¥ Client']/probegroup/probe=¥
   String probegroupName = probegroup.getName(); ¥
   String probeName = probe.getName();¥
   if( probegroupName.startsWith("cs") ) {¥
      uid=name="Sales";¥
      discoveryPolicies="applyAppFilterToProbeContents"; ¥
   } ¥
   else if (probegroupName.startsWith("is") ) { ¥
      uid=name="Information Systems";¥
      discoveryPolicies="applyAppFilterToProbeContents"; ¥
   }
```

このスクリプトは汎用的であり，環境間で交換できます。

可用性の高い Diagnostics サーバの準備

Diagnostics デプロイメントで，Diagnostics サーバに高い可用性が必要な場合は，各 Diagnostics サーバにスタンバイ Diagnostics サーバを作成できます。スタンバイは，Diagnostics サーバのホストにハードウェア障害などの問題が生じたときにいつでも使用できます。

スタンバイ Diagnostics サーバの作成

Diagnostics サーバをスタンバイ・マシンにインストールし，プライマリ Diagnostics サーバのデータをスタンバイ Diagnostics サーバに定期的に複製することで，各 Diagnostics サーバのスタンバイを作成できます。

スタンバイ Diagnostics サーバを設定するには，次の手順を実行します。

1 スタンバイ・マシンに Diagnostics サーバをインストールします。スタンバイ・サーバにインストールする Diagnostics サーバは，プライマリ・サーバの Diagnostics サーバと同じバージョンを使用してください。

2 スタンバイ Diagnostics サーバのホストから次のコマンドを使って，プライマリ・サーバからスタンバイ・サーバへの定期リモート・バックアップをスケジュールします。

```
% cd /opt/MercuryDiagnosticsServer/
% ./bin/remote-backup.sh -h < プライマリ・サーバのホスト > -o .
```

＜プライマリ・サーバのホスト＞ を複製する Diagnostics サーバのホスト名に置き換えます。

これらのコマンドは，Diagnostics データ，設定ファイル，カスタマイズされたビュー，警告およびコメントのインクリメンタルな複製をスタンバイ Diagnostics サーバに実行します。Windows でクローン・ジョブまたはスケジュールされたタスクを使って，定期バックアップをスケジュールできます。

注 : wget ユーティリティは，HTTP を通じてバックアップをダウンロードします。Windows では，Diagnostics サーバのホストに **cygwin** のインストールが必要です。**cygwin** は，http://www.cygwin.com/ から入手できます。

スタンバイ Diagnostics サーバのフェールオーバー

プライマリ Diagnostics サーバのホストで障害が発生した場合は，プライマリ Diagnostics サーバとして機能するようにスタンバイ Diagnostics サーバを設定します。

スタンバイ Diagnostics サーバをプライマリ Diagnostics サーバにするには, 次の手順を実行します。

1 障害が発生したプライマリ Diagnostics サーバのホストのホスト名に一致するように，スタンバイ Diagnostics サーバのホスト名を変更します。これにより，プローブが起動したときに Diagnostics サーバに再接続できるようになります。

2 スタンバイ Diagnostics サーバを Windows Service として起動するか，または **bin/server.sh** スクリプトか **bin¥server.cmd** スクリプトを使います。プローブは Diagnostics サーバに再接続します。プローブは，Diagnostics サーバへの接続が失われると，約 30 秒おきに必ず再接続を試みます。

3 以上で, スタンバイ Diagnostics サーバがプライマリ Diagnostics サーバになりました。445 ページ「スタンバイ Diagnostics サーバの作成」の説明に従って，新しいスタンバイ Diagnostics サーバを設定します。

注 : 障害が発生した Diagnostics サーバが復旧しても，復旧した Diagnostics サーバをプライマリにしないでください。新しいプライマリ Diagnostics サーバの使用中にプローブから収集したデータが失われます。

HP ServiceGuard（HA ソリューション）用の Diagnostics の設定

Diagnostics に，HA（高可用性）ソリューションとして HP ServiceGuard を設定できます。本項では，HP ServiceGuard（Linux）で実行されるように，HP Diagnostics（バージョン 7.50 以降）を設定するための手順の概要を示します。

注 : 読者は, Diagnostics および HP ServiceGuard に精通していることが前提となります。

このセクションに記載された設定手順は，ほかの HA ソリューション（Microsoft クラスタ・サービスなど）にも使用できます。

Diagnostics サーバは，Diagnostics 時系列データベース（TSDB）およびその他の構成アイテム（ユーザ権限，カスタム・ダッシュボード画面など）に使用できる十分な空き容量がある共有ディスク上にインストールする必要があります。

両方の Diagnostics サーバ（アクティブとスタンバイ）は NTP または Business Service Management を介して同期する必要があります。時刻同期メカニズムとして SYSTEM を使用することはお勧めしません。Diagnostics サーバで使用される「クロック」は，両方のサーバで同じである必要があります。

Diagnostics サーバはホスト名を，アーカイブおよびストレージ・ディレクトリのサブディレクトリのプレフィックスとして使用します。この動作を上書きするには，サーバを起動した Java コマンド・ラインで **-Dmediator.id=cluster -Dserver.id=＜クラスタ＞**（＜クラスタ＞はほかの一意の名前で置き換えることが可能）を指定する必要があります。

例 : ＜インストール・ディレクトリ＞/bin/server.sh

```
$JAVA1_5_HOME/bin/java -Dserver.id=cluster -Dmediator.id=cluster -server
-Xmx512m
$SERVER_BCP $JAVAOPTS -Dsun.net.client.defaultReadTimeout=70000
-Dsun.net.client.defaultConnectTimeout=30000
-classpath $SERVER_HOME/lib/mediator.jar$PATHSEP$SERVER_HOME/lib/
loading.jar$PATHSEP$SERVER_HOME/lib/common.jar$PATHSEP$SERVER_HOME/
lib/mercury_picocontainer-1.1.jar  com.mercury.opal.mediator.util.DiagnosticsServer
```

注 : すべてのコマンド・ライン・コンポーネントは同じ行に記述する必要があります。

ServiceGuard パッケージの場合は，アプリケーションの開始および停止コマンドが必要です。Diagnostics の開始コマンドは，**＜サーバのインストール・ディレクトリ＞/bin/server.sh** スクリプトです。

停止コマンドの場合は，次の内容を含む新しいスクリプトを，**＜サーバのインストール・ディレクトリ＞/bin** に配置する必要があります。

＜インストール・ディレクトリ＞/bin/stop.sh

```
#!/bin/sh
PID=`ps -ef | grep -v grep | grep DiagnosticsServer | awk '{ print $2 }' `
kill $PID
sleep 10
```

スクリプトに実行権限（chmod u+x stop.sh）があることを確認してください。

ServiceGuard パッケージ・スクリプトに，次の行を追加します。

```
stop_command:
  ＜インストール・ディレクトリ＞ /bin/stop.sh

start_command:
  ＜インストール＞ /bin/server.sh &
```

＜インストール・ディレクトリ＞ を Diagnostics のサーバ・インストール・ディレクトリで置き換えて，server.sh の末尾にアンパサンド（&）があることを確認します。

Diagnostics サーバの割り当て（LoadRunner/Performance Center の実行）

標準設定で，LoadRunner または Performance Center の実行用に選択したプローブは，**＜プローブのインストール・ディレクトリ＞/etc/dynamic.properties** で指定されている Diagnostics サーバを使用します。

実行のためにプローブを起動したときに，設定を変更できます。そのためには，Diagnostics コマンダ・サーバのマッピング・ファイルを変更します。このようにすると，Diagnostics サーバのプローブの割り当てをオーバーライドできます。

これは，LoadRunner / Performance Center と Business Service Management の複合環境で Diagnostics を実行している場合に便利です。Diagnostic サーバが LoadRunner / Performance Center の実行環境で稼働している場合は，Business Service Management にデータをレポートしている場合と異なる Diagnostic サーバをプローブに使用できます。

プローブの設定ファイルを編集するよりも，この方法を使った方が便利です。

注 : プローブが実行中でない場合は，**<プローブのインストール・ディレクトリ> etc/dynamic.properties ファイル**で指定した Diagnostics サーバが使用されます。

プローブに対して Diagnostics サーバの割り当てを変更する場合，Diagnostics サーバ（Commander モード）ホスト・マシンで **< Diagnostics サーバのインストール・ディレクトリ>¥etc** ディレクトリの **server_assignment.properties** ファイルを変更します。

server_assignment.properties ファイルの形式は次のとおりです。

<ProbeID> = <Server.id>

- **<ProbeID>** をプローブの ID に置き換えます。
- **<Server.id>** を Diagnostics サーバの ID に置き換えます。

LoadRunner / Performance Center の実行の冒頭で，**server_assignment.properties** が動的に読み取られます。このファイルに加えた変更は，Diagnostics サーバ（Commander モード）を再起動しなくても有効になります。

LoadRunner オフライン分析ファイル・サイズに対する Diagnostics サーバの設定

実行される LoadRunner シナリオまたは Performance Center テストのそれぞれに対して，Diagnostics サーバ（Mediator モード）は，シナリオの最中にキャプチャされる Java データが含まれる LoadRunner オフライン分析に必要なファイルを作成します。このファイルのサイズは，非常に大きくなることがあります。シナリオの実行中にファイルが一時的に保存される Diagnostics サーバ（Mediator モード）のホスト・マシンと，シナリオが完了したときにファイルが保存される LoadRunner コントローラ・ホスト・マシンの両方に，LoadRunner オフライン・ファイルを保持できる十分なディスク容量があることを確認してください。

LoadRunner オフライン・ファイルのサイズの予測

オフライン・ファイルのサイズの見積りは，キャプチャされるデータと，データがキャプチャされる頻度によって大きく異なります。

LoadRunner オフライン・ファイルのサイズを予測するには，次の手順を実行します。

1 負荷テストを 5 分間実行し，LoadRunner のシナリオが開始したときに Diagnostics サーバ（Mediator モード）によって作成されたオフライン・ファイルのサイズを監視します。

＜Diagnostics サーバのインストール・ディレクトリ＞/data/＜newest ディレクトリ＞ で，Diagnostics サーバ（Mediator モード）ホスト・マシンのオフライン・ファイルを見つけます。オフライン・ファイルには，**.inuse** という拡張子が付いています。

2 5 分たったら，オフライン・ファイルのサイズを書き留めます。

3 前の手順で得たオフライン・ファイルのサイズを 12 倍して，1 時間後のオフライン・ファイルのサイズを見積もります。

4 前の手順で計算した 1 時間後のファイル・サイズに，実際に実行する予定の負荷テストの時間数をかけて，負荷テスト終了時のオフライン・ファイルの推定サイズを特定します。

5 Diagnostics サーバ（Mediator モード）ホスト・マシンと Controller ホスト・マシンに，オフライン・ファイルの推定サイズを格納するのに十分なディスク容量があるかどうかを判断してください。

LoadRunner オフライン・ファイルのサイズの圧縮

オフライン・ファイルのサイズに心配がある場合，Diagnostics サーバ（Mediator モード）のオフライン集計時間を長くして，ファイル・サイズを小さくすることができます。これにより，オフライン・データの粒度が下がり，オフライン・ファイルのサイズが小さくなります。

これらのプロパティの標準設定は **5s**（5 秒）で，Diagnostics サーバ（Mediator モード）は 5 秒のタイム・スライスにすべてのデータを集計します。集計時間が長くなると保存に要するデータ・ポイントが少なくなるため，これらのプロパティの値を大きくするとオフライン・ファイルが小さくなります。たとえば，オフライン集計時間プロパティを 45s にすると，ファイル・サイズは 50～75% 小さくなります。

注：オフライン集計時間の調整によるオフライン・ファイル・サイズへの影響は，アプリケーションの動作と負荷テストの仕様によって非常に異なります。

次の手順を使って，**＜Diagnostics サーバのインストール・ディレクトリ＞/etc/server.properties** で，Diagnostics サーバ（Mediator モード）オフライン集計時間プロパティ **bucket.lr.offline.duration** および **bucket.lr.offline.sr.duration** を変更します。

Diagnostics サーバ（Mediator モード）のオフライン集計時間を長くして，オフライン・ファイルのサイズを小さくするには，次の手順を実行します。

1. Diagnostics サーバ（Mediator モード）が，LoadRunner / Performance Center のアクティブな実行に関係していないことを確認します。次の手順に記載されているプロパティの変更が有効になる前に，Diagnostics サーバ（Mediator モード）を再起動する必要があるため，必ず確認してください。
2. 次の URL に移動して，［Mediator Configuration］ページにアクセスします。

   http://<diagnostics_server_hostname>:2006/configuration/Aggregation?level=60
3. **LoadRunner / Performance Ceter オフライン VU 集計時間**プロパティの設定を大きくして，オフライン VU 集計時間を長くします。このプロパティの値は，5 の倍数である必要があります。たとえば，45s に設定します。
4. **LoadRunner / Performance Center オフライン・サーバ要求集計時間**プロパティの値を大きくして，オフライン・サーバ要求集計時間を長くします。このプロパティの値は，5 の倍数である必要があります。たとえば，45s に設定します。
5. ページ下部の［**送信**］をクリックして，Diagnostics サーバ（Mediator モード）を変更後のプロパティ値で更新します。

   ページの一番上に，変更が保存された旨を伝えるメッセージと，Diagnostics サーバ（Mediator モード）の再起動を求めるリマインダが表示されます。［**Restart Mediator**］ボタンも表示されます。

［Configuration］ページからプロパティ値を更新する方法，およびコマンド・ボタンを表す画面画像については，751 ページ「サーバ設定の変更」を参照してください。

設定の変更を有効にするには，［**Restart Mediator**］をクリックして Diagnostics サーバ（Mediator モード）を再起動します。

Business Service Management のサンプル・キュー・サイズと Web サービス CI の頻度の設定

次の設定は，Business Service Management 統合に適用できます。

Business Service Management のサンプル・キュー・サイズの設定

標準設定では，Business Service Management サンプル・キュー・サイズは 100 に設定されています。一度に 100 個以上のサンプルが作成された場合，サンプルのいくつかはドロップし，SOA 用の Application Management のデータが消失します。ログには次のメッセージを記録できます。BAC sample being dropped since too many are waiting for delivery。

サーバ・プロパティ **dispatcher.server.wdedelivery.max.queue.size** を設定することにより，サンプル・キュー・サイズを増やして WDEDelivery キュー・サイズを設定できます。

Web サービス CI の頻度

Web サービス CI は，Diagnostics によって標準設定の頻度で自動的に作成され，Run-time Service Model に追加されます。

Web サービス CI を Run-time Service Model に追加するプロセスのタイミングを変更できます。Web Service CI の作成プロセスでは，次の設定プロパティを **server.properties** で定義しています。

- **bac.webservice.CI.create.runfrequency** - 作成を実行する秒単位の間隔（標準設定は 300s，すなわち 5 分）
- **bac.webservice.CI.create.query.granularity** - 作成する Web サービス CI の特定に使用する Diagnostics クエリの粒度（標準設定は 1d）

Diagnostics サーバ設定ページを使用した Diagnostics の設定

Diagnostics サーバの設定ページでは, Diagnostics サーバとほかの Diagnostics コンポーネントの通信方法, およびプローブから受信したデータを Diagnostics サーバでどのように処理するのかを制御するプロパティ値を設定できます。

注 : 有効なプロパティ値を確実に入力するために, プロパティ・ファイルを直接編集するのではなく, これらのページを使って Diagnostics サーバ・プロパティを更新する必要があります。

Diagnostics サーバの設定ページを使って Diagnostics を表示および変更する方法については, 付録 A, 「Diagnostics 管理 UI」を参照してください。

実運用環境で処理できるプローブを増やすための Diagnostics サーバの最適化

1 つの Diagnostics サーバが処理できるプローブの数は, 5 分間隔の一意のサーバ要求の数と, 各サーバ要求のメソッドおよびレイヤの数によって大きく異なります。次の最適化を行うと, サーバ・プロセスごとに処理できるプローブ数が増加します。

- 標準設定で，Diagnostics サーバは，各プローブから 5 秒間隔でトレンドを取得し，45 秒間隔で各プローブのツリーを取得するように設定されています。1 つの Diagnostics サーバ・プロセスで 25 以上のプローブを処理している場合，トレンドとツリーの取得頻度が減るように最適化できます。実運用環境の推奨最適化設定は，トレンドの取得間隔が 30 秒，ツリーの取得間隔が 120 秒です。これらの値は，次のように **<Diagnostics サーバのインストール・ディレクトリ>¥Server¥etc¥server.properties** で設定できます。

```
# The interval at which to pull trends from probes
probe.trends.pull.interval = 30s

# The interval at which to pull trees from probes
probe.trees.pull.interval = 120s
```

- サーバ・プロセスの最大ヒープ・サイズは，サーバの起動スクリプトの **-Xmx** パラメータで決められます。最大ヒープ・サイズの標準設定は 512 MB です。プローブからの負荷に基づいて，最大ヒープ・サイズを増やします。処理されるプローブの数に基づく最大ヒープ・サイズの推奨値は，第1 章，「HP Diagnostics のインストールの準備」を参照してください。
- サーバが 30 以上のプローブを処理している場合，Diagnostics サーバの実運用環境では 1 Gbps リンクの使用を強くお勧めします。
- 1 つのサーバ・プロセスで 75 以上のプローブを処理している場合は，Jetty スレッドの数を増やします。スレッドの数を見極める一般的な目安は，プローブ数の 2 倍に 40 を足した数です。標準設定値は 200 です。Jetty スレッドの数は，**<Diagnostics サーバのインストール・ディレクトリ>¥Server¥etc¥webserver.properties** の **jetty.threads.max** プロパティを変更して増加することができます。次に例を示します。

```
jetty.threads.max=300
```

12

Java Agent およびアプリケーション・サーバの詳細構成

このセクションでは，Diagnostics Java Agent とアプリケーション・サーバ環境の詳細設定について説明します。詳細設定は，この製品に精通している熟練ユーザを対象にしています。コンポーネントのプロパティを変更する際は，十分に注意してください。

本章の内容

- SOAP の障害ペイロード・データの構成（491 ページ）
- REST サービスの構成（493 ページ）
- JMS 一時キュー / トピックのグループ化のカスタマイズ（493 ページ）
- SQL クエリの解析の設定（493 ページ）
- サーバ要求に対するアプリケーション名の表示構成（494 ページ）
- Java Profiler UI からのプローブ設定の保守（495 ページ）
- JUnit テスト用のパフォーマンス・レポートの生成（503 ページ）

詳細設定の概要

次の情報は，お使いの環境の設定に役立つプローブ設定情報です。

- Profiler モードでプローブを他者に使わせないようにするには，457 ページ「Java Diagnostics Profiler の無効化」を参照してください。
- ログ・メッセージを低レベル・メッセージのプローブ・ログに送るには，458 ページ「プローブの記録の制御」の手順を行なってログ・レベルを調整します。
- 同じホストに複数の Agent をインストールしている場合，459 ページ「プローブのログ・ディレクトリの変更」の手順を行なって，各 Agent のログ・メッセージを別のファイルに保存できます。
- Diagnostics で報告されたメトリックスから通常トリミングされる可能性のある処理のパフォーマンスを調べるため，464 ページ「Agent におけるメソッドの自動トリミングの制御」の手順を行なって，トリミング・レベルを下げたり，トリミング機能を完全にオフにできます。
- Agent と Diagnostics コマンド・サーバの間にプロキシが存在し，Agent に Diagnostics コマンド・サーバの URL を通知するように正しいプロパティを設定する必要がある場合は，467 ページ「プロキシ・サーバへの Agent の構成」を参照してください。
- HP Software as a Service（SaaS）環境に Java Agent をインストールし，Agent と Diagnostic メディエータ・サーバの間のリバース HTTP（RHTTP）通信を無効にする場合は，468 ページ「VMware 上で実行中のプローブの時刻の同期」を参照してください。

- 仮想環境で実行している場合は，468 ページ「VMware 上で実行中のプローブの時刻の同期」を参照してください。
- 例外データの数を制限する必要がある場合は，468 ページ「例外ツリー・データの制限」を参照してください。
- プロパティ・ファイルを変更するコレクション・オプションを使用する場合は，480 ページ「コンシューマ ID の構成」など，このセクションのほかのトピックを参照してください。

Java Diagnostics Profiler の無効化

Java Agent で Diagnostics Profiler for Java を無効にすることで，誤ったアクセスを回避できます。Java Diagnostics Profiler が無効になると，Java Diagnostics Profiler の URL http:// ＜プローブのホスト＞ : ＜プローブのポート＞ /profiler からユーザ・インタフェースにアクセスできなくなります。

Java Diagnostics Profiler を無効にするには，**＜プローブのインストール・ディレクトリ＞ /etc/probe.properties** の **disable.profiler** プロパティを true に設定します。

disable.profiler の標準設定値は，**false** です。無効にした Java Diagnostics Profiler を有効にするには，**disable.profiler** プロパティの値を **true** から **false** に変更します。

プローブの記録の制御

プローブ・プロパティを使用して，プローブが記録するメッセージのレベルを制御したり，ログ・メッセージを送る場所を変更できます。

ログ・メッセージ・レベルの制御

標準出力に記録されるプローブのメッセージ・レベルは，**<プローブのインストール・ディレクトリ> /etc/logging.properties** プロパティ・ファイルの **lowest_printing_level** プロパティで制御されます。このプロパティの標準設定は **OFF** です。この場合，ほとんどすべての Agent メッセージがコンソールにログ処理されません。

lowest_printing_level プロパティに割り当てられている値を変更して，ログ・レベルを動的に調整できます。記録されるメッセージのレベルは，プロパティ・ファイルを保存するとすぐに変更されます。

lowest_printing_level プロパティの有効な値は次のとおりです。

プロパティ値	説明
OFF	メッセージは記録されません。
DEBUG	メッセージはすべて記録されます。
INFO	Info，Severe および Warning メッセージが記録されます。
WARN	Warning および Severe メッセージが記録されます。
SEVERE	Severe メッセージが記録されます。

プローブのログ・ディレクトリの変更

プローブのログ・ディレクトリの標準設定の場所は**＜プローブのインストール・ディレクトリ＞ /log** です。同じホストに複数のプローブがある場合，**log.dir** プロパティを使用して，プローブごとにログ・ディレクトリの場所を変更できます。このプロパティは，次の 2 つの方法で設定できます。

- **log.dir** プロパティの値は，**＜プローブのインストール・ディレクトリ＞ / etc/probe.property** プロパティ・ファイルで設定できます。
- **log.dir** プロパティの値は，次の例のように，Java システム・プロパティとして，アプリケーション・サーバの起動コマンド・ラインで指定できます。

```
-Dprobe.log.dir=/path/to/log
```

起動コマンド・ラインで log.dir プロパティを指定する方法については，467 ページ「プロキシ・サーバへの Agent の構成」を参照してください。

プローブのホスト・マシン名の設定

Diagnostics コマンダ・サーバでは，プローブのホスト名によってプローブが登録されます。Diagnostics コマンダ・サーバでは，プローブのホスト名を使用してプローブと通信し，プローブが監視しているサーバのシステム・メトリックスとともに Diagnostics ビューに表示します。

通常，プローブは，そのホストであるマシンのホスト名を検出できます。ただし，サーバ設定にエラーがあるため，プローブが正しいホスト名を検出できないことがあります。ファイアウォールまたは NAT がある場合，または Agent ホスト・マシンがマルチホーム・デバイスとして設定されている場合，Agent はホストを検出できないことがあります。

プローブがホスト名を検出できない場合，ソケット接続に基づくリバース DNS 参照を介してホスト名を特定するように プローブに指示することも，またはプローブ・プロパティを使用してホスト名を指定することもできます。

プローブのリバース DNS ルックアップ使用の指定

プローブのホスト設定によってプローブでホスト名を検出できない場合，**server.host.name.lookup** プロパティを設定して，リバース DNS 参照を使用してホスト名を検出するようにプローブに指示できます。このプロパティは，**<プローブのインストール・ディレクトリ> /etc/dispatcher.properties** ファイルにあります。

server.host.name.lookup プロパティの標準設定値は‘false’です。これにより，リバース DNS を使わずに参照を実行するようにプローブに指示します。このプロパティを「true」に設定して，プローブにリバース DNS 参照を使用するように指示します。

プローブのホスト名の手動指定

registered_hostname を使用して，プローブのホスト・マシン名を手動で設定したり，プローブの自動ルックアップ機能を停止できます。

プローブの標準設定のホスト・マシン名を設定するには，**registered_hostname** プロパティ（プロパティ・ファイル **<プローブのインストール・ディレクトリ>/etc/dispatcher.properties** にある）をマシン名または IP アドレスに設定します。

registered_hostname プロパティを設定すると，ホスト名の自動ルックアップ機能が無効になります。

注：NAT またはファイアウォールが原因で **registered_hostname** プロパティを設定することは，LoadRunner，Performance Center，または Diagnostics Standalone を使用するテスト環境における唯一の問題です。

Business Service Management または Diagnostics Standalone を使用している実運用環境で **registered_hostname** を設定する必要がある場合，指定した名前はシステムの状況にホスト名として表示されます。

異なるプローブの IP アドレスの指定

probe.host.ip.address.override プロパティ（プロパティ・ファイル**＜プローブのインストール・ディレクトリ＞ /etc/dispatcher.properties** にある）により，プローブの IP アドレスを上書きできます。このプロパティは，異なる IP アドレス（トンネルを介してサーバとプローブが通信できるようにする仮想 IP など）をプローブからサーバに提供する場合に使用できます。

アクティブ・プロダクト・モードの設定

Java Agent モードは一般的にセットアップ・プログラムで入力したオプションに基づいて設定されます。しかし，本項で説明するように手動でこのモードを設定できます。

Java Agent は，次のことができるようにさまざまなモードで設定できます。

- 開発から実稼動前のテスト，実運用までアプリケーションを監視する
- ほかの HP ソフトウェア製品と連携する
- サーバまたはほかの HP ソフトウェア製品にレポートしないスタンドアロンの Diagnostics Java Profiler として使用する

Java Agent が動作するモードは，**＜プローブのインストール・ディレクトリ＞ /etc/probe.properties** プロパティ・ファイルにある **active.products** プロパティのモード値によって決まります。

active.products プロパティのモード値は，ライセンス・キャパシティに対する使用数を決定する場合にも使われます（83 ページ「現在接続されているプローブに基づくライセンス情報」を参照）。Diagnostics には 2 種類の LTU（使用ライセンス）があります。

- AM - Enterprise モード（通常は実運用環境）で製品を使用する場合
- AD - 製品を実稼動前の負荷テスト環境（LoadRunner または Performance Center でプローブを実行）で使用する場合

active.products プロパティの値は，Java Agent のインストール時に初期設定されます。

active.products プロパティの値を変更するには，プロパティ・ファイルを編集してアプリケーション・サーバを再起動します。または，Java Agent セットアップ・モジュールを再実行して，［変更］オプションでモードを［Diagnostics Profiler モード］(PRO)，［Diagnostics 用の Application Management/Enterprise モード］（Enterprise）や［TransactionVisions 用の Application Management/Enterprise モード］（TV），または［LoadRunner/Performance Center 用の Diagnostics モード］（AD）に設定します。

注 : スタンドアロンの Diagnostics Profiler for Java の評価版を Enterprise モードまたはほかの HP ソフトウェア製品と統合して使用するには，HP ソフトウェア・カスタマ・サポートに連絡して HP Diagnostics を購入してください。

インタフェースしている HP ソフトウェア製品のユーザ・インタフェースで Diagnostics データを表示するには，追加の設定手順を行う必要があります。Business Service Management，LoadRunner または Performance Center との統合の詳細については，691 ページ「ほかの HP ソフトウェア製品との統合のセットアップ」の項を参照してください。次の項では，**active.products** プロパティの各プロダクト・モードを設定するための手順を紹介します。

PRO プロダクト・モード ー Diagnostics Profiler for Java

PRO モードが設定されている場合，Agent はパフォーマンス・メトリックスを収集し，Agent ホストの URL を介して利用できるスタンドアロンの Diagnostics Java Profiler のユーザ・インタフェースに表示します。

Java Diagnostics Profiler の評価版の一部として Java Agent を実行している場合，Agent で処理可能な負荷を制限するための制限が Agent に設定されています。

完全な Diagnostics Enterprise 製品の一部として Java Agent を実行している場合，またはほかの HP ソフトウェア製品と連携して実行している場合，Profiler は負荷を制限せずに有効になります。

PRO モードは，ライセンス・キャパシティに対する使用数を決定する場合には使用されません。

Enterprise プロダクト・モード

Enterprise モードに設定されている場合，Agent は，Business Service Management，LoadRunner，Performance Center などの HP ソフトウェア製品と連携して，完全な Diagnostics Enterprise 製品として機能します。ただし，これらの統合を有効にするには，追加の設定を行う必要があります（詳細については，691 ページ「ほかの HP ソフトウェア製品との統合のセットアップ」の各項を参照）。

Enterprise モードでは，データも Diagnostics Java Profiler に送信されます。

Enterprise モードでは，Diagnostics サーバに Agent を登録する必要もあります（148 ページ「Diagnostics サーバへの Agent の登録」を参照）。

Enterprise モードは，Java Agent の標準設定です（AD または AM モードを指定していない場合）。Enterprise モードでは，Agent が AM ライセンス・キャパシティに対してカウントされます。

AM プロダクト・モード

AM モードの場合，Java Agent はあらゆるインストルメンテーション・データをキャプチャします。実運用環境 Business Service Management デプロイメントの Agent が LoadRunner または Performance Center の実行に誤って含まれないように保護するには，AM モードに設定します。AM モードでは，Agent は，LoadRunner または Performance Center の使用可能な Agent に表示されません。

AM モードの Agent は，常に AM ライセンス・キャパシティに対してカウントされます。

AM モードは，AD を除く，その他のすべてのモードよりも優先されます。

AD プロダクト・モード

AD モードの場合，Java Agent は LoadRunner または Performance Center の実行時にのみデータをキャプチャします。結果は，Default Client:21 など，この実行のための専用 Diagnostics データベースに格納されます。

AD モードでは，プローブが LoadRunner または Performance Center の実行の一部である場合を除いて，Agent はリソースを使用したり，サーバにデータを送信しません。

LoadRunner 統合のセットアップ方法については，第 23 章，「HP LoadRunner と HP Diagnostics の統合のセットアップ」を参照してください。Performance Center 統合のセットアップ方法については，第 24 章，「Diagnostics を使用するための Performance Center のセットアップ」を参照してください。

このモードを使用して, 負荷テストが実行されていないときに, QA 環境の Agent が追加リソースを使用して，継続的に Diagnostics サーバにデータを報告しないようにできます。

AD モードでプローブを実行する別の利点は，AD モードのプローブが LoadRunner または Performance Center の実行に含まれる場合にのみ AD ライセンス・キャパシティに対してプローブがカウントされるという点にあります。たとえば，LoadRunner/Performanace Center AD モードで 20 個のプローブがインストールされていて，5 個のみが実行されている場合，その 5 個のみが AD ライセンス・キャパシティに対してカウントされます。

TV プロダクト・モード

TV モードは Transaction Vision にイベントを送信します。このモードはほかのモードと組み合わせることができる。TV モードは，HP Diagnostics ライセンス・キャパシティの使用数の判別には使用されません。

Agent におけるメソッドの自動トリミングの制御

Agent の標準設定には，メソッドのトリミングを制御する設定が含まれます。トリミング機能は，レイテンシと呼ばれるメソッドが実行に要する時間に従って，メソッド呼び出しのスタックの深さで制御できます。標準設定では，レイテンシと深さの両方でトリミングを行うようにプローブに指示します。

トリミングのレベルを下げたり，トリミング機能を完全にオフにできます。**＜プローブのインストール・ディレクトリ＞ /etc/capture.properties** の **minimum.method.latency** プロパティと **maximum.stack.depth** プロパティを使用してトリミングを制御できます。

レイテンシのトリミングの制御

minimum.method.latency プロパティの値と同じか, またはそれ以上のレイテンシのあるメソッドはキャプチャされ，この制限よりも小さなレイテンシのあるメソッドは，あまり重要でないメソッドでオーバーヘッドが生じるのを避けるためにトリミングされます。

注 : 次の場合，レイテンシがトリミング・プロパティよりも小さいとき，メソッドのレイテンシはトリミングされません。

- 呼び出しツリーのルートであるメソッド。
- 例外を発生させるメソッド。

すべてのメソッドの情報をキャプチャする必要がある場合は，**minimum.method.latency** プロパティの値を低くするか，または 0（ゼロ）に設定します。

minimum.method.latency プロパティを設定する際は，次の点に留意してください。

- **minimum.method.latency** プロパティの値が小さくなるほど，アプリケーションのパフォーマンスに悪影響を与える可能性が大きくなります。
- プラットフォームの種類や，ネイティブのタイムスタンプを使用しているかどうかに応じて（**use.native.timestamps** = false），この値を 10 ms 未満の増分で指定しても意味がないことがあります。

深さのトリミングの制御

maximum.stack.depth プロパティの値に等しいか，またはその値よりも小さいスタックの深さで呼び出されるメソッドはキャプチャされます。この値よりも大きなスタックの深さで呼び出されるメソッドは，あまり重要でないメソッドでオーバーヘッドが生じるのを避けるためにトリミングされます。

次に例を示します。

maximum.stack.depth が 3 and /login.do calls a() calls b() calls c() の場合，/login.do, a, b だけがキャプチャされます。

maximum.stack.depth を低い値に設定すると，キャプチャのオーバーヘッドが大幅に減ります。

URI の切り捨て，マッピング，トリミングの構成

すべての HTTP/S サーバ要求 URI は，プローブでレポートされる前に変換することができます。変換の候補は，正規表現による照合と **dynamic.properties** の **uri.pattern.replace** プロパティによって制御される置換に基づいています。プロパティの値は，各 URI で試行されるパターン置換操作のカンマ区切りリストです。

これは，サーバ要求が多すぎるために多数のサーバ要求 URI を 1 つの簡素化されたサーバ要求 URI に置換してまとめる場合に使用できます。

s/ パターン / 置換先 / 構文を使用して URI を切り捨てまたはマッピングします。複数の操作を実行するには，カンマ区切りリストを使用します。操作は順番に実行されます。

たとえば，ある文字列の前で切り捨てを行う場合，その文字列とその後に続く任意の文字を照合して置換先を空のままにします。この例の「$」は改行です。

```
s/string.*$//
```

詳細と例については，dynamic.properties ファイルの「URI Truncation and Mapping」の下にあるコメントを参照してください。

重要 : この機能を多用するとパフォーマンスに影響します。

サーバ要求の数が多すぎる場合，**capture.properties** ファイルのプロパティ **maximum.uri.pathsegments** を使用して，サーバ要求を第 *n* パス・セグメントまでトリミングすることもできます。

標準設定は -1 です。この場合，プロパティが無効になります。SaaS（Java Agent のセットアップで SaaS を選択）環境でのプローブのレポートでは，HP でホストされるサーバに送信されるサーバ要求データ量が大きくなり過ぎないように maximum.uri.pathsegments が自動的に 2 に設定されます。

たとえば，2 に設定すると，第 2 パス・セグメント以内になります。そのため，http://localhost:8080/path1/path2/path3 は，http://localhost:8080/path1/path2/ までトリミングされます。

uri.pattern.replace を使用して，特定の数のパス・セグメントまでトリミングするように maximum.uri.pathsegments を設定できます。また，いずれかのプロパティのみを使用します。

プロキシ・サーバへの Agent の構成

重要！ このセクションは, Diagnostics サーバで Java Agent を使用している場合のみ必要になります。

Java Agent, Diagnostics コマンド・サーバの URL の指定には 2 つのプロパティを使用します。設定するプロパティは，プロキシがあるかどうかで異なります。

- **dispatcher.properties** の **registrar.url**

  Agent をインストールするときに，**＜プローブのインストール・ディレクトリ＞ ¥etc¥dispatcher.properties** の **registrar.url** プロパティが設定されます。Agent と Diagnostics コマンド・サーバの URL の間に直接接続がある場合，このプロパティの値は Agent によって使用されます。

- **webserver.properties** の **registrar.url**

  プロキシがある場合，**＜プローブのインストール・ディレクトリ＞ ¥etc¥ webserver.properties** ファイルで **registrar.url** プロパティを設定して，Diagnostics コマンド・サーバの URL を指定する必要があります。

VMware 上で実行中のプローブの時刻の同期

VMware ゲストで実行するプローブでは，VMware ゲストとそれを支える VMware ホストとの間で時刻を同期する必要があります。時刻が正しく同期していないと，Diagnostics UI は VMware ゲストで実行中のプローブから間違った測定値を表示したり，測定値がまったく表示されなくなったりすることがあります。

時刻は，時間管理に関する VMware のホワイトペーパー（http://www.vmware.com/pdf/vmware_timekeeping.pdf（英語サイト））の「Synchronizing Hosts and Virtual Machines with Real Time」の項にある推奨事項に従って同期化する必要があります。VMware ツールは Diagnostics プローブをホストする VMware ゲスト・オペレーティング・システムごとにインストールする必要があります。VMWare ツールの時刻同期オプションをオンにする必要があります。

VMware ツールのこのオプションは，ゲスト・オペレーティング・システムの時刻が VMware ホストよりも早い時刻に初期設定されている場合にかぎり機能します。VMware ツールのインストール方法については，VMware ESX Server の『基本システム管理』を参照してください。VMware 以外の時刻同期ソフトウェア（Network Time Protocol など）を使用している場合は，VMware ESX サーバ・サービス・コンソールでそれを実行してください。

プローブのプロパティ attempt.vmware.timestamp.adjustments を true に設定して VMware ゲストでプローブを実行している間にレイテンシが負の値になった場合は，プローブの **etc/capture.properties** ファイルに次のプロパティを設定します。

```
use.vmware.timestamp.workaround=true
```

use.vmware.timestamp.workaround が true に設定されているとき，プローブは代替呼び出しを使用して VMware ホスト・タイムスタンプを取得し，レイテンシが負の値である問題を回避します。

例外ツリー・データの制限

Agent は例外情報を収集し，それを使用して例外インスタンス・ツリーを構築します。例外インスタンス・ツリーは，スタック・トレースなど Diagnostics UI に表示されている例外情報のデータを提供します。

標準設定では，監視対象アプリケーションで発生するすべての例外が，例外インスタンス・ツリーの候補です。ただし，関心のない例外は，表示だけでなく，データ収集，転送処理にも負荷がかかるため，すべての例外情報を収集するのは，通常，望ましくありません。このため，収集するデータの例外タイプを制限できます。たとえば，**javax.naming.AuthenticationException** などのアプリケーション・サーバ・ベースのエラーをフィルタリングすると，より多くのアプリケーション固有のエラーを例外ツリーに含めることができます。

収集される例外ツリー・データを制御するには，特定の例外タイプを制限するか，または例外タイプの数を制限します。

特定の例外タイプの制限

＜プローブのインストール・ディレクトリ＞¥etc¥dispatcher.properties ファイルで次のように **exception.types.to.exclude** プロパティと **exception.types.to.include** プロパティを設定することにより，コレクションからどの例外タイプを除外するか，あるいは包含するかを制御できます。

- **exception.types.to.exclude**

  1 つ以上の指定したタイプの例外を無視する場合はこのプロパティを設定します。指定した各タイプのサブタイプも，**exception.types.to.include** プロパティでそのサブタイプを指定しない限りすべて無視されます。

- **exception.types.to.include**

  指定した除外される例外の中で包含するタイプ（またはそのサブタイプ）がある場合は，このプロパティを設定します。包含するように指定した例外タイプのサブタイプも包含されます。

どちらのプロパティも，完全に修飾された例外タイプ名のカンマで区切られたリストを受け付けます。**dispatcher.properties** ファイルを変更するとただちに有効になります。アプリケーションを再起動する必要はありません。

例外タイプの数の制限

また, **server.properties** で **exception.instance.tree.count** プロパティを設定し，さまざまなタイプの例外の数を指定して，収集する例外ツリー・データを制限することもできます。標準設定では，このプロパティは 4 に設定されています。これはプローブのデータ収集サイクル中に発生した最初の 4 つの例外タイプだけを使用して例外ツリーを構築することを示します。この設定は増減できます。

例

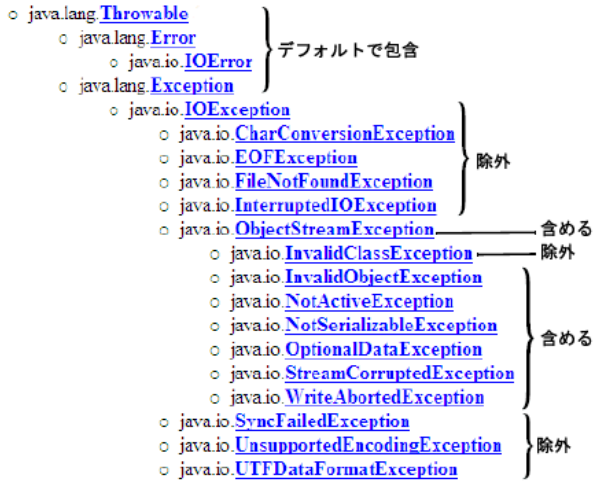
次の例では，タイプが ClassNotFoundException とそのすべてのサブタイプの例外が無視されます。

```
...
exception.types.to.exclude=javax.naming.AuthenticationException
```

次の例では，その次の図に示されているように，java.lang.IOException クラスのいくつかのサブタイプが除外されます。

```
...
exception.types.to.exclude=java.io.IOException,java.io.InvalidClassException
exception.types.to.include=java.io.ObjectStreamException
```

次の図は，java.io クラス階層のどの例外タイプが除外され，包含されるかを示しています。



Diagnostics プローブの［管理］ページ

Diagnostics プローブの［管理］ページを使用して，Java Agent と Profiler の設定を行うことができます。Diagnostics プローブの管理ページには，ブラウザから直接アクセスします。

Diagnostics プローブの［管理］ページへのアクセス

ブラウザで Diagnostics プローブの管理ページを開きます。

Diagnostics プローブの管理ページにアクセスするには，次の手順を実行します。

1 ブラウザで，http:// ＜プローブのホスト＞：＜プローブのポート＞にアクセスします。

プローブは，**35000** で始まる最初の空きポートに割り当てられます。

管理ページが開きます。

2 実行する活動項目に該当するメニュー・オプションを選択します。

- **Diagnostics Profiler を開きます**：Java Diagnostics Profiler を開きます。
- **高度なオプション**：［コンポーネント］ページを開きます。詳細については，472 ページ「Diagnostics プローブの［コンポーネント］ページ」を参照してください。
- **権限と認証を管理します**：プローブの設定方法に応じて，このオプションからさまざまなページにアクセスします。

- プローブが Diagnostics サーバと一緒に動作するように設定されている場合，プローブ（Profiler）の認証と承認設定は，このプローブが接続されている Diagnostics コマンド・サーバから管理されます。このオプションをクリックすると，その Diagnostics コマンド・サーバにリダイレクトされます。詳細については，付録 B，「ユーザの認証と承認」を参照してください。
- プローブが Profiler 専用で動作するように設定されており，Diagnostics サーバに接続されていない場合，このオプションによって［ユーザ管理］ページが開きます。このページで，ユーザを作成，編集，削除したり，ユーザの権限を変更できます。詳細については，474 ページ「Diagnostics Java Profiler での権限と認証」を参照してください。

Diagnostics プローブの［コンポーネント］ページ

［コンポーネント］ページから，Java Diagnostics Profiler を開いたり，［ユーザ管理］ページにアクセスできます。

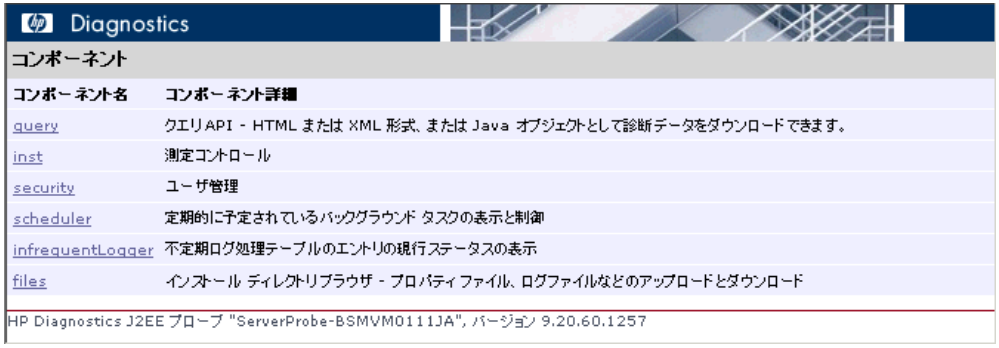
［コンポーネント］ページにアクセスするには，次の手順を実行します。

1 471 ページ「Diagnostics プローブの［管理］ページへのアクセス」の説明に従って，Diagnostics プローブの管理ページを開きます。

2 ［**高度なオプション**］をクリックします。

3 プロンプトが表示されたら，ユーザ名とパスワードを入力します。

［コンポーネント］ページが開きます。



4 次のオプションのいずれかをクリックします。

- **query :** 開発者による内部使用専用です。
- **inst :** さまざまなインストルメンテーション・オプションが含まれます。プローブのインストルメンテーションの詳細については，297 ページ「Java アプリーションのカスタム・インストルメンテーション」を参照してください。
- **security :** プローブの設定方法に応じて, このオプションからさまざまなページにアクセスします。
  - プローブが Diagnostics サーバと一緒に動作するように設定されている場合，プローブ（Profiler）の認証と承認設定は，このプローブが接続されている Diagnostics コマンド・サーバから管理されます。このオプションをクリックすると，その Diagnostics コマンド・サーバにリダイレクトされます。詳細については，755 ページ「ユーザの認証と承認」を参照してください。
  - プローブが Profiler 専用で動作するように設定されており, Diagnostics サーバに接続されていない場合，このオプションによって [ユーザ管理] ページが開きます。このページで，ユーザを作成，編集，削除したり，ユーザの権限を変更できます。詳細については，474 ページ「Diagnostics Java Profiler での権限と認証」を参照してください。
- **scheduler :** 定期的に予定されているバックグラウンド・タスクを表示し，制御できるようにします。ServerCommunication スケジューラまたは sharedInfrequentEventScheduler に対して, スケジューラ内のタスクの状態と数を表示できます。タスクごとに RUN NOW（すぐに実行）や DELETE（削除）などのアクションを選択できます。
- **infrequentLogger :** 不定期のログ記録テーブルでエントリの現在のステータスを参照します。
- **files :** プロパティ・ファイル，ログ・ファイルなどのアップロードとダウンロードに使用するためのインストール・ディレクトリ・ブラウザ。

Diagnostics Java Profiler での権限と認証

Java Agent を Profiler 専用でインストールしている（Diagnostics サーバに接続していない）場合，Diagnostics プローブの［ユーザ管理］ページから Profiler のユーザの認証と承認を管理できます。

注 : Java Agent が Diagnostics サーバと連携して動作するように設定されている場合，プローブ（Profiler）の認証と承認設定は，このプローブが接続されている Diagnostics コマンド・サーバから管理します。詳細については，755 ページ「ユーザの認証と承認」を参照してください。

スタンドアロンの Java Diagnostics Profiler のユーザの認証と承認を管理するには，次の手順を実行します。

1 Diagnostics プローブの管理ページにアクセスします。

ブラウザで，http:// ＜プローブのホスト＞：＜プローブのポート＞にアクセスします。プローブは，35000 で始まる最初の空きポートに割り当てられます。

Diagnostics プローブの管理ページが開きます。

2［権限と認証の管理］を選択して，［ユーザ管理］ページを開きます。

［ユーザ管理］ページでは，新しいユーザの作成，ユーザへの権限の割り当て，既存ユーザのパスワードの変更，ユーザの削除を実行できます。

新しいユーザを作成するには，次の手順を実行します。

1 ［**ユーザの作成**］をクリックして，［**新規ユーザ名**］ボックスにユーザ名を入力し，［**OK**］をクリックします。ユーザ名のリストに新しいユーザが表示されます。

2 新しいユーザを表す行で，［**パスワード**］ボックスにパスワードを入力し，［**パスワードの確認**］ボックスに再入力して確認します。

3 ［**<現在のユーザ>のパスワード**］ボックスに現在ログオンしているユーザのパスワードを入力して，［**変更を保存**］をクリックします。

ユーザに権限を割り当てるには

1 関連するユーザを表す行で，さまざまな権限を表す適切なチェック・ボックスを選択します。

Java Diagnostics Profiler ユーザに割り当てられる権限レベルは次のとおりです。

権限	説明
表示	ユーザは，UI から Profiler のデータを表示できます。
実行	ユーザは，ガベージ・コレクションを実行して，Profiler で保持しているパフォーマンス・データをクリアできます。
変更	ユーザは，ヒープダンプの取得やインストルメンテーションの変更といった，潜在的に危険を伴う操作を実行できます。

注 : 権限レベルの **rhttpout** と **system** は，内部使用のみを目的としています。

各権限レベルは独立しています。ある権限レベルから次の権限レベルへの継承はありません。1 人のユーザに，実行に必要なすべての権限レベルを付与できます。

2 [**<現在のユーザ>のパスワード**] ボックスに現在ログオンしているユーザのパスワードを入力して，[**変更を保存**] をクリックします。

既存ユーザのパスワードを変更するには，次の手順を実行します。

1 関連ユーザを表す行で，[**パスワード**] ボックスにパスワードを入力し，[**パスワードの確認**] ボックスに再入力して確認します。

2 [**<現在のユーザ>のパスワード**] ボックスに現在ログオンしているユーザのパスワードを入力して，[**変更を保存**] をクリックします。

ユーザを削除するには，次の手順を実行します。

1 ［**<現在のユーザ>のパスワード**］ボックスに現在ログオンしているユーザのパスワードを入力します。

2 削除するユーザに対応する［**ユーザの削除**］ボタン をクリックします。

選択したユーザを削除するかどうかを尋ねるメッセージ・ボックスが表示されます。

3 ［**OK**］をクリックしてユーザを削除します。

CPU 時間メトリックスの収集の構成

CPU 時間メトリックスは，Transaction ビュー，プローブ・ビュー，Call Profile ビュー，Portal Components ビューの［詳細］表示枠に表示されます。CPU 時間メトリックスの収集の有効化，無効化，設定を実行できます。CPU 時間測定値は **CPU（Avg）**と **CPU（Total）**です。CPU 時間測定値の収集が無効になっているか，メソッドが設定されていない場合，測定値には N/A が表示されます。

CPU 時間メトリックスは，Windows，Solaris，AIX，HP-UX，Linux kernels 2.6.10 以降（RedHat 5.x，SUSE 10.x など）のプラットフォームで一般にサポートされている CPU タイムスタンプを使用します。

注 : ただし，CPU タイムスタンプのサポートはオペレーティング・システムだけでなく，プラットフォーム・アーキテクチャ（SPARC，x86 など）によっても異なります。
特定のプラットフォーム・バージョンおよびアーキテクチャでの CPU 時間のサポートに関する最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

重要 : VMware では，CPU 時間の測定値はゲスト・オペレーティング・システム側から見た測定値であり，VMware 仮想タイマの影響を受けます。http://www.vmware.com/pdf/vmware_timekeeping.pdf にある時間管理に関する VMware のホワイトペーパー，および 468 ページ「VMware 上で実行中のプローブの時刻の同期」を参照してください。

標準設定では，CPU 時間測定値の収集は，サーバ要求に対して有効に設定されています。CPU 時間メトリックスの収集を無効にしたり，CPU 時間メトリックスの収集をプロパティ・ファイルまたは Java Diagnostics Profiler の UI で設定したりできます。次の CPU 時間メトリックスの収集を設定できます。

- サーバ要求のみ
- サーバ要求とポートレット・メソッド
- サーバ要求とすべてのメソッド

Java Agent の場合，CPU 時間メトリックスの収集は次の 2 つのプロパティによって制御されます。

- **＜プローブのインストール・ディレクトリ＞ ¥etc¥capture.properties** の **use.cpu.timestamps** プロパティ

  このプロパティは標準設定で **true** に設定されており, CPU 時間メトリックスの収集が有効になっています。CPU スタンプの収集は，次に示す 2 つ目のプロパティによって制御されます。use.cpu.timestamps プロパティを false に設定すると，CPU 時間メトリックスは，プローブによってレポートされたどのサーバ要求またはメソッドに対しても収集されません。

- **＜プローブのインストール・ディレクトリ＞ ¥etc¥dynamic.properties** の **cpu.timestamp.collection.method** プロパティ

注 : Diagnostics のオーバーヘッドが増加するため, CPU タイムスタンプの収集を設定にする際は十分に注意してください。オーバーヘッドの増加は，タイムスタンプの収集に必要な各メソッドの追加呼び出しによって生じます。

Cpu.timestamp.collection.method は，次のいずれかの値に設定できます。

- **0 –** CPU タイムスタンプはありません。
- **1 –** CPU タイムスタンプはサーバ要求に対してのみ収集されます。

  標準設定値は **1** です。これは，CPU 時間をトランザクション・レベルではなく，サーバ要求レベルでレポートできることを示します。ただし，プロパティ・ファイルから設定が削除されたりコメント・アウトされたりした場合，標準設定値は 0 になります。

- **2 –** CPU タイムスタンプはすべてのサーバ要求およびすべてのメソッドに対して収集されます。
- **3 –** CPU タイムスタンプはすべてのサーバ要求とポータル・コンポーネントでインストルメントされたライフサイクル・メソッドに対して収集されます。

cpu.timestamp.collection.method プロパティを設定するもう 1 つの方法としては，次のように Java Diagnostics Profiler の［構成］タブを使用します。

1. **Profiler** UI で，［**構成**］タブを選択します。このプローブ設定の変更を行うために Profiler を起動する必要はありません。
2. ［構成］画面で，ドロップダウン・リストから［**CPU タイムスタンプの収集**］オプションを選択します。

CPU タイムスタンプ収集メソッド	説明
なし	CPU タイムスタンプはありません。
サーバ要求に対してのみ	CPU タイムスタンプはサーバ要求に対してのみ収集されます。
サーバ要求とポートレット・メソッドに対して	CPU タイムスタンプはすべてのサーバ要求とポータル・コンポーネントでインストルメントされたライフサイクル・メソッドに対して収集されます。
サーバ要求とすべてのメソッドに対して	CPU タイムスタンプはすべてのサーバ要求とすべてのメソッドに対して収集されます。

3 変更を完了したら，［**変更の適用**］をクリックします。

注 : 変更はただちに有効になります。アプリケーション・サーバのホスト（またはプローブ）は再起動する必要がありません。

コンシューマ ID の構成

Web サービス・メトリックスは，Web サービスの特定のコンシューマ別にグループ化できます。メトリックスはその後，そのコンシューマに対して集計され，Services by Consumer（コンシューマ ID 別サービス）や，Operations by Consumer ID（コンシューマ ID 別動作）などの SOA サービス・ビューに表示されます。コンシューマ ID を定義するには，次のように複数の方法があります。

- SOAP ヘッダ内の値
- SOAP エンベロープ内の値
- SOAP ボディ内の値
- HTTP ヘッダ内の値
- SOAP over JMS Web サービスの JMS キュー名（またはトピック名）
- SOAP over JMS Web サービスの JMS メッセージ・プロパティ
- SOAP over JMS Web サービスの JMS メッセージ・ヘッダ
- 特定の IP アドレス
- IP アドレスの範囲

重要 : SOAP のヘッダ，エンベロープ，ボディに基づいてコンシューマ ID を定義するには，Java プローブの Diagnostics SOAP メッセージ・ハンドラが必要です。一部のアプリケーション・サーバについては，Diagnostics SOAP メッセージ・ハンドラを自動的に読み込むための特別なインストルメンテーションが Diagnostics に用意されています。

ただし, WebSphere 5.1 JAX-RPC と Oracle 10g JAX-RPC については, 手動設定が必要です。詳細については，237 ページ「Diagnostics SOAP メッセージ・ハンドラの読み込み」を参照してください。

Diagnostics SOAP メッセージ・ハンドラはすべてのアプリケーション・サーバに対して利用できるわけではありません。カスタム・インストルメンテーションは, SOAP ペイロードから SOAP エラーやコンシューマ ID をキャプチャするのには使用できません。したがって，この機能はアプリケーション・サーバやそのバージョンによっては使用できない場合があります。Diagnostics SOAP メッセージ・ハンドラのサポートの最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

コンシューマ ID 別のデータの集計は, 誰がどのサービスを使用しているかとその使用頻度を調べる場合に役立ちます。コンシューマ ID は Business Service Management に対しても有用です。Business Service Management ユーザは, コンシューマに基づいて同一のアプリケーションのパフォーマンスを表示して，パフォーマンスの特徴を比較できます。

コンシューマ ID の設定は任意です。標準設定では，SOAP over HTTP/S Web サービスのコンシューマ ID として IP アドレスが使用され, SOAP over JMS Web サービスの標準設定のコンシューマ ID として受信キュー名（またはトピック名）が使用されます。

本項の内容

コンシューマ ID 設定の基本手順

コンシューマ ID を設定するための基本手順は次のとおりです。

1 （この手順は省略可能です）。**consumer.properties** ファイルの ***dump-payload** を指定し，SOAP ペイロード全体を **consumer.log** ファイルに印刷します。この出力を使用して，SOAP ペイロード・キャプチャのコンシューマ ID を設定する特定のルールを作成する方法を計画します。

コンシューマ ID の設定前に，SOAP ペイロードデータに慣れ，コンシューマ ID 用の値を見つけるために Diagnostics が使用するルールの作成方法を決定します。

dump-payload オプションは，コンシューマ ID を含む要素を見つけるために補助が必要なときにのみ使用します。

このオプションは，使うときは，等号（=）の右側の唯一の値にする必要があります。DumpTest;HTTP_WS;TraderService = *dump-payload

---

重要 : 値を抽出し，同時にペイロードをダンプするために，同じサービス名を使用しないでください。

たとえば，次のように入力してこの機能を使用します。

SoapTest1;HTTP_WS;TraderService = *dump-payload

この結果，TraderService に一致するルールの SOAP ペイロードが印刷されます。consumer.log ファイルの内容は次のとおりです。

```
2009-01-15 14:42:13,653 INFO consumer [[ACTIVE] ExecuteThread: '0' for queue:
'weblogic.kernel.Default (self-tuning)'] [PAYLOAD:] <?xml version="1.0" encoding="UTF-8"
standalone="yes"?><soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:trad="http://
www.bea.com/examples/Trader"      xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/">
   <soapenv:Header>
    <CallerA>customerA</CallerA>
   </soapenv:Header>
   <soapenv:Body>
      <trad:buy soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <string xsi:type="xsd:string">hpq</string>
         <intVal xsi:type="xsd:int">11</intVal>
      </trad:buy>
   </soapenv:Body>
</soapenv:Envelope>
```

2 コンシューマ別にグループ化したメトリックスが必要な Java Agent ごとに，485 ページ「Java Agent に関するコンシューマ ID のルール構文と例」で説明したように **consumer.properties** ファイルを更新します。

3 6 つ以上のコンシューマ・タイプを追跡する場合は，**dispatcher.properties** ファイルの **max.tracked.ids.per.probe** 設定を更新します。

4 **probe/files/log** ディレクトリにある **＜プローブの名前＞_id.properties** ファイルを確認します。直前の手順で行った **consumer.properties** の変更に一致させるためには，**＜プローブの名前＞ _id.properties** ファイルを完全に削除するか，または変更する必要があります。このファイルは max.tracked.ids.per.probe（dispatcher.properties）設定と関連しており，プローブごとの制限に達すると，ほかのコンシューマはすべて「Other」としてグループ化されます。

コンシューマ ID のルール

コンシューマ ID の割り当ては，設定ファイル **consumer.properties** のコンシューマ ID ルールによって制御されます。

コンシューマ ID のカテゴリごとに独自のルールがあります。つまり，SOAP ルール，HTTP ヘッダ・ルール，JMS Web サービス・ルール，IP ルールです。ルールは，ルールが定義された方法に従って適用されるわけではありません。SOAP ヘッダ・ルールが最初に適用され，次に HTTP ヘッダ・ルール，次に JMS ルール，最後に IP ルールが適用されます。

重要 : ルール内のすべての設定項目で，大文字と小文字が区別されます。このため，たとえば TraderService の＜パターンの名前＞を入力した場合，パターンが一致するためには Web サービス名に大文字の T と大文字の S が必要です。

すべてのルール・タイプを使用する必要はありません。SOAP ルールがあり，HTTP ルールと IP ルールがない場合もあります。これらのルールのいずれにも一致するものがない場合，元の IP アドレスまたは JMS のキュー名がコンシューマ ID として使用されます。

SOAP ルールを使用して，SOAP ヘッダ，エンベロープ，または本文の XML 要素からコンシューマ ID を取得できます。ルールは，コンシューマによって呼び出される Web サービス名との照合に使われる正規表現を指定します。正規表現を使用したヘルプについては，882 ページ「正規表現の使用」を参照してください。

一致するものがあれば，プローブはルール内で指定されているテキスト要素を検索します。SOAP ヘッダ内にテキスト要素が見つからない場合，このルールはスキップされ，プローブは定義されている次のルールに進みます。

HTTP ヘッダ・ルールを使用すると，コンシューマ ID を HTTP 要求内の一連の HTTP のヘッダから取得できます。

JMS Web サービス・ルールを使用すると，コンシューマ ID を JMS キュー / トピック名，および JMS メッセージ・プロパティまたはメッセージ・ヘッダ（JMSReplyTo のみ）にできます。

IP ルールを使用すると，コンシューマ ID を IP アドレスとコンシューマ ID のマッピングから取得できます。ルールを使用して，コンシューマ ID に割り当てる IP アドレスや IP アドレスの範囲を定義します。

Java Agent に関するコンシューマ ID のルール構文と例

コンシューマ ID の割り当ては **consumer.properties** ファイルでルールを指定して制御します。

重要 : すべての設定項目で，大文字と小文字が区別されます。このため，たとえば TraderService の＜パターンの名前＞を入力した場合，パターンが一致するためには Web サービス名に大文字の T と大文字の S が必要です。

SOAP ヘッダ内の値

SOAP ヘッダ内の値に基づいてコンシューマ ID を割り当てるには，次の書式を使用します。

<rule-name>**;HTTP_WS;**<pattern-name> **= soap-header;**<element-value>

<rule-name> はルールを識別する文字列です。名前は consumer.properties ファイルにとって一意である必要があります。

<pattern-name> は, Web サービス名を検索する正規表現です。正確な Web サービス名を使用することもできます。

<element-value> は，コンシューマ ID として使用する値を含む SOAP エンベロープ内の要素です。

たとえば，次のルールは，サービス名が TraderService の Web サービスを検索し，CallerA 要素の値をコンシューマ ID として使用します。

```
SoapRule1;HTTP_WS;TraderService = soap-header;CallerA
```

TraderService Web サービスの呼び出し元が CallerA の値を定義しているとき，メトリックスは CallerA に対してさまざまな値でグループ化されます。soap ヘッダからの次の抜粋は，TraderService の呼び出し元の「Customer2」というコンシューマ ID にマップされます。

```
SoapTest1;WS<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/
envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <env:Header>
      <CallerA>Customer2</CallerA>  <---- The consumer id returned would be
                                              "Customer2"
   </env:Header>
   <env:Body   env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <m:sell    xmlns:m="http://www.bea.com/examples/Trader">
         <string     xsi:type="xsd:string">sample string</string>
         <intVal     xsi:type="xsd:int">100</intVal>
      </m:sell>
   </env:Body>
</env:Envelope>
```

標準設定で，Diagnostics は第 1 レベル要素（SOAP env:Header の真下の要素）の CallerA を探します。コンシューマ ID 用により深いレベルの xml 要素を調べるよう Diagnostics を設定できます。**consumer.properties** ファイル内の動的プロパティ **max.search.level.depth** は，コンシューマ ID を検索する深さを制御します（標準設定値の深さは 1 レベルです）。たとえば，max.search.level.depth = 2 の場合，コンシューマ ID が検出されます。

```
<env:Header>
   <test:id>
      <test:CallerA>consumerA</test:CallerA>
   </test:id>
</env:Header>
```

SOAP エンベロープ内の値

SOAP エンベロープ内の値に基づいてコンシューマ ID を割り当てるには，次の書式を使用します。

＜ルール名＞ ;HTTP_WS; ＜パターン名＞ = soap-envelope; ＜要素値＞

<rule-name> はルールを識別する文字列です。名前は consumer.properties ファイルにとって一意である必要があります。

<pattern-name> は，Web サービス名を検索する正規表現です。正確な Web サービス名を使用することもできます。

<element-value> は，コンシューマ ID として使用する値を含む SOAP エンベロープ内の要素です。

SOAP ボディ内の値

SOAP ボディ内の値に基づいてコンシューマ ID を割り当てるには，次の書式を使用します。

<rule-name>**;HTTP_WS;**<pattern-name> **= soap-body;**<element-value>

<rule-name> はルールを識別する文字列です。名前は consumer.properties ファイルにとって一意である必要があります。

<pattern-name> は，Web サービス名を検索する正規表現です。正確な Web サービス名を使用することもできます。

<element-value> は, コンシューマ ID として使用する値を含む SOAP ボディ内の要素です。

HTTP ヘッダ内の値

HTTP ヘッダ内の値に基づいてコンシューマ ID を割り当てるには，次の書式を使用します。

<rule-name>**;HTTP_WS;**<pattern-name> **= attribute;**<header-value>

<rule-name> はルールを識別する文字列です。名前は consumer.properties ファイルにとって一意である必要があります。

<pattern-name> は URI を検索する正規表現です。

<header-value> は，コンシューマ ID として使用する値を含む HTTP ヘッダです。

たとえば，次のルールは，URI が「/webservice/.*」の Web サービスを検索し，「User-Agent」ヘッダの値をコンシューマ ID として使用します。

```
WsRule1;HTTP_WS;/webservice/.* = attribute;User-Agent
```

Web サービスの呼び出し元が User-Agent の値を定義しているとき，メトリックスは User-Agent に対してさまざまな値でグループ化されます。HTTP ヘッダからの次の抜粋は，ヘッダのコンシューマ ID にマップされます。

```
GET /service/call HTTP/1.1
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 2000)
Host: ovrntt1
Caller: ovrntt1
Connection: Keep-Alive
```

JMS キュー名

JMS キュー/ トピック名の検索結果に基づいてコンシューマ ID を割り当てるには，次の書式を使用します。

<rule-name>;**JMS_WS;**<queue-name>=<consumerID-string>

<rule-name> はルールを識別する文字列です。名前は consumer.properties ファイルにとって一意である必要があります。

<queue-name> は，JMS キュー / トピック名を検索する正規表現です。

<consumerID-string> は，コンシューマ ID として使用するリテラル文字列です。

たとえば，次のルールは queue://sca_soapjms.* で始まる JMS キュー名を検索し，文字列「myJMSConsumer」をコンシューマ ID として使用します。

```
JMSTest3;JMS_WS;queue¥://sca_soapjms.*=myJMSConsumer
```

キューまたはトピックの後の「:」は，円マーク「¥:」を使用してエスケープする必要があります。

検索に使われる優先順位は，consumer.properties ファイルで指定された順序によって決まります。JMS_WS キューの検索は IP の検索より優先されます。JMS_WS プロパティの検索は JMS_WS ヘッダの検索より優先されます。JMS_WS ヘッダの検索は JMS_WS キュー名の検索より優先されます。

JMS メッセージ・プロパティ

JMS キュー/ トピック名の検索に基づいてコンシューマ ID を割り当て，JMS メッセージ・プロパティの値をコンシューマ ID として使用するには，次の書式を使用します。

<rule-name>;**JMS_WS;**<queue-name>**=jms-property**;<property-value>

<rule-name> はルールを識別する文字列です。名前は consumer.properties ファイルにとって一意である必要があります。

<queue-name> は，JMS キュー / トピック名を検索する正規表現です。

<property-value> は，コンシューマ ID として使用する値を含む JMS プロパティです。

たとえば，次のルールは queue://MedRec.* で始まる JMS キュー名を検索し，JMSXDeliveryCount プロパティの値をコンシューマ ID として使用します。

```
JMSTest1;JMS_WS;queue¥://MedRec.*=jms-property;JMSXDeliveryCount
```

キューまたはトピックの後の「:」は，円マーク「¥:」を使用してエスケープする必要があります。

JMS メッセージ・ヘッダ

JMS キュー / トピック名と JMS メッセージ・ヘッダの検索結果に基づいてコンシューマ ID を割り当てるには，次の書式を使用します。

<rule-name>;**JMS_WS**;<queue-name>**=jms-header**;<header-value>

<rule-name> はルールを識別する文字列です。名前は consumer.properties ファイルにとって一意である必要があります。

<queue-name> は，JMS キュー / トピック名を検索する正規表現です。

<header-value> は JMSReplyTo である必要があります。

たとえば，次のルールは queue://MedRec.* で始まる JMS キュー名を検索し，JMSReplyTo ヘッダの値をコンシューマ ID として使用します。

```
JMSTest1;JMS_WS;queue¥://MedRec.*=jms-header;JMSReplyTo
```

キューまたはトピックの後の「:」は，円マーク「¥:」を使用してエスケープする必要があります。

特定の IP アドレス

IP アドレスに基づいてコンシューマ ID を割り当てるには，次の形式を使用します。

<rule-name>; **IP;** <IP-address> = <consumerID-string>

たとえば，次のルールは，IP アドレス 123.456.567.8 を検索し，名前「CustomerA_IP」をコンシューマ ID として使用します。

```
IPRule1;IP;123.456.567.8 = CustomerA_IP
```

IP アドレスの範囲

IP アドレスの範囲に基づいてコンシューマ ID を割り当てるには, 次の書式を使用します。

<rule-name>; **IP**; <IP address range> = <consumerID-string>

<IP address range> は，整数，***** で指定されるワイルドカード，**-** で指定される整数範囲で定義できます。

次のルールでは, 最初のオクテットが 15 であるすべての IP アドレスを検索し, 名前「mySuperCluster」をコンシューマ ID として使用します。

```
IPRule2;IP;15.*.*.* = mySuperCluster
```

次のルールは，最初のオクテットが 15，2 番目のオクテットが 200 ～ 300 の間であるすべての IP アドレスを検索します。「コンシューマ ID」として Customer_IP を使用します。

```
IPRule3;IP;15.200-300.*.* = Customer_IP
```

SOAP の障害ペイロード・データの構成

SOAP の障害が検出された場合, SOAP の障害データに SOAP ペイロードを含めることができます。SOAP ペイロードは，SOAP の障害が発生した場合のみにキャプチャされます。

Diagnostics UI では，インスタンス・ツリーの一部としてペイロード情報を表示できます。JAX-WS および JAX-RPC Web サービスがサポートされています。

ペイロードにはクレジット・カード番号などの機密情報が含まれる可能性があるため, SOAP の障害でのペイロードのキャプチャは標準設定では無効になっています。

SOAP 障害のペイロード・キャプチャを有効にするには，Java Agent の **dispatcher.properties** ファイルの **max.soap.payload.bytes** をゼロよりも大き値（推奨値は 5000）に設定します。これは，キャプチャされるバイト数であるため，ペイロードが小さすぎることが UI で示されている場合，この数値を増やすことができます。標準設定では，この値はゼロに設定されており，ペイロード・キャプチャは無効になっています。

SOAP ペイロードをキャプチャするには，Java プローブの Diagnostics SOAP メッセージ・ハンドラが必要です。一部のアプリケーション・サーバについては，Diagnostics SOAP メッセージ・ハンドラを自動的に読み込むための特別なインストルメンテーションが Diagnostics に用意されています。
WebSphere 5.1 JAX-RPC と Oracle 10g JAX-RPC については，手動設定が必要です。詳細については，237 ページ「Diagnostics SOAP メッセージ・ハンドラの読み込み」を参照してください。

また，一部のアプリケーション・サーバでは Diagnostics SOAP メッセージ・ハンドラが使用できず，カスタム・インストルメンテーションを使用して SOAP ペイロードから SOAP エラーやコンシューマ ID をキャプチャすることもできません。したがって，この機能はアプリケーション・サーバやそのバージョンによっては使用できない場合があります。Diagnostics SOAP メッセージ・ハンドラのサポートの最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

Java Agent では，**＜プローブのインストール・ディレクトリ＞ ¥etc¥dispatcher.properties** ファイルを変更することによってペイロード・サイズの制限を定義します。指定したサイズよりも大きいペイロードは切り捨てられます。

たとえば，次のエントリは，SOAP のペイロードの長さを標準設定の 5000 から 10000 に増やしています。

```
max.soap.payload.bytes = 10000
```

この機能を無効にする場合はこのプロパティを 0 に設定します。

REST サービスの構成

REST スタイルの Web サービスを設定して通常の Web サービスとして Diagnostics UI に表示できます。設定の詳細については，**＜プローブのインストール・ディレクトリ＞ ¥etc¥rest.properties** ファイルのコメントを参照してください。

現在，HTTP のみがサポートされています（JMS なし）。

JMS 一時キュー / トピックのグループ化のカスタマイズ

Diagnostics のレポートでは，SOAP over JMS の一時キューは 1 つのノードにグループ化されます。Diagnostics は，キュー / トピック名を正規表現のリストと照合して一時キュー / トピック名を見つけます。一致したものは，その種類に従って **queue: ＜ Probe ID ＞ ¥TEMPORARY** または **topic: ＜ Probe ID ＞ ¥ TEMPORARY** に置き換えられます。

この照合に使用される正規表現のリストは，**＜プローブのインストール・ディレクトリ＞ /etc/capture.properties** ファイルにあります。正規表現のリストは，プロパティ **grouped.temporary.jms.names** でカスタマイズできます。

SQL クエリの解析の設定

リテラルを使用する SQL クエリが多数存在する場合，サーバの記号テーブルが過負荷になる可能性があります。このような場合，Java Agent の **dispatcher.properties** ファイルの **sql.parsing.mode** プロパティを設定できます。次のモード設定を使用できます。

1 - メソッドのみ，SQL クエリなし。

2 - SQL クエリのメイン・カテゴリ（select/update/insert/delete/...）。

3 -（標準設定）SQL クエリ全体で類似ステートメントを単一測定値に集約する測定値（リテラル，キーワードの大文字と小文字などは無視）。

4 - SQL クエリ全体で同一ステートメントのみを集約する測定値。

```
sql.parsing.mode = 3
```

一時データベース・テーブルの場合，**dispatcher.properties** ファイルの別のプロパティを使用して，収集されるさまざまな SQL ステートメントの数を制限できます。これにより, SQL ステートメントの正規表現による代替を使用してテーブル名をまとめることができます。このプロパティは **sql.pattern.replace** です（詳細については，**dispatcher.properties** ファイルのコメントを参照）。

サーバ要求に対するアプリケーション名の表示構成

Diagnostics UI の［サーバ要求］詳細表示枠に表示される［**展開先**］値で，ほとんどのアプリケーション・サーバのアプリケーション名を表示できます。Diagnostics 9.0 より以前は，この情報は WebLogic アプリケーション・サーバについてしか入手できなかったため, WebLogic Probe のみがサーバ要求でアプリケーション名識別子を入力できました。

サーバ要求トレンド・ラインとの下位互換性を確実にするため，WebLogic サーバ要求を除いて，アプリケーション名はサーバ要求については入力しません（標準設定）。

これは **capture.properties** ファイルの **keep.fragment.data.compatible** プロパティを使用して設定できます。標準設定では, keep.fragment.data.compatible=**true** です。これは WebLogic サーバ要求を除いて，サーバ要求に対してアプリケーション名は入力されないという意味です。

Diagnostics UI に各サーバ要求の J2EE アプリケーション・サーバ名を表示する場合は，このプロパティを **false** に設定します（［サーバ要求］ビューの詳細表示枠に［展開先］として表示されます）。

Java Profiler UI からのプローブ設定の保守

Java Diagnostics Profiler の［構成］タブを使用して，インストルメンテーション・ポイントを保守し，Java Agent キャプチャ・ポイント・ファイルまたはプロパティ・ファイルを手動で編集することなくプローブ設定を編集できます。プロファイル処理を開始しているかどうかに応じて，Java Diagnostics Profiler から［構成］タブにアクセスできます。

Java Diagnostics Profiler の［構成］タブの［プローブ設定］セクションを使用して，スレッドのスタック・トレースのサンプリング，CPU 時間メトリックスの収集（タイムスタンプを使用），コレクション・リークの報告についてプローブを設定できます。

Java Diagnostics Profiler の［構成］タブで［**変更を適用**］をクリックすると，［構成］タブの［プローブ設定］セクションで行ったすべての更新が，キャプチャ・ポイント・ファイルとプロパティ・ファイルに適用されます。

注 : 変更はただちに有効になります。アプリケーション・サーバのホスト（またはプローブ）を再起動する必要はありません。

次の項では，［プローブ設定］の各セクションについて説明します。

➤ 496 ページ「スレッドのスタック・トレースのサンプリング構成」
➤ 501 ページ「CPU タイムスタンプの収集の制御」
➤ 502 ページ「コレクション・リーク・レポートの有効化と構成」

スレッドのスタック・トレースのサンプリング構成

非同期のスレッド・サンプリングを有効にすると，長期間実行されたフラグメントの中で実行されたメソッドが呼び出しプロファイル・ビューに表示されます。これは，この期間中にインストルメンテーション対象のメソッドがヒットしなかった場合でも有効です。スレッドのサンプリングに基づいて追加されたノードが表示された画面ショットについては，『HP Diagnostics ユーザー・ガイド』の呼び出しプロファイルに関する章を参照してください。

複数のプロパティによって，スレッドのスタック・トレースのサンプリングが有効になり設定されます。

次のプロパティが **dynamic.properties** にあります。

➤ **enable.stack.trace.sampling** – 非同期スレッドのスタック・トレースのサンプリングが有効になります。取り得る値には，false，auto（標準設定），true があります。

動的なプロパティである enable.stack.trace.sampling を auto に設定すると，選択した（認定した）プラットフォームと JVM 上でプローブが実行されている場合にのみ，スタック・トレースのサンプリングが有効になります。ほかの JVM では，明示的に true に設定する必要があります。JVM ではエラーや中途終了が発生する可能性があるため，十分に注意してください。Diagnostics の Readme を参照してください。

➤ **tardy.method.latency.threshold** – このメソッドのスタック・トレースのサンプリングを試行する前に，インストルメンテーション・ポイントにヒットせずにインストルメンテーション対象のメソッドを実行する必要がある最短時間。このプロパティの主な目的は，スタック・トレースの収集を最も注目すべきケースに限定することで，サンプリングのオーバーヘッドを制御することにあります。

➤ **stack.trace.sampling.rate** – 連続する次のサンプリング試行を行うまでに経過する必要がある時間。

stack.trace.sampling.rate に小さい値を指定すると，サンプリングが頻繁に発生し，データが豊富になりますが，その代償としてオーバーヘッドが増加します。

頻繁なサンプリングによるオーバーヘッドは，主にサーバ要求のレイテンシに影響します。プローブによる CPU の全体的な使用率も増加しますが，この影響はレイテンシの増加ほど深刻ではありません。多くの CPU を持つシステムでは，実際にはプロセスの CPU 消費量は減少します（これは良いことではありません）。

- **stack.trace.depth.max** – JVM によって取得されるスタック・トレースの深さに対する制限。ほとんどの場合，この値を調整する必要はありません。

次のプロパティが **dispatcher.properties** にあります。

- **enable.stack.trace.aggregation** – ブール型のプロパティで，収集された 1 つ以上の連続するスタック・トレースで観察されるノードの結合を相関スレッドに許可します。ただし，それらのノードが単独のメソッド呼び出しを表していないという証拠がある場合は除きます。true に設定すると，追加作成される呼び出しツリー・ノードの数は減りますが，追加ノードの呼び出し数が既知で多いという誤解を与える可能性があります。false に設定すると，各メソッドとメソッドが表示された各スタックに対して 1 つのノードが作成されるため，ノードの呼び出し数が既知で大きいという誤解を与える可能性があります。実際には，スタック・トレースのサンプリングで呼び出しの数を表示することができません。
- **aggregated.stack.trace.validity.threshold** – enable.stack.trace.aggregation プロパティが true に設定されている場合に，**aggregated.stack.trace.validity.threshold** を上回る数の個別のスタック・トレースから作成された呼び出しツリー・ノードだけが報告されます。この設定により，特にサーバ側のノイズ除去とメモリ使用量が制御されます。

プロパティはすべて動的に変更できるので，アプリケーションを再起動する必要はありません。

最初の（dynamic.properties の）4 つのプロパティは，Diagnostics Java Profiler の［構成］タブを使用してリモートで変更できます。変更の後，［**変更を適用**］をクリックし，［構成］タブを使用して加えた変更をすべて適用します。

スレッド・サンプリングの設定例

使用例 1 : あるメソッドの平均レイテンシは 170 ミリ秒ですが，このメソッドの完了までに 1.4 秒かかることが時々あります。任意のフラグメントの呼び出しプロファイルに表示されるメソッドのほとんどは，550 ミリ秒以下で実行されます。問題のメソッドは呼び出される側に対して呼び出しを複数回行うので，呼び出しをインストルメントする必要はありません。

代わりに，スタック・トレースのサンプリングを有効にして，実行時間が長くなる原因を調べます。オーバーヘッドを最小限に抑えるため，tardy.method.latency.threshold を 600 ミリ秒に設定します。こうすると，ほとんどのメソッドはこの時間が経過する前に完了する可能性が高いため，まったくサンプリングされなくなります。ただし，問題の長時間実行されるメソッドを含めて，この値より長く実行されるメソッドは，そのメソッドがインストルメント対象のどのメソッドに対する呼び出しも行わずに 600 ミリ秒（以上）実行されたときに，サンプリングされます。

また，**stack.trace.sampling.rate** の値を 100 ミリ秒に設定すると，理論的には 1.4 秒間続くメソッド呼び出しごとに最大 8 個のサンプルが取得されます（（1400-600）/100）。このメソッドが呼び出される側に対して多くの呼び出しを行うことがわかっているため，**aggregated.stack.trace.validity.threshold** をゼロに設定することもできます。これにより，収集される個々のスタック・トレースが全く異なる場合でも，そのすべてが報告されます。

サーバ要求の長時間実行されるインスタンスの呼び出しプロファイルを調べると，スタック・トレースのサンプリングによって表示された追加ノードがわかります。

使用例 2 : デプロイメント用のカスタム・アプリケーションを準備しますが，呼び出しプロファイルの多くにはアプリケーション固有の動作に関する見通しを与えてくれるメソッドがほとんど含まれていないため，Diagnostics Agent で提供される標準設定のインストルメンテーションではうまく機能しないことがわかっています。パフォーマンスやメモリ消費量に問題があるため，カスタム・アプリケーションに属するすべてのクラスとメソッドに対してインストルメンテーションを追加するのにも抵抗があります。

そこで，スタック・トレースのサンプリングを有効にします。十分に詳細な呼び出しツリーの情報がない一般的なサーバ要求は約 2 秒間実行されると仮定して，**stack.trace.sampling.rate** を 200 ミリ秒に設定します。これにより，一般的なサーバ要求ごとに最大 10 個のスタック・トレースが取得されます。ただし，スタック・トレースに表示されるメソッドの中には瞬時に完了するものもあり，サーバ要求の全体的なレイテンシにはほとんど影響しないため，すべてのスタック・トレースを報告する必要はありません。このため，**aggregated.stack.trace.validity.threshold** を 2 に設定します。これにより，3 個以上の連続するスタック・トレースに表示されるメソッド，または予想されるレイテンシが 600 ミリ秒以上であるメソッドだけが報告されます。

サンプリングで取得された追加ノードを含む呼び出しプロファイルを表示した後は，デプロイメントのプローブ設定へのインストルメンテーション・ポイントの追加に関して，十分な情報に基づいた決定を行うことができます。

スレッドのスタック・トレースのサンプリングに関するトラブルシューティング

質問 1: スタック・トレースのサンプリングを有効にしても，呼び出しプロファイルに新しいノードが表示されないのはなぜですか。

回答 : 次のチェックリストを調べて, 該当する項目があるかどうかを確認してください。

- ❑ Java 1.5 以降を使用していますか。スタック・トレースのサンプリングは，以前のバージョンの Java では機能しません。
- ❑ 呼び出しプロファイルに表示される最後のメソッドは発信呼び出しですか。発信としてマークされたメソッドはサンプリングされません （メソッドが発信としてマークされているかどうかを確実に確認するには，detailReport.txt ファイルでそのメソッドを見つけ，それに対応するインストルメンテーション・ポイントの detail に「outbound」キーワードがあるかどうかを確認します)。
- ❑ 呼び出しプロファイルに表示される最後のメソッドは no-layer-recurse としてマークされていましたか。そのようなメソッドはサンプリングされません（メソッドが no-layer-recurse かどうかを確認するには，直前の項目と同じ手順を使用します)。
- ❑ tardy.method.latency.threshold または minimum.method.latency を小さくしてみますか。呼び出しプロファイルに表示される最後のメソッドがトリミングされた呼び出しを行う可能性がありますが，呼び出し側の tardy.method.latency.threshold のアクティブでない期間がまったくないため，そのような呼び出しではサンプリングが実行されません。

- ❑ aggregated.stack.trace.validity.threshold の値を小さくしてみるか，probe.log ファイルにスタックの深さ不足に関する警告があるかどうかを確認するかしてみましたか。観察するスタック・トレースの変化が速すぎて報告されていない可能性があります。
- ❑ stack.trace.sampling.rate を小さくしようとしましたか。メソッドをサンプリングする機会が失われているだけである可能性があります。
- ❑ 呼び出しプロファイルの最後に表示されるメソッドのレイテンシがガベージ・コレクタの実行によるものではないことを確認しましたか。スタック・トレースのサンプリング・コードを含む Java code はベージ・コレクションの間は実行されません。

質問 2: stack.trace.sampling.rate の最小値として使用できる値は何でしょうか。

回答 : 任意の正数を使用できますが, 各プラットフォームで可能な頻度を超えるサンプリングは拒否されることに留意してください。この場合，sleep() の利用可能な最小粒度，タイマの精度，1 つのサンプル・セットを収集するのにかかる実際の時間という 3 つの要因が影響します。

質問 3: stack.trace.sampling.rate の最大値として使用できる値は何でしょうか。

回答 : 制限はありません。大きな設定値がどの程度有用かは，アプリケーションのサーバ要求のレイテンシに全面的に依存します。結果を得るには，関心のあるサーバ要求ごとに少なくともいくつかのサンプルを計画してください。それにさえ，ほかのサンプリング・パラメータのチューニングが必要な可能性があります。

CPU タイムスタンプの収集の制御

CPU タイムスタンプは，メソッドが使用する独占的な CPU 時間を計算します。この情報は, Java Diagnostics Profiler の［**ホットスポット**］タブに表示できます。

重要 : VMware では，CPU 時間の測定値はゲスト・オペレーティング・システム側から見た測定値であり，VMware 仮想タイマの影響を受けます。http://www.vmware.com/pdf/vmware_timekeeping.pdf にある時間管理に関する VMware のホワイトペーパー，および 468 ページ「VMware 上で実行中のプローブの時刻の同期」を参照してください。

標準設定では，CPU 時間測定値の収集は，サーバ要求に対して有効に設定されています。

CPU 時間メトリックスの収集は，プロパティ・ファイルで（477 ページ「CPU 時間メトリックスの収集の構成」参照）または下記の Java Diagnostics Profiler UI を使用して設定できます。

1 **Profiler** UI で，［**構成**］タブを選択します。このプローブ設定の変更を行うために Profiler を起動する必要はありません。

2 ［構成］画面で，ドロップダウン・リストから［**CPU タイムスタンプの収集**］オプションを選択します。

CPU タイムスタンプ収集メソッド	説明
なし	CPU タイムスタンプはありません。
サーバ要求に対してのみ	CPU タイムスタンプはサーバ要求に対してのみ収集されます。
サーバ要求とポートレット・メソッドに対して	CPU タイムスタンプはすべてのサーバ要求とポータル・コンポーネント（layertype=portlet）でインストルメントされたライフサイクル・メソッドに対して収集されます。
サーバ要求とすべてのメソッドに対して	CPU タイムスタンプはすべてのサーバ要求とすべてのメソッドに対して収集されます。

3［構成］タブの変更が完了したら，［**変更を適用**］をクリックします。

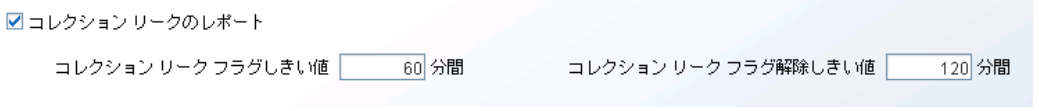
注 : 変更はただちに有効になります。アプリケーション・サーバのホスト（またはプローブ）を再起動する必要はありません。

コレクション・リーク・レポートの有効化と構成

データ収集とコレクション・リークの指摘のレポートは標準設定でプローブ用に有効にされます。プローブの **etc/auto_detect.points** ファイルの **[Collection Leak Pinpointing] keyword = clp** ポイントは標準設定で true に設定されます。

注 : Java Agent でコレクション・リークの指摘（CLP）機能を使用する場合は，アプリケーション・サーバの適切なモードを使用して JRE Instrumenter を実行する必要があります。

Java Profiler の［構成］タブを使用して，コレクション・リーク・レポートのために次の構成アイテムを設定できます。



- **コレクション・リークのレポート :** UI でこの機能のレポートを無効にするには，チェックボックスをオフにします。

- **コレクション・リーク・フラグしきい値 :** コレクションのサイズが増大する継続時間のしきい値。コレクションのサイズが増大する継続期間がこのしきい値を超える場合，プローブによってメモリ・リークとしてフラグを設定されます。
- **コレクション・リーク・フラグ解除しきい値 :** すでにフラグが設定されているリーク中のコレクションで，このしきい値期間について継続的にサイズの増大が停止した場合，プローブはリークとしてのフラグを設定解除します。

これら同じ値は，プローブについて **dynamic.properties** ファイルにも設定できます。**clp.diagnostics.reporting**，**clp.diagnostics.growth.time**，**clp.diagnostics.nongrowth.time**。

JUnit テスト用のパフォーマンス・レポートの生成

JUnit テストを実行する場合，すべてのユニット・テストのパフォーマンス・レポートを生成できるように Java Agent を有効化および設定できます。これは，ある期間に特定のテストのパフォーマンス（レイテンシ /CPU）が変更されているかどうかを調べる場合に便利です。

ユニット・テストが完了したら，Java Agent によってテスト・メソッド（サーバ要求）ごとに CSV ファイルが作成されます。この CSV ファイルには，各 JVM インスタンスで実行されたすべてのテスト・メソッドの完全なリストが含まれています（通常はテスト・クラスごと）。スプレッドシート・プログラムで CSV ファイルを開いて，パフォーマンスの特徴を分析および視覚化できます（Excel のフィルタ機能は，特定のメソッドを選択するのに非常に便利です）。

次に，CSV ファイルの例を示します。

```
Date,Server Request,Avg Latency,Count,Min Latency,Max Latency,Cpu
Time,Exceptions
Fri Sep 23 12:55:22 PDT
2011,UT_SiSXmlDataReader.testDataSample(),1068.81,1,1068.81,1068.81,374.403,0
Fri Sep 23 12:55:40 PDT
2011,UT_SiSXmlDataReader.testDataSample(),1064.845,1,1064.845,1064.845,405.60
2,0
Fri Sep 23 12:55:57 PDT
2011,UT_SiSXmlDataReader.testDataSample(),1141.689,1,1141.689,1141.689,358.80
2,0
Fri Sep 23 12:56:27 PDT
2011,UT_SiSXmlDataReader.testDataSample(),1474.81,1,1474.81,1474.81,468.003,0
```

レイテンシ時間はミリ秒単位です。

標準設定では，各テストの実行データは CSV ファイルに追加されます。これは，継続的な統合サイクルの一環としてテストを実行する場合に特に便利です。これにより，ある期間の結果をキャプチャできます。

この機能を使用するには，JUnit テストの実行で次の JVM パラメータを指定して Java Agent を有効にします。

JVM パラメータ	説明
-javaagent: < Java Agent のホーム> /DiagnosticsAgent/lib/probeagent.jar（UNIX） または -javaagent: < Java Agent のホーム> \DiagnosticsAgent\lib\probeagent.jar（Windows）	Agent の JAR ファイルへのパスを指定して Agent を有効にします。JRE 1.4 を使用している場合，代わりに **-Xbootclasspath/p: < Java Agent のインストール・ディレクトリ> /DiagnosticsAgent/classes/boot** を使用できます。
-Ddispatcher.ac.autostart=true	すぐにプロファイル処理を開始するように Agent に指示します。
-Dcapture.exit_report=dir=perftest:append	指定したディレクトリにパフォーマンス・レポートを生成して結果を追加するように Agent に指示します（ファイルを上書きするには，**append** を **override** に置き換えます）。
-Ddispatcher.minimum.fragment.latency=1ms	レイテンシが 1 ミリ秒以上のサーバ要求（JUnit テスト・メソッドの実行など）のみを収集します。

次の例は，ANT への統合を示しています。

```
<junit dir="${build}" fork="yes" forkmode="perTest" printsummary="yes"
jvm="${env.JAVA_HOME}/bin/java">

    ...

    <jvmarg value="-javaagent:C:/MercuryDiagnostics/JavaAgent/DiagnosticsAgent/lib/
probeagent.jar"/>
    <jvmarg value="-Ddispatcher.ac.autostart=true"/>
    <jvmarg value="-Dcapture.exit_report=dir=<dir_name>:append"/>
    <jvmarg value="-Ddispatcher.minimum.fragment.latency=1ms"/>

    ...

</junit>
```

上記の設定のほかに，**＜ Java Agent のホーム＞ /DiagnosticsAgent/etc/auto_detect.points** で JUnit ポイントをアクティブ化（**active=true** に設定）する必要があります。

```
[JUnit]
class      = junit.framework.TestCase
method     = !test.*
signature = !.*
deep_mode  = hard
layer      = JUnit
active     = true
```

注 : JUnit 4.x を使用していて，ユニット・テスト・クラスが junit.framework.TestCase のサブクラスでない場合，ユニット・テスト・クラスに合うように上記の JUnit ポイントのクラス定義を変更する必要があります。

13

.NET Agent 設定ファイルについて

設定ファイル **<プローブのインストール・ディレクトリ>/etc/probe_config.xml** の要素と属性を変更して，.NET Agent の設定を制御します。

本章の内容

.NET Agent 設定ファイルについて

このセクションのトピックでは, .NET Agent の設定ファイルである **<プローブのインストール・ディレクトリ> /etc/probe_config.xml** を構成する要素と属性について説明します。

各要素は，その目的，属性，親要素，子要素の説明で定義されます。
TransactionVision に固有の .NET プローブの追加設定要素については，『HP TransactionVision Deployment Guid』を参照してください。

.NET Agent の設定要素

<appdomain> 要素

目的

複数のアプリケーション・ドメインをホストするプロセスのための AppDomain 包含リストを作成します。プロセスに対して appdomain 要素が定義されていない場合は，このプロセスのすべてのアプリケーション・ドメインが含まれます。

属性

属性	有効な値	標準設定	説明
enabled	true false	true	AppDomain をインストルメントするかどうかを判別する。process 要素の enableallappdomains 属性でオーバーライドされる。
name	文字列	なし	.NET AppDomain の名前（IIS 修飾パス，下の例を参照）
website	文字列	なし	Web サイトである appdomains の Web サイト名（情報のみ）

要素

発生数	0 以上
親要素	process
子要素	bufferpool, credentials, diagnosticsserver, mediator, id, ipaddress, logging, lwmd, modes, points, profiler, sample, trim, webserver, symbols, filter, topology

例

```
<appdomain enabled="true" name="1/ROOT/MSPetShop"/>
```

1/ROOT は Web サイト ID で，MsPetShop は仮想ディレクトリ名

```
<appdomain enabled="false" name="1/ROOT" website="Default Web Site">
        <points file="Default Web Site.points"/>
        <id probeid="Default Web Site" />
</appdomain>
```

<authentication> 要素

目的

認証されたユーザ名とパスワードを一覧表示します。

属性

属性	有効な値	標準設定	説明
username	文字列	admin	ユーザ名アカウント
password	文字列	admin	**<プローブのインストール・ディレクトリ>¥bin** ディレクトリで passgen ユーティリティを使って，パスワードを作成する必要がある。

要素

発生数	0 ～多数
親要素	プロファイラ
子要素	なし

例

```
<profiler authenticate="true">
    <authentication username="Test" password="uU8X9zOtl6Twi7TkGAhQ="/>
</profiler>
```

<bufferpool> 要素

目的

バッファ・プールの動作を設定します。

属性

属性	有効な値	標準設定	説明
size	数値	65536	各バッファのサイズ。
buffers	数値	512	プールのバッファ数。
sleep	数値	1000	フラッシュ・チェック間のミリ秒数。
expires	数値	1000	バッファの有効期限が切れる前のミリ秒数。

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	なし

例

```
<bufferpool size="65536" buffers="512" sleep="1000" expires="1000" />
```

<captureexceptions> 要素

目的

例外のスタック・トレース・キャプチャを有効にして制御します。

属性

属性	有効な値	標準設定	説明
enabled	true false	true	例外のキャプチャを有効にします。
max_per_request	数値	4	1 つのサーバ要求に対してキャプチャされる最大例外数。
max_stack_size	数値	0（最大数なし）	キャプチャした例外の呼び出しスタックの最大サイズ。

要素

発生数	1
親要素	probeconfig
子要素	include, exclude

例

```
<captureexceptions enabled="true" max_per_request="4">
```

<consumeridrules> 要素

目的

これはコンシューマ ID ルールを設定するためのルート要素です。

属性

属性	有効な値	標準設定	説明
enabled	true false	false	コンシューマ ID ルールの評価を行う。

要素

発生数	1
親要素	probeconfig
子要素	httpheaderules，Ciprules，Csoaprules

例

```
<consumeridrules enabled="false">
```

<cputime> 要素

目的

cputime 設定プロパティを制御します。

属性

属性	有効な値	標準設定	説明
mode	none, serverrequest, method	serverrequest	

要素

発生数	1
親要素	probeconfig, process または appdomain
子要素	なし

例

```
<cputime mode="serverrequest"/>
```

<credentials> 要素

目的

Diagnostics サーバとの通信の検証に使われる資格情報を提供します。

属性

属性	有効な値	標準設定	説明
username	文字列	なし	Diagnostics サーバで検証するユーザ名
password	文字列	なし	Diagnostics サーバで検証するパスワード
authenticate	true, false	true	認証を有効 / 無効にします。

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	なし

例

```
<credentials username="test" password="diag" authenticate="true"/>
```

＜ demomode ＞要素

目的

デモ・モードを設定します。デモ・モードでは定義が必要なカスタム・ポイント数が少なくなるため，.NET Agent の機能や値を表示しやすくなります。demomode が有効な場合，その他のインストルメンテーションに関係なく，すべての発信呼び出しが表示されます。

関心のある発信呼び出しの呼び出し元を特定したら，demomode を無効にして，発信呼び出しの起点となる呼び出しスタックが明らかになるように「カスタム」インストルメンテーションを追加します。

注 : 実運用環境では，これを無効にすることをお勧めします。

---

demomode は，主に作成するメソッドがインストルメントされていないときに発信呼び出し（webserver, http, remoting, msmq）を検索するために使用されます。これは，アプリケーション固有のメソッドをインストルメントすることなく，アプリケーションがどのように接続されているかをすばやく見つける方法として提供されています。実運用環境ではノイズが多すぎる状態になる可能性がありますが，アップストリーム・インストルメンテーションが不足していて，発信呼び出しの呼び出し元が不明な場合には役立ちます。ASP.NET を含むすべての種類のアプリケーションに使用できます。

属性

属性	有効な値	標準設定	説明
enabled	true, false	false	デモ・モードを有効または無効にする。

要素

発生数	ゼロまたは 1
親要素	probeconfig
子要素	なし

例

```
<demomode enabled="false"/>
```

<depth> 要素

目的

深さのトリミングを設定します。

属性

属性	有効な値	標準設定	説明
enabled	true false	true	深さのトリミングを有効にする。
depth	数値	25	深さのトリミングの深さを設定する。

要素

発生数	1
親要素	trim
子要素	なし

例

```
<trim>
    <depth enabled="true" depth="25"/>
</trim>
```

<diagnosticsserver> 要素

目的

enterprise モードで使われる Diagnostics サーバに関する接続情報と設定情報が含まれます。

属性

属性	有効な値	標準設定	説明
url	レジスタの URL : http://＜ホスト＞:＜ポート＞	なし	レジスタに接続するための URL。
delay	数値	2	登録の前に待機する秒数。
keepalive	数値	15	keepalive 間の秒数。
proxy	プロキシの URL	なし	レジストラの接続プロキシ。
proxyuser	プロキシのユーザ ID	なし	プロキシ・ユーザ・アカウント
proxypassword	プロキシのパスワード	なし	プロキシ・ユーザ・アカウントのパスワード
registered_hostname	文字列	なし	（ファイアウォールのトラバースの外部名）として登録するホストの名前。
register_byip	true, false	false	ホスト名の代わりに IP アドレスを使って登録する。
timeskewcheckinterval	数値	60	Diagnostics サーバからタイム・スキューを取得するまで待機する秒数。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig
子要素	なし

例

次の一般的な例では，< diagnosticsserver >要素の設定を示します。疑問符 (?) は，適切な値で置き換える必要があることを示します。

```
<diagnosticsserver url="http://localhost:2006/commander" delay="2"
keepalive="15" proxy="?" proxyuser="?" proxypassword="?"
registerhostname="?" register_byip="false"/>
```

registered_hostname 属性を使用して標準設定のプローブ・ホスト・マシン名をオーバーライドする手順については，612 ページ「標準設定のプローブホスト・マシン名の変更」を参照してください。

<exceptiontype> 要素

目的

例外タイプを定義します。

属性

属性	有効な値	標準設定	説明
name	文字列	なし	例外のクラス名。

要素

発生数	0 ～多数
親要素	include, exclude
子要素	なし

例

```
<exceptiontype name="System.DivideByZeroException"/>
```

<exclude> 要素（親が captureexecptions の場合）

目的

除外する例外のリストを定義します

属性

なし

要素

発生数	1
親要素	captureexceptions
子要素	exceptiontype

例

```
<exclude>
    <exceptiontype name="System.DivideByZeroException"/>
</exclude>
```

<exclude> 要素（親が lwmd の場合）

目的

.NET Profiler の［コレクション］タブおよび Diagnostics ユーザ・インタフェースの［コレクション］ビューの［Collections by Growth］および［Collections by Size］テーブルから除外するコレクション・クラスを定義します。

指定したコレクション・クラスに，**ICollection** を実装するクラスが含まれることがあります。この設定は，LWMD ポイントのインストルメンテーションに影響しないことに注意してください。この設定が影響するのは，LWMD データの表示や Diagnostics サーバに送信される LWMD データの量のみです。

属性

なし

要素

発生数	0 ～多数
親要素	lwmd
子要素	なし

例

```
<lwmd enabled="true" sample="15s" autobaseline="1h" growth="10" size="10">
    <exclude>System.Collections.ArrayList</exclude>
    ＜ exclude ＞ System.Data.DataView ＜ /exclude ＞
</lwmd>
```

System.Data.DataView は System.Collections.ICollection を実装することに注意してください。

＜ excludeassembly ＞要素

目的

アセンブリのインストルメンテーションを実行します。アセンブリは .exe または .dll ファイルです。インストルメンテーションから重要なアセンブリを除外する機能を提供します（たとえば，重要なアセンブリのコードを暗号化および暗号するために製品を使用していて，インストルメント時に例外をスローする場合）。

process 要素の子として＜ excludeassembly name= ＜ AssemblyNameToExclude ＞を追加します。

属性

属性	有効な値	標準設定	説明
name	文字列	なし	除外するアセンブリの名前（ファイル拡張子は含まない）

要素

発生数	0 ～多数
親要素	process
子要素	なし

例

```
<process enablealldomains="true" name="ASP.NET">
   <logging level="" />
   <points file="ASP.NET.points" />
   <points file="ADO.points" />
   <points file="WCF.points" />

   <excludeassembly name="Acme.Encryption" />

  <appdomain enabled="false" name="TestWebService">
     <points file=" TestWebService .points" />
   </appdomain>
</process>
```

<filter> 要素

目的

結果を歪曲することがある，または監視中の処理を表していない特定のメトリックスをフィルタリングします。

属性

属性	有効な値	標準設定	説明
firstserverrequest	true，false	false	アプリケーションの起動後，特定のサーバ要求（URL）が初めて実行したときに，測定値のコレクションのスキップを有効 / 無効にします。

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	なし

例

```
<filter firstserverrequest="false"/>
```

＜ httpclient ＞要素

目的

HTTP 発信呼び出しの ID として URL を含めるかどうかを設定します。標準設定は true です。発信 HTTP 呼び出しに多数の異なる URL が対応している場合を除いて，標準設定を維持する必要があります。発信呼び出しで作成される番号（URL ごとに 1 つずつ）が原因で，Diagnostics サーバのパフォーマンスが低下することがあります。HTTP 発信呼び出しの URL を考慮しない場合は，この機能を無効にすることもできます。HTTP 発信呼び出しの ID は，要求の送信先となるサーバおよびポート番号になります。

属性

属性	有効な値	標準設定	説明
showurl	true，false	true	HTTP を使ってクライアントから送信される発信呼び出しの ID に URL を含める機能を有効 / 無効にする。 false に設定すると，極めて多数の異なる HTTP クライアント呼び出しが発生した場合に，サーバ / Agent の記号テーブルが急激に増大しないように保護できる。 REST サービスのクライアント・アプリケーションでは，値を false に設定する必要があります。

要素

発生数	0 ～ 1
親要素	probeconfig, process, appdomain
子要素	なし

例

```
<httpclient showurl="true"/>
```

<gentvhttpeventforwcf> 要素

目的

このオプションを設定すると，IIS（HTTP ベース）ホスティングを使用するバインディングで WCF サービスの TransactionVision イベントの生成が有効になります。一部の WCF サービスでは，実際の Web サービスとしてはサポートされていないカスタムまたはプライベート・バインディングを使用している可能性があり，このような場合はこのオプションを有効にしないと TransactionVision Web サービス・イベントは生成されません。

属性

属性	有効な値	標準設定	説明
enabled	true，false	false	IIS（HTTP ベース）ホスティングを使用するバインディングで WCF サービスの HTTP イベントの生成を有効 / 無効にします。有効にした場合，TransactionVision Web サービス・イベントを指定します。

要素

発生数	0 ～ 1
親要素	probeconfig, process, appdomain
子要素	なし

例

```
<gentvhttpeventforwcf enabled="true"/>
```

＜ httpheaderrule ＞要素

目的

HTTP ヘッダのコンシューマ ID を定義します。

属性

属性	有効な値	標準設定	説明
id	文字列	なし	ルールの ID
rule	文字列	なし	コンシューマによって HTTP 要求が送信される URL との照合に使われる正規表現。
consumeridfield	文字列	なし	コンシューマ ID として使うヘッダの名前。

要素

発生数	0 ～多数
親要素	httpheaderrules
子要素	なし

例

```
<httpheaderrule id="httpHeader 1" rule="/Webservice/.*"
consumeridfield="Caller"/>
```

<httpheaderrules> 要素

目的

この要素には < httpheaderrule >要素がすべて含まれています。

属性

なし

要素

発生数	1
親要素	consmeridrule
子要素	httpheaderule

例

```
<httpheaderrules>
</httpheaderrules>
```

<id> 要素

目的

プローブの ID およびグループ ID を提供します。

属性

属性	有効な値	標準設定	説明
probeid	次の文字を含む文字列 : 文字, 数字, アンダースコア, ダッシュ, ピリオド, 内部的に定義された次の $() 変数値 : $(APPDOMAIN), $(MACHINENAME), $(WEBSITENAME), $(PID)	$(APPDOMAIN).NET	LoadRunner / Performance Center およびシステムの状況で認識されるプローブの名前
probegroup	文字列	標準設定	システム・メトリックスおよびプローブ・メトリックスを報告するために Diagnostics サーバで認識されるグループを定義します。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig, process, appdomain
子要素	なし

例

次に，標準設定の例を示します。

```
<id probeid="$(APPDOMAIN).NET" probegroup="Default"/>
```

例

LoadRunner 8.1 環境で動作していて，「myDiagServer」に報告しているプローブの例を示します。この例では，プローブの名前は有効な文字列，アプリケーションがデプロイされている Web サイトの名前，およびアプリケーションがデプロイされているマシンの名前で構成されます。

```
<id probeid="LR_81_$(WEBSITENAME)_$(MACHINENAME).NET"
probegroup="LR_81_myDiagServer"/>
```

<include> 要素（親が captureexceptions の場合）

目的

包含する例外のリストを定義します。

属性

なし

要素

発生数	1
親要素	captureexceptions
子要素	exceptiontype

例

```
<include>
    <exceptiontype name="System.DivideByZeroException"/>
</include>
```

<include> 要素（親が lwmd の場合）

目的

その他のすべてのトリミングに含むコレクション・クラスを定義します。

属性

なし

要素

発生数	0 ～多数
親要素	lwmd
子要素	なし

例

＜ include ＞ System.Collections.Hashtable ＜ /include ＞
<include>System.Collections.ArrayList</include>

<instrumentation> 要素

目的

Instrumenter のログ処理設定が含まれます。

属性

なし

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig, process
子要素	ログ処理

例

```
<instrumentation>
    <logging level="property lwmd" />
</instrumentation>
```

<iprule> 要素

目的

IP アドレスのコンシューマ ID を定義します。

属性

属性	有効な値	標準設定	説明
id	文字列	なし	コンシューマ ID ルールの評価を行う。
rule	文字列	なし	コンシューマ ID に割り当てる IP アドレスまたは IP アドレスの範囲を定義します。
consumerid	文字列	なし	ルールに一致するものがある場合に使用するコンシューマ ID。

要素

発生数	0 ～多数
親要素	iprules
子要素	なし

例

```
<iprule id="IpTest1" rule="43.*.1-20.*" consumerid="HP"/>
```

<iprules> 要素

目的

この要素には < iprule >要素がすべて含まれています。

属性

なし

要素

発生数	1
親要素	consumeridrules
子要素	iprule

例

```
<iprules>
</iprules>
```

<latency> 要素

目的

レイテンシのトリミングを設定します。

属性

属性	有効な値	標準設定	説明
enabled	true false	true	レイテンシのトリミングを有効にする。
throttle	true false	true	レイテンシ・トリミングのスロットルを有効にする。
min	数値	2	レイテンシの最大しきい値。
max	数値	100	レイテンシの最大しきい値。
increment	数値	2	しきい値の増分。
increment threshold	数値	75	スロットルを増やす前のバッファの使用割合。
decrement threshold	数値	50	スロットルを減らす前のバッファの使用割合。

要素

発生数	1
親要素	trim
子要素	なし

例

```
<trim>
    <latency enabled="true" throttle="true" min="2" max="100" increment="2"
incrementthreshold="75" decrementthreshold="50"/>
</trim>
```

<logdirmgr> 要素

目的

ログ・ディレクトリ・マネージャの設定が含まれます。logdirmgr では，ログ・ディレクトリを監視して，アンバウンドを増やさないようにします。logdirmgr は，scaninterval で示したとおりにログを定期的にスキャンします。サイズが maxdirsize で示したサイズを超えると，logdirmgr は，サイズが maxdirsize を超えなくなるまで古いファイルから削除します。

重要 : .NET プロセスを実行しているアカウント（IIS の場合は AppPool アカウント）にログ・フォルダへの「**削除**」権限が付与されている必要があります。これは NETWORK SRERVICE アカウントまたは App Pool ID アカウント（標準設定のアプリケーション・プール・アカウント）の標準設定では利用できません。

属性

属性	有効な値	標準設定	説明
enabled	true false	true	
maxdirsize	数値	1024 MB	ログ・ディレクトリのサイズ制限にする最大サイズ。
scaninterval	数値	30m	マネージャでログをスキャンして，増加およびサイズをチェックする頻度。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig
子要素	なし

例

```
<logdirmgr enabled="true" maxdirsize="1024 MB" scaninterval="30m"/>
```

<logging> 要素（親が instrumentation の場合）

目的

.NET Agent のインストルメンテーション・プロセスのログ処理レベルを設定します。

属性

属性	有効な値	標準設定	説明
level	off assert break severe warning info _____ debug points eh sig chi cil classmap ilasm symbols deepmode load all checksum property remoting lwmd http	"" これは「info」と同等。	ログ処理のレベル。
threadids	true false	true	ログにはスレッド ID を含める必要がある。

注 : info よりも下にある有効な値は，通常は使用しないでください。診断設定によっては，極めて大きなログ・ファイルが生成される可能性があるためです。

要素

発生数	0 ～多数
親要素	instrumentation
子要素	なし

例

```
<instrumentation>
    <logging level="warning" />
</instrumentation>
```

<logging> 要素（親が appdomain，probeconfig または process の場合）

目的

アプリケーション・パフォーマンスを監視および報告するための .NET Agent 処理のログ処理レベルを設定します。

属性

属性	有効な値	標準設定	説明
level	off severe warning info _____ debug events property webserver http symbols probemetrics registrar threadpool authentication bufferpool rum bacforsoa vmware exceptions| tvdebug	"" これは「info」と同等。	
max	数値	10	プローブ・ログ・ファイルの最大サイズ。ログがこのサイズに達するとログは記録されなくなります。

注 : info よりも下にある有効な値は，通常は使用しないでください。診断設定によっては，極めて大きなログ・ファイルが生成される可能性があるためです。

要素

発生数	
親要素	appdomain, probeconfig, process
子要素	なし

例

```
<logging max="10" level="INFO"/>
```

<lwmd> 要素

目的

ライトウェイトなメモリ診断（LWMD）機能を設定します。

属性

属性	有効な値	標準設定	説明
enabled	true false	false	lwmd キャプチャのサンプリングを有効にする。
sample	文字列	1m	サンプリング間隔（h は時間 /m は分 / s は秒）。
autobaseline	文字列	1h	自動ベースラインの間隔。
manualbaseline	文字列	なし	手動ベースライン時間。
growth	数値	15	増加トラックのコレクション数。
size	数値	15	サイズ・トラックのコレクション数。

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	exclude, include

例

```
<lwmd enabled="false" sample="1m" autobaseline="1h" manualbaseline= "?"
growth="15" size="15"/>
```

<mediator> 要素

目的

Enterprise モードでイベントが送信される，Mediator モードの Diagnostics サーバを指定します。

属性

属性	有効な値	標準設定	説明
host	ホスト名	なし	Mediator の名前
port	数値	2612	Mediator のポート
ssl	true/false	false	Diagnostics サーバの URL が http で始まる場合，デフォルト値は false です。Diagnostics サーバの URL が https で始まる場合，デフォルト値は true です。
metrichost	文字列		測定値データの送信先ホスト
metricport	数値	2006	ヒープの使用量や可用性など，プローブがプローブ・メトリックスを送信するポート。
block	true/false	false	Mediator 接続が確立するまでブロックする。
ipaddress			イベント・サーバに接続するときに使うローカル IP アドレス。

属性	有効な値	標準設定	説明
localportstart	数値	4000	Diagnostic メディエータ・サーバへの TCP イベント・チャネル接続に使用するポート範囲の始め。IP アドレスが指定されている場合のみ使用されます。
localportend	数値	5000	Diagnostic メディエータ・サーバへの TCP イベント・チャネル接続に使用するポート範囲の終わり。IP アドレスが指定されている場合のみ使用されます。

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	なし

例

```
<mediator host="localhost" port="2612" ssl="false" metricport="2006"
block="false" ipaddress="16.255.18.99" localportstart="4000"
localportend="5000"/>
```

＜ metrics ＞要素

目的

この要素には ＜ metric ＞要素がすべて含まれています。

属性

なし

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, process
子要素	metric

例

```
<metrics>
  <metric name="% Time in GC" group="Memory" units="percent"
category=".NET CLR Memory" counter="% Time in GC"/>
</metrics>
```

<metric> 要素

目的

Diagnostics .NET を使用して perfmon から収集する追加のプローブ・メトリックスを指定します。追加情報については，622 ページ「その他のプローブ・メトリックスの収集またはプローブ・メトリックスの変更」を参照してください。

属性

属性	有効な値	標準設定	説明
name	文字列		Diagnostics UI に表示する測定値の名前
group	文字列		Diagnostics UI に表示する測定値のグループ（カテゴリ）
units	microseconds, milliseconds, seconds, minutes, hours, days, bytes, kilobytes, megabytes, gigabytes, count, percent, fraction_percent, load, status		perfmon 測定値の測定単位
category	文字列		perfmon で指定されたパフォーマンス・カウンタのカテゴリ
counter	文字列		perfmon で指定されたパフォーマンス・カウンタ

注 : カウンタのインスタンスは，カウンタのプロセス・インスタンスとして，または ASP.NET アプリケーション・カウンタのアプリケーション・ドメイン・インスタンスとして自動的に割り当てられます。プロセス・インスタンスまたはアプリケーション・ドメイン・インスタンスのないカウンタは，収集されません。代わりに，システム・メトリックスを定義する必要があります。

要素

発生数	親ごとに 1 つ以上
親要素	メトリックス
子要素	なし

例

```
<metrics>
  <metric name="% Time in GC" group="Memory" units="percent"
category=".NET CLR Memory" counter="% Time in GC"/>
</metrics>
```

<modes> 要素

目的

.NET Agent を実行するプロダクト・モードを指定します。各モードの使用法の詳細については，593 ページ「Agent が連携して動作できる HP ソフトウェア製品の制御」を参照してください。

<modes> 要素は，HP Diagnostics ライセンス・キャパシティの使用数を判別するためにも使用されます（ライセンスの詳細については，83 ページ「現在接続されているプローブに基づくライセンス情報」を参照）。

<modes> 要素の値は，最初に Agent をインストールするときに設定されます。

.NET Agent は，次を実行する異なるモードに設定できます。

- 開発から実稼動前のテスト，実運用までアプリケーションを監視する
- ほかの HP ソフトウェア製品で使用する
- サーバまたはほかの HP ソフトウェア製品にレポートしないスタンドアロンの Diagnostics Java Profiler として使用する

属性

属性	有効な値	標準設定	説明
enterprise	true false	インストール時に選択したモードによります。 ➤ pro が false の場合は true ➤ pro が true の場合は false	Enterprise モードで実行する Agent を設定する（プローブは，Diagnostics サーバと連動する）。 Enterprise モードは *AD*，*AM*，および *PRO* モードを組み合わせたものと同様である。LoadRunner 実行データおよび LoadRunner 実行に含まれないデータをキャプチャする。 Enterprise モードが .NET Agent の標準設定です（AD または AM モードを指定していない場合）。Enterprise モードでは，Agent が AM ライセンス・キャパシティに対してカウントされます。
ent	true false	インストール時に選択したモードによります。 pro が false の場合は true pro が true の場合は false	enterprise 属性の省略形

属性	有効な値	標準設定	説明
ad	true false	false	*AD* モードは，その他のすべてのモードよりも優先する。*AD* モードとその他のモードが設定されている場合，モードは *AD* に設定される。 *AD* モードの場合，.NET Agent は LoadRunner からの実行のみをキャプチャし，結果をその実行専用のデータベース（Default21 など）に格納する。 AD モードの Agent は，プローブが LoadRunner または Performance Center でテスト実行されている場合のみに AD ライセンス・キャパシティに対してカウントされます。テスト実行中でない Agent は，ライセンス・キャパシティに対してカウントされません。 たとえば，LoadRunner/ Performanace Center AD モードで 20 個のプローブがインストールされていて，5 個のみが実行されている場合，その 5 個のみが AD ライセンス・キャパシティに対してカウントされます。

属性	有効な値	標準設定	説明
am	true false	➤ false	*AM* モードは，*AD* を除く，その他のすべてのモードよりも優先する。*AM* モードの場合，.NET Agent は実行を無視する。LoadRunner がアプリケーションを実行している場合，データは標準 Diagnostics データベースに格納される。 AM モードの Agent は，常に AM ライセンス・キャパシティに対してカウントされます。

属性	有効な値	標準設定	説明
pro	true false	インストール時に選択したモードによります。 ➤ enterprise が false の場合は true ➤ enterprise が true の場合は false	Profiler モードで実行するように Agent を設定する。 このモードの場合，データは Profiler に送信される。このモードはほかのモードと組み合わせることができる。pro モードの Agent は，ライセンス・キャパシティに対してカウントされません。
tv	true false	false	TransactionVision イベントのキャプチャを有効にします。トランスポートおよびその他の TV オプションの設定の詳細については，279 ページ「TransactionVision 用の .NET Agent 設定について」を参照してください。このモードは TransactionVision にイベントを送信する。このモードはほかのモードと組み合わせることが可能で，tv モードの Agent は Diagnostics のライセンス・キャパシティに対してカウントされません。

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	なし

例

```
<modes enterprise="false" ad="false" am="false" pro="true"/>
```

<points> 要素

目的

インストルメンテーションに使用するキャプチャ・ポイント・ファイルを指定します。

属性

属性	有効な値	標準設定	説明
file	文字列	なし	インストルメンテーション・キャプチャ・ポイント・ファイルの名前。

要素

発生数	0 以上
親要素	appdomain, process
子要素	なし

例

```
<points file="ASP.NET.points"/>
```

<probeconfig> 要素

目的

.NET Agent 設定に包含ルート要素を 1 つ提供します。

属性

なし

要素

発生数	1
親要素	なし
子要素	appdomain, bufferpool, captureexceptions, consumeridrules, credentials, diagnosticsserver, eventserverhost, id, instrumentation, ipaddress, logging, lwmd, mediator, modes, points, process, profiler, rum, sample, soappayload, trim, webserver, topology, vmware, xvm

例

```
<probeconfig>
</probeconfig>
```

<process> 要素

目的

プロセスを監視する包含フィルタ・リストを提供します。

process 要素が定義されていない場合，プロセスは監視されません。

属性

属性	有効な値	標準設定	説明
enablealldomains	true false	true	true に設定すると，プロセスの一部であるすべての appdomain の属性が変更されるため，すべて有効になります。
name	文字列	なし	これらの設定が適用される .NET プロセスの名前。

次に，< process >要素の enablealldomains attribute 属性に関するルールを示します。

- enablealldomains = false : < appdomain >のリスト内にドメインがない場合は，どのドメインも有効になりません。
- enablealldomains = false : < appdomain >のリスト内にドメインがある場合は,「enable」属性が true に設定されているか, < appdomain >の enable 属性で定義されていないときに，ドメインが有効になります。
- enablealldomains = true : < appdomain >のリスト内にドメインがある場合は,「enable」属性に関係なく，リスト内のドメインのみが有効になります。
- enablealldomains = true : < appdomain >のリスト内にドメインがない場合は，すべてのドメインが有効になります。
- enablealldomains 属性が定義されていない : enablealldomains = true の場合と同様です。

要素

発生数	0 以上
親要素	probeconfig
子要素	appdomain, bufferpool, credentials, diagnosticsserver, mediator, id, instrumentation, ipaddress, logging, lwmd, modes, points, profiler, sample, trim, webserver, filter, symbols, topology

例

```
<process enablealldomains="true" name="ASP.NET">
```

<profiler> 要素

目的

Profiler 機能の設定が含まれます。

属性

属性	有効な値	標準設定	説明
authenticate	true, false	なし	受信 Profiler 接続要求の認証を有効 / 無効にします。
register	true, false	false	プローブが Profiler 専用モードの場合でも登録するようにプローブに指示します。
samples	数値	60	lwmd / heap トレンドを保持するためのサンプル数を Profiler に伝えます。
best	数値	1	保持する一番早いインスタンス・ツリーの数。
worst	数値	3	保持する一番遅いインスタンス・ツリーの数。
inactivitytimeout	文字列	10m	ユーザが Profiler とのやり取りを終えた後に，Profiler が実行を続ける時間。
disableremoteaccess	true, false	false	Profiler へのリモート・アクセスを無効にする。このため，ユーザ / パスワードは公開されなくなるが，マシンへの telnet / RemoteDeskTop を実行して Profiler をローカルで実行することは可能である。

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	authentication

例

```
<profiler authenticate="true" register="false" samples="60" best="1" worst="3"
inactivitytimeout="10m">
    <authentication username="admin" password="admin"/>
</profiler>
```

<rum> 要素

目的

Real User Monitoring の設定を制御する。

属性

属性	有効な値	標準設定	説明
enable	true false	true	RUM 統合機能を有効または無効にします。
responseheader	文字列	X-HP-CAM-COLOR	値に Diagnostics と RUM の統合情報が含まれている http ヘッダの名前。
encryptedkey	文字列		＜プローブのインストール・ディレクトリ＞¥bin ディレクトリで passgen ユーティリティを使って，暗号キーを作成する必要がある。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig
子要素	なし

例

```
<rum enabled="true" responseheader="X-HP-CAM-COLOR"
encryptedkey="OBF:3pe941vx43903wre40303xxz3q6r42ob43n93wre3io03xjs4
0h940pc3wir3q233jur3zir3yi03zir3vc03wre3xpi3r8o3olr44na3zor3v6m3vc03zir4
4u03ohb3rdi3xjs3wx03v6m3zor3yc63zor3jqz3q6r3wd740vi40b53xpi3ike3wx04
3gp42ur3q233y3r3zwy3wx0432i42293p9p"/>
```

暗号キーを作成するには，次のように PassGen ユーティリティを使用します。

```
cd <installdir>/bin
PassGen /system encryptionKey
```

encryptionKey は最大 128 文字までの英数字から成る文字列です。暗号キーは標準出力に表示されます。

passgen の例：

```
PassGen /system TheLazyFoxJumpedHigh
```

以下が返されます：

```
OBF:3q6r3xxz3y3r3xjs3wx03yc63n0r3lbr3vc03wd745893wre44u0413j3kn93zw
y40vi432i44fr3m453m894493439040pc40303kjd419r44na3wx0451h3wir3v6m3
lfr3mwj3yi03wre3xpi3xxz3y3r3q23
```

<sample> 要素

目的

サンプリングのタイプとレートを設定します。

属性

属性	有効な値	標準設定	説明
method	percent, count, period	percent	サンプリング方法を設定する。 ➤ percente レートは，0 ～ 100 になります。 ➤ count レートは，1 未満になります。 ➤ period レートは，標準の Diagnostics 時間文字列のいずれかになります（3 分は 3m，4 秒は 4s など）。
rate	数値	0	パーセント・タイプにサンプリング・レートを設定する。

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, Cprobeconfig, Cprocess, Cws
子要素	なし

例

<xvm><ws><sample method="percent" rate="50"/></ ws ></xvm>
サンプリングはランダムなパーセント率です。

<xvm>< ws ><sample method="count" rate="50"/></ ws ></xvm>
サンプリングは率ごとに 1 回です。

<xvm>< ws ><sample method="period" rate="60000"/></ ws ></xvm>

＜ soapcapture ＞要素

目的

SOAP 要求および応答をキャプチャするかどうかを設定します。

属性

属性	有効な値	標準設定	説明
enabled	true false	true	SOAP 要求および応答のキャプチャを有効または無効にします。無効な場合は，次の項目に影響します。 ➤ SOAP に失敗した場合の SOAP 要求キャプチャ ➤ TV モードでの SOAP 要求および応答のキャプチャ ➤ SOAP ルールを通して割り当てられたコンシューマ ID
maxsize	数値	0	これは，キャプチャされた SOAP 要求または応答の最大文字数を指定する任意指定の属性である。 0 は無限を示す。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig
子要素	なし

例

```
<soapcapture enabled="true" maxsize="0" />
```

<soappayload> 要素

目的

この要素は廃止されていて，< soaprequestforsoapfault >で置き換えられています。

SOAP 障害時の SOAP ペイロード・キャプチャ機能を設定して，SOAP の障害に関連する SOAP ペイロードを提供します。ここでは，SOAP ペイロードは SOAP エンベロープ全体であると定義されます。

属性

属性	有効な値	標準設定	説明
enabled	true false	true	SOAP ペイロード・キャプチャ機能を有効または無効にする。
maxsize	数値	5000	これは，すべてのペイロード・キャプチャの最大サイズの文字数を指定する任意指定の属性です。指定されていなければデフォルト値が使われます。設定にエラーがある場合は，標準設定値が使われる。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig
子要素	なし

例

```
<soappayload enabled="true" maxsize="5000" />
```

＜ soaprequestforsoapfault ＞要素

目的

SOAP の失敗時の SOAP 要求キャプチャ機能（ペイロードを含む）を設定します。ペイロードにはクレジット・カード番号などの機密情報が含まれる可能性があるため，この要素は標準設定では無効になっています。

注 : <soapcapture> 要素が無効な場合，<soaprequestforsoapfault> 設定はオーバーライドされます。＜soapcapture＞要素のドキュメントを参照してください。

属性

属性	有効な値	標準設定	説明
enabled	true false	false	SOAP 障害時の SOAP 要求キャプチャ機能を有効または無効にします。標準設定では無効です。
maxsize	数値	5000	これは，SOAP 要求キャプチャの最大文字数を指定する任意指定の属性である。指定されていなければデフォルト値が使われます。設定にエラーがある場合は，標準設定値が使われる。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig
子要素	なし

例

```
<soaprequestforsoapfault enabled="true" maxsize="5000" />
```

<soaprule> 要素

目的

SOAP ヘッダのコンシューマ ID を定義します。

属性

属性	有効な値	標準設定	説明
id	文字列	なし	ルールの ID
rule	文字列	なし	コンシューマによって呼び出される Web サービス名との照合に使われる正規表現。
consumeridfield	文字列	なし	コンシューマ ID として使用する値を取得する SOAP ヘッダ内の要素。

要素

発生数	0 ～多数
親要素	soaprules
子要素	なし

例

```
<soaprule id="SOAP1" rule="TestService2" consumeridfield="Caller"/>
```

<soaprules> 要素

目的

この要素には < soaprule >要素がすべて含まれている。

属性

なし

要素

発生数	1
親要素	consumeridrules
子要素	soaprules

例

```
<soaprules>
</soaprules>
```

<sqlparsing> 要素

目的

この要素は，SQL クエリで解析するモードを示すために使用します。リテラルを使用する多数の SQL クエリがある場合はサーバの記号テーブルが過負荷になる可能性があるため，この問題を避けるために標準設定ではモード 3 に設定されています。

属性

属性	有効な値	標準設定	説明
mode	1，2，3，4	3	モードは SQL クエリを解析する方法を示します。 1 - メソッドのみで SQL クエリなし 2 - SQL クエリのメイン・カテゴリ（select/update/insert/delete/...） 3 -（標準設定）SQL クエリ全体で類似ステートメントを単一測定値に集約する測定値（リテラル，キーワードの大文字と小文字などは無視） 4 - SQL クエリ全体で同一ステートメントのみを集約する測定値
keywordsfile	文字列	なし	必要に応じて，Agent で SQL ステートメント内を検索するキーワードを含むファイルを指定し，Diagnostics で保存または表示されるときに大文字で強調表示できます。これにより，類似したクエリが大文字と小文字を区別せずに同じクエリとして認識されます。

要素

発生数	1
親要素	probeconfig
子要素	

例

```
<sqlparsing mode="4" keywordsfile="C:\myfolder\mykeyword.txt"/>
```

<symbols> 要素

目的

メモリ消費量を制御するためにキャプチャ可能な一意の URI および SQL 文字列の数を制限します。

属性

属性	有効な値	標準設定	説明
maxuri	数値	1000	キャプチャ可能な一意の SQL の数の上限を設定します。
maxuriname	文字列	一意の URI の最大数を超えました。	
maxsql	数値	1000	キャプチャ可能な一意の SQL の数の上限を設定します。
maxsqlname	文字列	一意の SQL の最大数を超えました。	

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	urireplacepattern

例

```
<symbols maxuri="1000" maxuriname="Maximum number of unique URIs exceeded" maxsql="1000" maxsqlname="Maximum number of unique SQLs exceeded"/>
```

<timeskew> 要素

目的

HP TransactionVision の設定に使用します。タイム・サーバと .NET Agent が実行されているホスト間の時間差を計算します。タイム・サーバによるチェック頻度を設定できます。

属性

属性	有効な値	標準設定	説明
historysize	数値	24	（起動時に読み取り）保存して最良のサンプルと比較するタイム・スキュー・サンプル数。
checkinterval	数値	300,000 ミリ秒	（動的）タイム・サーバをチェックしてスキュー・タイムを計算する前の待機時間（単位ミリ秒）。
latencythreshold	数値	100 ミリ秒	（動的）有効なタイム・スキュー値としてタイム・サーバからの応答に認められる最大時間（単位ミリ秒）。
retrythreshold	数値	8	（動的）タイム・サーバのリクエストが失敗したときの試行回数。

要素

発生数	1（1 回）
親要素	tv
子要素	なし

例

```
<timeskew historysize="24" checkinterval="300000" latencythreshold="100"
retrythreshold="8"/>
```

<topology> 要素

目的

トポロジ情報を収集して Diagnostics サーバに送信するかどうかを制御します。

属性

属性	有効な値	標準設定	説明
enable	true false	true	トポロジ情報の収集と，それらの情報を Diagnostics サーバへの引き渡しを有効にします。

要素

発生数	1
親要素	<probeconfig>, <process>, または <appdomain>
子要素	なし

例

```
<topology enable="true">
```

<transport> 要素

目的

TransactionVision が使用するイベント・チャネルを設定します。

属性

属性	有効な値	標準設定	説明
type	mqseries sonicmq	sonicmq	Agent が使用するイベント配送プロバイダ
connectionString	下記参照。		イベント配送プロバイダに関する接続情報。

type=sonicmq の場合の conectionString 構文

```
broker = <broker>; port = <port>; user = <user>; password =<password>;
configurationQueue = <configurationQueue>
```

パラメータ	説明
broker	Sonic ブローカが実行されるホスト名。普通は Analyzer ホスト名です。
port	ブローカが通信を行うポート。標準設定では 21111 です。
user	SonicMQ インストールで接続に必要な場合は，ユーザ ID。標準設定では，ユーザ名は不要です。
password	SonicMQ インストールで必要とされる場合は，パスワード。これは，PassGen ユーティリティを使用して作成される難読化形式になります。標準設定では，パスワードは不要です。**PassGen** の詳細については，『BSM Application Administration User Guide』の「Administration Utilities」を参照してください。
configurationQueue	.NET TransactionVision Agent に関する構成メッセージを含むキュー名。

type=mqseries の場合の conectionString 構文

```
host= <host>; queuemanager=<queuemanager>; port= <port>; channel=,channel>
configurationQueue = <configurationQueue>
```

パラメータ	説明
host	TransactionVision 構成キューをホスティングするホスト。
queuemanager	キューマネージャ名。
port	QueueManager が通信を行う MQSeries ポート。
channel	通信に使用される MQSeries チャネル。
configurationQueue	.NET TransactionVision Agent に関する構成メッセージを含むキュー名。

要素

発生数	1（1 回）
親要素	tv
子要素	なし

例

SonicMQ の場合

```
<transport type="sonicmq" connectionstring="broker=brokerHost;
    port=21111; user=; password=;
    configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
```

MQ Series の場合

```
<transport type="mqseries" connectionstring="host=mqHost;
    queuemanager=; port=1414; channel=TRADING.CHL;
    configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
```

<trim> 要素

目的

トリミング機能を設定して プローブと Diagnostics サーバの間で転送されるデータ量を減らします。

Profiler ではデータを Diagnostics サーバに送信する必要がないため，Profiler ユーザ・インタフェースは, 深さのトリミングやレイテンシのトリミングといった, 設定されているすべてのトリミング設定を無視します。

属性

なし

要素

発生数	親 1 つあたり 1 回
親要素	appdomain, probeconfig, process
子要素	depth，latency

例

```
<trim>
</trim>
```

<tv> 要素

目的

TransactionVision で使用する .NET Agent の設定

属性

属性	有効な値	標準設定	説明
eventthreads	数値	3	（起動時に読み取り）Agent によって生成され，Analyzer へ送信されるスレッド数。
eventthreadsleep	数値	100	（動的）メッセージ（イベント・パッケージ）送信後のイベント・スレッドのスリープ時間（単位ミリ秒）。
eventmemorythreshold	数値	25,000,000	（動的）内部バッファ（Q）により消費されるメモリ容量。その後で Agent がアプリケーション・スレッドについてメッセージの送信を試みます。
configthreadsleep	数値	10,000	（動的）構成キューをブラウンジングした後のイベント・スレッドのスリープ時間（単位ミリ秒）。

要素

発生数	1（1 回）
親要素	ProbeConfig
子要素	transport, timeskew

例

```
<tv eventthreads="3" eventthreadsleep="80"
eventmemorythreshold="25000000" configthreadsleep="10000" >
    <timeskew historysize="24" checkinterval="300000" latencythreshold="100"
        retrythreshold="8"/>
        <transport type="sonicmq"
        connectionstring="broker=myhost.mydomain.com;
        port=21111; user=; password=;
        configurationqueue=TVISION.CONFIGURATION.QUEUE"/>
</tv>
```

<urireplacepattern> 要素

目的

多数のサーバ要求を集約して簡素化されたサーバ要求 URI に置き換えることで，サーバ要求数を減らすために使用します。正規表現パターン照合を使用します。詳細については，603 ページ「URI の切り捨ておよびマッピングの設定」を参照してください。

属性

属性	有効な値	標準設定	説明
enabled	true false	false	URI パターンの置換を有効にします。
pattern value	s/ 文字列 / 文字列 /	有効にした場合は 2 つの標準設定パターンが定義されています。	パターン値の構文は s/ search_pattern/ replace_pattern/ です。 パターンで / が使用されている場合は，区切り文字として / の代わりに # を使用します。 パターンはすべてのサーバ要求に適用され，probe_config.xml で指定された順序で適用されます。

要素

発生数	1
親要素	probeconfig, symbols
子要素	なし

例

```
<symbols maxuri="" maxsql="">
        <urireplacepattern enabled="true">
         <pattern value="s/TestService1/CommonService/"/>

        <pattern value="s/TestService2/CommonService/"/>
   </urireplacepattern>
 </symbols>
```

<vmware> 要素

目的

VMware 環境での実行時に，タイムスタンプをより正確に調節する機能を制御します。

属性

属性	有効な値	標準設定	説明
attempttime stampadjustments	true false	false	VMware 環境でのタイムスタンプの調節を可能にします。
useworkaround	true false	false	attempttimestampadjustments 属性を true に設定した状態で VMware ゲスト上で .NET Agent を実行しているときに，好ましくないレイテンシ問題が発生する場合は，この属性を true に設定する必要があります。この属性が true に設定されている場合，.NET Agent は代替呼び出しを使用して，VMware ホストのタイムスタンプを取得することによって，好ましくないレイテンシ問題に対処する。
disableperfcounters	true false	false	.NET Agent によって VMWare 環境の IIS ワーカ・プロセスがクラッシュする場合は，このオプションを true に設定します。これは，特定の VMWare 環境での perfmon カウンタへのアクセスに関する Microsoft - VMWare 環境の問題の回避策です。

要素

発生数	1
親要素	probeconfig
子要素	なし

例

```
<vmware attempttimestampadjustments="false"/>
```

<webserver> 要素

目的

プローブとの通信のためのローカル Web サーバ・プロパティを指定します。

属性

属性	有効な値	標準設定	説明
start	数値	35000	Web サーバの開始ポート。
end	数値	35100	Web サーバの終了ポート。
ipaddress	IP アドレス		Web サーバを実行するローカル IP アドレス。

例

<webserver start="35000" end="35100" ipaddress="16.255.18.99"/>

<ws> 要素

目的

Web サービスの関連付けサンプリングを制御します。

属性

なし

要素

発生数	1
親要素	<xvm>
子要素	<sample>

例

```
<xvm><ws><sample method="percent" rate="50"/></ ws ></xvm>
```

<xvm> 要素

目的

VM 間設定を制御します。

属性

なし

要素

発生数	1
親要素	probeconfig, process または appdomain
子要素	<ws>

例

```
<xvm></xvm>
```

14

.NET Agent の詳細設定

.NET Agent の詳細設定について説明します。詳細設定は，この製品に関する深い知識を有する熟練ユーザを対象にしています。Diagnostics コンポーネントのプロパティを変更する際は，十分に注意してください。

本章の内容

➤ SOAP の障害データの設定（621 ページ）

➤ その他のプローブ・メトリックスの収集またはプローブ・メトリックスの変更（622 ページ）

VMware で実行中の .NET Agent の時刻の同期

VMware ホストで実行する .NET Agent には，時刻の同期化に関する追加の要件があります。VMware ゲストで実行する Agent では，VMware ゲストとそれを支える VMware ホストとの間で時刻を同期する必要があります。時刻が正しく同期していないと，Diagnostics UI は VMware ゲストで実行中のプローブから間違った測定値を表示したり，測定値がまったく表示されなくなったりすることがあります。

時刻は，時間管理に関する VMware のホワイトペーパー（http://www.vmware.com/pdf/vmware_timekeeping.pdf）の「Synchronizing Hosts and Virtual Machines with Real Time」のセクションにある推奨事項に従って同期化する必要があります。要約すると，VMware Tools は Diagnostics プローブをホストする VMware ゲスト・オペレーティング・システムごとにインストールする必要があり，VMWare Tools の時刻同期オプションをオンにしておく必要があります。VMware Tools のこのオプションは，ゲスト・オペレーティング・システムの時刻が VMware ホストよりも早い時刻に初期設定されている場合にかぎり機能します。VMware ツールのインストール方法については，VMware ESX Server の『基本システム管理』を参照してください。また，VMware 以外の時刻同期ソフトウェア（Network Time Protocol など）を使っている場合は，VMware ESX サーバ・サービス・コンソールでそれを実行してください。

ASP.NET アプリケーションのインストルメンテーションのカスタマイズ

.NET Agent をインストールすると，監視対象サーバにおいて Agent がすべての ASP.NET 処理に適用する標準のインストルメンテーションで，**ASP.NET.points** ファイルが作成されます。

アプリケーション特有のインストルメンテーション・ポイントを作成して，アプリケーション特有のクラスおよびメソッドを通じて実装されたビジネス・ロジックのパフォーマンス測定値をキャプチャする必要があります。アプリケーション特有のインストルメンテーション・ポイントは，**<プローブのインストール・ディレクトリ> /etc/probe_config.xml** ファイルの属性を使って，アプリケーションに関連付けられるカスタム・キャプチャ・ポイント・ファイルに保存する必要があります。アプリケーションが，インストール中または IIS の再スキャン中に自動検出された場合，同時に，カスタム・キャプチャ・ポイント・ファイルがアプリケーションに自動的に作成されています。

注 : 監視するアプリケーションのクラスおよびメソッドがわからない場合は，.NET Agent と一緒にインストールされた Reflector ツールを使って，アプリケーションの .dll ファイルを分析し，クラスとメソッドを検出できます。Reflector の使用方法については，590 ページ「アプリケーションのクラスとメソッドの検出」を参照してください。

カスタム・ポイント・ファイルのインストルメンテーション・ポイントをアプリケーションに適用させることが必要な旨を .NET Agent に伝えるには，**probe_config.xml** ファイルの **appdomain** 要素の **points** 属性を更新する必要があります。

カスタム・キャプチャ・ポイント・ファイルをアプリケーションに関連付けるには

1 アプリケーション特有のクラスに，インストルメンテーションを使ってキャプチャ・ポイントを作成します。キャプチャ・ポイント・ファイルを作成するには，**<プローブのインストール・ディレクトリ> /etc** ディレクトリに既存のキャプチャ・ポイントをコピーします。

注 : インストール中または IIS の再スキャン中にアプリケーションが自動検出された場合，そのアプリケーションのキャプチャ・ポイント・ファイルは，ポイント・ファイル・エントリの一部またはすべてコメントアウトされた状態ですでに存在します。

---

2 Agent でアプリケーションのカスタム・ビジネス・ロジックをキャプチャできるように，インストルメンテーション・ポイントを追加してキャプチャ・ポイント・ファイルをカスタマイズします。

次の例では，Agent で IBuySpy カスタム・コードをキャプチャできるように，キャプチャ・ポイント・ファイルを変更する方法を示します。

```
[IBuySpy Callee]
class = !IBuySpy.*
method = !.*
signature =
scope =
ignoreScope =
layer = Custom.IBuySpy
```

インストルメンテーションの詳細については，第 10 章，「.NET アプリーションのカスタム・インストルメンテーション」を参照してください。

3 **probe_config.xml** の .NET Agent の設定を更新して，変更したキャプチャ・ポイント・ファイルが正しく参照されるようにします。

ASP.NET **<process>** タグ内で，アプリケーションに **<appdomain>** タグを追加します。**file** 属性と **enabled** 属性に **<points>** タグを含めます。その他の例については，588 ページ「異なる IIS パスにある同じ名前の仮想ディレクトリ（AppDomain）」を参照してください。

```
<appdomain name="1/ROOT/your_app_name" website="Default Web Site"
enabled="true">
      <points file="DefaultWebsite-your_app.capture points"/>
</appdomain>
```

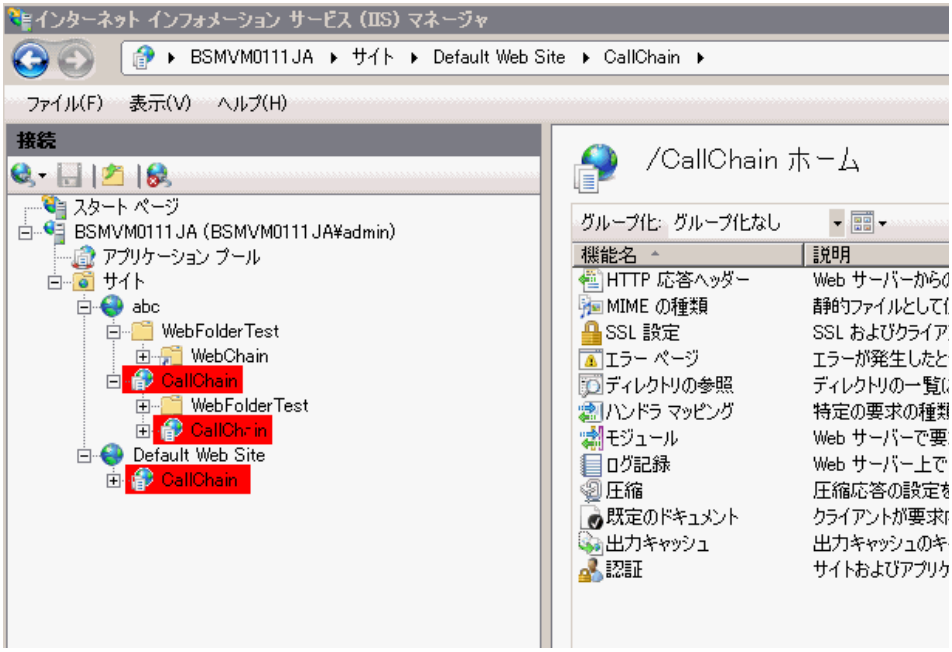
以下に，この手順の例を示します。MSPetsShop アプリケーションに，カスタム・キャプチャ・ポイント・ファイルが作成されています。ファイルには，**MSPetShop.points** という名前が付けられています。このアプリケーションの **< appdomain >** タグ，およびキャプチャ・ポイント・ファイルが，**probe_config.xml** ファイルの ASP.NET **< process >** タグに追加されました。appdomain タグに IIS パスが含まれています。

```
<?xml version="1.0" encoding="utf-8"?>

<probeconfig>
 <id probeid="" probegroup="Umatilla"/>

 <credentials username="" password=""/>
 <profiler authenticate=""><authentication username="" password=""/></profiler>

 <diagnosticsserver url="http://issaquah:2006"/>
 <mediator host="issaquah" port="2612"/>
 <webserver start="35000" end="35100"/>
 <modes am="true"/>

 <instrumentation><logging level="" threadids="no"/></instrumentation>

 <lwmd enabled="true" sample="1m" autobaseline="1h" growth="10" size="10"/>

 <process name="ASP.NET", enablealldomains="false">
      <logging level=""/>
      <points file="ASP.NET.points"/>
      <appdomain name="1/ROOT/MSPetShop" website="Default Web Site"
enabled="true">
           <points file="DefaultWebsite-MSPetShop.points"/>
      </appdomain>
 </process>
</probeconfig>
```

4 281 ページ「検出および標準インストルメンテーション」の手順に従って IIS を再起動します。

異なる IIS パスにある同じ名前の仮想ディレクトリ（AppDomain）

同じ IIS サーバ上にある複数の同じ名前の appdomain を区別できます。次に示すように，「CallChain」という名前の 3 つの仮想ディレクトリ（AppDomain）が設定されているとします。



probe_config.xml ファイルに IIS 設定パスを含めることで，これらの AppDomain を区別できます。

上の例の 3 つの CallChain アプリケーションの設定は，次のようになります。

```
<appdomain enabled="false" name="1/ROOT/CallChain/CallChain" website="Default
Web Site">
     <points file="Default Web Site-CallChain-CallChain.points" />
</appdomain>
<appdomain enabled="false" name="1/ROOT/CallChain" website="Default Web Site">
    <points file="Default Web Site-CallChain.points" />
</appdomain>
<appdomain enabled="false" name="2/ROOT/CallChain" website="WebSite2">
    <points file="WebSite2-CallChain.points" />
</appdomain>
```

この結果プローブは IIS パスを使用して区別され，Enterprise UI に 1ROOTCallChain.NET，1ROOTCallChainCallChain.NET，2ROOTCallChain.NET として表示されます。

9.01 未満のリリースとの下位互換性

下位互換性を保つため，9.01 以降のバージョンの Agent は，ASP.NET AppDomain に対する 9.01 未満のバージョンのプローブ設定の読み取り，処理を行うことができます。以前の形式では，次の例のようになります。

```
<appdomain name="CallChain">
    <points file="CallChain.points" />
</appdomain>
```

以前の形式を使用すると，Agent の動作は前のバージョンの動作に戻ります。

- 名前が「CallChain」（この例の場合）の AppDomain は，すべて同時に有効または無効になります。
- すべての CallChain のプローブインスタンスは，サーバでは 1 つのプローブとみなされます。
- プローブとサーバ要求のトレンド・ラインは，前のバージョンから継続されます。

下位互換性が不要な場合（トレンド・ラインなど），以前の形式の設定を使用しないことをお勧めします。

以前の形式で設定された appdomain で新しい動作を可能にするには，旧形式のエントリを probe_config.xml ファイルから削除する必要があります。次に，プローブ・システムの［スタート］メニューから［**Rescan ASP.NET Applications**］を実行します。これにより新しい形式の AppDomain エントリが追加され，同じ IIS サーバ上にある同じ名前の異なるプローブを区別できるようになります。

アップグレード・インストールでは，以前のバージョンの appdomain 設定が保持され，リストにない AppDomain には新しい形式の設定を追加するように probe_config.xml が変更されます。

アプリケーションのクラスとメソッドの検出

よく知らないアプリケーションのパフォーマンスを監視する場合，.NET Agent と一緒にインストールされた Reflector 自動検出ツールを使って，プローブで使用されるインストルメンテーションに追加するアプリケーションのクラスとメソッドを見つけます。Reflector の実行可能ファイルは，**＜プローブのインストール・ディレクトリ＞ ¥bin¥reflector.exe** にあります。

Reflector を使ってクラスとメソッドを検出するには，次の手順を実行します。

1. 監視するアプリケーションのインストール・ディレクトリを見つけます。
2. .dll ファイルが保存されているアプリケーション・インストール・ディレクトリのフォルダを見つけます。
3. コマンド・プロンプトを開いて，アプリケーションの .dll ファイルが保存されているフォルダにディレクトリを変更します。
4. コマンド・プロンプトで次のコマンドを実行して，現在のディレクトリのすべての .dll ファイルおよび .exe ファイルに対して Reflector を実行します。

```
＜プローブのインストール・ディレクトリ＞ ¥bin¥Reflector.exe
```

コマンドに追加パラメータを追加して，特定の .dll および .exe ファイルに Reflector を制限します。次の例は，前の例のコマンドを入力するもう 1 つの方法を示します。

```
＜プローブのインストール・ディレクトリ＞ ¥bin¥Reflector.exe *.dll *.exe
```

このコマンドは，Reflector に，対象のディレクトリのすべての .dll および .exe ファイルを確認するように指示します。

Reflector を特定のファイルに制限する場合，次のように入力します。

```
＜プローブのインストール・ディレクトリ＞ ¥bin¥Reflector.exe WorkHorse.dll
Utility.dll
```

このコマンドは，Reflector に，指定した 2 つの .dll ファイルだけを確認するように指示します。

次の例は, bin フォルダに .dll ファイルがある PetShop というアプリケーションがインストールされている場合に実行可能なコマンドを示します。

```
C:¥ ＞ cd "c:¥Program Files¥Microsoft¥PetShop¥Web¥bin"

C:¥Program Files¥Microsoft¥PetShop¥Web¥bin ＞
C:¥MercuryDiagnostics¥".NET Probe"¥bin¥Reflector.exe
```

5 Reflector は，指定した .dll ファイルで見つかったアセンブリ，名前空間，クラスおよびメソッドに関するレポートを表示します。

```
--------------------------------------------------------------------------------
Assemblies:
--------------------------------------------------------------------------------
C:\Program Files\Microsoft\PetShop\Web\bin\PetShop.BLL.dll
C:\Program Files\Microsoft\PetShop\Web\bin\PetShop.DAL.dll
C:\Program Files\Microsoft\PetShop\Web\bin\PetShop.Web.dll


--------------------------------------------------------------------------------
Namespaces:
--------------------------------------------------------------------------------
(8  classes)  PetShop.BLL
(6  classes)  PetShop.DALFactory
(17 classes)  PetShop.Web
(11 classes)  PetShop.Web.Controls
(2  classes)  PetShop.Web.ProcessFlow
(1  classes)  PetShop.Web.WebComponents


--------------------------------------------------------------------------------
(8 classes)  Namespace: PetShop.BLL
--------------------------------------------------------------------------------|
PetShop.BLL.Account (10 Methods)
    Equals              System.Boolean(System.Object)
    Finalize            System.Void()
    GetAddress          PetShop.Model.AddressInfo(System.String)
    GetHashCode         System.Int32()
    GetType             System.Type()
    Insert              System.Void(PetShop.Model.AccountInfo)
    MemberwiseClone     System.Object()
    SignIn              PetShop.Model.AccountInfo(System.String,System.String)
    ToString            System.String()
    Update              System.Void(PetShop.Model.AccountInfo)

PetShop.BLL.Cart (17 Methods)
    Add                 System.Void(System.String)
    Equals              System.Boolean(System.Object)
    Finalize            System.Void()
    get_Count           System.Int32()
    get_Item            PetShop.Model.CartItemInfo(System.Int32)
    get_Total           System.Decimal()
    GetCartItems        System.Collections.ArrayList()
    GetEnumerator       System.Collections.IEnumerator()
    GetHashCode         System.Int32()
    GetInStock          System.Int32(System.String)
    GetOrderLineItems   System.Collections.ArrayList()
    GetType             System.Type()
    MemberwiseClone     System.Object()
```

注 : 次の例のように, Reflector からの出力をファイルにリダイレクトできます。

＜プローブのインストール・ディレクトリ＞ ¥bin¥Reflector.exe sys*.dll ＞ ＜レポート名＞ .txt

Reflector からの出力は，指定したファイルにリダイレクトされます。

584 ページ「ASP.NET アプリケーションのインストルメンテーションのカスタマイズ」の手順に従って，レポートの情報を使用し，アプリケーションのインストルメンテーションをカスタマイズします。

Agent が連携して動作できる HP ソフトウェア製品の制御

.NET Agent は，次を実行する異なるモードに設定できます。

- 開発から実稼動前のテスト，実運用までアプリケーションを監視する
- ほかの HP ソフトウェア製品で使用する
- サーバまたはほかの HP ソフトウェア製品にレポートしないスタンドアロンの Diagnostics Java Profiler として使用する

  .NET Agent の動作モードは, **＜プローブのインストール・ディレクトリ＞ /etc/probe_config.xml** ファイルで設定される **<modes>** 要素で決定されます。

  <modes> 要素は，ライセンス・キャパシティの使用数を判別するためにも使用されます（83 ページ「現在接続されているプローブに基づくライセンス情報」を参照）。Diagnostics には 2 種類の LTU（使用ライセンス）があります。

- AM - Enterprise モード（通常は実運用環境）で製品を使用する場合
- AD - 製品を実稼動前の負荷テスト環境（LoadRunner または Performance Center でプローブを実行）で使用する場合

  <modes> 要素の値は，最初に .NET Agent をインストールするときに設定されます。第 8 章，「.NET Agent のインストール」を参照してください。

<modes> 要素の値を変更するには，probe_config.xml ファイルを編集します。または，.NET Agent インストーラを再実行して，［変更］オプションでモードを［Diagnostics Profiler モード］（PRO），［Diagnostics 用の Application Management/Enterprise モード］（Enterprise）や［TransactionVisions 用の Application Management/Enterprise モード］（TV），または［LoadRunner/Performance Center 用の Diagnostics モード］（AD）に設定します。

注 : スタンドアロンの Diagnostics Profiler for .NET を Enterprise モードで使用する場合, またはほかの HP ソフトウェア製品と統合して使用する場合は, HP ソフトウェア・カスタマ・サポートに連絡して HP Diagnostics を購入してください。

インタフェースしている HP ソフトウェア製品のユーザ・インタフェースで Diagnostics データを表示するには，追加の設定手順を行う必要があります。Business Service Management，LoadRunner または Performance Center との統合の詳細については，691 ページ「ほかの HP ソフトウェア製品との統合のセットアップ」の項を参照してください。

次の項では，**<modes>** 要素の各プロダクト・モードを設定するための手順を紹介します（548 ページ「<modes> 要素」も参照）。

PRO モード - Diagnostics Profiler for .NET

PRO モードが設定されている場合，Agent はパフォーマンス・メトリックスを収集し，Agent ホストの URL を介して利用できるスタンドアロンの Diagnostics Profiler for .NET ユーザ・インタフェースに表示します。

このモードの場合，Profiler UI を使用していないときでも常にデータが収集されます。このモードはほかのモードと組み合わせることができる。

PRO モードは，ライセンス・キャパシティの使用数の判別には使用されません。

Enterprise モード

Enterprise モードに設定されている場合，Agent は，Business Service Management，LoadRunner，Performance Center などの HP ソフトウェア製品と連携して，完全な Diagnostics Enterprise 製品として機能します。LoadRunner/Performance Center 実行のデータを独立したデータベースにキャプチャし，LoadRunner/Performance Center 実行以外のデータもキャプチャします。

AD と AM のモードはいずれもこのモードより優先します。

Enterprise モードでは，Diagnostics .NET Profiler にもデータが送信されます。PRO モードと Enterprise モードが設定されている場合，Profiler UI が使用中でない場合も，.NET Agent は Profiler のデータを継続的に収集します。PRO モードが設定されていない場合，Agent は Profiler UI が起動するまでデータ収集を開始しません。

Enterprise モードが .NET Agent の標準設定です（AD または AM モードを指定していない場合）。Enterprise モードでは，Agent が AM ライセンス・キャパシティに対してカウントされます。

AM モード

AM モードの場合，.NET Agent はあらゆるインストルメンテーション・データをキャプチャします。実運用環境 Business Service Management デプロイメントの Agent が LoadRunner または Performance Center の実行に誤って含まれないように保護するには，AM モードに設定します。AM モードでは，Agent は，LoadRunner または Performance Center の使用可能な Agent に表示されません。

AM モードの Agent は，常に AM ライセンス・キャパシティに対してカウントされます。

AM モードは，AD を除く，その他のすべてのモードよりも優先されます。

AD モード

AD モードの場合，.NET Agent は LoadRunner/Performance Center から実行しているときのみデータをキャプチャします。結果は，Default Client:21 など，この実行のための専用 Diagnostics データベースに格納されます。

このモードでは，プローブが LoadRunner または Performance Center の実行の一部である場合を除いて，Agent はリソースを使用したり，サーバにデータを送信しません。

AD モードは，その他のすべてのモードよりも優先します。したがって，AD モードとその他のモードが設定されているような場合，モードは AD に設定されます。

LoadRunner 統合のセットアップ方法については，第 23 章，「HP LoadRunner と HP Diagnostics の統合のセットアップ」を参照してください。Performance Center 統合のセットアップ方法については，第 24 章，「Diagnostics を使用するための Performance Center のセットアップ」を参照してください。

このモードを使用して，負荷テストが実行されていないときに，QA 環境の Agent が追加リソースを使用して，継続的に Diagnostics コンソール・データセットにデータを報告しないようにできます。

AD モードでプローブを実行する別の利点として，AD モードのプローブは LoadRunner または Performance Center でテスト実行されている場合のみに AD ライセンス・キャパシティに対してカウントされます。たとえば，LoadRunner/Performanace Center AD モードで 20 個の Agent がインストールされていて，5 個のみが実行されている場合，その 5 個のみが AD ライセンス・キャパシティに対してカウントされます。

TV モード

TV モードは Transaction Vision にイベントを送信します。このモードはほかのモードと組み合わせることができる。TV モードは，HP Diagnostics ライセンス・キャパシティの使用数の判別には使用されません。

AD モードと Enterprise モードに関する注意事項

.NET Agent は，Diagnostic Mediator から LoadRunner / Performance Center 実行に関する通知を受けます。

LoadRunner / Performance Center が，動作していないインストルメンテーション済みアプリケーション（最初の一致を取得する Web アプリケーションなど）のテストを開始した場合，そのアプリケーションが Diagnostics の実行を開始した時点で Agent はその実行に関する通知を受け取りません。これは，初期化し，この通知に関して Mediator をリッスンするために必要な時間が Agent にないためです。

この問題を回避するには，LoadRunner / Performance Center の実行を開始する前に Web アプリケーションに対する呼び出しを行って，.NET Agent を「準備」（初期化）する必要があります。このようにすると，Web アプリケーションのプロセス（ワーカ・プロセス）およびプローブが初期化されるため，Mediator から実行に関する情報を受信する準備が整います。

MSMQ ベース通信のサポートの設定

MSMQ ベース通信をサポートするように .NET Agent を設定するには，**<プローブのインストール・ディレクトリ> /etc/probe_config** ファイルの例にあるように，appdomain の範囲に **msmq.points** を含めます。

```
<process name="SimplestQueuingSender">
<points file="msmq.points"/>
<modes enterprise="true"/>
</process>
```

レイテンシのトリミングおよびスロットリングの設定

プローブでキャプチャしているデータの量に Diagnostics サーバが対応しきれないために，リソース不足が生じていることを .NET Agent で特定すると，レイテンシのトリミングというプロセスを使ってプローブがキャプチャするメソッドの数を自動的に減らすことができます。標準設定では，プローブが必要に応じて作業負荷を調整できるように，レイテンシのトリミングは有効になっています。

レイテンシのトリミングが有効になると，.NET Agent は，合計レイテンシが最小レイテンシしきい値を下回っているメソッドを無視して，プローブがキャプチャするメソッドの数を減らします。トリミングは，プローブがフリーズしたり実行を停止するよりも，レイテンシが小さく，関心が低いメソッドをキャプチャしない方がいいという考えに基づきます。トリミングによって，プローブは実行を継続し，レイテンシが長く，関心の高いメソッドをキャプチャできるようになります。

注 : スレッド処理とバッファリングが原因で，トリミングされたメソッドに関する一部の情報が Diagnostics サーバに送信されることがあります。Diagnostics サーバでメソッドの情報を一部だけ受信したことを検出すると，警告メッセージを発行します。すべてのメソッドの情報がキャプチャされるはずだった場合を除いて，この警告メッセージは無視してください。

注 :

- レイテンシのトリミングおよびスロットリングは，Profiler のユーザ・インタフェースに無視されます。
- プローブ・データのトリミングをさらに適用して Diagnostics ユーザ・インタフェースに表示されるデータの粒度を変更するように，Diagnostics サーバを設定できます。

レイテンシのトリミングの無効化

デフォルトで，トリミングは .NET Agent で有効になっています。トリミングを無効にするには，設定を変更する必要があります。

レイテンシのトリミングを無効にするには，次の手順を実行します。

次の例のように，**＜プローブのインストール・ディレクトリ＞ /etc/probe_config.xml** 設定ファイルに **latency** タグを追加します。

```
<trim>
 <latency enabled="false" />
</trim>
```

レイテンシのトリミングをオンにする latency 要素の属性は，**enabled** です。**enabled** を true に設定すると，レイテンシのトリミングが有効になります。**enabled** 属性を false に設定すると，レイテンシのトリミングが無効になります。この属性の標準設定値は true です。

latency 要素の属性と要素については，第 13 章，「.NET Agent 設定ファイルについて」を参照してください。

レイテンシのトリミングの有効化

デフォルトで，トリミングは .NET Agent で有効になっています。後でトリミングを無効にした場合，再び有効にするには，設定を変更する必要があります。

レイテンシのトリミングを有効にするには，次の手順を実行します。

次の例のように，**＜プローブのインストール・ディレクトリ＞ /etc/probe_config.xml** 設定ファイルの **latency** 要素の有効な属性の値を変更します。

```
<trim>
 <latency enabled="true" />
</trim>
```

レイテンシのトリミングをオンにする latency 要素の属性は，**enabled** です。**enabled** を **true** に設定すると，レイテンシのトリミングが有効になります。**enabled** 属性を **false** に設定すると，レイテンシのトリミングが無効になります。この属性のデフォルト値は **true** です。

latency 要素の属性と要素については，第 13 章，「.NET Agent 設定ファイルについて」を参照してください。

レイテンシのトリミングのしきい値の設定

デフォルトで，レイテンシのトリミングのしきい値は，2 ms 未満のレイテンシを持つメソッドがトリミングされ，100 ms より長いレイテンシを持つメソッドをトリミングしないように設定されています。

min 属性の値を調整して，トリミングの最小しきい値を設定することができます。**max** 属性の値を調整して，トリミングの最大しきい値を設定することができます。これらの属性は，**＜プローブのインストール・ディレクトリ＞ /etc/probe_config.xml** 設定ファイルの **latency** 要素で指定されます。

```
<trim>
 <latency enabled="true" min="50" max="100" />
</trim>
```

トリミングしきい値を制御する latency 要素の属性は次のとおりです。

- **min**

  レイテンシの最小しきい値を設定します。レイテンシのトリミングが有効になると，この属性の値に等しいか，それよりも小さなレイテンシを持つメソッドがトリミングされます。この属性に値を指定しない場合，標準設定値 2 ms が使用されます。

  min 属性の値が小さくなるほど，アプリケーションのパフォーマンスに悪影響を与える可能性が大きくなります。値が小さいと，レイテンシの小さなメソッドがキャプチャされるため，トリミングされるメソッドが少なくなります。

  すべてのメソッドの情報をキャプチャする必要がある場合は，**latency enabled** を false に設定してレイテンシのトリミングを無効にします。

- **max**

  レイテンシの最大しきい値を設定します。レイテンシのトリミングが有効になると，この属性の値に等しいか，それよりも大きなレイテンシを持つメソッドがトリミングされます。値を指定しない場合，この属性の標準設定値は 100ms になります。

latency 要素の属性と要素については，第 13 章，「.NET Agent 設定ファイルについて」を参照してください。

レイテンシのトリミングのスロットリングの設定

デフォルトで，レイテンシのトリミングはスロットリングされます。スロットルを有効にすると，実行されるトリミングの量は，Diagnostics サーバの処理中バックログによって使われているプローブ・リソースの割合に基づいて自動的に調整されます。

スロットリングを使わないと，メソッドの最小レイテンシしきい値を下回ったメソッドが必ずトリミングされます。

プローブで使われるリソースの割合が，設定スロットルの増分しきい値を超えた場合，長いレイテンシを持つメソッドがトリミングされるように，有効なトリミングしきい値が高くなります。使われるプローブ・リソースの割合が再びしきい値を超えると，さらに長いレイテンシを持つメソッドがトリミングされるように，有効なトリミングしきい値が再度高くなります。使われるプローブ・リソースの割合がスロットルの減分しきい値を下回ると，短いレイテンシを持つメソッドが再度キャプチャされるように，有効なトリミングしきい値が低くなります。

有効なトリミングしきい値は，最大のメソッド・トリミング時間しきい値を超えられず，メソッドの最小有効時間しきい値を下回ることもできません。

以下は，スロットリング属性が含まれる **probe_config.xml** 設定ファイルの **latency** 要素の例です。

```
<trim>
 <latency enabled="true" min="50" max="100"
 throttle="true" incrementthreshold="75"
 decrementthreshold="50" increment="2"/>
</trim>
```

スロットリングを制御する **latency** 要素の属性は次のとおりです。

- **throttle**

  この属性を true に設定すると，スロットリングが有効になります。この属性を false に設定すると，スロットリングが無効になります。この属性の標準設定値は true です。

- **increment**

  使われるプローブ・リソースの割合が **incrementthreshold** を超えたときに，有効なトリミングしきい値が高くなる量を設定します。**decrementthreshold** が渡されたときに，有効なトリミングしきい値が低くなる量を設定します。この属性のデフォルト値は 2 です。

- **incrementthreshold**

  使われるプローブ・リソースの割合が，この属性の値以上になったとき，有効なトリミングしきい値が高くなるようにスロットルがトリガされます。この属性のデフォルト値は **75**% です。

➤ **decrementthreshold**

使われるプローブ・リソースの割合が，この属性の値以下になったとき，有効なトリミングしきい値が低くなるようにスロットルがトリガされます。この属性のデフォルト値は **50**% です。

latency 要素の属性と要素については，第 13 章，「.NET Agent 設定ファイルについて」を参照してください。

深さのトリミングの設定

.NET Agent は，深さのトリミングというプロセスを使って，キャプチャするメソッドの数を自動的に減らします。Diagnostics サーバが，プローブがキャプチャしているデータの量に対応できない場合，プローブは深さのトリミングを使って，リソース不足を防止します。デフォルトで，深さのトリミングは有効になっています。

注 : 深さのトリミングは，Profiler ユーザ・インタフェースに無視されます。

深さのトリミングが有効になると，.NET Agent は，スタックの最大深さしきい値を超えるスタックの深さで呼び出されるメソッドを無視して，キャプチャされたメソッドの数を減らします。スタックの深さしきい値以下のスタックの深さで呼び出されるメソッドはキャプチャされます。トリミングは，呼び出しスタックの下部にある，関心が低いメソッドをキャプチャしないことによって，プローブが実行を続け，呼び出しスタックの上部で発生する関心の高いメソッドをキャプチャできるようにする方がいいという考えに基づきます。

たとえば，スタックの深さしきい値が 3 の場合，次のメソッド呼び出しが行われます。

```
/login.do calls a() calls b() calls c()
```

/login.do，a および b メソッドだけがキャプチャされる場合，メソッド c はトリミングされます。

以下は，トリミング属性が含まれる **probe_config.xml** 設定ファイルの **depth** 要素の例です。

```
<trim>
 <depth enabled="true" depth="10" />
</trim>
```

トリミングを制御する **depth** 要素の属性は次のとおりです。

- **enabled**

  この属性を true に設定すると，深さのトリミングが有効になります。この属性を false に設定すると，深さのトリミングが無効になります。この属性の標準設定値は true です。

- **depth**

  深さのトリミングに使われるしきい値を設定します。深さのトリミングが有効になると，この属性以下で呼び出されるメソッドはトリミングされます。この属性のデフォルト値は **25** です。

  depth を低い値に設定すると，キャプチャのオーバーヘッドが大幅に減少します。**depth** 要素の属性と要素については，第 13 章，「.NET Agent 設定ファイルについて」を参照してください。

URI の切り捨ておよびマッピングの設定

すべての HTTP/S サーバ要求 URI は，プローブでレポートされる前に変換することができます。この変換は，正規表現照合および **probe_config.xml** 設定ファイルの **urireplacepattern** 要素で制御される置換に基づきます。この変換は標準設定では無効になっています。

サーバ要求が多すぎるため，多数のサーバ要求 URI をまとめて 1 つのサーバ要求 URI に置き換える場合にこれが役立ちます。

重要 : この機能を多用するとパフォーマンスに影響します。

次に，例を示します。

```
<symbols maxuri="" maxsql="">
        <urireplacepattern enabled="true">
         <pattern value="s/TestService1/CommonService/"/>

        <pattern value="s/TestService2/CommonService/"/>
   </urireplacepattern>
 </symbols>
```

パターン値に使用する構文は s/search_pattern/replace_pattern/ です。

search_pattern と replace_pattern は / で囲みます。パターンで / が使用されている場合は，区切り文字として / の代わりに # を使用します。

パターンはすべてのサーバ要求に適用され，probe_config.xml ファイルで指定された順序で URI に適用されます。

urireplacepattern が有効になっている場合，2 つの標準設定パターンが設定されます。

最初の標準設定パターンは, ; または /! を含むサーバ要求をトリミングするために使用されます。これらのトークンの後のすべてのコンテンツがサーバ要求から削除されます。

使用されるパターン : s#(;|/?¥¥!).*$##"

2 番目の標準設定パターンは，画像，pdf，doc の読み込みを固定トークン（「/Static Content」）で置き換えます。

使用されるパターン :

s#(?<word1>^.*)(/.*¥¥.js|css|jpg|gif|png|pdf|html|doc|docx)#${word1}/Static Content#

両方のパターンがカスタマイズ可能です。

ライトウェイトなメモリ診断用の .NET Agent の設定

ライトウェイトなメモリ診断（LWMD）機能は，コレクションに関連する使用データをキャプチャし，分析する機能です。コレクションは，**System.Collections.ICollection** インタフェースまたは **System.Collections.Generic.ICollection** インタフェースを実装するすべてのクラスを表します。このようなコレクションの例は，ArrayList，HashTable，DataView などです。.NET メモリ・リークが最も発生しやすいのは，コレクションが適切に維持されていない場合です。

.NET Agent がインストールされている場合，.NET Agent の標準設定では，プローブの LWMD はオフです。LWMD 機能を有効にするには，**probe_config.xml** ファイルに 2 つの変更を施す必要があります。

- <lwmd> 要素を有効にする（542 ページ「<lwmd> 要素」を参照）。
- **Lwmd.points** ファイルに対する参照を 1 つ以上追加する（次の説明を参照）。

注 : コレクション・メトリックスをキャプチャできるようにプローブを設定すると，アプリケーションのホストのオーバーヘッドが増加する可能性があります。

プロセスまたはアプリケーションのコレクション・メトリックスをキャプチャできるようにするには，次の手順を実行します。

Lwmd.points ファイルの **points** タグを **process** タグ，または **probe_config.xml** 設定ファイルの 1 つ以上の **appdomain** タグに追加します。

.NET Agent をインストールしている場合，**ASP.NET.points** および **ADO.points** ファイルと一緒に，**Lwmd.points** ファイルが **<プローブのインストール・ディレクトリ> /etc/** ディレクトリにインストールされます。**Lwmd.points** ファイルには，コレクション・インストルメンテーション値のキャプチャを有効にするのに必要なインストルメンテーション命令が含まれます。

プロセスに従って実行される有効なすべての appdomain に対して LWMD インストルメンテーションを有効にするには，**probe_config.xml** 設定ファイル内のプロセス・タグに points タグを追加します。たとえば，有効なすべての ASP.NET appdomain に対して LWMD インストルメンテーションを有効にするには，次の手順を実行します。

```
< process name="ASP.NET", < enablealldomains="false" >
    <points file="ASP.NET.points" />
    <points file="ADO.points" />
    <points file="Lwmd.points"/>
    <appdomain name="1/ROOT/your_app_name" website="Default Web Site"
enabled="true">
        <points file="DefaultWebsite-your_app.capture points" />
    </appdomain>
</process>
```

プロセスに従って実行される有効な特定の appdomain に対して LWMD インストルメンテーションを有効にするには，**probe_config.xml** 設定ファイル内の appdomain タグに points タグを追加します。points タグは 1 つ以上の appdomain タグに追加できます。たとえば，ASP.NET プロセスで実行中の「your_app_name」という appdomain に対して LWMD インストルメンテーションを有効にするには，次の手順を実行します。

```
    < process name="ASP.NET", < enablealldomains="false" >
    <points file="ASP.NET.points" />
    <points file="ADO.points" />
    <appdomain name="1/ROOT/your_app_name" website="Default Web Site"
enabled="true">
        <points file="DefaultWebsite-your_app.capture points" />
        <points file="Lwmd.points"/>
    </appdomain>
</process>
```

LWMD を無効にするには，次の手順を実行します。

LWMD 機能を無効にするには，**probe_config.xml** ファイルに 2 つの変更を施す必要があります。

- <lwmd> 要素を無効にする（542 ページ「<lwmd> 要素」を参照）
- すべてのプロセス・タグおよび該当する appdomain タグから，**Lwmd.points** ファイルの points タグを削除する

設定ファイルに LWMD points タグがないと，プローブは Lwmd.points ファイルに含まれている LWMD インストルメンテーションに関する指示を見つけることができないため，コレクション使用量に関するインストルメンテーションを実行しません。

LWMD インストルメンテーションを制御するには，次の手順を実行します。

.NET Agent がインストールされている場合，Lwmd.points ファイルの標準設定には，さまざまなアセンブリ，appdomain，名前空間，およびクラスでのコレクション使用量をインストルメンテーションするための指示が含まれています。アプリケーションの points ファイルを変更すると，検査するコレクションの範囲を絞ることができます。LWMD インストルメンテーションは，呼び出し側インストルメンテーションとして実装されます。このインストルメンテーションの仕組みについては，398 ページ「呼び出し側のインストルメンテーション」を参照してください。

注 : 推奨されるベスト・プラクティスは，LWMD インストルメンテーションの範囲を絞ることです。

検査するコレクションの範囲を絞るには，次の手順を実行します。

1. プロセス・タグおよび該当する appdomain タグから，Lwmd.points ファイルの points タグを削除します。このようにすると，広範なインストルメンテーション範囲を指定する LWMD 設定が削除されます。
2. プロセスまたは appdomain の points ファイルに LWMD セクションを追加します。そのためには，例として，次の内容を **your_app.points** ファイルにコピーして貼り付けます。

```
[LWMD]
keyWord = lwmd
scope =
ignoreScope =
```

3 検査するコレクションの範囲を絞るには，LWMD セクションで scope および ignoreScope 引数を設定します。例：

```
[LWMD]
keyWord = lwmd
scope = !my_namespace¥..*
ignoreScope = !my_namespace.my_class1¥..*
```

上記の例では，my_namespace.my_class1 クラス内の任意のメソッドから構築されたコレクションを除いて，my_namespace 名前空間から構築されたすべてのコレクションをインストルメンテーションします。

LWMD インストルメンテーションの場合，ignoreScope に関する内部的な標準設定値があります。この値は非公開であり，常にユーザ入力値によって追加されます。標準設定値には .NET インフラストラクチャに関連した名前空間およびクラスが含まれています。これらをインストルメンテーションした場合は，アプリケーションに悪影響を与えます（!System.*，!Microsoft.* など）。

例外スタック・トレース・データの制限

Agent はサーバ要求をスローする例外については例外データを収集し，情報を Diagnostics UI に表示します。収集した例外データには，必要に応じてスタック・トレースを含めることができます。

ただし，注目していない例外スタック・トレースは，表示，データ収集，転送処理の負荷がかかるため，すべての例外についてスタック・トレース・データを収集するのは通常は望ましいことではありません。このため，スタック・トレース・データを収集する例外タイプを制限できます。たとえば，**System.Security.Authentication.AuthenticationException** などのアプリケーション・サーバ・ベースのエラーをフィルタリングすると，よりアプリケーション固有のエラーを対象にスタック・トレースが使われます。

収集するスタック・トレース・データは，次の 3 つの方法で制御されます。すなわち，特定の例外タイプを制限する方法，収集するスタック・トレース・データの例外の数を制限する方法，スタック・トレース・データを制限する方法です。

注 : probe_config.xml ファイルで **captureexceptions enabled**="false" を設定することにより，すべてのスタック・トレースの収集を無効にできます。デフォルトでは，スタック・トレースの収集は有効になっています。

本項の内容

- 特定の例外タイプの制限
- サーバ要求ごとの例外数の制限
- スタック・トレースのサイズの制限
- 例

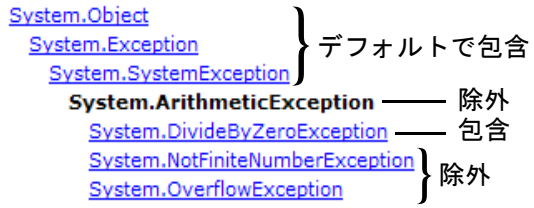
特定の例外タイプの制限

スタック・トレース・データが収集される例外は，次の例に示すように，**probe_config.xml** ファイルの **exclude** プロパティと **include** プロパティを設定することで制限されます。

```
<exclude>
      <exceptiontype name="System.ArithmeticException"/>
</exclude>
<include>
      <exceptiontype name="System.DivideByZeroException"/>
</include>
```

除外または包含として指定された例外タイプのサブタイプも，包含リストまたは除外リストで明示的に指定されない限り，除外または包含されます。

次の図は，前記の例に基づいてどの例外タイプが包含され，除外されるかを示します。



probe-config.xml ファイルへの変更は即座に有効になり，アプリケーションを再起動する必要はありません。

サーバ要求ごとの例外数の制限

標準設定では，.NET Agent のプローブはサーバ要求中に見つかった最初の 4 つの例外に対してのみスタック・トレース・データを収集します。スタック・トレース情報を確認する必要がある例外がアプリケーションにもっと多くある場合は，**probe_config.xml** ファイルで **max_per_request** プロパティの値を増やすことができます。収集するすべての測定値と同様に，収集するデータの量が増えると，Diagnostics サーバの負荷は高くなります。

スタック・トレースのサイズの制限

デフォルトでは，キャプチャするスタック・トレース・データのサイズは任意です。スタック・トレース文字列のサイズを制限すると，[Exceptions] タブを読みやすくできます。**probe_config.xml** ファイルで，**max_stack_size** プロパティの値を最大スタック・トレース文字列に設定します。収集するすべてのデータと同様に，収集するデータの量が増えると，Diagnostics サーバの負荷は高くなります。デフォルトではこのプロパティは 0（ゼロ）に設定されています。これは，スタック・トレース・サイズに制限がないことを意味します。

例

次の設定で，最小スタック・トレース文字列サイズを 2048 として，例外スタック・トレースを有効にできます。

```
<captureexceptions enabled="true" max_per_request="4" max_stack_size="2048">
  <exclude>
      <exceptiontype name="System.ArithmeticException"/>
  </exclude>
  <include>
      <exceptiontype name="System.DivideByZeroException"/>
 </include>
</captureexceptions>
```

記録の無効化

次の例のように，**probe_config.xml** ファイルの ASP.NET プロセス・セクションの **logging level** タグを変更して，アプリケーションの記録機能を無効にできます。

```
<process name="ASP.NET">
      <logging level="off"/>
</process>
```

次の例のように，インストルメンテーション・セクションの **logging level** タグを変更して，インストルメンテーションの記録機能を無効にできます。

```
<instrumentation>
      <logging level="off" />
</instrumentation>
```

標準設定のプローブホスト・マシン名の変更

registered_hostname プロパティを使って，プローブが Diagnostics コマンド・サーバへの登録に使用する標準設定のホスト・マシン名を変更できます。ファイアウォールまたは NAT がある場合，またはプローブ・ホスト・マシンがマルチホーム・デバイスとして設定されている場合，標準設定のホスト・マシン名を変更しないかぎり，Diagnostics コマンド・サーバはプローブと通信できません。

プローブの標準設定のホスト・マシンを変更するには，次の 3 つの手順で行います。

1 まず，**probe_config.xml** ファイルの .NET Agent **<diagnosticsserver>** 要素にある **registered_hostname** 属性を，Diagnostics コマンド・サーバがプローブと通信できる別のマシン名または IP アドレスに設定します。

次に例を示します。

```
<diagnosticsserver url="http://localhost:2006/commander"
registered_hostname=" my_host_name "/>
```

2 次に，.NET Metrics Agent にホストの代替マシン名または IP アドレスを登録します。そのためには，**metrics.config** ファイル内に **metrics.agent.registered_hostname** エントリを作成します。このエントリは，metrics.systemgroup エントリの真下に追加できます。

次に例を示します。

```
metrics.systemgroup = Default
metrics.agent.registered_hostname = my_host_name
```

3 最後に，.NET Agent と .NET Metrics Agent を両方とも再起動して，この変更を有効にする必要があります。

注 :

- IIS ホスト・ヘッダ・テクノロジを使って適切に処理するように，**registered_hostname** 属性を設定する必要があります。
- NAT またはファイアウォールが原因で **registered_hostname** 属性を設定することは，LoadRunner，Performance Center，または Diagnostics Standalone を使用するテスト環境における唯一の問題です。
- ただし，Business Service Management または Diagnostics Standalone を使用している実稼動環境で registered_hostname を設定する必要がある場合，指定した名前はシステムの状況にホスト名として表示されます。

ホストで実行中のプローブの一覧表示

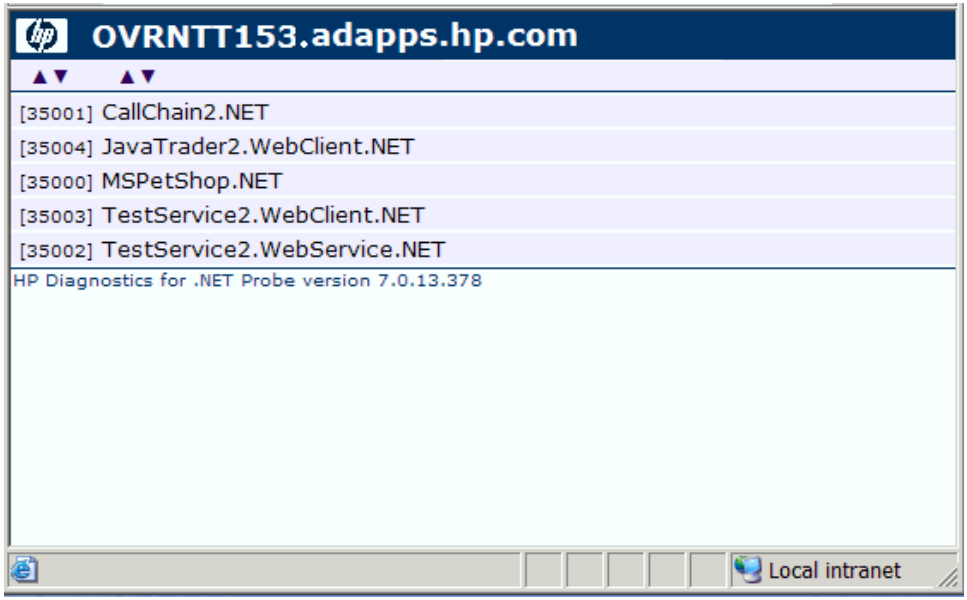
1 台のホストで複数のプローブを実行している場合, 割り当てられているポートが，アプリケーション（および プローブ）の起動時に使用可能なものに基づくため，各プローブで使用するポートを把握できません。アプリケーションを起動および停止するときに，特定のアプリケーションのプローブに割り当てられているポートが変更されることがあります。

次の URL にアクセスして，ホストで実行されているプローブと，それらが使用しているポートを特定できます。

```
http:// ＜プローブのホスト＞ : ＜ポート＞
```

ポート値に，ポート番号 35000 または 35001 を入力します。どのポート番号を入力してもかまいません。

プローブとポートの一覧は，次の例のように表示されます。



.NET Profiler での権限と認証

Profiler のユーザの権限と認証は，**＜プローブのインストール・ディレクトリ＞/etc/probe_config.xml** ファイルで管理できます。

注 : .NET Agent が Diagnostics サーバと連携して動作するように設定されている場合，プローブ（Profiler）の認証と承認設定は，このプローブが接続されている Diagnostics コマンド・サーバから管理します。詳細については，755 ページ「ユーザの認証と承認」を参照してください。

Diagnostics サーバからプローブにアクセスする場合，標準設定のユーザ名は admin で，標準設定のパスワードは admin です。

.NET Agent が Profiler 専用でインストールされている場合，標準設定で，ユーザは Profiler にアクセスするのにユーザ名とパスワードを入力する必要がありません。

ただし，ユーザ認証を求めるように Profiler を設定できます。ユーザ認証を要求するように Profiler を設定すると，Profiler へのアクセスに必要なパスワードを定義できます。

ユーザ認証を要求するように Profiler を設定するには，次の手順を実行します。

➤ **＜プローブのインストール・ディレクトリ＞/etc/probe_config.xml** ファイルを開き，**profiler authenticate** の値を **true** に設定します。

```
<profiler authenticate="true">
      <authentication username="Test" password="uU8X9zOtl6Twi7TkGAhQ="/>
</profiler>>
```

ユーザ名とパスワードを設定していない場合，デフォルトのユーザ名は admin で，デフォルトのパスワードは admin です。

.NET Diagnostics Profiler のユーザに新しいユーザ名とパスワードを作成するには，次の手順を実行します。

1 **＜プローブのインストール・ディレクトリ＞ /bin** ディレクトリにある **PassGen.exe** ユーティリティを使って，新しいユーザ名とパスワードを作成します。暗号化のためのユーザ名とパスワードを入力します。ユーザに生成される暗号化されたパスワードは，FIPS-2 に準拠しています。

2 **＜プローブのインストール・ディレクトリ＞ >/etc/probe_config.xml** ファイルで，**＜ profiler authenticate="true" ＞**行の後に，次の形式で各新規ユーザのユーザ名とパスワードを入力します。

```
<profiler authenticate="true">
      <authentication username="" password=""/>
</profiler>
```

- **authentication username** には，PassGen ユーティリティを実行するときに選択したユーザ名を入力します。
- **password** には，**PassGen.exe** ユーティリティが返したエンコードされた文字列を入力します。

注意：Profiler へのアクセスに新しいユーザ名とパスワードを定義すると，デフォルトのユーザ名とパスワード（admin，admin）は使用できなくなります。定義した新しいユーザ名のいずれかを使用する必要があります。

コンシューマ ID の設定

Web サービス測定値は，Web サービスの特定のコンシューマ別にグループ化できます。測定値はその後，そのコンシューマに対して集計され，Services by Consumer（コンシューマ ID 別サービス）ビューや，Operations by Consumer ID（コンシューマ ID 別動作）ビューのように表示されます。

コンシューマ ID 別のデータの集計は，誰がどのサービスを使用しているかとその使用頻度を調べる場合に役立ちます。コンシューマ ID は Business Service Management に対しても有用です。BSM ユーザは，コンシューマに基づいて同一のアプリケーションのパフォーマンスを表示して，パフォーマンスの特徴を比較できます。

コンシューマ ID の設定は任意です。デフォルトでは，監視対象の Web サービスのコンシューマ ID は，Web サービスのコンシューマの IP アドレスとしてレポートされます。

コンシューマ ID の定義方法は次の 3 とおりあります。

- SOAP 要求に表示される値
- HTTP ヘッダに表示される値
- 特定の IP アドレスまたは IP アドレスの範囲

コンシューマ ID 設定の基本手順

コンシューマ ID を設定するための基本手順は次のとおりです。

1. 617 ページ「コンシューマ ID ルールの構文と .NET Agent の例」の説明に従って，コンシューマ別に測定値をグループ化する .NET プローブごとに，probe_config.xml ファイルを更新します。
2. 6 個以上のコンシューマ・タイプを設定している場合は，**server.properties** ファイルで **max.tracked.ids.per.probe** の設定を書き換えます。

コンシューマ ID のルール

コンシューマ ID の割り当ては，**probe_config.xml** ファイル**のコンシューマ ID ルールによって制御されます。**

コンシューマ ID のカテゴリごとに独自のルールがあります。すなわち，SOAP ルール，HTTP ヘッダ・ルール，IP ルールです。ルールは，それが定義された順番に関係なく順序どおりに適用されます。SOAP ヘッダ・ルールが最初に適用され，次に HTTP ヘッダ・ルールが適用され，最後に IP ルールが適用されます。

すべてのルール・タイプを使用する必要はありません。SOAP ルールがあり，HTTP ルールと IP ルールがない場合もあります。これらのルールのいずれにも一致するものがない場合，元の IP アドレスがコンシューマ ID として使用されます。

SOAP ルールを使用すると，コンシューマ ID を SOAP ヘッダ，エンベロープ，または本体の XML 要素から取得できます。ルールは，コンシューマによって呼び出される Web サービス名との照合に使われる正規表現を指定します。正規表現を使用したヘルプについては，882 ページ「正規表現の使用」を参照してください。

Web サービス名との一致がある場合，Agent / Probe は SOAP ルールで定義された該当する SOAP の場所で，consumeridfield で定義された要素を検出しようとします。要素が見つからない場合，このルールはスキップされ，Agent / Probe は定義されている次のルールに進みます。

HTTP ヘッダ・ルールを使用すると，コンシューマ ID を HTTP 要求内の一連の HTTP のヘッダから取得できます。

IP ルールを使用すると，コンシューマ ID を IP アドレスとコンシューマ ID のマッピングから取得できます。ルールを使用して，コンシューマ ID に割り当てる IP アドレスや IP アドレスの範囲を定義します。

コンシューマ ID ルールの構文と .NET Agent の例

ルール構文と例は，コンシューマ ID の定義方法によって異なります。

SOAP ルール

SOAP ルールを使用すると，コンシューマ ID を SOAP ヘッダ，エンベロープ，または本体の XML 要素から取得できます。

次に，SOAP ヘッダの値に基づいてコンシューマ ID を設定する例を示します。

```
<consumeridrules enabled="true">
  <soaprules>
     <soaprule id="SOAP1" rule="TestService" location="soap-header"
consumeridfield="Caller"/>
  </soaprules>
</consumeridrules>
```

id= 属性には，ユーザがルールの識別に使用する任意の名前を指定できます。この属性は .NET プローブには使用されません。

soaprule では rule= 属性を定義する必要があります。ルールは，コンシューマによって呼び出される Web サービス名との照合に使われる正規表現です。正確な Web サービス名を使用することもできます。

location= は「soap-header」，「soap-envelope」，「soap-body」に設定できます。location を指定しない場合は，標準設定の「soap-header」が使用されます。いずれかの SOAP ルールで location を設定した場合は，すべての SOAP ルールで location を設定する必要があります。設定しない場合，重大なエラーが発生し，SOAP ロジックに基づくコンシューマ ID が無効になります。

soaprule では consumeridfield= 属性を定義する必要があります。コンシューマ ID として値を使用する SOAP ヘッダ，エンベロープ，または本体の要素です。

rule= 属性で指定されたパターンと一致するものがあれば，.NET Agent は consumeridfield で定義された要素のテキスト要素を探そうとします。consumeridfield 内の要素には，修飾された名前（名前空間名とローカル部分で構成），または修飾されていない名前（名前空間が関連付けられていない）を指定できます。指定場所に要素が見つからない場合，このルールはスキップされ，プローブは定義されている次のルールに進みます。

たとえば，次のルールはサービス名が TestService の Web サービスを照合し，Caller 要素の値をコンシューマ ID として使用します。

```
<soaprule id="SOAP1" rule="TestService" location="soap-header"
consumeridfield="Caller"/>
```

TestService Web サービスの呼び出し元が Caller の値を定義しているかぎり，メトリックスは Caller に対してさまざまな値でグループ化されます。次に，TraderService のこの呼び出し元の Customer2 というコンシューマ ID にマップされる soap ヘッダからの抜粋です。

```
SoapTest1;WS<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/
envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <env:Header>
      <Caller>Customer2</Caller>  <-- The consumer id returned is
                                                "Customer2"
   </env:Header>
   <env:Body   env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <m:sell    xmlns:m="http://www.bea.com/examples/Trader">
         <string     xsi:type="xsd:string">sample string</string>
         <intVal     xsi:type="xsd:int">100</intVal>
      </m:sell>
   </env:Body>
</env:Envelope>
```

SOAP キャプチャの有効化

SOAP エンベロープはサイズが巨大になることがあるため，SOAP の要求と応答をキャプチャする際のオーバーヘッド（主にメモリ・オーバーヘッド）を制御するための< soapcapture >要素が用意されています。

<soapcapture enabled="true">

< soapcapture >要素は，SOAP 要求および応答をキャプチャするかどうかを制御します。無効な場合，SOAP 要求および応答はキャプチャされません。つまり，TransactionVision イベントに含まれる SOAP 要求または応答がなくなるか，SOAP の失敗時に使用できる SOAP 要求がなくなります。ユーザは SOAP ヘッダ，エンベロープ，または本体に基づいてコンシューマ ID を設定できません。

< soapcapture >設定は，SOAP の失敗が発生したときに SOAP ペイロード・キャプチャを制御する< soaprequestforsoapfault >設定をオーバーライドします。詳細については，621 ページ「SOAP の障害データの設定」を参照してください。

HTTP ヘッダ・ルール

HTTP ヘッダ・ルールを使用すると，コンシューマ ID を HTTP 要求内の一連の HTTP のヘッダから取得できます。ルールと consumeridfield 属性はどちらも HTTP ルール要素で定義する必要があり，個々のルールをユーザが識別するために id 属性も定義できます。

ルールは，コンシューマによって HTTP 要求が送信される URL との照合に使われる正規表現です。一致するものがあれば，.NET プローブは consumeridfield で定義されたヘッダ名に対応する HTTP ヘッダ名を探そうとします。一連の HTTP ヘッダ内にヘッダ名が見つからない場合，このルールはスキップされ，プローブは定義されている次のルールに進みます。

httpheader ルールの例 :

```
<consumeridrules enabled="true">
  <httpheaderrules>
      <httpheaderrule id="httpHeader 1" rule="/Webservice/.*
consumeridfield="Caller"/>
  </httpheaderrules>
</consumeridrules>
```

IP アドレス・ルール

IP ルールを使用すると，コンシューマ ID を IP アドレスとコンシューマ ID のマッピングから取得できます。ルールと consumerid 属性はどちらも IP ルール要素で定義する必要があり，個々のルールをユーザが識別するために id 属性も定義できます。

ルールを使用して，コンシューマ ID に割り当てる IP アドレスや IP アドレスの範囲を定義します。このルールは，19.225.17.125 のように 1 つの IP アドレスとして定義できます。また，19.255.17.125,19.255.17.255 のように範囲を定義することもできます。

さらに，19.255.17.* のように，IP アドレスのオクテットにアスタリスクを使用して，そのオクテット内のすべてと一致させることもできます。19.255.17.20-255 のように，1 つのオクテット内で範囲を定義し，そのオクテット内の値の範囲と一致させることもできます。19.*.17.20-255, 20.*.10-55.* のように，これらの組み合わせを使用することもできます。一致するものがある場合，.NET プローブはそのコンシューマ ID をルールで定義されたコンシューマ ID に設定します。

例：

```
<consumeridrules enabled="true">
  <iprules>
      <iprule id="IpTest1" rule="18.*.1-20.*" consumerid="Client1"/>
      <iprule id="IpTest2" rule="17.*.*.*" consumerid="Client2"/>
      <iprule id="IpTest3" rule="19.255.17.125,19.255.17.255" consumerid="Client3"/>
  </iprules>
</consumeridrules>
```

SOAP の障害データの設定

SOAP の障害が検出された場合, SOAP の障害データに SOAP ペイロードを含めることができます。SOAP ペイロードは，SOAP の障害が発生した場合のみにキャプチャされます。

Diagnostics UI では，SOAP 障害インスタンス・ツリー（呼び出しプロファイル）の一部としてペイロード情報を表示できます。

ペイロードにはクレジット・カード番号などの機密情報が含まれる可能性があるため, SOAP の失敗でのペイロードのキャプチャは標準設定では無効になっています。SOAP の障害でのペイロードのキャプチャを有効にするには，.NET プローブ・システムの **probe_config.xml** ファイルに **<soaprequestforsoapfault enabled="true"/>** を設定します。

<soaprequestforsoapfault> 要素の **maxsize** 属性を使用して，ペイロード・サイズの制限を定義することもできます。たとえば，次のエントリは，SOAP のペイロードの長さを標準設定の 5000 から 10000 に増やしています。

```
<soaprequestforsoapfault enabled="true" maxsize="10000"/>
```

< soapcapture >要素は< soaprequestforsoapfault >要素をオーバーライドします。したがって, < soapcapture >が無効な場合は, < soaprequestforsoapfault >が true に設定されていても, < soaprequestforsoapfault >は無効になります。また，< soapcapture >の maxsize の設定値は，< soaprequestforsoapfault >の maxsize をオーバーライドします。したがって，< soapcapture >の maxsize が 5000 に設定されていて，< soaprequestforsoapfault >の maxsize が 10000 に設定されている場合，ペイロード・サイズの最大値は 5000. になります。

その他のプローブ・メトリックスの収集またはプローブ・メトリックスの変更

＜プローブのインストール・ディレクトリ＞ ¥etc¥probe_config.xml ファイルの <metrics> および <metric> 要素を使用すると，perfmon カウンタに基づいて追加のプローブ・メトリックスを収集するように .NET Agent を設定できます。詳細については，546 ページ「<metric> 要素」および 546 ページ「<metric> 要素」を参照してください。

＜ metric ＞要素を使用して プローブ・メトリックスを変更することもできます。ただし，次に示す特殊な場合に注意する必要があります。

- メトリックス・カテゴリ間でメトリックスを移動する場合は，メトリックスのグループ属性，およびメトリックスの名前属性を変更する必要があります。これは，既存の測定値名が Diagnostics Mediator の古いグループに登録されていて，この割り当てを変更できないためです。
- 既存のプローブ測定値を事前に定義する場合は，測定値に別の perfmon カウンタを割り当てないで，まったく新しい測定値エントリを作成することをお勧めします。このようにすると，異なるデータが集計されることがなくなります。

パフォーマンス・カウンタのセキュリティ

.NET Agent ではパフォーマンス・カウンタを使用してプローブ・メトリックスを収集します。このためには，.NET Agent で監視されるアプリケーション・プロセスに，パフォーマンス・カウンタへのアクセス権を設定する必要があります。各プロセスはユーザ・アカウントとして実行されるため，このユーザ・アカウントにパフォーマンス・カウンタへのアクセス権を設定する必要があります。そのための最も簡単な方法は，プロセスを実行するユーザ・アカウントを **Performance Monitor Users** グループに追加することです。

ただし，Microsoft では Windows Vista SP2，Windows Server 2008 SP2，Windows 7，および Windows Server 2008 R2 に**仮想アカウント**の概念が導入されています（詳細については，http://technet.microsoft.com/en-us/library/dd548356(WS.10).aspx（英語サイト）を参照してください）。これらのオペレーティング・システムは IIS 内で仮想アカウントの概念を使用しています。標準設定では，IIS のアプリケーション・プールは **ApplicationPoolIdentity** として実行されます。このユーザ・アカウントは仮想アカウントであるため，Performance Monitors Users グループにユーザ・アカウントを追加するには，特殊な手順が必要です。

Windows 2008 R2 および Windows 7 では，次の手順を実行します。

1 **[サーバ マネージャ]** ツールを起動します。このツールはさまざまな方法で起動できますが，［管理ツール］を使用することもできます。

2 左側の表示枠の［構成］で，［**ローカル ユーザとグループ**］を検索します。

3 **+** をクリックして展開します。

4 ［グループ］をダブルクリックします。

5 ［**Performance Monitor Users**］グループをダブルクリックします。

6 ［**追加**］ボタンをクリックします。

7 ［**場所**］ボタンをクリックします。

8 ローカル・コンピュータを選択します。

9 ［**OK**］ボタンをクリックします。

10 オブジェクト・タイプに**ビルトイン セキュリティ プリンシパル**が含まれていることを確認します。

11 テキスト・ボックスに **IIS APPPOOL¥ ＜アプリケーション・プールの名前＞**と入力します（IIS APPPOOL¥My WebService App Pool など。My WebService App Pool はアプリケーション・プールの名前）。

12 ［**OK**］ボタンをクリックします。

Windows 2008 SP2 および Windows Vista SP2 では，次の手順を実行します。

1 コマンド・プロンプト・ウィンドウを開きます。

2 **net localgroup "Performance Monitor Users" "IIS APPPOOL¥ ＜アプリケーション・プールの名前＞ /ADD** と入力します（＜アプリケーション・プールの名前＞は，アプリケーション・プールの名前）。

3 この操作に成功すると，「**コマンドは正常に完了しました**」と表示されます。

第 V 部

プロキシおよびファイアウォールを介した通信の設定

本項の内容

- HTTP プロキシ用の Diagnostics サーバおよび Agent の設定
- ファイアウォール環境で動作するための Diagnostics の設定

15

HTTP プロキシ用の Diagnostics サーバおよび Agent の設定

Diagnostics コンポーネント間で HTTP プロキシ通信を有効にするための設定手順について説明します。これらの設定手順は，Diagnostics に精通している熟練管理者を対象にしています。Diagnostics コンポーネントの構成設定を変更する際は十分に注意してください。

本章の内容

Diagnostics サーバでの HTTP プロキシ通信の有効化

次の項では，Diagnostics コマンダ・サーバと Diagnostic メディエータ・サーバを設定して，HTTP プロキシを通じて相互通信を行う手順について説明します。

Diagnostics サーバに HTTP プロキシ通信を設定するには，次の手順を実行します。

1 **＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/server.properties** で次のプロパティを設定します。

- **proxy.host** をプロキシ・サーバのホスト名に設定します。
- **proxy.port** をプロキシ・サーバのポートに設定します。
- **proxy.protocol** をプロキシ・サーバ（HTTP）で使用するプロトコルに設定します。
- **proxy.user** をプロキシ・サーバの認証に使うユーザに設定します。
- **proxy.password** をプロキシ・サーバの認証に使うパスワードに設定します。
- プロキシで実行する Diagnostics コマンダ・サーバで，ホスト名がローカルホストではなく実際のホスト名になるように **commander.url** を設定します。

2 Diagnostics サーバを再起動します。詳細については，68 ページ「Diagnostics サーバの起動および停止」を参照してください。

Java Agent での HTTP プロキシ通信の有効化

次の項では，HTTP プロキシを通じて Diagnostics コマンダ・サーバと通信するように Java Agent を設定する手順について説明します。または，Java Agent のインストールおよびセットアップ・プログラムで［Use Proxy Server］チェック・ボックスを選択することで，プロキシ通信を設定することもできます。

Java Agent に HTTP プロキシ通信を設定するには，次の手順を実行します。

1 **＜プローブのインストール・ディレクトリ＞ /etc/dispatcher.properties** で次のプロパティを設定します。

- **proxy.enabled** を true に設定して Java Agent のプロキシ通信を有効にし，次のオプションを入力します。
- **proxy.host** をプロキシ・サーバのホスト名に設定します。
- **proxy.port** をプロキシ・サーバのポートに設定します。
- **proxy.protocol** をプロキシ・サーバ（HTTP）で使用するプロトコルに設定します。
- **proxy.user** をプロキシ・サーバの認証に使うユーザに設定します。
- **proxy.password** をプロキシ･サーバの認証に使うパスワードに設定します。

2 インストルメント対象のアプリケーション VM を再起動します。

.NET Agent での HTTP プロキシ通信の有効化

次の項では，HTTP プロキシを通じて Diagnostics コマンダ・サーバと通信するように .NET Agent を設定する手順について説明します。

.NET Agent に HTTP プロキシ通信を設定するには，次の手順を実行します。

1 **＜プローブのインストール・ディレクトリ＞ /etc/probe_config.xml** ファイルの次の **proxy** プロパティを設定して，Diagnostics サーバ・ホストを指定します。

- **uri** を Diagnostics サーバのホストに設定します。
- **proxy** をプロキシ URL に設定します。
- **proxy.user** をプロキシ・サーバの認証に使うユーザに設定します。
- **proxy.password** をプロキシ・サーバの認証に使うパスワードに設定します。

次の例は，これが **probe_config.xml** ファイルでどのように表示されるのかを示します。

```
<diagnosticsserver url="http://<diagserver_host_name>:2006/registrar/"
proxy="http://proxy:8080" proxyuser= "<username>" proxypassword="<password>"/>
```

2 **＜プローブのインストール・ディレクトリ＞ /etc/metrcis.config** ファイルの次の **proxy** プロパティを設定して，システム測定値でプロキシを使うように指定します。

- **proxy.uri** をプロキシ URL に設定します。
- **proxy.user** をプロキシ・サーバの認証に使うユーザに設定します。
- **proxy.password** をプロキシ・サーバの認証に使うパスワードに設定します。

3 インストルメント対象のアプリケーション・プロセスを再起動します。

16

ファイアウォール環境で動作するための Diagnostics の設定

ファイアウォールがある環境で HP Diagnostics を正常に動作させるための基本設定について説明します。ファイアウォールによってプローブと Diagnostics コンポーネント，あるいは LoadRunner，Performance Center または Business Service Management のコンポーネントが分離される場合は，追加の設定が必要になります。詳細については，LoadRunner，Performance Center，および Business Service Management のドキュメントを参照してください。

本章の内容

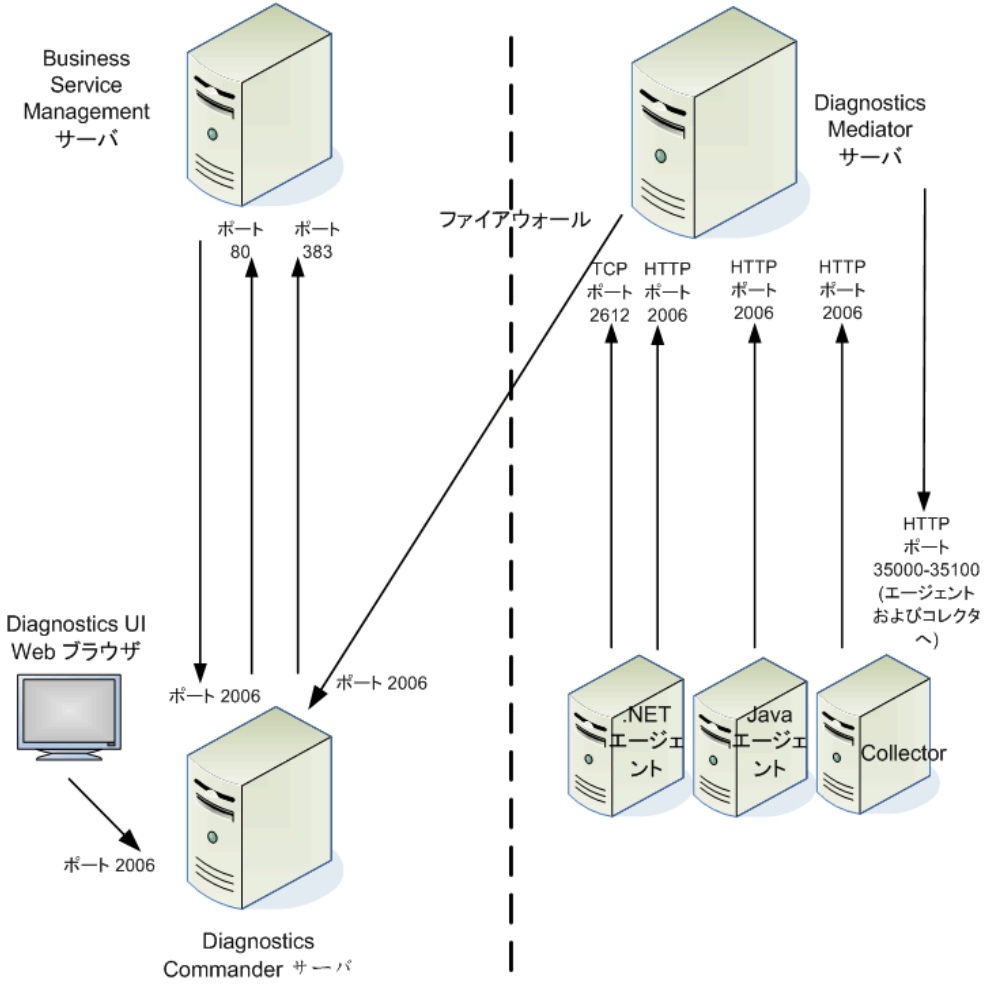
注： これらの設定手順は，HP Diagnostics に関する深い知識を有する熟練ユーザだけが使用してください。Diagnostics コンポーネントの構成設定を変更する際は十分に注意してください。

ファイアウォールへの Diagnostics の設定について

ファイアウォールへの Diagnostics の設定は，どの HP ソフトウェア製品が Diagnostics 統合の一部かによって異なります。

Business Service Management

下の図は，ファイアウォールによってプローブがほかの Diagnostics および Business Service Management コンポーネントから隔離される一般的な Diagnostics デプロイメントを示します。

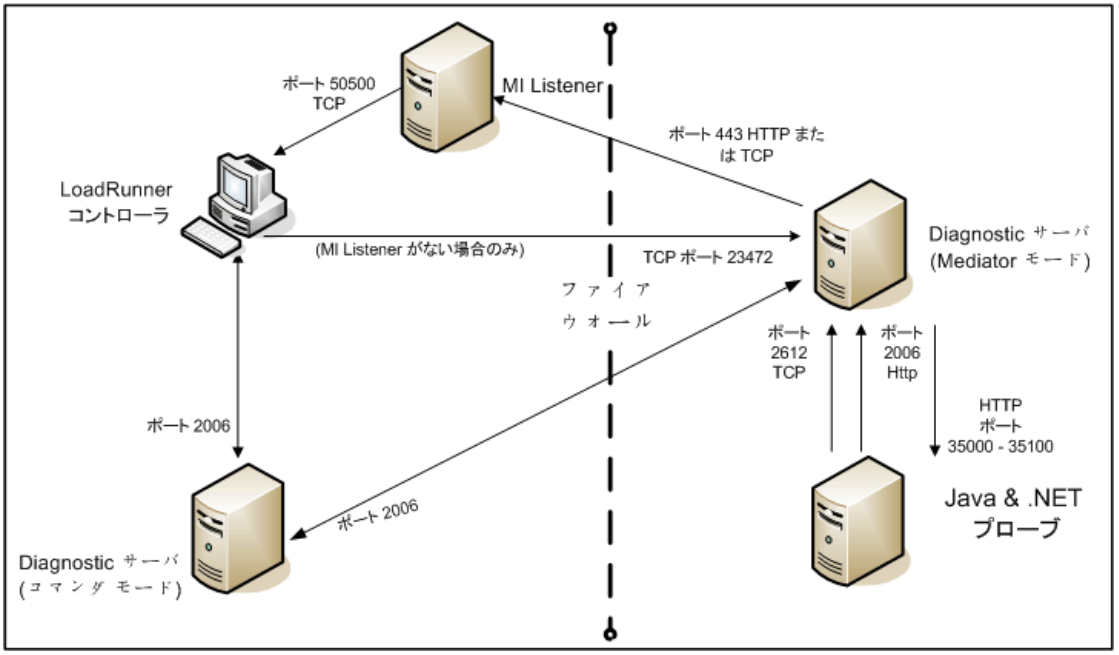


Diagnostics コンポーネントと Business Service Management 間で通信できるようにファイアウォールを設定するには，ポートを開いて次の要求を許可します。

- Business Service Management サーバから，ポート 2006 の Diagnostics コマンダ・サーバへの HTTP 要求。
- Diagnostics コマンダ・サーバから，ポート 80 の Business Service Management サーバへの HTTP 要求。Diagnostics コマンダ・サーバから，ポート 383 の Business Service Management への HI アップデート。
- Diagnostics UI の Web ブラウザ・クライアント・マシンから，ポート 2006 の Diagnostics コマンダ・サーバへの HTTP 要求。Diagnostics UI からは Java または .NET Profiler へもアクセスできます。
- Diagnostics メディエータ・サーバから，ポート 2006 の Diagnostics コマンダ・サーバへの HTTP 要求。
- Diagnostics メディエータ・サーバからプローブのポート 35000 ～ 35100 への HTTP 要求。通信を許可する実際のポートは，プローブを設定したときに有効にしたポート番号とインストルメント対象の VM の数によって異なります。プローブのポート範囲の設定については，231 ページ「アプリケーション・サーバの複数の Java プロセスの監視設定」を参照してください。
- .NET Agent から，ポート 2612 の Diagnostics メディエータ・サーバへの TCP 要求。
- Java Agent，Collector，.NET Metrics Collector から，ポート 2006 の Diagnostics メディエータ・サーバへの HTTP 要求。

LoadRunner と Performance Center

下の図は，ファイアウォールによってプローブがほかの Diagnostics および LoadRunner コンポーネントから隔離される一般的な Diagnostics トポロジを示します。



注 : この図では, 例として LoadRunner が使われています。Performance Center にも同様に考えることができます。

ファイアウォールを設定して，Diagnostics コンポーネント間で相互通信できるようにする必要があります。

Diagnostics コンポーネント間で通信できるようにファイアウォールを設定するには，ポートを開いて次の要求を許可します。

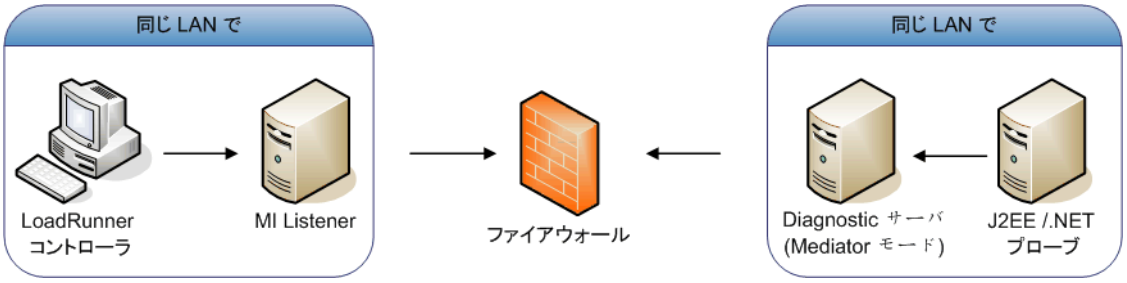
- Diagnostics メディエータ・サーバから，ポート 2006 の Diagnostics コマンダ・サーバへの HTTP 要求。
- プローブから，ポート 2612 の Diagnostics メディエータ・サーバへの TCP 要求。
- プローブから，ポート 2006 の Diagnostics メディエータ・サーバへの HTTP 要求。

注 : 上記のトポロジのほかに，LoadRunner Analysis Tool を使ってオフラインで J2EE の結果を表示している場合，「ファイアウォールを通じたオフライン分析ファイルの照合」の説明に従って Controller と Diagnostics サーバ（Mediator モード）を正しく設定し，オフラインでのファイル取得を可能にする必要があります。

ファイアウォールを通じたオフライン分析ファイルの照合

LoadRunner / Performance Center の負荷テストの間，報告を行うプローブのある Diagnostics サーバは，ホスト・マシンでオフライン分析ファイルを作成します。オフライン分析ファイルは，負荷テストの結果を照合する際，LoadRunner / Performance Center によって取得されます。

LoadRunner / Performance Center Controller と負荷テストに関係のある Diagnostics サーバの間にファイアウォールがある場合，Controller と Diagnostics サーバを設定し，MI Listener ユーティリティを使ってオフライン分析ファイルの転送を有効にする必要があります。MI Listener ユーティリティは，LoadRunner / Performance Center に付属し，下図のようにファイアウォール内部のマシンにインストールする必要があります。



Controller を設定して，ファイアウォールの背後にある Diagnostics サーバにアクセスするには，次の項を参照してください。

- MI Listener のインストールと設定
- ファイアウォールと連携するための Diagnostic メディエータ・サーバ の設定
- Diagnostics ファイアウォールと連携するための LoadRunner および Performance Center の設定

MI Listener のインストールと設定

MI Listener コンポーネントは，ファイアウォールの外にある Load Generator をサポートするのに使われるものと同じコンポーネントです。MI Listener for LoadRunner の設定方法の詳細については，『HP LoadRunner Controller ユーザーズ・ガイド』を参照してください。MI Listener for Performance Center の設定方法の詳細については，『HP Performance Center 管理者ガイド』を参照してください。

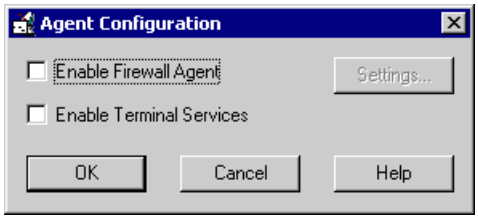
ファイアウォールと連携するための Diagnostic メディエータ・サーバ の設定

Diagnostic メディエータ・サーバをファイアウォールを通じて機能するように設定するには，次の追加手順を行う必要があります。Diagnostic メディエータ・サーバをインストールして設定していない場合は，次の手順を行う前にインストールと設定を行ってください。Diagnostic メディエータ・サーバのインストールについては，第 2 章，「Diagnostics サーバのインストール」を参照してください。

Windows マシンで，ファイアウォールに Diagnostic メディエータ・サーバを設定するには，次の手順を実行します。

1 **＜ Diagnostics サーバのインストール・ディレクトリ＞ /nanny/windows/bin/AgentsConfig.exe** を実行して，Agent Configuration を起動します。

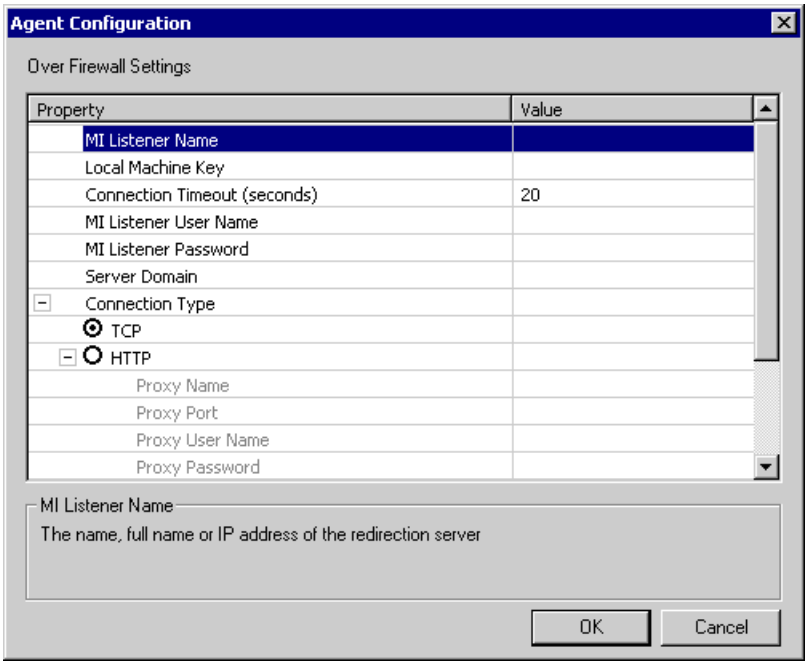
［Agent Configuration］ダイアログ・ボックスが開きます。



2 ［**Enable Firewall Agent**］を選択します。［**Settings**］ボタンが有効になります。

3 ［**Settings**］をクリックします。Agent Configuration プロセスは，［Agent Configuration Settings］ダイアログ・ボックスを開きます。

4 MI Listener 名プロパティの［Value］列に，MI Listener がインストールされているマシンのホスト名または IP アドレスを入力します。



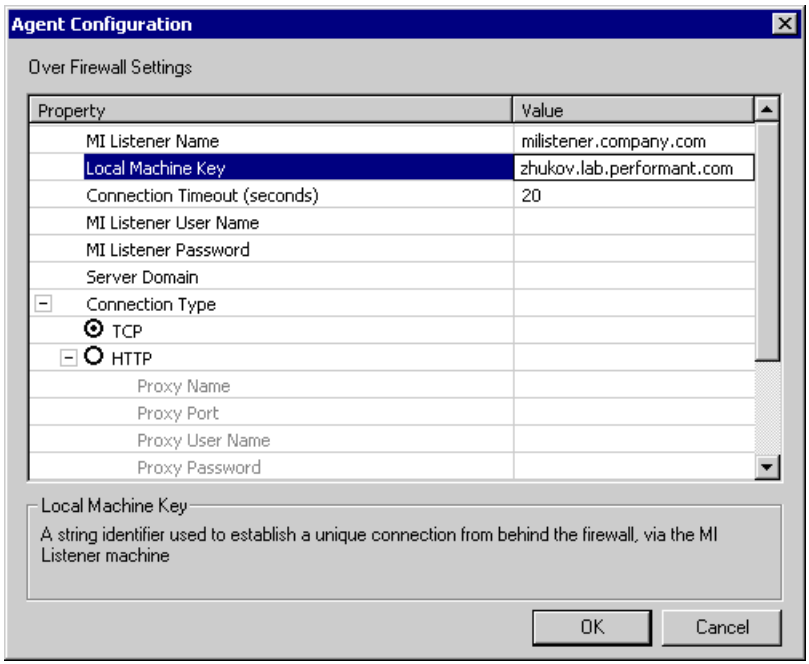
5［**Local Machine Key**］プロパティで，Diagnostic メディエータ・サーバのホストのマシン名を入力します。

重要 :

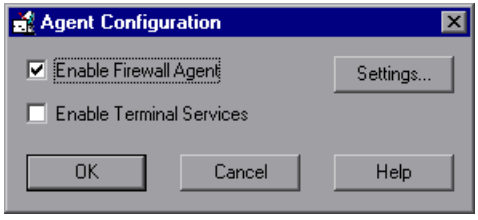
- Diagnostics コンポーネントのホスト名を入力する際, 完全修飾ホスト名, つまりマシン名とドメイン名を必ず使用してください。
- Diagnostics UI のシステムの状況ビューを使って，Diagnostic メディエータ・サーバのホストのマシン名を特定します。システムのビューの詳細については, 付録 D,「管理者用のシステム・ビューの使用」を参照してください。

---

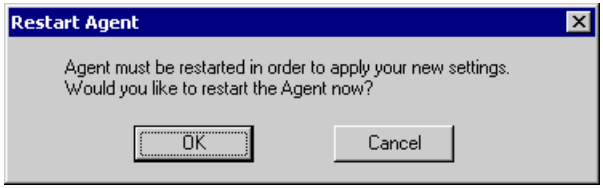
6 ［**OK**］をクリックして，ダイアログ・ボックスを閉じます。



7 もう一度［**OK**］をクリックして，［Agent Configuration］ダイアログ・ボックスを閉じます。



8 ［Restart Agent］ダイアログ・ボックスが開きます。［**OK**］をクリックして Agent を再起動します。



UNIX/Linux マシンで，ファイアウォールに Diagnostic メディエータ・サーバを設定するには，次の手順を実行します。

1 **＜Diagnostics サーバのインストール・ディレクトリ＞/nanny/＜プラットフォーム＞ /dat/br_lnch_server.cfg** ファイルを変更します。

FireWallServiceActive プロパティの値を 1 に変更します。

2 次のコマンドを実行して，Agent Configuration ユーティリティを起動します。

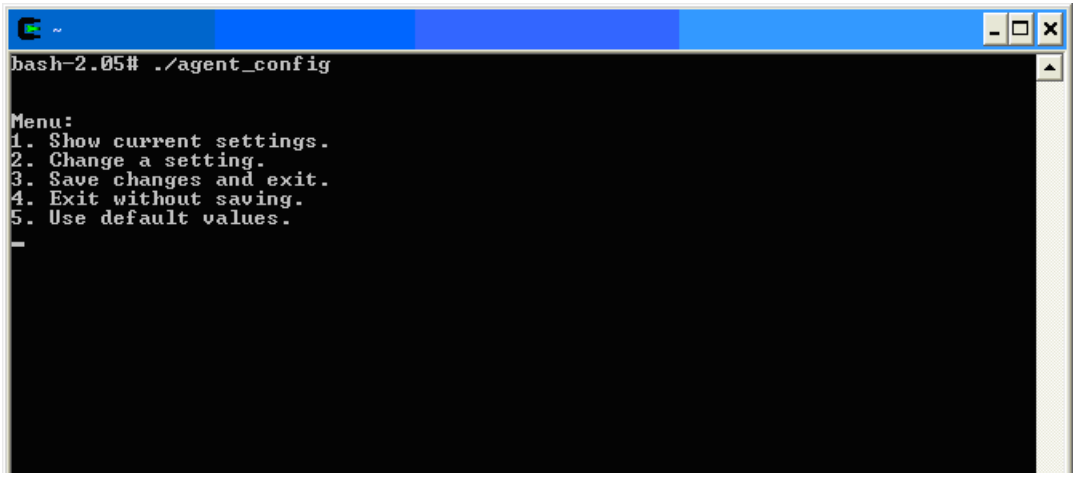
Solaris および Linux の場合（＜プラットフォーム＞ は solaris または linux になります）：

```
export LD_LIBRARY_PATH=.
export M_LROOT= ＜ Diagnostics サーバのインストール・ディレクトリ＞ /nanny/
＜プラットフォーム＞
cd $M_LROOT/bin
./agent_config
```

HP-UX の場合：

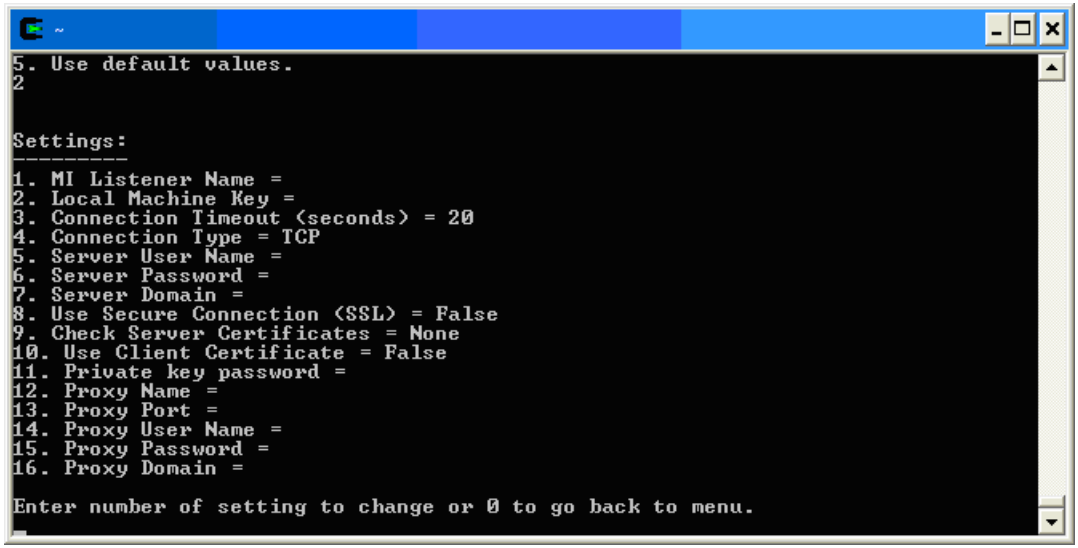
```
export SHLIB_PATH=.
export M_LROOT= ＜ Diagnostics サーバのインストール・ディレクトリ＞ /nanny/
hpux
cd $M_LROOT/bin
./agent_config
```

3 ［Agent Configuration Utility］ウィンドウで，**2** を押して［**Change a setting**］を選択します。



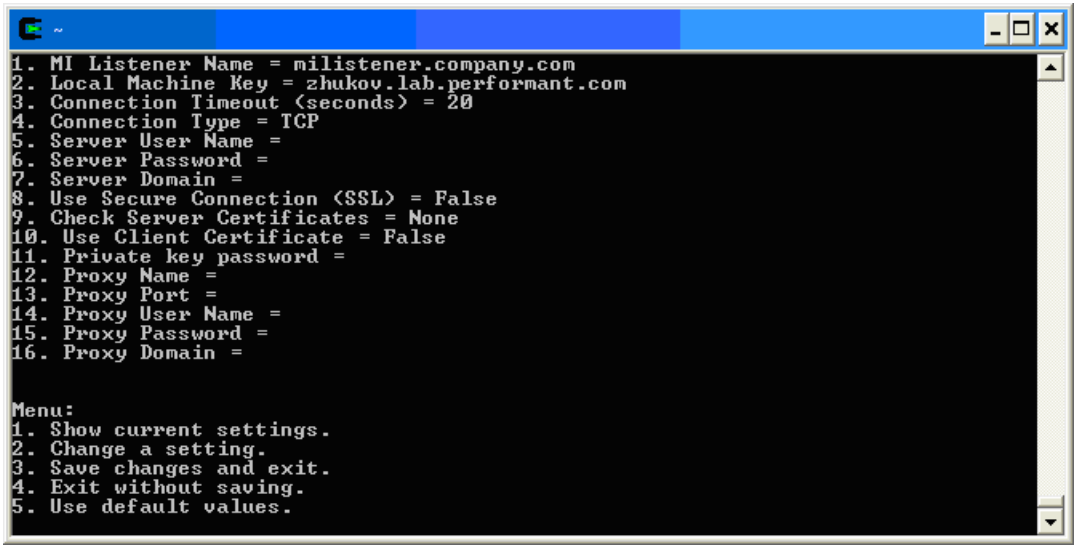
4 設定リストが表示されます。

1 を押して［**MI Listener Name**］を選択し，MI Listener ホストのマシン名と IP アドレスを入力します。



5 **2** を押して［**Change a setting**］を選択し，Diagnostic メディエータ・サーバのホストのマシン名を入力します。

Diagnostics UI のシステムの状況ビューを使って，Diagnostic メディエータ・サーバのホストのマシン名を特定します。システムのビューの詳細については，付録 D,「管理者用のシステム・ビューの使用」を参照してください。



6 **3** を押して［**Save changes and exit**］を選択し，更新を完了します。

7 Diagnostic メディエータ・サーバを再起動します。

./m_daemon_setup -remove（これでサーバと MI Agent が停止します）

./m_daemon_setup -install（これでサーバと MI Agent が起動します）

Linux でエラーが発生した場合は，libstdc++.so.5 共有ライブラリのインストールが必要になる可能性があります。

Diagnostics ファイアウォールと連携するための LoadRunner および Performance Center の設定

MI Listener をインストールして Mediator マシンの設定が完了したら，LoadRunner / Performance Center で Diagnostics Configuration を更新して，ファイアウォールの外部の Mediator からオフライン・データを転送しているときに，MI Listener を使うようにアプリケーションに指示する必要があります。

Performance Center の場合：

ファイアウォール経由でアプリケーションの診断データを収集するように設定された MI Listener マシンの IP アドレスを指定したことを確認します。詳細については，『HP Performance Center 管理者ガイド』の MI Listerner に関するセクションを参照してください。Performance Center で Diagnostics が有効になっていることも確認します。詳細については，『Performance Center ユーザーズ・ガイド』の HP Diagnostics に関するセクションを参照してください。

LoadRunner の場合：

LoadRunner で Diagnostics が有効になっていることを確認します。Diagnostics と連携して機能するように負荷テスト・シナリオを設定するときは，ファイアウォール経由で動作するオプションを選択し，関連する MI Listener サーバの名前を指定します。詳細については，『HP LoadRunner Controller ユーザーズ・ガイド』の HP Diagnostics に関するセクションを参照してください。

第 VI 部

Diagnostics メトリックス・コレクタの設定

本項の内容

- .NET System Metrics Agent - システム・メトリックスのキャプチャ
- Java Agent メトリックス・コレクタ
- Java Agent ー システム・メトリックス・キャプチャ
- Java Agent - JMX メトリックス・キャプチャ

17

.NET System Metrics Agent - システム・メトリックスのキャプチャ

システム・メトリックス・キャプチャの概要，および .NET Agent と一緒にインストールされたシステム測定値コレクタの設定方法について説明します。

本章の内容

.NET System Metrics Agent について

システム・メトリックス・コレクタは .NET Agent と一緒にインストールされ，Windows サービス（HP Diagnostics Metrics Agent）として実行されます。.NET System Metrics Agent は CPU の利用状況やメモリの利用状況といったシステム・レベルのメトリックスを Agent のホストから収集します。この Agent は設定可能なため，収集するメトリックス，および測定値の収集方法や公開方法を制御できます。

ホストで起動しているプローブのインスタンスの数とは無関係に，.NET System Metrics Agent のインスタンスは，特定のホストで 1 つだけ実行されます。

注 : .NET Agent に追加のプローブ・メトリックス・キャプチャ機能（ここで説明するシステム・メトリックス・キャプチャ以外の機能）を設定するには，622 ページ「その他のプローブ・メトリックスの収集またはプローブ・メトリックスの変更」を参照してください。

標準設定でキャプチャされるシステム・メトリックス

次に，サポートされているすべてのプラットフォーム（z/OS を除く）で，.NET System Metrics Agent が標準設定で収集するシステム測定値を示します。

- CPU
- MemoryUsage
- VirtualMemoryUsage
- ContextSwitchesPerSec
- DiskBytesPerSec
- DiskIOPerSec
- NetworkBytesPerSec
- NetworkIOPerSec
- PageInsPerSec
- PageOutsPerSec

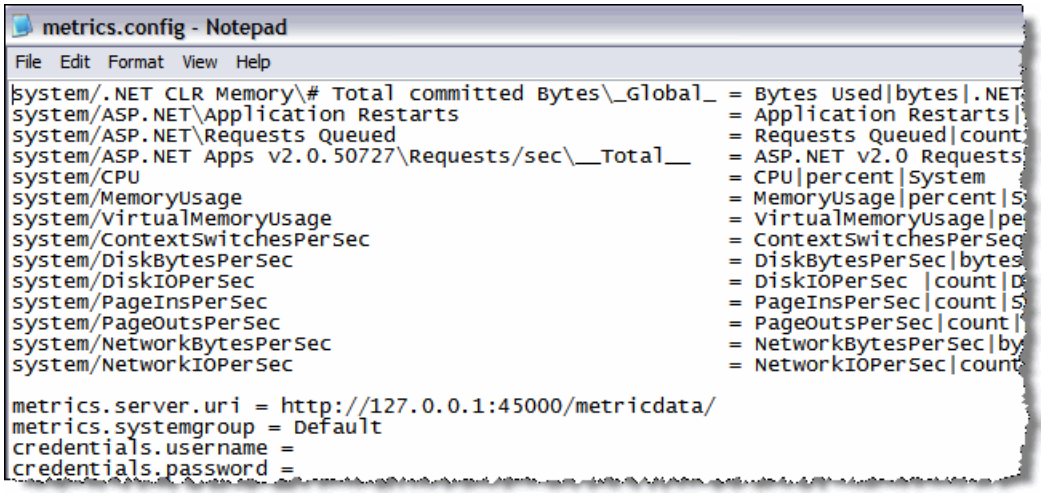
上記の標準設定システム・メトリックスのほかに，.NET Agent システムでは標準設定で次のシステム・メトリックスもキャプチャされます （これらのエントリのレイアウトについては，650 ページ「システム / メトリックス・コレクタのエントリについて」を参照してください)。

```
.NET CLR Memory¥# Total committed Bytes¥_Global_
ASP.NET¥Application Restarts
ASP.NET¥Requests Queued
ASP.NET Apps v2.0.50727¥Requests/sec¥__Total__
```

.NET System Metrics Agent で収集される標準設定のシステム・メトリックスを制御し，.NET System Metrics Agent でカスタムのシステム・メトリックスをキャプチャできます。

.NET システム・メトリックスのキャプチャの設定

.NET System Metrics Agent の設定ファイルは，**<プローブのインストール・ディレクトリ> /etc/metrics.config** ファイルです。metrics.config ファイルを変更すると，.NET Agent によって変更が動的に処理されます。



注 : Java Agent では別の metrics.config ファイルがあります（第 18 章，「Java Agent メトリックス・コレクタ」を参照）。

システム / メトリックス・コレクタのエントリについて

metrics.config ファイル内のメトリックス・コレクタのエントリは，特定の測定値を収集するように .NET System Metrics Agent に指示します。**system/** で開始するエントリは，Windows パフォーマンス・モニタ・カウンタとして処理されます。

これらのシステム・メトリックス・コレクタのエントリでは，次のレイアウトを使用します。

```
system/ ＜カウンタ名＞ ¥ ＜パフォーマンス・オブジェクト＞ ¥ ＜インスタンス＞ ¥
＜リモート・マシン＞ = ＜測定値 ID ＞ | ＜測定値の単位＞ | ＜カテゴリ ID ＞
```

＜インスタンス＞フィールドと＜リモート・マシン＞フィールドは省略可能です。それ以外のすべてのフィールドは必須です。

パラメータ

- **カウンタ名** : Windows パフォーマンス・モニタ・カウンタを指定します。Windows パフォーマンス・モニタ UI のカウンタ，パフォーマンス・オブジェクト，およびインスタンスの識別方法の詳細については，652 ページ「Windows パフォーマンス・モニタを使用したシステム・メトリックスの追加」を参照してください。
- **パフォーマンス・オブジェクト** : カウンタ名に関連付けられた，Windows パフォーマンス・モニタのパフォーマンス・オブジェクトを指定します。
- **インスタンス** : カウンタの Windows パフォーマンス・モニタ・インスタンスを指定します。ワイルドカード（*****）を使用すると，すべてのインスタンスが必要であることを指定できます。すべてのインスタンスを具体的に列挙する場合は，列挙インデックス番号の前にハッシュ記号（#1）を付加します。列挙インデックスの値は正数である必要があります。
- **リモート・マシン** : Windows のパフォーマンス・モニタ・カウンタが .NET System Metrics Agent が実行されているマシンと異なる（リモートの）マシンで動作している場合のみ必須です。この設定が機能するための最小要件は，.NET System Metrics Agent が実行されているマシン上のネットワーク・サービス・ユーザに，リモート・マシンから Windows パフォーマンス・モニタ・カウンタを読み取るための権限があることです。

- **＜測定値 ID ＞**: Diagnostics UI の測定値を表す名前を指定します。metric_id は，**metrics.config** ファイルで一意である必要があります。metric_id の値が標準設定の測定値と同じ場合，Diagnostics は，エントリの metric_id を UI の測定値を参照するのに使われる標準の名前に置き換えます。metric_id の値が標準設定のメトリックスと異なる場合，metric_id は，エントリに示されるように，UI に測定値の名前として使用されます。
- **＜測定値の単位＞** は，測定値が報告される測定単位を示します。これは必須パラメータで，次のいずれかの測定値を含む必要があります。
  - マイクロ秒，ミリ秒，秒，分，時，日
  - バイト，キロバイト，メガバイト，ギガバイト
  - パーセント，割合
  - 数
  - load
- **＜カテゴリ ID ＞**: Diagnostics UI の［詳細］表示枠内で，一連のメトリックスを同じ見出しでグループ化します。このパラメータは，Diagnostics ビューに表示されるデータに影響しません。

＜インスタンス＞を含まない例

```
system/ASP.NET¥Requests Queued = Requests Queued|count|ASP
```

＜インスタンス＞を含む例

```
system/Processor¥% Processor Time¥_Total = CPU|percent|System
```

正数の＜インスタンス＞を含む例

```
system/Processor¥% Processor Time¥#1 = CPU 1|percent|System
```

＜インスタンス＞を含まない，＜リモート・マシン＞で実行する例

```
system/ASP.NET¥Requests Queued¥¥IISAQUAH = Requests
Queued(IISAQUAH)|count|ASP
```

Windows パフォーマンス・モニタを使用したシステム・メトリックスの追加

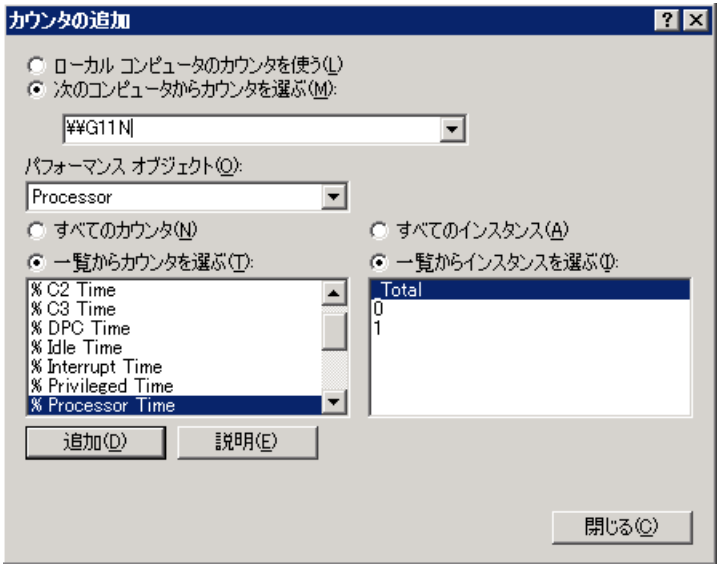
metrics.config ファイルにシステム測定値カウンタを追加するには，最初に Windows パフォーマンス・モニタ（Perfmon）を使用して定義を検索する必要があります。次の例ではバージョン 5.x の Perfmon を使用します。バージョン 6.x は似ていますが，UI は少し異なります。

Perfmon にカウンタを追加するには，次の手順を実行します。

1 Windows パフォーマンス・モニタを起動します。たとえば，［**スタート**］＞［**コントロール パネル**］＞［**管理ツール**］＞［**パフォーマンス**］を選択します。

2 Perfmon の［Performance］ダイアログ・ボックスが開き，システム・モニタ・グラフと，グラフの下に現在のカウンタの表が表示されます。システム・モニタ・グラフを右クリックして，ポップアップ・メニューから［**カウンタの追加**］を選択します。

［カウンタの追加］ダイアログ・ボックスが表示されます。



3 ［**次のコンピュータからカウンタを選ぶ**］エントリを選択して，ドロップ・ダウン・リストでホスト・コンピュータが選択されていることを確認してください。

4 ［パフォーマンス オブジェクト］リストで，カウンタが属しているオブジェクトを選択します。

5 ［**一覧からカウンタを選ぶ**］を選択し，インスタンス・リストからインスタンスを選択します。

6 ［**追加**］ボタンをクリックして，カウンタを追加します。次の手順では，650 ページ「システム / メトリックス・コレクタのエントリについて」に記載されたシステム / メトリックス・エントリを使用して，カウンタのエントリ作成方法を示します。

Perfmon カウンタのメトリックスを収集するには，次の手順を実行します。

1 .NET Agent システムの**＜プローブのインストール・ディレクトリ＞ /etc/ metrics.config** ファイルを開きます。

2 650 ページ「システム / メトリックス・コレクタのエントリについて」に記載されたレイアウトを使用して，カウンタの **system/** メトリックス・エントリを作成します。

このエントリはファイル内のどこにでも追加できますが，ベスト・プラクティスは，これらの種類のエントリからなる既存のコレクションの一番下に追加することです。上記のスクリーン・ショットの例では，次のようになります。

- 選択したホスト・コンピュータは ROS59524ART
- 選択したパフォーマンス・オブジェクトは Processor
- 選択したカウンタは % Processor Time
- 選択したインスタンスは _Total

ホスト・コンピュータがローカルな場合，パフォーマンス・モニタ・カウンタに対応する metrics.config ファイル内のエントリは次のようになります。

```
system/Processor¥% Processor Time¥_Total = CPU|percent|System
```

ホスト・コンピュータがリモートな場合，パフォーマンス・モニタ・カウンタに対応する metrics.config ファイル内のエントリは次のようになります。

```
system/Processor¥% Processor Time¥_Total¥ROS59524ART =
CPU(ROS59524ART)|percent|System
```

パフォーマンス・カウンタのセキュリティ

.NET Metrics Agent はパフォーマンス・カウンタを使用してシステム・メトリックスを収集します。Metrics Agent はネットワーク・サービスとして実行されるため，このアカウントを **Performance Monitor Users** グループに追加する必要があります。

追加されたシステム・メトリックスカウンタのトラブルシューティング

指定した新しいカウンタが機能していないように見える場合は, Windows イベント・ビューアを使用して，.NET System Metrics Agent の診断ログを参照して，エラーや警告を確認できます。

次に例を示します。

「**Could not locate Performance Counter with specified category name**」という警告エントリは，通常，カウンタ名の入力ミスの可能性があることを示します。入力ミスが発生するのは，［PerfMon Performance］表示枠から空白が埋め込まれたカウンタ名を読み取る場合などです。PerfMon で使用される標準設定のフォントは等幅フォントでないため，カウンタ名，カテゴリ，およびインスタンスの名前に埋め込まれている空白を確認することは困難です。フォントを等幅のフォント・タイプに変更すると，カウンタ名の正確な形式を判別しやすくなります。

次に例を示します。

「**Instance does not exist in the specified Category**」という警告エントリは，通常，選択したインスタンスがその時点でアクティブでないことを示します。一時的なインスタンスは使用しないことをお勧めします。__Total__ などの永続的なインスタンスが適しています。

.NET Agent metrics.config ファイルの標準設定エントリ

＜プローブのインストール・ディレクトリ＞ /etc/metrics.config ファイルには，インストール時に 3 つのエントリが格納されています。

- PerfMon カウンタの標準設定の **system/** エントリのグループ
- **metrics.server.uri** エントリは，.NET System Metrics Agent のデータ公開方法を指定します。
- 標準設定の **metrics.systemgroup** エントリ

その他の追加エントリは，これらの標準設定エントリの後に追加できます。

metrics.config ファイルのキーワード

＜プローブ・インストール・ディレクトリ＞ /etc/metrics.config ファイルのエントリで使用されるキーワードは，次のとおりです。

- credentials.password
- credentials.username
- default.sampling.rate
- metrics.server.uri
- metrics.systemgroup
- metrics.agent.publish.interval
- metrics.agent.registered_hostname
- proxy.password
- proxy.user
- proxy.uri
- system/

system/ キーワードの使用については，649 ページ「.NET システム・メトリックスのキャプチャの設定」を参照してください。

その他の各キーワードの使用法は，次のセクションに記載されています。

credentials.password	この設定は，probe_config.xml ファイル内にある＜ credentials ＞要素の password 属性の設定と一致する必要があります。詳細については，514 ページ「<credentials> 要素」を参照してください。
credentials.username	この設定は，probe_config.xml ファイル内にある＜ credentials ＞要素の username 属性の設定と一致する必要があります。詳細については，514 ページ「<credentials> 要素」を参照してください。
default.sampling.rate	この設定は，.NET System Metrics Agent が設定済みのシステム・メトリックス・カウンタをサンプリングするレートを定義します。標準設定レートは 5 秒おきです。値は秒，分，時間，または日の値として表されます（nS，nM，nH，または nD など）。次に，レートを 10 秒おきに設定する例を示します。 default.sampling.rate = 10s

metrics.server.uri	この設定はインストール時に自動的に生成されます。.NET System Metrics Agent がシステム測定値カウンタを Diagnostic メディエータ・サーバに公開する場合に使用する URL を定義します。 次に，Diagnostic メディエータ・サーバが my_diag_server マシンで実行されていて，メトリックスを公開するために metricport 値に 2006 を使用する例を示します。 metrics.server.uri = http:// < Diagnostics Server > :2006/metricdata/?sleep=false probe_config.xml 設定内で< mediator >要素の metrichost 属性または metricport 属性に加えた変更は，metrics.server.uri 設定にも同時に反映する必要があります。 ?sleep 設定は，公開されたメトリックスを受信した Diagnostic メディエータ・サーバが即座に応答するのか，または .NET System Metrics Agent への応答を遅らせるのかを制御します。?sleep=false を設定すると即座に応答し，?sleep=true を設定すると，標準設定の 5 秒間だけ応答が遅れます。
metrics.systemgroup	この設定はインストール時に自動的に生成されます。この設定は変更しないでください。
metrics.agent.publish.interval	この設定は，.NET System Metrics Agent が Diagnostic メディエータ・サーバにシステム測定値カウントの現在値を公開する間隔を定義します。標準設定の間隔は 5 秒です。設定値は秒または分の値として表されます（nS または nM. など）。次に，公開間隔を 10 秒に設定する例を示します。 metrics.agent.publish.interval = 10S
metrics.agent.registered_hostname	この設定を使用するタイミングおよび使用方法については，612 ページ「標準設定のプローブホスト・マシン名の変更」を参照してください。
proxy.password	この設定は，probe_config.xml ファイル内にある < diagnosticsserver >要素の proxypassword 属性の設定と一致する必要があります。詳細については，518 ページ「<diagnosticsserver> 要素」を参照してください。第 15 章，「HTTP プロキシ用の Diagnostics サーバおよび Agent の設定」も参照してください。

proxy.user	この設定は，probe_config.xml ファイル内にある＜ diagnosticsserver ＞要素の proxyuser 属性の設定と一致する必要があります。詳細については，518 ページ「<diagnosticsserver> 要素」を参照してください。第 15 章,「HTTP プロキシ用の Diagnostics サーバおよび Agent の設定」も参照してください。
proxy.uri	この設定は，probe_config.xml ファイル内にある＜ diagnosticsserver ＞要素の proxy 属性の設定と一致する必要があります。詳細については，518 ページ「<diagnosticsserver> 要素」を参照してください。第 15 章,「HTTP プロキシ用の Diagnostics サーバおよび Agent の設定」も参照してください。

18

Java Agent メトリックス・コレクタ

このセクションでは，Java Agent メトリックスのキャプチャと，測定値コレクタの設定方法について説明します。

本章の内容

測定値のキャプチャについて

Java Agent では，メトリックス設定ファイル **<プローブのインストール・ディレクトリ> /etc/metrics.config.** のエントリを変更することでメトリックス・コレクタを設定できます。

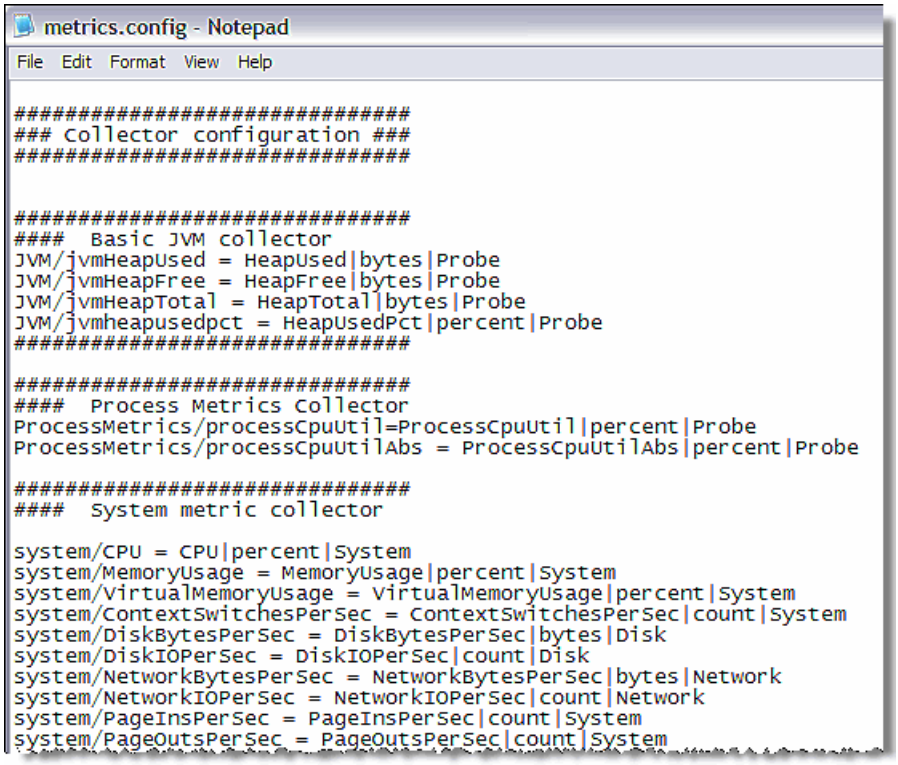
注 : .NET Agent とともに含まれるほかの metrics.config ファイルがあります (647 ページ「.NET System Metrics Agent - システム・メトリックスのキャプチャ」を参照)。

Agent インストールのシステムおよび JMX メトリックス・コレクタは，メトリックス設定ファイルで定義されます。メトリックス設定ファイル **<プローブのインストール・ディレクトリ> /etc/metrics.config** のプロパティとエントリを使って，測定値コレクタを制御できます。

注 : メトリックス設定ファイルを更新すると，測定値コレクタが自動的に再起動して変更を有効にします。

Java Agent が収集しているメトリックス

metrics.config ファイルに，Java Agent が収集しているメトリックスが表示されます。



利用可能な測定値のリスト

Java Agent の metrics.config ファイルには，各 JMX コレクタの利用可能な全メトリックスのリストを 1 つのファイルに書き込む機能があります。metrics.config ファイルの **default.dump.available.metrics** プロパティが true に設定されているとき，プローブは，利用可能なメトリックスのこのリストをプローブのログ・ディレクトリにあるテキスト・ファイルに書き込みます。ファイルの名前は次のとおりです。**＜プローブのインストール・ディレクトリ＞ /log/ ＜ Probe ID ＞ / jmx_metrics ＜コレクタの名前＞ .txt。**追加測定値キャプチャの設定用テンプレートとしてこの情報を使用する方法の詳細と例については，681 ページ「利用可能な JMX または WebSphere PMI メトリックスのリストの取得」を参照してください。

測定値コレクタのエントリについて

測定値コレクタのエントリは，Java Agent 測定値コレクタに特定のメトリックスを収集するよう指示します。エントリの左側にあるパラメータは，プローブがホストまたは JVM からメトリックスを収集する方法を制御し，エントリの右側のパラメータは，収集されたメトリックスを Diagnostics で処理する方法とユーザ・インタフェースで表示する方法を定義します。

エントリの形式は，次のいずれかです。

```
＜コレクタ名＞ / ＜測定値の設定＞ = ＜測定値 ID ＞ | ＜測定値の単位＞ |
  ＜カテゴリ ID ＞
```

または

```
  ＜ Collector 名＞ / ＜測定値設定＞ =
  RATE ＜レートの乗数＞ ( ＜測定値 ID ＞ | ＜測定値の単位＞ | ＜カテゴリ ID ＞ )
```

説明：

- **＜コレクタ名＞** は，Diagnostics 測定値コレクタの名前を示します。コレクタは，**metrics.config** で定義されます。

  システム測定値の場合，このパラメータの値は system になります。JMX 測定値の場合，通常，このパラメータの値は，WebSphere5 といったアプリケーション・サーバのタイプやバージョンの名前で定義されます。

  コレクタ名およびメトリックス名は，Diagnostics UI の［Advanced Query］ページ（http:// ＜ Diagnostics Server ＞ :2006/query）でも確認できます。

- **＜測定値の設定＞** は，アプリケーション・サーバのホスト・システムまたは JVM で監視される測定値を特定します。このパラメータの形式は，システム測定値と JMX 測定値のどちらのエントリを作成しているかによって異なります。システム測定値コレクタの **metric_config** プロパティの形式については，671 ページ「追加カスタム・システム・メトリックスのキャプチャ」を参照してください。JMX 測定値の **metric_config** プロパティの形式については，684 ページ「新しい JMX または WebSphere PMI メトリックス・エントリの作成」を参照してください。

- **RATE(...)** は，サンプリング中に測定値の値が割合（1 秒あたりの単位）に変換されることを示します。

たとえば，**Rate** パラメータを測定値「**起動時からの統計サーブレット要求**」で使用すると，収集された測定値の単位は，「サーブレット要求数」から「1 秒あたりのサーブレット要求数」に変換されます。

Rate を使用しない場合は，先述の最初の例のように括弧をつけません。

注 : このパラメータは，非減少の値を持つ測定値だけで使います。

- **＜ rate_multiplier ＞** は，任意のパラメータで，割合に **＜ rate_multiplier ＞** で掛けて，割合を調整することを示します。

  たとえば，**Rate** パラメータと **rate_multiplier** が，測定値「**合計 GC 時間**」（ms）で使われる場合，収集された測定値の値は「GC の合計時間」から「GC で費やした時間の割合」に変換されます。

- **＜測定値 ID ＞** は，UI の測定値を表す名前を示します。metric_id は，**metrics.config** ファイルで一意である必要があります。metric_id の値が標準設定の測定値と同じ場合，Diagnostics は，エントリの metric_id を UI の測定値を参照するのに使われる標準の名前に置き換えます。metric_id の値が標準設定のメトリックスと異なる場合，metric_id は，エントリに示されるように，UI に測定値の名前として使用されます。
- **＜測定値の単位＞** は，測定値が報告される測定単位を示します。これは必須パラメータで，次のいずれかの測定値を含む必要があります。
  - マイクロ秒，ミリ秒，秒，分，時，日
  - バイト，キロバイト，メガバイト，ギガバイト
  - パーセント，割合
  - 数
  - load

➤ **<カテゴリ ID >**は, Java Diagnostics Profiler の［測定値］タブのサイドバーにあるツリーの同じヘッダに，一連のメトリックスをグループ化します。このパラメータは，Diagnostics ビューの［詳細］表示枠に表示されるデータに影響しません。

注 : 測定値コレクタのエントリを作成した後，円マーク「¥」，スペース，またはコロン「:」の前に，拡張文字「¥」を追加します。これは，ファイルから読み込まれる Java プロパティで必須です。

追加プローブ測定値の収集について

追加メトリックスの情報を収集するには，662 ページ「測定値コレクタのエントリについて」で説明した構文を使用し，**metrics.config** ファイルに適切なメトリックス・コレクタにメトリックスのエントリを追加します。

追加システム測定値のキャプチャの詳細については，671 ページ「追加カスタム・システム・メトリックスのキャプチャ」を参照してください。

追加 JMX 測定値のキャプチャの詳細については，681 ページ「追加カスタム JMX メトリックス」を参照してください。

キャプチャ済みプローブ測定値の変更

metrics.config ファイルで，測定値コレクタの標準設定およびカスタムの両方の測定値エントリを更新できます。

測定値のキャプチャの停止

metrics.config に記載されている測定値の収集から測定値コレクタを停止する場合，測定値のエントリを削除するか，または先頭に「#」を追加して測定値のエントリをコメント行にできます。

システム上の複数の JVM アプリケーション用にカスタマイズされた metrics.config ファイルの使用

複数の JVM があるシステムで実行中の特定の JVM アプリケーションに対して，特定のメトリックスのみの収集またはメトリックス・コレクタのプロパティのカスタマイズを行うと，そのシステムで実行されているほかのインストルメントされた JVM に悪影響を及ぼします。このような場合は，次の手順に従うことで，異なる **metrics.config** 設定ファイルを作成，カスタマイズして，カスタマイズした設定を使用するようにそれらの JVM アプリケーションを設定できます。

注 : カスタマイズした **metrics.config** ファイルが必要な JVM アプリケーションのみを設定する必要があります。それ以外の JVM アプリケーションでは，標準の **metrics.config** 設定を使用できます。

---

1 特殊なカスタマイズが必要な各 JVM アプリケーション用に **etc/metrics.config** ファイルをコピーし，そのファイルに **metrics_ ＜アプリケーション名＞ .config** などの名前を付けます。このファイルは，元の **metrics.config** ファイルと同じ **＜プローブのインストール・ディレクトリ＞ /etc** フォルダに置く必要があります。このファイルを必要に応じてカスタマイズします。

2 作成した各 **metrics_ ＜アプリケーション名＞ .config** ファイル用に **lib/modules.properties** ファイルをコピーし，そのファイルに **modules_ ＜アプリケーション名＞ .properties** などの名前を付けます。このファイルは，元の **modules.properties** ファイルと同じ**＜プローブのインストール・ディレクトリ＞ /lib** フォルダに置く必要があります。

次の例のように，新しい **metrics_ ＜アプリケーション名＞ .config** ファイルを指すようにこの新しいファイルの **metrics.properties** プロパティを変更します。

```
################################################################################
## Metrics capture module
################################################################################
metrics.class.name=com.mercury.diagnostics.capture.metrics.MetricsModule
metrics.class.loader=probeLoader
metrics.properties=metrics_ ＜アプリケーション名＞ .config
```

3 対応する新しい **lib/modules_ ＜アプリケーション名＞ .properties** ファイルを使用するため，カスタマイズされたメトリックス収集が必要な各 JVM 起動スクリプトを更新するには，次を JVM プロパティ定義に追加します。

-Dmodules.properties.file=module_ ＜アプリケーション名＞ .properties

19

Java Agent — システム・メトリックス・キャプチャ

システム・メトリックスのキャプチャ・プロセスと，Java Agent システム測定値コレクタを設定してシステム・メトリックスをキャプチャする手順について説明します。

本章の内容

システム測定値について

システム測定値コレクタは，Java Agent と一緒にインストールされます。システム測定値コレクタでは，CPU の利用状況やメモリの利用状況といったシステム・レベルのメトリックスを Agent のホストから収集します。システム測定値コレクタは設定可能であるため，ユーザは収集するシステム測定値を制御できます。

ホストで起動されたプローブのインスタンスの数とは無関係に，システム測定値コレクタのインスタンスは，特定のホストで 1 つだけ実行されます。プローブのインスタンスが起動すると，測定値プロパティで指定した UDP ポートに接続しようとします。接続が確立すると，システム測定値コレクタのインスタンスが起動します。接続できない場合，システム測定値コレクタのインスタンスが，ホストでほかのプローブのインスタンスによってすでに起動されており，新しいインスタンスを起動できません。

各プローブは，定期的にポートに接続して，システム測定値コレクタが常に実行していることを確認します。システム測定値コレクタを起動したプローブを停止すると，プローブのほかのいずれかのインスタンスがポートが使用可能なことを確認したときに，システム測定値コレクタの新しいインスタンスを起動します。

標準設定でキャプチャされるシステム測定値

次は，サポートされているすべてのプラットフォーム（z/OS を除く）で，測定値コレクタが標準設定で収集するシステム測定値です。

- CPU
- MemoryUsage
- VirtualMemoryUsage
- ContextSwitchesPerSec
- DiskBytesPerSec
- DiskIOPerSec
- NetworkBytesPerSec
- NetworkIOPerSec
- PageInsPerSec
- PageOutsPerSec

システム測定値コントローラで収集する標準設定のシステム測定値を制御したり，ほかのプラットフォーム特有の測定値を追加して，コレクタでそれらの情報も収集するようにできます。詳細については，669 ページ「システム・メトリックス・コレクタの設定」を参照してください。Windows，Solaris，Linux などの特定のプラットフォームの場合，システム測定値コレクタで収集できるカスタム・システム・メトリックスを作成できます。詳細については，671 ページ「追加カスタム・システム・メトリックスのキャプチャ」を参照してください。

z/OS システム測定値については，677 ページ「z/OS システム測定値のキャプチャの有効化」を参照してください。

システム・メトリックス・コレクタの設定

メトリックス設定ファイル **＜プローブのインストール・ディレクトリ＞ /etc/metrics.config** のエントリを変更して，お使いの環境で実行し，関心のあるシステム・メトリックスを収集，報告するシステム測定値キャプチャ・プロセスを設定できます。メトリックス・コレクタの一般的な情報については，第 18 章，「Java Agent メトリックス・コレクタ」を参照してください。メトリックス・コレクタ・エントリと構文については，662 ページ「測定値コレクタのエントリについて」を参照してください。

注 : .NET Agent とともに含まれるほかの metrics.config ファイルがあります（647 ページ「.NET System Metrics Agent - システム・メトリックスのキャプチャ」を参照）。

注 : メトリックス設定ファイルを更新すると，システムのメトリックス・コレクタが自動的に再起動して変更を有効にします。

システム・メトリックス・コレクタ・エントリの例

次の例は，システム測定値に測定値コレクタのエントリを作成する方法を示します。ホスト・プラットフォームに CPU というシステム測定値にエントリを作成する場合，次のように入力します。

```
system/CPU = CPU|percent
```

説明：

- **system** は, 測定値がシステム測定値コレクタによって収集されることを示します。
- 最初の **CPU** は，プラットフォームで **CPU** という測定値が監視されていることを示します。
- 2 つ目の **CPU** は，測定値にラベルを付けるために UI で使われる名前です。
- **percent** は，測定値がホストで測定され，UI に報告される単位を示します。

標準設定のポートの変更

測定値コレクタの標準設定のポートは **35000** です。Agent ホストの設定でほかのポートを使用する必要がある場合，**system.udp.port** プロパティを使って，この値を変更できます。

標準設定のポートを変更するには，次の手順を実行します。

1. **metrics.config** で **system.udp.port** プロパティを見つけます。
2. **system.udp.port** プロパティの値を，システム測定値コレクタで使用するポート番号に変更します。標準設定のポートは **35000** です。

注 : システム測定値コレクタに割り当てられているポートは，Agent の Web サーバのポートとは関係ありません。

システム測定値の収集の無効化

収集または UI に表示されないようにシステム測定値の収集を無効にするには，**system.udp.port** プロパティの値を -1 に設定します。

追加カスタム・システム・メトリックスのキャプチャ

Java Agent システム測定値コレクタを使って，Windows，Solaris，Linux のプラットフォームでカスタム・システム測定値をキャプチャできます。

次のセクションでは，メトリックスをキャプチャしたり，システム測定値コレクタのエントリを更新して，カスタム・メトリックスを監視する手順について説明します。

本項の内容

- Windows ホストでのカスタム・システム測定値のキャプチャ
- Solaris ホストでのカスタム・システム測定値のキャプチャ
- Linux ホストでのカスタム・システム測定値のキャプチャ

Windows ホストでのカスタム・システム測定値のキャプチャ

Windows システム・モニタの機能を使って，カウンタを追加し，システムまたはサービスの特定の状況のパフォーマンスを表すことができます。カウンタは，Windows システム・モニタで追跡，報告され，Java Agent システム測定値コレクタで監視できます。

Windows システム・モニタを使ってカウンタを追加するには，次の手順を実行します。

1 Windows パフォーマンス・モニタを起動します。

   a ［スタート］メニューから［**スタート**］＞［**ファイル名を指定して実行**］を選択します。

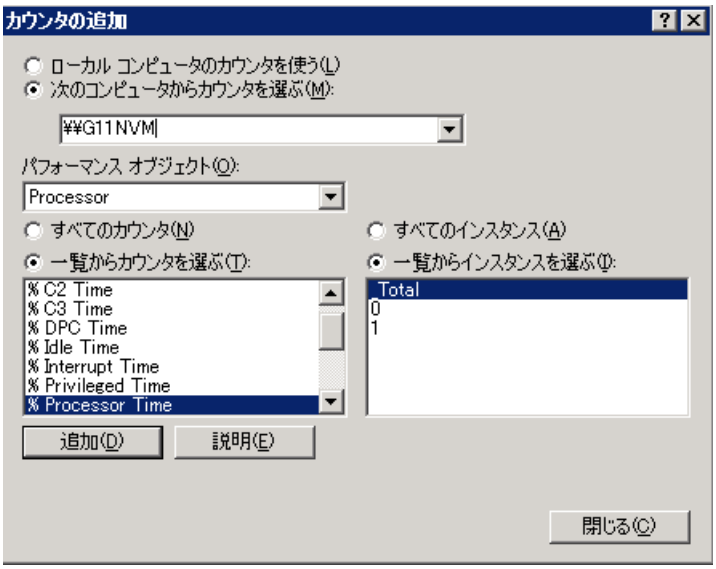
   b ［**ファイル名を指定して実行**］ダイアログ・ボックスの［**名前**］ボックスに，perfmon と入力します。

   ［**Performance**］ダイアログ・ボックスが開き，**システム・モニタ**・グラフと，グラフの下に現在のカウンタの表が表示されます。

2 ［カウンタの追加］ダイアログ・ボックスを表示します。

システム・モニタ・グラフを右クリックして，ポップアップ・メニューから［**カウンタの追加**］を選択します。

Windows に［**カウンタの追加**］ダイアログ・ボックスが表示されます。



3 ［**次のコンピュータからカウンタを選ぶ**］リストでホスト・コンピュータが選択されていることを確認します。

4 ［**パフォーマンス オブジェクト**］リストで，カウンタに属すオブジェクトを選択します。

5 ［**一覧からカウンタを選ぶ**］を選択し，下のカウンタ・リストからカウンタを指定します。

6 ［**一覧からインスタンスを選ぶ**］を選択し，下のインスタンス・リストからインスタンスを指定します。

7 ［**追加**］をクリックします。

システム・モニタにカウンタを追加すると，そのカウンタの測定値を収集するようにシステム測定値コレクタを設定できます。次は，次のテンプレートに基づいて **metrics.config** のエントリを作成する手順です。

```
＜コレクタ名＞/＜測定値設定＞=＜測定値 ID＞|＜測定値の単位＞
```

このテンプレートについては，662 ページ「測定値コレクタのエントリについて」で説明します。

Windows システム・モニタのカウンタの測定値を収集するには，次の手順を実行します。

1 **＜プローブのインストール・ディレクトリ＞ /etc/metrics.config** を開きます。

2 次のテンプレートを使って，エントリの <metric_config> 部分を作成し，カウンタのエントリを入力します。

```
¥＜パフォーマンス・オブジェクト＞（＜インスタンス＞）¥＜カウンタ＞
```

前ページの画面例の場合

- 選択されているパフォーマンス・オブジェクトは %Processor
- 選択されているインスタンスは _Total
- 選択されているカウンタは Processor Time

この例で作成されるエントリの＜測定値設定＞部分は次のようになります。

```
¥Processor(_Total)¥% Processor Time
```

3 次の例のように，残りのシステム測定値エントリ・テンプレートを入力します。

```
system/¥Processor(_Total)¥% Processor Time = ProcessorTime|percent
```

4 最初のエントリをフォーマットします。最初のエントリで，円マーク「¥」，スペース，またはコロン「:」の前に，拡張文字「¥」を追加します。

この手順に従うと，前の手順の最初のエントリは次のようになります。

```
system/¥¥Processor(_Total)¥¥%¥ Processor¥ Time = ProcessorTime|percent
```

これは，システム測定値コレクタで Windows システム・モニタのカウンタの測定値を収集するために，**metrics.config** を正しくフォーマットしたエントリです。

```
system/¥¥¥¥RemoteMachine¥¥Processor(_TOTAL)¥¥%¥ Processor¥ Time=
Processor¥ Time(Remote Machine)|percent
```

注 : リモート・マシンで **perfmon** が正しくセットアップされている場合，次の例のように Performance オブジェクト名の前に **¥¥MachineName** を追加して，リモート・マシンから測定値を取得できます。

system/¥¥¥¥RemoteMachine¥¥Processor(_TOTAL)¥¥%¥ Processor¥ Time=Processor¥ Time(Remote Machine)|percent

---

Solaris ホストでのカスタム・システム測定値のキャプチャ

システム測定値コレクタで監視可能な Solaris システム測定値は，**kstat** コマンドを使って特定します。システム測定値コレクタで監視できるのは，**kstat** コマンドを使って見つかった測定値のサブセットだけです。

Solaris システム測定値の測定値を収集するには，次の手順を実行します。

1 **kstat** コマンドを実行して，監視する測定値を特定します。

Solaris システム測定値は次の形式です。

```
module:instance:name:statistic
```

次に例を示します。

```
vmem:35:ptms_minor:free
```

2 測定値コレクタで追加システム測定値の測定値を収集するには，次のテンプレートを使って，**metrics.config** ファイルで測定値のエントリをシステム測定値コレクタに追加します。

```
＜コレクタ名＞/＜測定値設定＞=＜測定値 ID ＞|＜測定値の単位＞
```

このテンプレートについては，662 ページ「測定値コレクタのエントリについて」で説明します。

このテンプレートを使うと，前の手順の例は，最初に次のようになります。

```
system/vmem:35:ptms_minor:free = Virtual Memory (35) Free | count
```

3 最初のエントリをフォーマットします。円マーク「¥」，スペース，またはコロン「:」の前に，拡張文字「¥」を追加します。

この手順に従うと，前の手順の最初のエントリは次のようになります。

```
system/vmem¥:35¥:ptms_minor¥:free = Virtual¥ Memory¥ (35)¥ Free | count
```

これは，システム測定値コレクタで Solaris システム測定値の測定値を収集するために，**metrics.config** を正しくフォーマットしたエントリです。

Linux ホストでのカスタム・システム測定値のキャプチャ

システム測定値コレクタで監視可能な Linux システム測定値は，**/proc** ファイル・システムで特定します。カスタム Linux 測定値を収集するようにシステム測定値コレクタを設定するには，**/proc** ファイル・システムをスキャンして目的の測定値を見つけ，測定値情報の場所に従って **metrics.config** の測定値に，システム測定値コレクタのエントリを作成する必要があります。

Linux システム測定値の測定値を収集するには，次の手順を実行します。

1 **/proc** ファイル・システムをスキャンして，Diagnostics システム測定値コレクタで監視する測定値を特定します。

Linux 測定値に，**metrics.config** でシステム測定値設定エントリを作成するには，システム測定値の値がある場所を明確に指定する必要があります。その場所は，次の値を使って指定します。

- **File name :** **/proc** ディレクトリからのパスを含む，測定値情報があるファイルの名前。
- **Line offset :** ファイルの中で，システム測定値がある行までの行数のカウント。最初の行は 0 行目とカウントされます。
- **Word offset :** ファイルの行の中で測定値の値がオフセットである語数のカウント。行の最初の語は 0 行目とカウントされます。指定したオフセットの値は符号なし整数である必要があります。

たとえば，Diagnostics ビューで表示して確認できるように，システム測定値コレクタで SwapFree システム測定値を監視する場合，**/proc** ディレクトリをスキャンして測定値を特定し，測定値が **meminfo** ファイルにあることを検出します。このファイルのレイアウトは次のとおりです。

```
MemTotal: 515548 kB
MemFree: 1552 kB
Buffers: 41616 kB
Cached: 152084 kB
SwapCached: 46064 kB
Active: 402720 kB
Inactive: 75328 kB
HighTotal: 0 kB
HighFree: 0 kB
LowTotal: 515548 kB
LowFree: 1552 kB
SwapTotal: 1048568 kB
SwapFree: 779192 kB
Dirty: 4544 kB
Writeback: 0 kB
Mapped: 300056 kB
Slab: 28764 kB
Committed_AS: 801364 kB
PageTables: 3184 kB
VmallocTotal: 499704 kB
VmallocUsed: 2184 kB
VmallocChunk: 497324 kB
HugePages_Total: 0
HugePages_Free: 0
Hugepagesize: 4096 kB
```

このファイルの SwapFree 測定値の場所は，次の値になります。

- **File name :** meminfo
- **Line offset :** 12
- **Word offset :** 1

2 追加システム測定値の測定値を収集するには，次のテンプレートを使って，**metrics.config** ファイルで測定値のエントリをシステム測定値コレクタに追加します。

```
＜ collector_name ＞ / ＜ line ＞＜ word ＞＜ file ＞ = ＜測定値 ID ＞ | ＜測定値単位＞
```

このテンプレートは，662 ページ「測定値コレクタのエントリについて」に記載されているテンプレートの 1 つのバージョンです。**＜測定値設定＞**プロパティは，プロパティ **＜ line ＞ : ＜ word ＞ : ＜ file ＞** に置き換えられています。

このテンプレートを使うと，前の手順の例は，最初に次のようになります。

```
system/12:1:meminfo = Swap Free | kilobytes
```

3 最初のエントリをフォーマットします。円マーク「¥」，スペース，またはコロン「:」の前に，拡張文字「¥」を追加します。

この手順に従うと，前の手順の最初のエントリは次のようになります。

```
system/12¥:1¥:meminfo = Swap¥ Free | kilobytes
```

これは，システム測定値コレクタで Solaris システム測定値の測定値を収集するために，**metrics.config** を正しくフォーマットしたエントリです。

z/OS システム測定値のキャプチャの有効化

z/OS プラットフォームの次のシステム測定値を収集できます。

- CPU
- DiskIOPerSec
- DiskBytesPerSec

システム設定をいくつか変更する必要があるため，標準設定ではシステム・メトリックスはキャプチャされません。したがって，システム設定の変更が必要になります。z/OS システム・メトリックスのキャプチャを有効にするには，次の設定手順を行う必要があります。

z/OS システム測定値のキャプチャを有効にするには，次の手順を実行します。

1. **＜プローブのインストール・ディレクトリ＞ /bin/** ディレクトリの権限を実行を再帰的に許可するように変更します。これを実行するには，次のコマンドを使用します。

```
chmod -R 770...
```

2. **＜プローブのインストール・ディレクトリ＞ /bin/390-zos/systemmetrics** ディレクトリの権限を実行を許可するように変更します。これを実行するには，次のコマンドを使用します。

```
chmod -R 0+x ...
```

3. RMF Monitor III を起動して，SMF レコード 70 ～ 79 が収集されていることを確認します。
4. sysplex の 1 つ以上のシステムで RMF Data Buffer を起動します。
5. ERBDSQRY サービスに渡されたシステム名のリストを確認します。
6. システムがサブセット 5 のある SMF レコード 92 を収集していることを確認します。

20

Java Agent - JMX メトリックス・キャプチャ

JMX 測定値のキャプチャ・プロセスと，Java Agent メトリックス・コレクタを設定して JMX メトリックスをキャプチャする手順について説明します。

本章の内容

JMX 測定値について

Java Agent には，あらかじめ定義された JMX メトリックス・コレクタが付属しており，次のアプリケーション・サーバから JMX 測定値にアクセスします。

- IBM WebSphere
- BEA Weblogic
- SAP NetWeaver
- Oracle AS
- Apache Tomcat
- JBoss J2EE サーバ
- TIBCO BusinessWorks

また，Java Agent は，JMX スタンダードをサポートしている J2EE サーバから JMX データを収集することもできます。

Java Agent では，JMX 測定値コレクタを定期的に実行して，アプリケーション・サーバから測定値を収集します。収集されたメトリックスは，HP Diagnostics および Diagnostics Java Profiler 両方のユーザ・インタフェースに表示されます。

JMX メトリックの収集のための WebSphere の設定

WebSphere JMX メトリックス収集について，場合によっては，JMX メトリックスの受信を開始するために，WebSphere サーバに Performance Monitoring Infrastructure（PMI）を設定する必要があります。

JMX メトリックス収集用に WebSphere 5.x，6.x，7.0 サーバを設定する方法の詳細については，209 ページ「JMX メトリックの収集のための WebSphere の設定」を参照してください。

JMX メトリックス収集のための TIBCO の設定

TIBCO JMX メトリックス収集では，JMX メトリックス収集を有効にする必要があります。手順については，182 ページ「JMX メトリックス収集のために TIBCO BusinessWorks を設定するには，次の手順を実行します。」を参照してください。

JMX メトリックス・コレクタの設定について

JMX メトリックス・コレクタは設定可能であり，ユーザはどの JMX メトリックスを収集するか制御できます。JMX 測定値コレクタは，**<プローブのインストール・ディレクトリ> /etc/metrics.config** ファイルに定義されます。

通常，各アプリケーション・サーバの主要バージョンごとに個別のコレクタが定義されます。

メトリックス・コレクタの一般的な情報については，第 18 章，「Java Agent メトリックス・コレクタ」を参照してください。メトリックス・コレクタ・エントリと構文については，662 ページ「測定値コレクタのエントリについて」を参照してください。

追加カスタム JMX メトリックス

Java Agent には，679 ページ「JMX 測定値について」に記載されているアプリケーション・サーバのためのさまざまな定義済み JMX メトリックス・コレクタが一緒にインストールされています。metrics.config ファイルのエントリを定義してこれらのコレクタを設定します。662 ページ「測定値コレクタのエントリについて」を参照してください。また，既存の測定値コレクタにエントリを作成したり，Diagnostics で監視する追加 JMX メトリックスがある場合に，新しいコレクタを作成することもできます。

JMX メトリックス・コレクタに新しいエントリを作成するため，利用可能な JMX メトリックスおよび WebSphere Performance Monitoring Infrastructure（PMI）メトリックスのリストを取得できます。次に，**metrics.config** ファイルに新しいメトリックス・エントリを作成できます。次の項では，追加 JMX メトリックスおよび PMI メトリックスを監視できるように，JMX メトリックス・コレクタに新しいエントリを作成する方法について説明します。

利用可能な JMX または WebSphere PMI メトリックスのリストの取得

Java Agent にインストールされている測定値コレクタには，各アプリケーション・サーバで使用可能なさまざまな JMX 測定値のためのエントリが含まれます。ただし，監視するほかの JMX メトリックスまたは WebSphere PMI メトリックスがある場合や，またはアプリケーション・サーバのベンダによって新しいメトリックスが公開されている場合もあります。

収集用の新規 / 追加 JMX/PMI メトリックスの設定を簡単にするため，metrics.config ファイルには，各 JMX コレクタに使用可能な全メトリックスのリストを 1 つのファイルに書き込む機能があります。**metrics.config** ファイルの **default.dump.available.metrics** プロパティが true に設定されている場合，プローブは，利用可能なメトリックスのこのリストをプローブのログ・ディレクトリにあるテキスト・ファイルに書き込みます。ファイルの名前は次のとおりです。**＜プローブのインストール・ディレクトリ＞ /log/ ＜ Probe ID ＞ / jmx_metrics ＜コレクタの名前＞ .txt。**

プローブの **metrics.config** ファイルの **default.dump.available.metrics** プロパティは実行時に変更できます。使用可能な JMX/PMI メトリックスのリストを書き込めるようにプロパティを true に設定するのは一時的だけにすることをお勧めします。メトリックス・リストがファイルに書き込まれた後は，メトリックス・リストをファイルに定期的に書き込むプローブのオーバーヘッドをなくすために，プロパティは false に再設定（またはコメント・アウト）してください。

メトリックス・リスト・ファイルのいくつかの例を次に示します。このタイプの情報を使用して，プローブの **etc/metrics.config** ファイルに追加の JMX または PMI メトリックスを設定できます。

次の例は，利用可能な MBean ObjectNames とその収集可能属性を示しています。

```
======= MBean ObjectNames and Available Attributes =======
MBean ObjectName:
WebSphere:J2EEServer=server1,JDBCProvider=Derby JDBC
Provider,JDBCResource=Derby JDBC
Provider,Server=server1,cell=yli87Node01Cell,diagnosticProvider=true,j2eeType=JDB
CDataSource,mbe
anIdentifier=cells/yli87Node01Cell/nodes/yli87Node01/servers/server1/
resources.xml#DataSource_12442
31364323,name=WST_PriceGen,node=yli87Node01,platform=dynamicproxy,process=
server1,spec=1.0,
type=DataSource,version=6.1.0.0
Available Attributes:
name: loginTimeout, type: int
name: statementCacheSize, type: int
name: testConnectionInterval, type: java.lang.Integer
.....................
```

次の例は，利用可能な MBean ObjectNames とその収集可能属性およびフィールドを示しています。

```
======= MBean ObjectNames and Available Attributes and Fields =======
MBean ObjectName:
java.lang:name=PS Old Gen,type=MemoryPool
Available Metrics:
Attribute: CollectionUsage type: javax.management.openmbean.CompositeData
    Field: committed, type: java.lang.Long
    Field: init, type: java.lang.Long
    Field: max, type: java.lang.Long
    Field: used, type: java.lang.Long
```

次の例は，利用可能な MBean ObjectNames とその収集可能操作およびフィールドを示しています。

```
======= MBean ObjectNames and Available Operations and Fields =======
MBean ObjectName:
com.tibco.bw:key=engine,name="MortgageBroker-BrokerService"
Available Metrics:
Operation: java.lang.Integer GetActiveProcessCount()
Operation: javax.management.openmbean.CompositeData GetExecInfo()
    Field: Threads, type: java.lang.Integer
    Field: Uptime, type: java.lang.Long

Operation: javax.management.openmbean.CompositeData GetMemoryUsage()
    Field: FreeBytes, type: java.lang.Long
    Field: PercentUsed, type: java.lang.Long
    Field: TotalBytes, type: java.lang.Long
    Field: UsedBytes, type: java.lang.Long
```

WebSphere JMX コレクタの場合，汎用 MBean JMX メトリックスのほかに，利用可能な WebSphere 固有の PMI メトリックスも WebSphere コレクタのダンプ・ファイルにダンプされます。これには，次の例に示すように，PMI ツリー・インスタンス・パスと利用可能な統計，および PMI モジュール設定情報が含まれます。

```
======= PMI Tree and Available PMI Statistics =======
connectionPoolModule
Available Statistics:
CreateCount, CloseCount, AllocateCount, ReturnCount, PoolSize, FreePoolSize,
WaitingThreadCount, FaultCount, PercentUsed, PercentMaxed, UseTime, WaitTime,
ManagedConnectionCount, ConnectionHandleCount, PrepStmtCacheDiscardCount,
JDBCTime
connectionPoolModule->Derby JDBC Provider
Available Statistics:
CreateCount, CloseCount, AllocateCount, ReturnCount, PoolSize, FreePoolSize,
WaitingThreadCount, FaultCount, PercentUsed, PercentMaxed, UseTime, WaitTime,
ManagedConnectionCount, ConnectionHandleCount, PrepStmtCacheDiscardCount,
JDBCTime
connectionPoolModule->Derby JDBC Provider->jdbc/ALBUM
Available Statistics:
CreateCount, CloseCount, AllocateCount, ReturnCount, PoolSize, FreePoolSize,
WaitingThreadCount, FaultCount, PercentUsed, PercentMaxed, UseTime, WaitTime,
ManagedConnectionCount, ConnectionHandleCount, PrepStmtCacheDiscardCount,
JDBCTime
```

新しい JMX または WebSphere PMI メトリックス・エントリの作成

ここで，次のテンプレートに基づいて JMX または PMI メトリックス・エントリを作成するプロセスを紹介します。

```
＜コレクタ名＞/＜測定値設定＞=＜測定値 ID ＞|＜測定値の単位＞
```

このテンプレートについては，662 ページ「測定値コレクタのエントリについて」で説明します。

JMX または WebSphere PMI メトリックスをキャプチャするには，次の手順を実行します。

1 **＜プローブのインストール・ディレクトリ＞ /etc/metrics.config.** を開き，Java Agent の監視対象のアプリケーションに適切な JMX 測定値コレクタを見つけます。

2 **＜コレクタ名＞** パラメータは，コレクタのほかのエントリと同じです。WebLogic にエントリを作成している場合，このパラメータの値は WebLogic になります。

3 **＜測定値設定＞** パラメータを作成します。

   a JMX 測定値の場合，**＜測定値設定＞** パラメータは，コレクタが一致する MBean を検索するのに使うパターンです。パターンは，「.」文字で区切った 2 つのコンポーネントで構成されます。下記の構文を参照してください。

   MBean オブジェクトおよび属性

   ```
   ＜ MBean オブジェクト名パターン＞ . ＜属性名＞
   ```

   MBean オブジェクト，属性，フィールド

   ```
   ＜ MBean オブジェクト名パターン＞ . ＜属性名＞ # ＜フィールド名＞
   ```

   MBean オブジェクトおよび操作

   ```
   ＜ MBean オブジェクト名パターン＞ .( ＜操作名＞ ())
   ```

   MBean オブジェクト，操作，フィールド

   ```
   ＜ MBean オブジェクト名パターン＞ .( ＜操作名＞ ()# ＜フィールド名＞ )
   ```

- **＜MBean オブジェクト名パターン＞**は, MBean のオブジェクト名を表す文字列です。測定値パターンの詳細については，687 ページ「メトリックス・パターンについて」を参照してください。JMX メトリックスをグループ化する方法の詳細については，688 ページ「JMX GROUPBY および EXPAND_PMI 修飾子」を参照してください。

- **＜属性名＞** は，測定値を表わす MBean 属性の名前です。＜属性名＞に「.」が含まれる場合は，丸括弧で囲む必要があります。**＜ MBean オブジェクト名パターン＞ .(＜属性名＞)**

  たとえば，WebLogic アプリケーション・サーバの場合，すべての**実行キュー**のスループットの＜測定値設定＞パラメータは次のように設定されます。

  ```
  *:Type=ExecuteQueueRuntime,*.ServicedRequestTotalCount
  ```

  利用可能な属性を表示するメトリックス・ダンプの例については，681 ページ「利用可能な JMX または WebSphere PMI メトリックスのリストの取得」を参照してください。

- **＜属性名＞ # ＜フィールド名＞** : 複合データを返す JMX 属性には，メトリックスとして使用する数値フィールドを含めることができます。単に MBean 名の後に，記号 # に続けてフィールド名を追加します。

  次に例を示します。

  ```
  Java¥ Platform/java.lang¥:type¥=MemoryPool,name¥=Perm¥
  Gen.Usage#used
  ```

  この例では，＜ Perm Gen ＞ MBean の＜ Usage ＞複合データ属性の＜ used ＞フィールドを追跡します。

- **(＜操作名＞ ())** : 操作名の後に始め括弧と終わり括弧が続きます。さらに，操作名全体を括弧で囲みます。操作で複合属性が返される場合，() の後に複合属性フィールドを追加します。

  次に例を示します。

  ```
  Tibco/com.tibco.bw¥:key¥=engine,name¥=*.(GetActiveProcessCount()) =
  Active Process Count|count|Tibco
  ```

  引数を使用しない操作のみがサポートされています。

➤ **(＜操作名＞()# ＜フィールド名＞)**: 複合データを返す JMX 操作には，メトリックスとして使用する数値フィールドを含めることができます。単に MBean 名の後に，記号 # に続けてフィールド名を追加します。

次に例を示します。

```
Tibco/com.tibco.bw¥:key¥=engine,name¥=*.(getStatus()#Total¥ Errors) =
Total Errors|count|Tibco
```

この例では，getStatus() 操作で返された複合データ・オブジェクトの「Total Errors」フィールドを追跡します。

b WebSphere PMI メトリックスの場合，**＜測定値設定＞**パラメータは，一致する PMI 統計を検索するのにコレクタが使うパターンです。パターンは，「.」文字で区切った 2 つのコンポーネントで構成されます。

```
＜ PMI 統計記述子＞ . ＜統計名＞
```

➤ **＜ PMI 統計記述子＞**は，WebSphere PMI ツリーの特定の状態を見つけ，アクセスに使います。それは PMI モジュール名（webAppModule など），または PMI モジュール分岐（[webAppModule][AccountManagement#AccountManagementWar.war] など）のどちらかの可能性があります。

➤ **＜統計名＞**は，測定値を表わす PMI 統計の名前です。統計名に「.」が含まれる場合は，丸括弧で囲む必要があります。
[webAppModule][AccountManagement#AccountManagementWar.war].
(webAppModule.numLoadedServlets)

PMI モジュールと PMI モジュール分岐の例およびそれらの利用可能な統計名の例については，681 ページ「利用可能な JMX または WebSphere PMI メトリックスのリストの取得」を参照してください。

PMI メトリックスをグループ化する方法の例については，688 ページ「JMX GROUPBY および EXPAND_PMI 修飾子」を参照してください。

4 次の例のように，残りの JMX 測定値エントリ・テンプレートを入力します。

```
WebLogic/*:Type=ExecuteQueueRuntime,*.ServicedRequestTotalCount =
RATE(Execute Queues Requests / sec|count|Execute Queues)
```

5 最初のエントリをフォーマットします。円マーク「¥」，スペース，またはコロン「:」の前に，拡張文字「¥」を追加します。

この手順に従うと，前の手順の最初のエントリは次のようになります。

```
WebLogic/*¥:Type¥=ExecuteQueueRuntime,*.ServicedRequestTotalCount =
RATE(Execute Queues Requests / sec|count|Execute Queues)
```

これは，コレクタで WebLogic JMX 測定値の収集を有効にする JMX 測定値コレクタの正しくフォーマットされたエントリです。

メトリックス・パターンについて

JMX メトリックスの場合, <メトリックス設定>パラメータは, 一致する MBean を検索するために Collector が使用するパターンです。次に例を示します。

```
*:Type=ExecuteQueueRuntime,*.ServicedRequestTotalCount
```

上の例で，オブジェクト名は ***:Type=ExecuteQueueRuntime,*** です。これは，名前に **ExecuteQueueRuntime** と同じ **Type** コンポーネントがある数多くの MBean に解決します。**ServicedRequestTotalCount** は，JMX 測定値コレクタによって測定値が収集される属性名です。

注 : JMX コレクタの現在の実装では，タイプが数値（ロング，整数など）の属性だけがサポートされます。

JMX 測定値コレクタは，まず MBeanServer のクエリ機能を使って，設定に指定されている各オブジェクト名に一致する MBean を検索します。JMX 測定値の場合，オブジェクト名は，コレクタが一致する MBean を検索するのに使うパターンです。オブジェクト名の詳細については，
http://java.sun.com/j2ee/1.4/docs/api/javax/management/ObjectName.html を参照してください。

MBean オブジェクト名は複数の MBean に解決可能なパターンであるため，JMX コレクタは，エントリのすべての属性名を，パターンに一致するすべての MBean で検証し，一致する MBean のセットで属性値を集計します。ただし，常にオブジェクト名が複数の MBean に解決するとは限りません。たとえば，次のオブジェクト名は（WebLogic アプリケーション・サーバの）1 つの MBean に解決します。

```
*¥:Name¥=weblogic.kernel.Default,Type¥=ExecuteQueueRuntime,
*.ServicedRequestTotalCount
```

JMX GROUPBY および EXPAND_PMI 修飾子

省略可能な GROUPBY 修飾子を使用し，GROUPBY の指定するキーと値が同じである MBean ObjectNames の各グループについて別々のメトリックスを作成できます。MBean オブジェクト名パターンを記述する JMX 測定値用に，プローブの etc/metrics.config ファイルには，追加できる省略可能な修飾子 GROUPBY があります。GROUPBY は JMX ベースのコレクタに metric_config をマルチインスタンス式として取り扱うよう指示します。

```
collector_name/GROUPBY[oname_key]/metric_config = ...
```

コレクタは metric_config と一致するすべての MBeans を検出し，オブジェクト名キー oname_key を使って各 MBeans に対応するメトリックスを作成し，それを category_id に追加して一意の名前を作成します。

```
WebSphere6/GROUPBY[name]/
WebSphere¥:type¥=DataSource,*.statementCacheSize = JDBC Statement
Cache Size|bytes|JDBC DataSource
```

次に例を示します。

```
WebSphere6/connectionPoolModule.CreateCount = JDBC Connection
Creates|count|JDBC ConnectionPools
```

```
WebSphere6/[connectionPoolModule][Derby¥ JDBC¥ Provider][jdbc/
ALBUM].AllocateCount = JDBCConnection Allocates|count|JDBC
ConnectionPools
```

または，省略可能な EXPAND_PMI 修飾子を使用し，JMX メトリックスをグループ化する方法と同じようにして PMI メトリックスをグループ化できます。

PMI の場合，EXPAND_PMI 修飾子は，指定のモジュールまたは StatDescriptor 分岐から指定レベルだけ PMI ツリーを展開します。拡張レベル「n」は 1，2，...，または * にできます。標準設定レベルは 1 です。* はすべてを展開するという意味です。

collector_name/EXPAND_PMI[n]/metric_config = ...

次に例を示します。

WebSphere6/EXPAND_PMI[*]/connectionPoolModule.AllocateCount = JDBC Connection Allocates|count|JDBC ConnectionPools

これは，各 JDBC 接続プール・プロバイダについて，およびプロバイダの各 DataSource について「JDBC Connection Allocates」測定値を作成します。

第 VII 部

ほかの HP ソフトウェア製品との統合のセットアップ

本項の内容

- Business Service Management と Diagnostics 間の統合のセットアップ
- LoadRunner Diagnostics Add-in のインストール
- HP LoadRunner と HP Diagnostics の統合のセットアップ
- Diagnostics を使用するための Performance Center のセットアップ

21

Business Service Management と Diagnostics 間の統合のセットアップ

HP Business Service Management と Diagnostics 間の統合をセットアップする方法について説明します。

注： このドキュメントは, 特に記述がないかぎり, Diagnostics の Business Service Management 9.x との統合を対象としています。サポートされている統合の最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

本章の内容

Business Service Management と Diagnostics 間の統合のセットアップについて

HP Diagnostics と Business Service Management を連携して使用する前に，Diagnostics コンポーネントとの通信に必要な情報を Business Service Management に入力する必要があります。

Business Service Management に Diagnostics をセットアップするには，次の手順を実行します。

1 Diagnostics サーバの情報を指定します。

Business Service Management に Diagnostics サーバの情報を入力します。詳細については，696 ページ「Business Service Management での Diagnostics サーバの登録」を参照してください。

2 関連する権限を割り当てます（任意）。

さまざまな Diagnostics ユーザにそれぞれ権限を与えます（この手順は任意です）。詳細については，704 ページ「Business Service Management での Diagnostics ユーザへの権限の割り当て」を参照してください。

3 Windows 2003 のみ : インターネット・ブラウザの設定を変更します。

インターネット・ブラウザを Windows 2003 環境で実行している場合，Business Service Management の［Diagnostics 設定］ページおよび［アプリケーション］ページにアクセスするためにインターネット・ブラウザの設定を変更する必要があります。詳細については，707 ページ「Windows 2003 での Diagnostics のページへのアクセス」を参照してください。

4 Diagnostics コマンダ・ホストでクッキーを有効にします。

Business Service Management に Diagnostics データを表示するには，クッキーを有効にする必要があります。これは通常，登録されている Diagnostics コマンダ・サーバを信頼されるサイトとしてブラウザ設定に追加することによって解決できます。

Business Service Management での Diagnostics サーバの登録

Business Service Management から HP Diagnostics にアクセスできるようにするには，Diagnostics サーバを登録します。次の項では，Business Service Management での登録手順について説明します。BAC 8.x との違いも示します。

重要 : Business Service Management をアップグレードしたら, Business Service Management と Diagnostics の統合を再登録する必要があります。

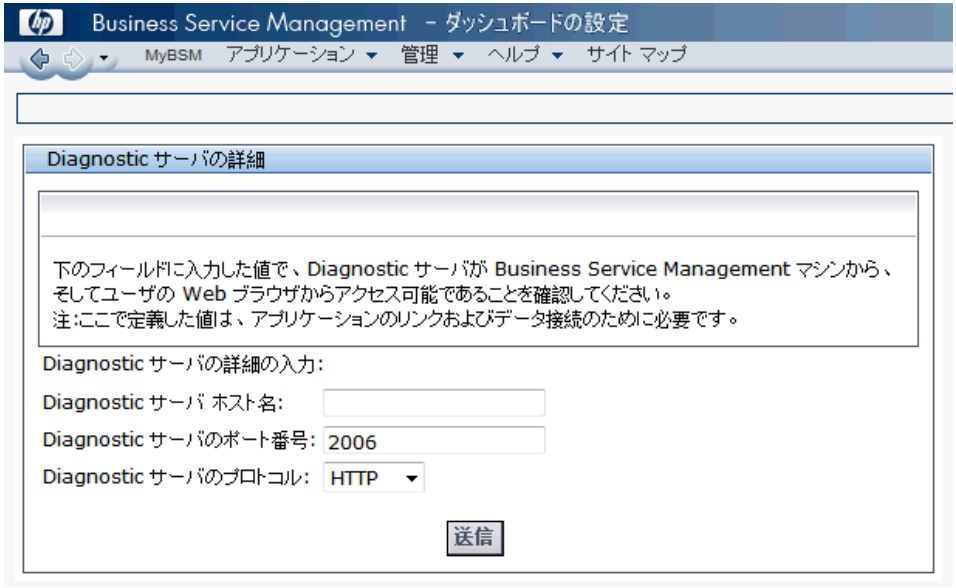
注 : Windows 2003 を使用している場合は，Diagnostics の［管理］ページにアクセスするようにインターネット・ブラウザを設定する必要があります。詳細については，707 ページ「Windows 2003 での Diagnostics のページへのアクセス」を参照してください。

Diagnostics が Business Service Management と統合されてから Diagnostics コマンダ・サーバをアップグレードした場合，**RegistrarPersistence.xml** ファイルを **etc.old** フォルダから新しい **etc** フォルダにコピーする必要があります。次に，［**BSM**］>［**管理**］>［**Diagnostics**］ページで Diagnostics の統合を確認し，正常に動作していない場合は BSM で Diagnostics サーバを再登録します。

Business Service Management で Diagnostics サーバを最初に登録するには，次の手順を実行します。

1 Business Service Management にログオンします。

2 ［**管理**］＞［**Diagnostics**］を選択します。統合を最初にセットアップする場合は，［Diagnostics サーバの詳細］ページが表示されます。



注 : Diagnostics サーバを設定する前に，([サイトマップ]ページの[**Diagnostics**]をクリックするか，または［**アプリケーション**］＞［**Diagnostics**］をクリックして）HP Diagnostics にアクセスしようとすると，Diagnostics サーバに登録するように指示するメッセージが表示されます。リンクをクリックして，［ダッシュボードの設定］ページを開きます。

---

3 Diagnostics コマンド・サーバの情報を入力します。

➤ **Diagnostics サーバ・ホスト名 :** Diagnostics コマンド・サーバのホスト・マシンの名前を入力します。

Diagnostics サーバが Business Service Management と同じシステムにインストールされている場合も，［**Diagnostics サーバ ホスト名**］ボックスに実際のホスト名を入力する必要があります。この場合，**localhost** はホスト名の代わりになりません。

Business Service Management が完全修飾ドメイン名を通じてアクセスされる場合は，完全修飾ドメイン名で Diagnostics サーバ・ホストを登録します。

- **Diagnostics サーバのポート番号 :** Diagnostics コマンド・サーバ で使用するポート番号を入力します。デフォルトのポート番号は **2006** です。
- **Diagnostics サーバのプロトコル :** Business Service Management が Diagnostics に接続する際の通信プロトコルを **HTTP** または **HTTPS** のいずれかから選択します。

注 : 通信プロトコルに **HTTPS** を選択した場合, 追加の設定手順が必要になります。必要な手順の詳細は，第 C 章,「コンポーネント間での HTTPS 有効化」を参照してください。

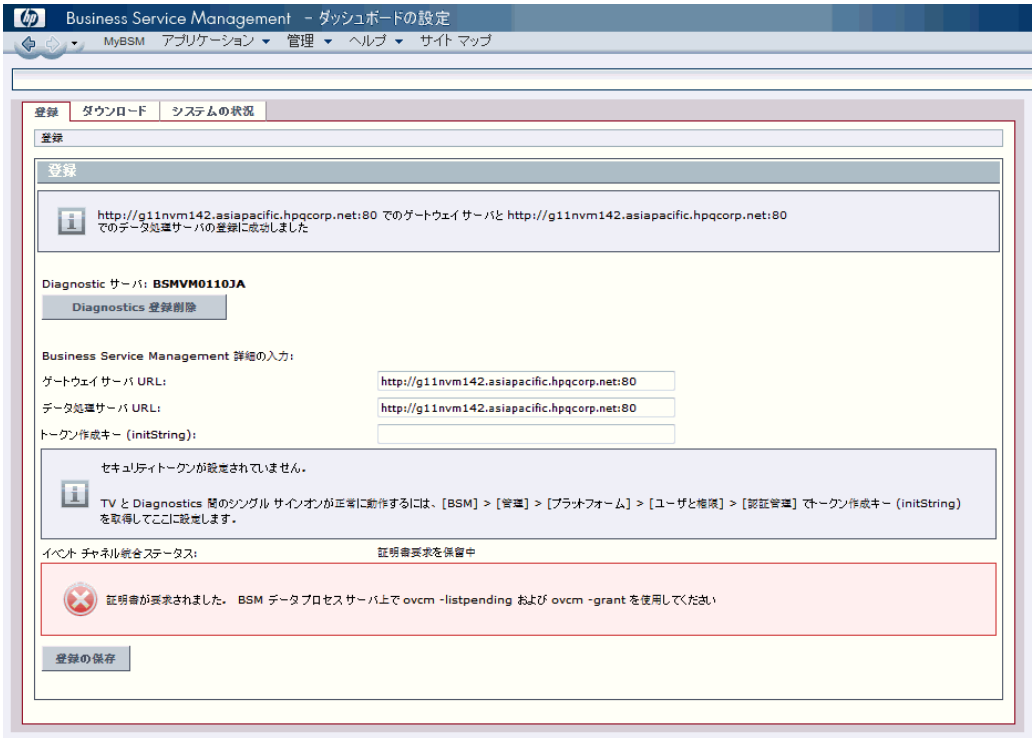
---

4 Diagnostics サーバの情報を入力して, それらの情報が正しいことを確認したら，［**送信**］をクリックして Diagnostics サーバの設定プロセスを完了します。

入力したサーバ名が間違っているか，またはサーバが使用不可の場合は，エラー・メッセージが表示されます。

［**送信**］をクリックすると，Diagnostics サーバ の情報が Business Service Management に保存され，Business Service Management のサーバ情報が Diagnostics サーバ・マシンに自動的に登録されます。

5 ［ダッシュボートの設定］ページの［**登録**］タブが開き，使用可能だった Business Service Management サーバの情報が表示されます。



Business Service Management の詳細

必要に応じて，［登録］タブの［**Business Service Management 詳細の入力**］セクションで Business Service Management のサーバ情報を手動で変更できます。

ゲートウェイ・サーバ URL : ルート URL が，Business Service Management へのアクセスに使用するルート URL と一致していることを確認してください。

データ処理サーバ URL : ルート URL が，Business Service Management へのアクセスに使用するルート URL と一致していることを確認してください。データ処理サーバの URL が，ゲートウェイ・サーバの URL と異なる場合は，通常，この URL でポート 8080 を使用します。

注 : Diagnostics サーバから BSM 処理サーバにアクセスできず，処理サーバに証明書要求をトンネリングするように BSM ゲートウェイ・サーバが設定されている場合（ゲートウェイ・サーバと処理サーバがロード・バランサまたは SSL アクセラレータの反対側にある場合など），この Diagnostics サーバの登録ページで BSM ゲートウェイ・サーバと処理サーバの両方に BSM ゲートウェイを使用します。この処理を行っても，次のページで説明されているように，処理サーバで手動で証明書を設定する必要がある可能性もあります。

トークン作成キー（nitString） TransactionVision を使用している場合に，TransactionVision から Diagnostics へのドリルダウン時に再ログインしなくてもよいようにするには，フィールドに Business Service Management トークン作成キーを入力します。トークン・キーを入力すると，このキーが Diagnostics に渡されます。TransactionVision を使用していない場合は，このキーを入力する必要はありません。

トークン作成キー（initString）は，Business Service Management の［**管理**］>［**プラットフォーム**］>［**ユーザおよび権限**］>［**認証管理**］で入手できます。登録が完了すると，このトークン作成キーが Diagnostics の **lwsso.properties** ファイルに書き込まれます。

イベント・チャネル統合ステータス : Diagnostics 9.x と Business Service Management 9.x を統合している場合，イベント・チャネルは，状況インジケータ・ステータス・イベントを Business Service Management ゲートウェイ・サーバに送信するために Diagnostics で使用されます（実際は OM エージェントおよび IAPA コンポーネントが使用されます）。

登録中に，イベント・チャネル統合を行うスクリプト（**＜ Diagnostics のインストール・ディレクトリ＞ /server/bin/switch_ovo_agent.sh** または **.vbs**）が実行されます。このスクリプトを実行するには，UNIX でのルート・アクセスが必要です。Diagnostics コマンダ・サーバが UNIX で root として実行されていない場合，BSM での Diagnostics の登録は権限の拒否エラーで失敗します。このエラーの発生後，Diagnostics コマンダ・サーバで手動で root としてスクリプトを実行する必要があります。手動でスクリプトを実行する前に，BSM の登録手順が実行されている必要があります。

重要 : イベント・チャネル統合での BSM ゲートウェイ・サーバと BSM 処理サーバ間の通信では，これらのサーバが異なるシステム上にある場合は，マシン間に信頼関係が確立されている必要があります。この設定方法については，716 ページ「DPS とゲートウェイに対する個別の BSM サーバの設定」を参照してください。

イベント・チャネル統合ステータスが赤の場合，証明書要求が保留中であるメッセージが表示されているかどうかを確認します。登録中に，イベント・チャネル統合を行うスクリプト（**＜ Diagnostics のインストール・ディレクトリ＞ / server/bin/switch_ovo_agent.vbs** または **.sh**）が実行されます。ただし，証明書を設定する手順を手動で実行する必要があるため，ステータスに証明書要求が保留中と示されます。

この場合，次の処理を実行して，イベント・チャネル統合を完了する必要があります。

証明書を手動で設定するには，次の手順を実行します。

1 Business Service Management データ処理サーバに移動して，OM エージェントの証明書を設定します。複数の証明書が一覧表示されている場合は，適切な証明書を選択します。

ovcm -listpending -l

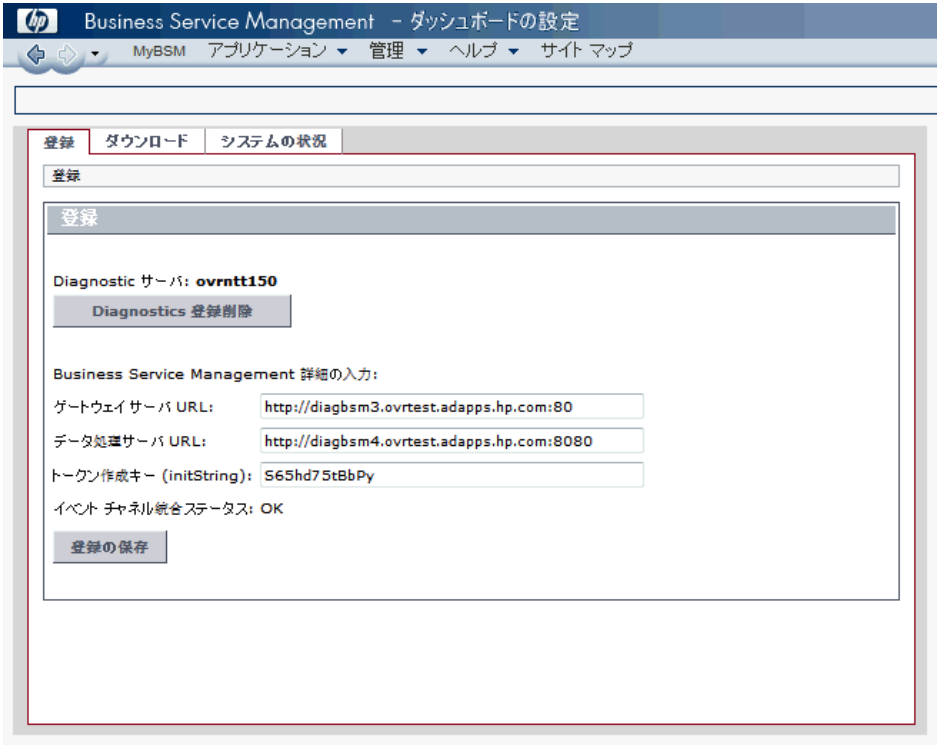
ovcm -grant ＜上記出力内のコア ID＞

OM エージェントおよび IAPA のインストール, または証明書に問題が発生する場合は，875 ページ「OM エージェントのトラブルシューティング」を参照してください。

注 : Diagnostics と Business Service Management 8.x を統合する場合は, イベント・チャネル統合ステータスに N/A と表示されます。これは，Diagnostics が Business Service Management 8.x と統合されている場合, OM エージェントおよび IAPA コンポーネントが使用されないためです。

---

2 ［**登録の保存**］をクリックして，イベント・チャネル統合ステータスが OK になっていて，その他の値が正しく設定されていることを確認します。



Diagnostics の登録の削除

Diagnostics の登録は完全に削除できます。

Diagnostics の登録を削除するには，次の手順を実行します。

1 ［**管理**］＞［**Diagnostics**］を選択します。

2 ［**登録**］タブで，［**Diagnostics 登録削除**］ボタンをクリックします。

3 開いたメッセージで，［**OK**］をクリックして，Diagnostics の登録の削除を確定します。

Diagnostics の登録が正常に削除されたことを伝えるメッセージが表示されます。

新しい Diagnostics サーバを登録するには，［**管理**］＞［**Diagnostics**］を選択して，696 ページ「Business Service Management での Diagnostics サーバの登録」の手順に従います。

Diagnostics の［管理］ページについて

［**管理**］＞［**Diagnostics**］を選択して，［Diagnostics 設定］ページにアクセスします。Business Service Management の Diagnostics［管理］ページは，本項で説明する次の 3 つのタブで構成されています。

［登録］タブ

［登録］タブには，次の情報が表示されます。

- ユーザが Business Service Management で登録した Diagnostics サーバの詳細。情報を変更する場合は，703 ページ「Diagnostics の登録の削除」を参照してください。
- Diagnostics サーバ・マシンに自動的に登録された Business Service Management のサーバ情報。［**Business Service Management 詳細の入力**］セクションで，Business Service Management のサーバ情報を手動で変更できます。

Business Service Management で初めて Diagnostics を登録する方法の詳細については，696 ページ「Business Service Management での Diagnostics サーバの登録」を参照してください。

[ダウンロード] タブ

[ダウンロード] タブには Diagnostics Agent および Collector インストーラへのリンクがあり，関連するプラットフォームに適した Agent または Collector インストール・ファイルをダウンロードできます。

Diagnostics サーバ のインストール中に Agent または Collector インストーラへのパスを指定しなかった場合，[ダウンロード] タブにはコンポーネントが表示されません。詳細については，第 C 章,「Diagnostics サーバのインストール」を参照してください。

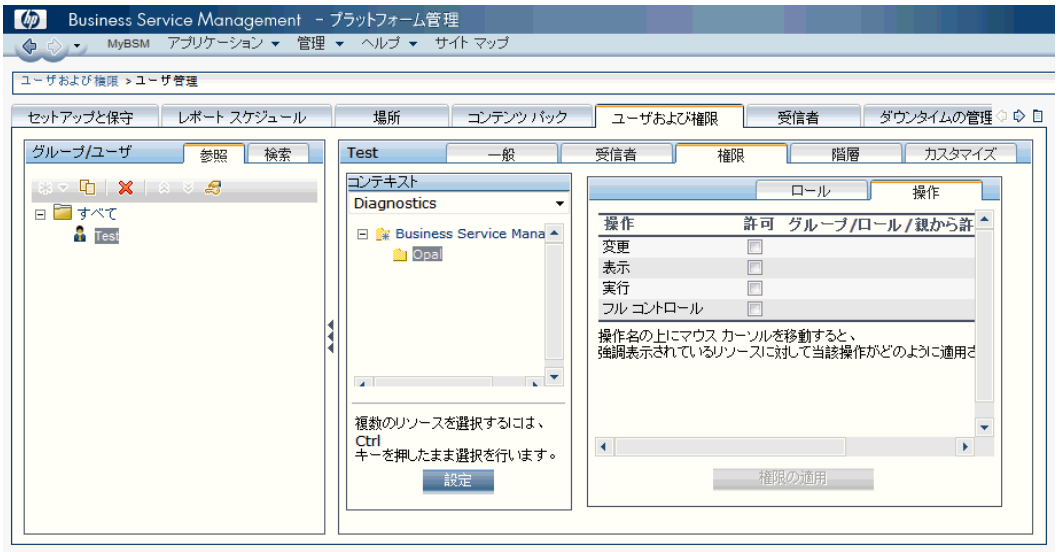
[システムの状況] タブ

HP Diagnostics デプロイメントのすべてのコンポーネントのマップを提供し，各コンポーネントのリアルタイム・ステータスと状況を知らせます。

Business Service Management での Diagnostics ユーザへの権限の割り当て

Business Service Management では，システムに定義されている特定のリソースのユーザおよびユーザ・グループに権限を適用できます。管理者が Diagnostics ユーザに与えられる特定の種類の操作権限があります。

次に，Business Service Management の［**管理**］>［**プラットフォーム**］>［**ユーザおよび権限**］>［**ユーザ管理**］ページの例を示します。



Diagnostics の権限は，「Diagnostics」というコンテキストに表示されています。ツリーには Business Service Management と，その下に Diagnostics が表示されています。Business Service Management 8.x では，Diagnostics の権限は Monitors コンテキストに表示されます。

Business Service Management で権限を適用する場合，管理者は，次の操作権限を Diagnostics ユーザに与えることができます。

- **変更 :** Diagnostics 管理の表示と Diagnostics の設定を有効にします。
- **表示 :** Business Service Management から Diagnostics にアクセスしたときに，Diagnostics アプリケーションを表示できるようにします。
- **実行 :** Diagnostics に設定されたしきい値を有効にします。
- **フル・コントロール :** Diagnostics ですべての操作を実行できるようにし，それらの操作の権限を割り当てたり解除できます。

Diagnostics の権限は BSM の役割から継承できます。

Business Service Management でユーザ権限を割り当てる方法の詳細については，HP Business Service Management 文書ライブラリの『プラットフォームの管理』を参照してください。

RTSM にアクセスするためのデータ・コレクタのパスワード

Business Service Management 構成サーバの起動中に，RTSM にアクセスするための Diagnostics などのデータ収集の標準設定パスワードを上書きできます。Business Service Management のセットアップおよびデータベース設定ユーティリティの［ログイン設定］ページでこのパスワードを上書きした場合，すべてのデータ・コレクタ（TV，BPI，RUM，Diagnostics）に同じパスワードが使用されます。

BSM の標準設定パスワードを上書きする場合，Diagnostics のパスワードもその新しいパスワードに一致するように変更する必要があります。Diagnostics サーバの **etc/cmdbProperties.xml** ファイルで変更します。

```
<customer>
        <!-- customerId is an Integer -->
        <customerId>1</customerId>
        <customerName>Default Client</customerName>
        <userName>diagnostics</userName>
        <!-- userPassword may be obfuscated -->
        <userPassword>integration</userPassword>
    </customer>
<customer>
```

Windows 2003 での Diagnostics のページへのアクセス

インターネット・ブラウザを Windows 2003 環境で実行している場合, Business Service Management の［Diagnostics 設定］ページおよび［アプリケーション］ページにアクセスするためにインターネット・ブラウザの設定を変更する必要があります。

Windows 2003 環境で Diagnostics のページにアクセスするには，次の手順を実行します。

1 Internet Explorer で，［**ツール**］＞［**インターネット オプション**］を選択して［インターネット オプション］ダイアログ・ボックスを開きます。

2 ［**プライバシー**］タブの［**サイト**］で，［サイトごとのプライバシーの操作］ダイアログ・ボックスを開きます。

3 ［**Web サイトのアドレス**］ボックスで, Diagnostics サーバの名前を入力します。

- Business Service Management で Diagnostics サーバを登録したときに IP アドレスを入力した場合は, その IP アドレスを入力します。Business Service Management にホスト名を入力した場合は，そのホスト名を入力します。
- 次の例のように，「http://」または「https://」プレフィクス，およびポート番号を含めます。

  http:// ＜ Diagnostics Commander ＞ :2006/

4 ［**許可**］をクリックします。

5 ［**OK**］をクリックして，［サイトごとのプライバシーの操作］ダイアログ・ボックスを閉じます。

6 ［**OK**］をクリックして，［インターネット オプション］ダイアログ・ボックスを閉じます。

Business Service Management からの Diagnostics アプリケーションへのアクセス

Diagnostics を使用するように Business Service Management をセットアップすると，Business Service Management から Diagnostics UI にアクセスできるようになります。

Business Service Management で［**アプリケーション**］＞［**Diagnostics**］を選択して，Diagnostics UI を開きます。

Business Service Management と Diagnostics が統合されている場合は，Diagnostics のデータが Business Service Management に送信されます。

アプリケーション・インフラストラクチャの要素およびビジネス・トランザクションに関するこの Diagnostics データは，Business Service Management 内のさまざまなビューで表示できます。また，Diagnostics と Business Service Management は一般的なデータ・モデルを共有しているため，Business Service Management からコンテキストをドリルダウンして，Diagnostics を直接選択できます。

Business Service Management で選択した CI が Diagnostics によって作成されている場合は，右クリックして［**Diagnostics へドリルダウンする**］を選択できます。Business Service Management のさまざまなレポートでは，アイコンを選択することにより，コンテキストをドリルダウンして Diagnostics を表示できます。

Business Service Management に送信されるデータ・サンプル

Diagnostics と Business Service Management を統合すると，Diagnostics はエンタープライズ・アプリケーションを監視し，アプリケーションのパフォーマンスと可用性データを**データ・サンプル**として Business Service Management に送信します。詳細については，『HP Diagnostics ユーザー・ガイド』の統合に関するセクションを参照してください。

Diagnostics は次のデータ・サンプルを Business Service Management に提供します。

- ws_perf_aggr_t（SOA サンプル）
- ws_event_aggr_t（SOA サンプル）
- appmon_vu_t（Transaction（BPM）サンプル）
- dg_trans_t（ビジネス・トランザクション（Diagnostics）サンプル）

データ・サンプルの詳細については，HP Business Service Management 文書ライブラリを参照してください。

Business Service Management での Diagnostics の CI 作成およびモデル化

Diagnostics 9.0 以降の場合，Diagnostics はアプリケーション・インフラストラクチャの要素とビジネス・トランザクションに対して，Business Service Management Run-time Service Model（RTSM）内に CI とモデルの関係を作成します。詳細については，『HP Diagnostics ユーザー・ガイド』の統合に関するセクションを参照してください。

ASP.NET アプリケーションで既存の ASP.NET アプリケーション・デプロイメントを変更する場合に，Run-time Service Model 作成や再スキャンの実行に必要となる IIS メタデータを検出する方法の詳細については，282 ページ「IIS によって配備された ASP.NET アプリケーションの CI 作成用 IIS メタデータの検出」を参照してください。

まれなケースですが，これらの CI を Run-time Service Model に追加するプロセスのタイミングを変更する場合は，いくつかのプロパティが Diagnostics の **server.properties** ファイルに指定されています。

Diagnostics と Business Service Management 間の CI の同期

Diagnostics によって作成されたモデル（CI）に対して Diagnostics と Business Service Management を強制的に同期させる場合は，**synchronize** 関数を Diagnostics サーバで使用できます。

メイン Diagnostics UI で，［**Diagnostics の設定**］を選択すると（または任意の Diagnostics ビューの右上角にある［**メンテナンス**］リンクを選択すると），［コンポーネント］ページが表示されます。［**synchronize**］リンクを選択して，モデルを同期化するページを表示します。

Business Service Management システムをアップグレードまたは再インストールしたときは，Diagnostics の CI を Business Service Management に転送する前に，厳密な同期を手動で行う（または 12 時間待機する）必要があります。厳密な同期を実行するには，［**ハード**］を選択します。

Diagnostics による Business Service Management の KPI / HI の色分け

ビジネス・トランザクションおよび Diagnostics で作成された Web サービス CI の状況インジケータ・ステータス（色分け）は，**測定値ベース**です。測定値ベースの KPI および状況インジケータのステータスは，データ・サンプルに格納されて Diagnostics から Business Service Management に送信されます。Diagnostics はデータ・サンプルを Business Service Management に送信し，Business Service Management のルールを使用してこれらのデータを評価し，インジケータのステータスを設定します。

ビジネス・トランザクションおよび Web サービス状況インジケータの標準設定の目標は，Business Service Management の［**管理**］＞［**サービス状況**］で変更できます。サービス状況管理の使用法については，『Business Service Management 文書ライブラリ』を参照してください。

Diagnostics で作成されたアプリケーション・インフラストラクチャ CI の状況インジケータ・ステータス（色分け）は，**イベントベース**です。関連するメトリックスに対するしきい値違反がある場合は，イベントベース状況インジケータのステータスが Diagnostics から Business Service Management に送信されます。

しきい値違反イベント・データは，Diagnostics コマンダ・サーバがインストールされた OM エージェントおよび IAPA コンポーネントによって，Business Service Management に送信されます。

OM エージェント / IAPA コンポーネントのインストールの詳細については，第 2 章，「Diagnostics サーバのインストール」を参照してください。イベント・チャネル・ステータスの確認方法の詳細については，696 ページ「Business Service Management での Diagnostics サーバの登録」を参照してください。問題がある場合は，872 ページ「イベント・ベース状況インジケータ・ステータスのトラブルシューティング・フロー」を参照してください。

KPI と HI ステータスの色分け，および Diagnostics アプリケーションおよびインフラストラクチャ・パフォーマンス・データが入力された Business Service Management ビューの概要については，『HP Diagnostics ユーザー・ガイド』の統合に関するセクションを参照してください。

Diagnostics と BSM の Service Health Analyzer の統合の有効化

Diagnostics と BSM の Service Health Analyzer（SHA）間の統合を有効にできます。この統合では，ホスト・メトリックスとプローブ・メトリックスを含むサンプルが，Diagnostics メディエータ・サーバから Diagnostics コマンダ・サーバを中継して BSM に送信され，BSM SHA データベースにメトリックスが追加されます。

SHA アプリケーションは，これらの Diagnostics プローブ・メトリックスとホスト・メトリックスに加えて，ほかのサンプルのメトリックスも使用してベースラインを作成します。SHA アプリケーションはメトリックスをベースラインと比較し，パフォーマンスの問題が検出されたときに異常を報告します。異常では, 詳しい Diagnostic データを確認するために Diagnostics プローブ・ビューまたはホスト・ビューにドリルダウンできます（SHA の使用方法の詳細については，BSM の Service Health Analyzer ドキュメントを参照してください）。

Diagnostics と SHA の統合は，標準設定では無効となっています。

統合を有効にするには，次の手順を実行します。

1 各 Diagnostics Mediator で，**/etc/server.properties** ファイルを見つけます。

2 次の変更を加えます。

```
# Send host metrics for Service Health Analyzer (SHA)
bac.diag.sha.host.metric.create.samples=true
# Send probe metrics for Service Health Analyzer (SHA)
bac.diag.sha.probe.metric.create.samples=true
```

3 各 Diagnostics Mediator を再起動します。

4 統合が有効にされたら，Diagnostics のホスト・メトリックスとプローブ・メトリックスを Business Service Management の SHA データベースで利用できます。次に，異常の検出に使用するため，SHA の管理アプリケーションでこれらの CI を選択します。

Diagnostics でフィルタを定義して, SHA のデータベースに送信されたホスト・メトリックスとプローブ・メトリックスを決定することもできます。これらのメトリックスをフィルタするには，Diagnostics サーバの **/etc** ディレクトリにある次の XML ファイルを使用します。フィルタは，データのエクスポートに似た正規表現の一致に基づきます。

- **shaHostMetrics.xml :** ホスト・メトリックスのフィルタを含める / 除外する
- **shaProbeMetrics.xml :** プローブ・メトリックスのフィルタを含める / 除外する

Diagnostics および OM サーバの共存

Diagnostics 9.x は，状況インジケータ（HI）の更新イベントを BSM 9.x に送信するためにバンドルされている OM エージェントを使用します。Diagnostics コマンダ・サーバで使用するシステムに OM エージェントがすでに含まれていて，OM サーバおよび BSM サーバへのレポートを設定する場合に必要な変更の手順を次に示します。

信頼済み証明書の設定

複数の BSM/OM サーバがある環境では，各サーバにほかのサーバが発行した証明書を信頼するよう設定する必要があります。このタスクでは，すべてのサーバの信頼済み証明書をエクスポートし，その信頼済み証明書をほかのすべてのサーバにインポートします。また，エージェントが BSM/OM サーバを信頼できるようにするため，エージェントの信頼済み証明書も更新する必要があります。

すべての BSM/OM に信頼済み証明書を設定するには，次の手順を実行します。

1 すべての BSM/OM サーバで，次のコマンドを使用して信頼済み証明書をファイルにエクスポートします。

ovcert -exporttrusted -file ＜ファイル名＞

このコマンドで，指定した名前のファイルが生成されます。

2 各ファイルをほかのすべてのサーバにコピーし，次のコマンドを使用して信頼済み証明書をインポートします。

ovcert -importtrusted -file ＜ファイル名＞

ovcert -importtrusted -ovrg server -file ＜ファイル名＞

3 Diagnostics システムで（エージェントがすでにインストールされている場合），次のコマンドを使用して信頼済み証明書を更新します。

ovcert -updatetrusted

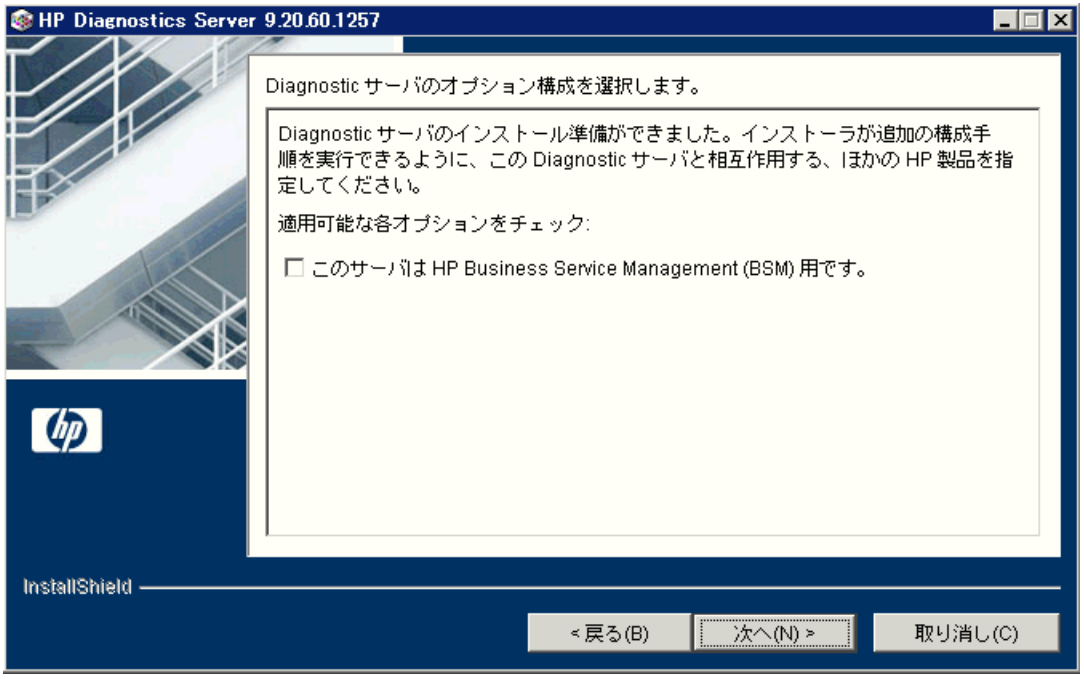
Diagnostics のインストール前にインストールされた OM エージェント

このシナリオは，OM エージェントがすでに存在していて，Diagnostics コマンダ・サーバをインストールするシステムですでに OM エージェントが OM サーバにレポートしていることを前提としています。

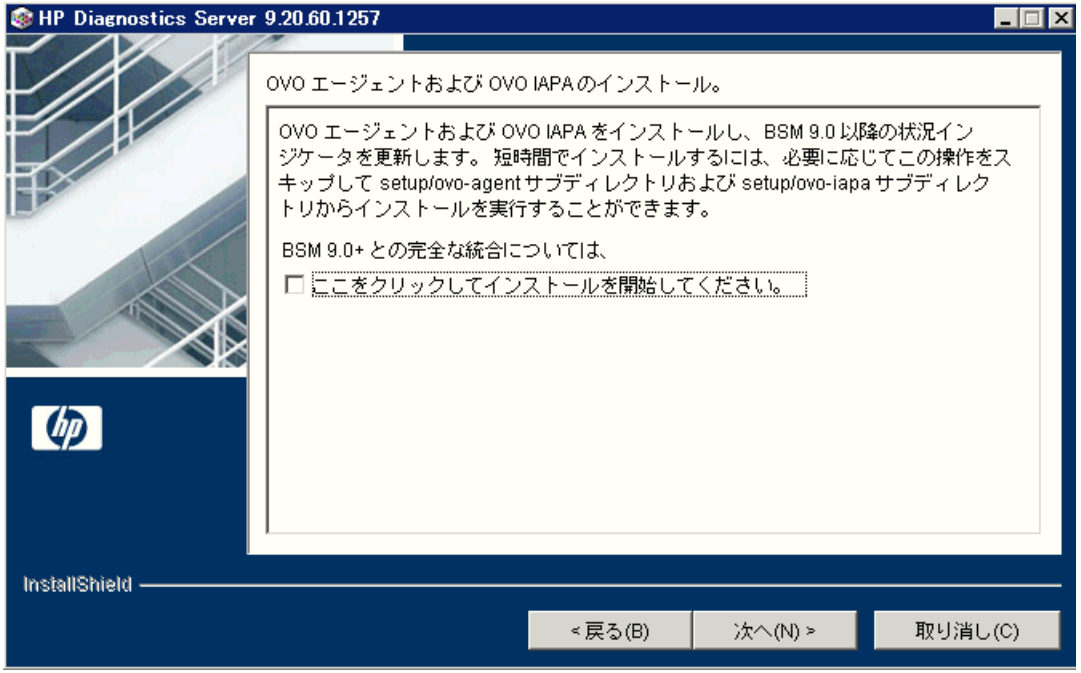
OM エージェントが先にインストールされている場合に共存をセットアップするには，次の手順を実行します。

1 Diagnostics コマンダ・サーバをインストールします（51 ページ「Diagnostics サーバのインストール」を参照）。

ここで，BSM の統合チェック・ボックスを選択します。



インストールの次の画面ではチェック・ボックスは選択しません。



2 Diagnostic コマンダ・サーバで，IAPA コンポーネントをインストールします（手順については，72 ページ「OM エージェントおよび IAPA コンポーネントの手動インストール」を参照）。

3 BSM の Diagnostics の［管理］ページで，BSM に Diagnostics を登録します（693 ページ「Business Service Management と Diagnostics 間の統合のセットアップ」を参照）。Diagnostics コマンダ・サーバが Linux にあり，Diagnostics サーバがルートとして実行されていない場合，BSM での Diagnostics の登録中にエラーが発生します。このエラーは無視して続行してください。

4 Diagnostics コマンダ・サーバで，**＜Diagnostics サーバのインストール・ディレクトリ＞¥bin** に移動して，**switch_ovo_agent.vbs**（UNIX の場合は **switch_ovo_agent.sh**）を実行し，-server と -cert_srv のターゲットとして OM サーバを指定します。Linux では，このコマンドをルートとして実行する必要があります。

次に例を示します。

cscript switch_ovo_agent.vbs -server ovruxt65.rose.hp.com -cert_srv ovruxt65.rose.hp.com

5 BSM ゲートウェイ・サーバと OM サーバのコア ID を特定します。Diagnostics Commander で次のコマンドを実行します。

bbcutil -ping ＜OM サーバ＞

bbcutil -ping ＜BSM ゲートウェイ・サーバ＞

6 **＜Diagnostics サーバのインストール・ディレクトリ＞¥newconfig¥ovo-agent¥policies¥mgrconf** ディレクトリを**＜Diagnostics サーバのインストール・ディレクトリ＞¥newconfig¥ovo-agent¥policies¥tmp** にコピーします。

mgrconf ディレクトリが存在しない場合は，サポートに連絡してこのディレクトリのコンテンツを入手してください。セットアップがより複雑な場合（複数の OM マネージャがある場合など），次のファイルへの追加変更が必要な場合があります。

7 **＜Diagnostics サーバのインストール・ディレクトリ＞¥newconfig¥ovo-agent¥policies¥tmp¥mgrconf¥FF9A8F04-B5E3-43C3-999B-7A9492C35014_data** ファイルを編集します。

- 文字列 ${OM_MGR_SRV} を検索して，そのすべてを HPOM 管理サーバの FQDN に置き換えます。
- 文字列 ${OM_MGR_SRV_ID} を検索して，そのすべてを HPOM 管理サーバのコア ID に置き換えます。
- 文字列 ${OMi_MGR_SRV} を検索して，そのすべてを BSM ゲートウェイ・サーバの FQDN に置き換えます。
- 文字列 ${OMi_MGR_SRV_ID} を検索して，そのすべてを BSM ゲートウェイ・サーバのコア ID に置き換えます。

OM セットアップがより複雑な場合，このファイルへのエントリの追加が必要な場合があります。

8 **＜Diagnostics サーバのインストール・ディレクトリ＞¥newconfig¥ovo-agent¥policies¥tmp** ディレクトリに移動して，ポリシーをインストールします。

```
ovpolicy -install -dir mgrconf
```

9 エージェントに付属された Diagnostics 固有のログファイル・カプセル化テンプレートは BSM サーバにレポートされるようになり，ほかのすべてのポリシーは OM サーバにレポートされます。

すでにインストール済みの Diagnostics

最初に OM エージェントをアンインストールしてから上記の手順を実行することをお勧めします。

Diagnostics 9.x は，状況インジケータ（HI）の更新イベントを BSM 9.x に送信するためにバンドルされている OM エージェントを使用します。Diagnostics, および Diagnostics に付属された OM エージェントと IAPA コンポーネントがすでにインストール済みで, OM サーバおよび BSM サーバに送信されるイベントを設定する場合に必要な変更の手順を次に示します。

Diagnostics がすでにインストールされている場合に共存をセットアップするには，次の手順を実行します。

1 Diagnostics でインストールした OM エージェント・コンポーネントをアンインストールします（手順については，74 ページ「OM エージェントおよび IAPA コンポーネントの手動アンインスト ール」を参照)。

2 OM サーバに移動し，OM エージェントを Diagnostics コマンダ・サーバにデプロイします。

3 前のページの手順 2 ～ 9 を実行します。

DPS とゲートウェイに対する個別の BSM サーバの設定

Diagnostics 9.0 以降と Business Service Management 9.0 以降を統合している場合，OMi イベント・チャネルは，状況インジケータ・ステータス・イベントを Diagnostics から Business Service Management ゲートウェイ・サーバに送信するために Diagnostics で使用されます（実際は OM エージェントおよび IAPA コンポーネントが使用されます)。

Diagnostics をセットアップするには，OMi 通信チャネルが必要です。BSM でイベント・チャネル統合ステータスを確認する方法については，699 ページ「Business Service Management の詳細」を参照してください。

BSM ゲートウェイ・サーバと BSM データ処理サーバは，個別のシステムにセットアップできます（このセットアップ方法については，BSM のドキュメントを参照してください)。これらのサーバが異なるシステム上で実行されていてイベント・チャネル統合が設定されている場合，BSM ゲートウェイ・サーバと BSM 処理サーバ間の通信では，マシン間に信頼関係が確立されている必要があります。

別のゲートウェイ・サーバに資格情報を設定するには，次の手順を実行します。

1 BSM ゲートウェイ・サーバで，次のコマンドを実行します。

ovconfchg -ns sec.cm.client -set CERTIFICATE_SERVER ＜処理サーバ＞

および

ovcert -certreq

2 BSM 処理サーバで，次のコマンドを実行します。

ovcm -listpending -l

および

ovcm -grant ＜要求 ID＞

3 BSM ゲートウェイ・サーバで，次のコマンドを実行します。

ovcert -list

および

bbcutil -ping ＜処理サーバ＞

詳細については，Business Service Management のオンライン・ヘルプを参照してください。

統合に関する追加情報

Diagnostics と Business Service Management 間の統合のセットアップに関する追加情報を次に示します。

認証ダイアログの表示

Diagnostics サーバが Business Service Management サーバとは異なるドメインにインストールされている場合，Lightweight Single Sign-On（LWSSO）が信頼される側のドメインとして Diagnostics サーバ・ドメインを追加するようにセットアップされていないと，Diagnostics ダッシュボード・アプレットが表示される前に MyBSM Diagnostics ダッシュボードに認証ダイアログが表示される可能性があります。

この問題を解決するには，Diagnostics サーバが実行されているドメインが Business Service Management の Single Sign-On ページに表示されていることを確認します。

1 Business Service Management で，［**管理**］>［**プラットフォーム**］>［**ユーザおよび権限**］>［**認証管理**］>［**Single Sign-On 設定**］を選択します。

2 ［**設定**］ボタンをクリックします。

3 ウィザードで［**次へ**］をクリックして，Single Sign-On ページに移動します。

4 ［**信頼されたホスト / ドメインを追加します**］アイコンをクリックして，Diagnostics サーバのドメインを入力します。

5 ［**次へ**］をクリックします。

6 ［**次へ**］をクリックします。

7 ［**終了**］をクリックします。Business Service Management からログアウトされます。再度 Business Service Management にログインして，MyBSM Diagnostics ダッシュボードを開きます。

Diagnostics UI の欠落しているリンク

Diagnostics UI が Business Service Management から起動されていて，同時に同じシステムでスタンドアロン・モードの Diagnostics UI が起動されている場合，スタンドアロン・モードの Diagnostics UI では［メンテナンス］リンクが利用できません。

この問題を解決するには，Diagnostics UI の両方のインスタンスを閉じて，スタンドアロンの Diagnostics UI を再起動します。

HI イベントが Business Service Management に送信されない

Business Service Management 9.x と統合された Diagnostics は，Business Service Management ゲートウェイ・サーバに状況インジケータ・ステータス・イベントを送信します。Business Service Management への HI イベントの送信に問題がある場合は，付録 H,「HP Diagnostics のトラブルシューティング」の項の「OM エージェント」,「BSM ゲートウェイの信頼関係」および「イベント・ベース状況インジケータ・ステータスのトラブルシューティング・フロー」を参照してください。

第 VIII 部

Diagnostics サーバおよび Java と .NET Agent の詳細設定

本項の内容

- Diagnostics サーバの詳細設定
- Java Agent およびアプリケーション・サーバの詳細構成
- .NET Agent 設定ファイルについて
- .NET Agent の詳細設定

22

LoadRunner Diagnostics Add-in のインストール

LoadRunner Diagnostics Add-in によって, LoadRunner 内部から Diagnostics UI にアクセスできるようになります。LoadRunner Diagnostics Add-in をインストールすると，LoadRunner を設定して，負荷テストの際に Diagnostics コンポーネントを使ってパフォーマンス・メトリックスを収集したり，Diagnostics UI に接続したりできるようになります。

本章の内容

LoadRunner Diagnostics Add-in のインストールの前に

LoadRunner Diagnostics Add-in をインストールする前に, Diagnostics コマンダ・サーバと LoadRunner をインストールする必要があります。LoadRunner のインストールについては,『HP LoadRunner インストール・ガイド』を参照してください。

Diagnostics バージョン 8.0x，9.0x，9.10 以降では，最新の Diagnostics 9.10 LoadRunner アドインを使用する必要があります。

LoadRunner Diagnostics Add-in のインストール

LoadRunner Diagnostics Add-in は，LoadRunner Controller のホスト・マシンにインストールします。

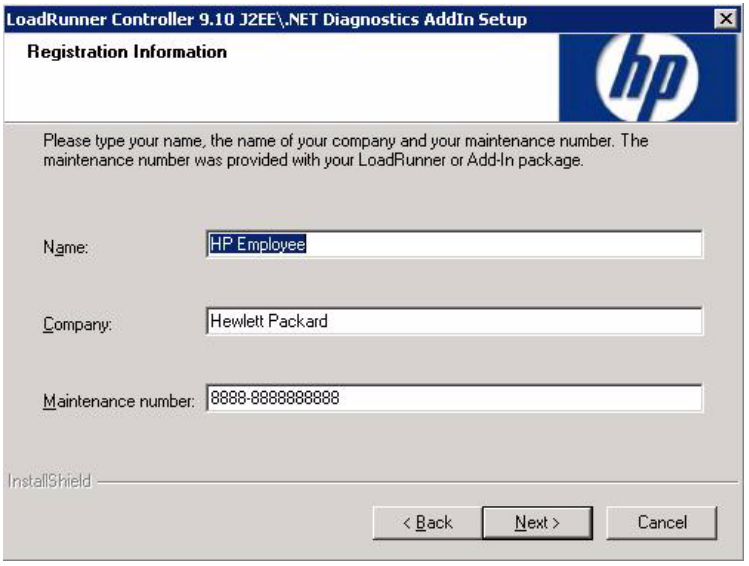
注 : LoadRunner Diagnostics Add-in は，最初の実行時に，小さいブートストラップ・プログラムを使って Diagnostics サーバから必要なソフトのほとんどを動的にダウンロードするようになっています。Diagnostics を更新すると，次の LoadRunner の実行時に新しい Diagnostics ファイルが自動的に選択されます。

LoadRunner Diagnostics Add-in をインストールするには，次の手順を実行します。

1. LoadRunner Controller システムで実行中の LoadRunner または LoadRunner 関連プロセス（LoadRunner Agent など）を閉じます。
2. Diagnostics インストール・ディスクの LR_AddIn ディレクトリにある **setup.exe** ファイルを実行します。setup インストール・プログラムが起動します。

3 ソフトウェア使用許諾契約書が表示されます。契約書を読み，［**Yes**］をクリックして同意します。

［Registration Information］ダイアログ・ボックスが開きます。



［Registration Information］ダイアログ・ボックスに，名前，会社名，お手元の LoadRunner メンテナンス番号を入力します。メンテナンス番号は，LoadRunner に同梱のメンテナンス・パックに記載されています。

［**Next**］を選択してインストール・プロセスを開始します。インストール・プロセスが始まります。

4 インストール・プロセスが完了すると，インストール・ウィザードに確認メッセージが表示されます。

インストール・プロセスを終了するには，［**Finish**］をクリックしてください。

コンピュータで LoadRunner に関連するプロセスが実行中の場合（LoadRunner Agent など），コンピュータを再起動して LoadRunner Add-in インストール・プロセスを完了する必要があります。

注 : LoadRunner Diagnostics Add-in 用のアンインストール・ユーティリティはありません。

注 : サービス・パック 1 と Windows XP Hotfix Q328310 を適用した Windows XP マシンに LoadRunner Diagnostics Add-in をインストールしている場合は，**iKernal.exe** のアプリケーション・エラー・メッセージが表示されます。このメッセージは，Windows XP Hotfix Q328310 に，InstallShield エンジンの予想どおりに実行しない Win32 API が含まれていることが原因で表示されます。この問題を解決するには，Java Technology Help Web サイト http://java.com/en/download/help/ikernel.jsp（英語サイト）で推奨されている解決方法を参照してください。

LoadRunner から Diagnostics UI にアクセスするには，LoadRunner を設定して，Diagnostics コンポーネントとの通信を有効にするのに必要な情報を指定する必要があります。LoadRunner を Diagnostics と統合するための設定については，第 23 章，「HP LoadRunner と HP Diagnostics の統合のセットアップ」を参照してください。

23

HP LoadRunner と HP Diagnostics の統合のセットアップ

負荷テストの実行とオフライン分析に関して HP LoadRunner と HP Diagnostics を統合するためのセットアップについて，一般的な情報を示します。

本章の内容

LoadRunner で HP Diagnostics を使用する方法

LoadRunner 診断モジュールと HP Diagnostics との統合により，LoadRunner で詳細なパフォーマンス情報が得られ，Siebel，Oracle，SAP，J2EE，.NET 環境のパフォーマンスの問題を迅速に特定できます。

LoadRunner では，HP Diagnostics によって **J2EE/.NET 診断**機能が提供され，J2EE および .NET アプリケーションのテスト環境の複雑なパフォーマンスの問題を監視，分析，解決できます。

LoadRunner と HP Diagnostics の統合をセットアップしたら，さまざまな方法で HP Diagnostics のデータを LoadRunner で表示できます。

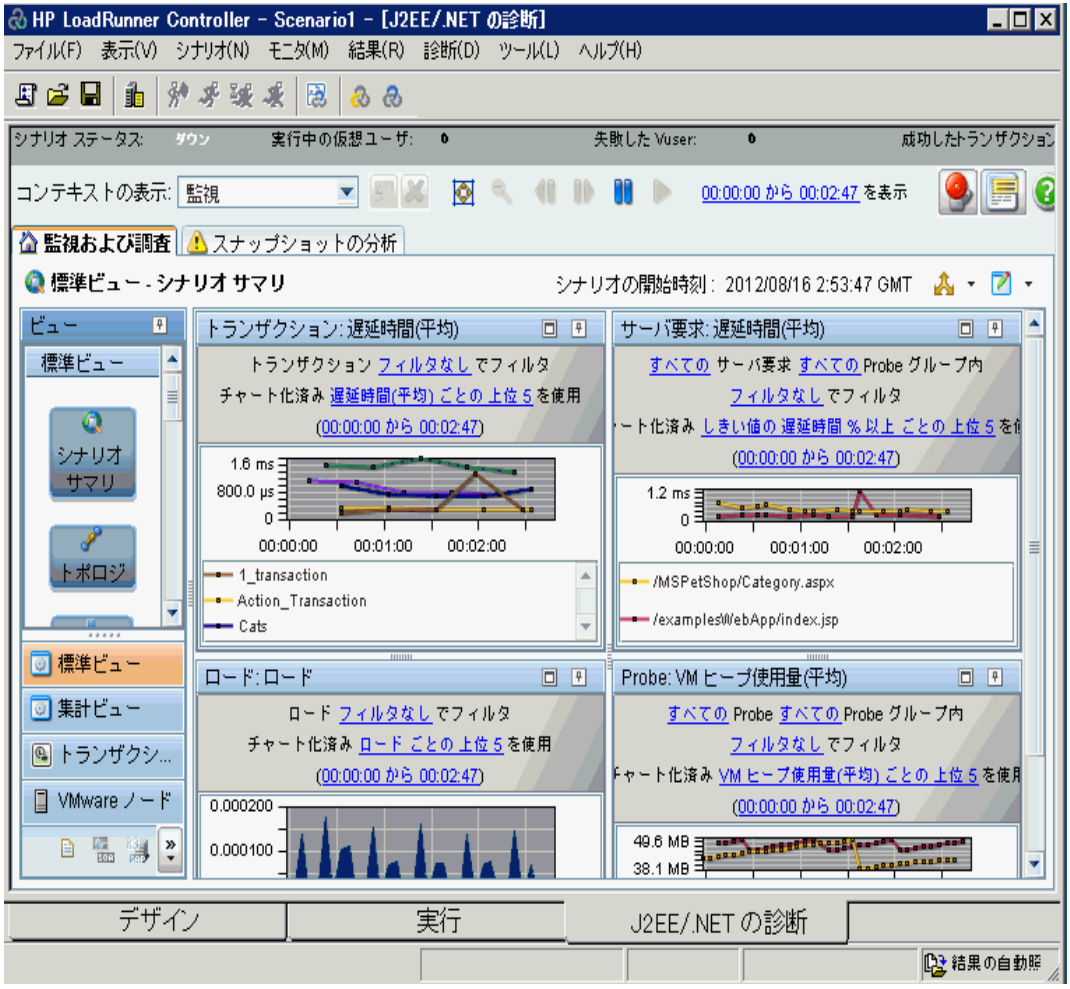
- LoadRunner シナリオの Diagnostics UI ビュー
- トランザクションの詳細を得るための Diagnostics UI へのドリルダウン
- LoadRunner Analysis の J2EE および .NET 診断グラフ

LoadRunner シナリオの Diagnostics UI ビュー

LoadRunner の負荷テスト・シナリオでは，HP Diagnostics UI を開いて，シナリオ全体の詳細なパフォーマンス・データを取得できます。

LoadRunner のシナリオ・ウィンドウの下部にある［J2EE/.NET 診断］タブを選択すると，HP Diagnostics UI が開き，シナリオの詳細な診断データが表示されます。HP Diagnostics UI では，ほかのビューに移動して実行時に検出されたパフォーマンスの問題を特定，隔離，分析，解決できます。

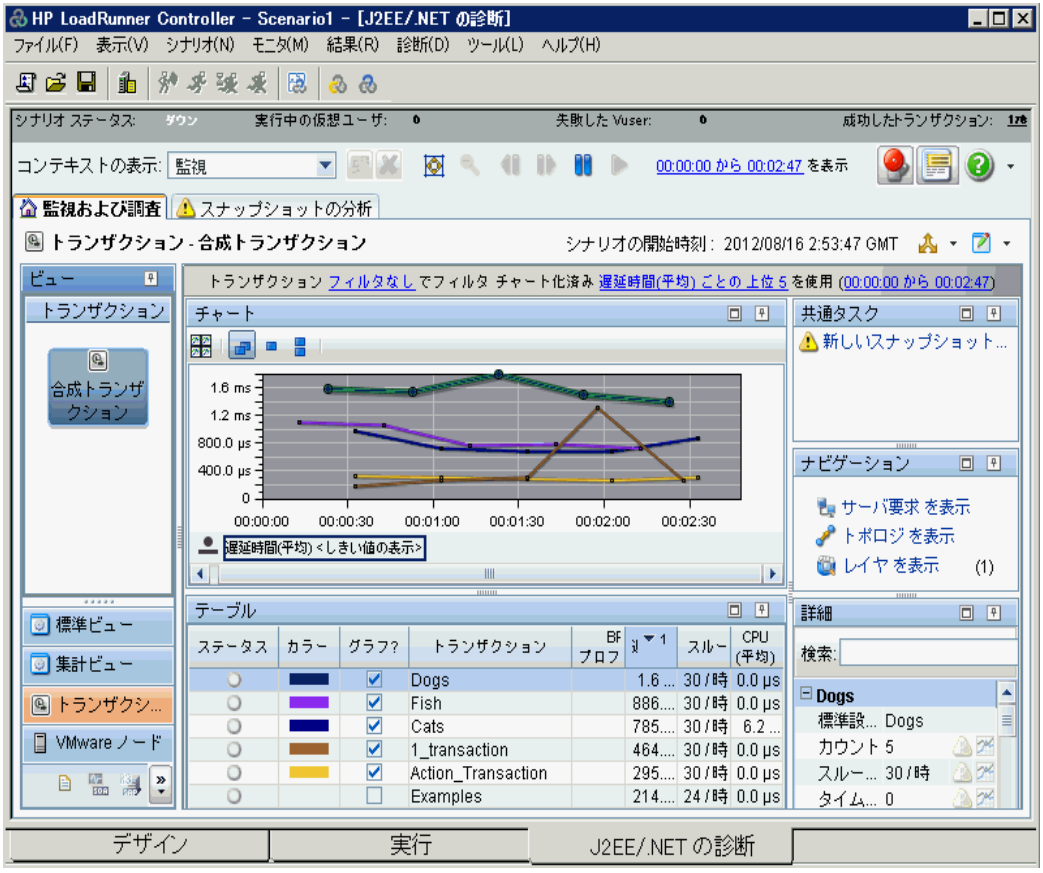
次に，HP Diagnostics UI の例を示します。



トランザクションの詳細を得るための Diagnostics UI へのドリルダウン

LoadRunner では，特定のトランザクションの HP Diagnostics UI にドリルダウンして，そのトランザクションの診断データを取得できます。

LoadRunner で，いずれかのトランザクションのグラフ（トランザクション応答時間など）を選択し，そのグラフから HP Diagnostics UI にドリルダウンすると，HP Diagnostics UI が開いて［トランザクション］ビューが表示されます。ここから，HP Diagnostics UI のほかのビューに移動して，トランザクションに関連する問題をトラブルシューティングできます。



LoadRunner Analysis の J2EE および .NET 診断グラフ

LoadRunner Analysis の **J2EE および .NET 診断グラフ**は，HP Diagnostics から提供されるデータに基づいています。これらのグラフを使用して，J2EE および .NET Web サーバ，アプリケーション・サーバ，データベース・サーバを介した個々のトランザクションやサーバ要求をトレース，時間計測，トラブルシューティングできます。また，サーブレットや JDBC の呼び出しの問題をすばやく特定でき，ビジネス・プロセスのパフォーマンス，スケーラビリティ，効率性を最大化できます。

LoadRunner の J2EE および .NET 診断グラフは，J2EE および .NET 診断グラフと J2EE および .NET サーバ診断グラフの 2 つのグループで構成されています。

LoadRunner で HP Diagnostics のデータを表示する方法については，『HP LoadRunner Controller ユーザーズ・ガイド』および『HP LoadRunner Analysis ユーザーズ・ガイド』を参照してください。

HP Diagnostics と統合するための LoadRunner のセットアップについて

LoadRunner から Diagnostics UI にアクセスするには，あらかじめ LoadRunner が HP Diagnostics コンポーネントと通信するのに必要な情報を LoadRunner に指定する必要があります。

LoadRunner から Diagnostics UI にアクセスできるようにするには，Diagnostics サーバとの統合を設定する必要があります。Diagnostics サーバの詳細については，初めて Diagnostics で LoadRunner を使用するときだけ指定する必要があります。Diagnostics サーバにアクセスできるように LoadRunner を設定する方法の詳細については，『HP LoadRunner Controller ユーザーズ・ガイド』を参照してください。

注 : Diagnostics サーバの詳細を指定する前に，LoadRunner Controller が閉じていることを確認してください。Controller が開いていると，Diagnostics の設定を表示することはできますが，変更できません。

HP Diagnostics を使用するための LoadRunner シナリオの設定

負荷テスト・シナリオで Diagnostics メトリックスをキャプチャするたびに，シナリオに Diagnostics パラメータを設定し，シナリオに含まれるプローブを選択する必要があります。LoadRunner Controller から Diagnostics 用のシナリオを設定します。詳細については，『HP LoadRunner Controller ユーザーズ・ガイド』を参照してください。

注 : 設定済みの Diagnostics 設定でシナリオを保存した場合は，このシナリオを実行するたびに Diagnostics パラメータを再設定する必要はありません。

オフライン分析ファイルに追加するプローブ・メトリックスの選択

LoadRunner オフライン分析に使用する Diagnostics プローブ・メトリックス・データを追加できます。

標準設定では，HeapUsed，GC コレクション / 秒，および コレクションに費やされた GC 時間のメトリックス・データのみがオフライン分析に追加されます。オフライン分析（.eve ファイル）に追加するプローブ・メトリックスを選択するには，Diagnostics 設定ファイル **etc/offline.xml** を使用します。

Diagnostics Mediator の offline.xml ファイルを設定して，オフライン分析に追加する Diagnostics プローブ・メトリックスを指定します。このファイルでは，実行に参加するプローブを保持しているすべての Mediator を設定します。

プローブ・メトリックスを使用できるのは，Java プローブのみです。.NET プローブでは使用できません。

注 : Diagnostic サーバと LoadRunner でクロックが同期していることを確認してください。

有効なすべての Diagnostics メトリックスのリストは，Diagnostics プローブ・インストール・ディレクトリの **metrics.config** ファイルに記述されています。このファイルにはすべてのメトリックスが含まれていますが，プラットフォーム（Java プローブのみ），アプリケーション・サーバの種類，監視対象のバージョンによっては，一部の測定値をオフラインイン分析に使用できないことがあります。

一般に，Diagnostics UI の［詳細］表示枠の Java プローブの下に表示されているすべてのメトリックスは，関連する Mediator の offline.xml ファイルで使用できます。**metrics.config** ファイルに含まれている「system」および「Mercury System」Collector は，offline.xml ファイルで使用できません。

次に，**offline.xml** ファイルの例を示します。

```
<?xml version="1.0" encoding="UTF-8" ?>
<probeMetrics xmlns="http://hp.com/diagnostics/offline/1.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="offline.xsd">
<metric>
   <name>HeapUsed</name>
 </metric>
 <metric>
   <name>GC Collections/sec</name>
 </metric>
 <metric>
   <name>GC Time Spent in Collections</name>
 </metric>
</probeMetrics>
```

＜metric＞要素は一致条件を指定します。測定値の名前（＜name＞），カテゴリ（＜category＞），およびコレクタ（＜collector＞）に関する一致を使用できます。＜name＞，＜category＞，および＜collector＞要素は組み合わせることができます。組み合わせた場合は，すべての要素が一致する必要があります。

一致の例：

「HeapUsed」という名前の測定値を照合して，追加しします。

```
<metric>
 <name>HeapUsed</name>
</metric>
```

「JVM」コレクタが公開しているすべての測定値を照合して，追加します。

```
<metric>
  <collector>JVM</collector>
</metric>
```

「JVM」コレクタが公開している「HeapFree」という名前の測定値を追加して，照合します。

```
<metric>
  <name>HeapFree</name>
  <collector>JVM</collector>
</metric>
```

標準設定では，照合は部分文字列に対して実行されます。つまり，＜name＞，＜category＞，＜collector＞の間にある指定のテキストは，実際の測定値の名前，カテゴリ，またはコレクタの一部である（含まれている）必要があります。たとえば，＜name＞に「HeapFree」というテキストを指定すると，名前に「HeapFree」が含まれるすべての測定値が一致します（「MyHeapFreeMetric」，「YourHeapFreeMetric」など）。

さらに，＜metric＞の match 属性を使用して，＜name＞，＜category＞，および＜collector＞の正規表現を指定することもできます（＜metric match= "regex"＞など）。正規表現は効率が悪く，out .eve ファイルの記述時間に影響します。

次のように，LoadRunner にすべてのプローブ・メトリックスを送信します。

```
<metric match="regex">
  <collector>.*</collector>
  </metric>
```

注 : offline.xml に多数のメトリックスを追加すると，オフライン（.eve）ファイルのサイズが大きくなり，LoadRunner 分析アプリケーションでオフライン・ファイルを分析する場合の所要時間が影響を受けます。

15 秒おきに **offline.xml** ファイルの変更が自動的に検出されて，適用されます。設定エラー（**offline.xsd** に対して検証）は，**server.log** ファイルに記録されます。

大きなオフライン分析ファイルの転送の改善

LoadRunner または Performance Center のテスト中に Diagnostics によって生成されるオフライン分析ファイル（.eve）は，かなり大きくなる可能性があります。実行終了時には，相関や分析のためにこれらのファイルが Diagnostics サーバから LoadRunner/Performance Center コントローラに転送されます。Diagnostics のデータを含むオフライン分析ファイルの転送時間とロード時間を改善するには，.eve ファイルの精度を下げます。

Diagnostics サーバの **server.properties** ファイルにある **bucket.lr.offline.duration** プロパティと **bucket.lr.offline.sr.duration** プロパティを使って，集計期間を増やします（たとえば，5 秒から 15 秒へ）。これらのプロパティを使って，オフライン分析用のサンプルを 1 つ作成するのに 5 秒のトレンド・ポイントを何個集計するかを定義できます。

LoadRunner Controller の Diagnostics UI におけるメモリ不足の問題

LoadRunner UI ログ（Mercury_Diagnostics_UI.log）で OutOfMemory エラーが見つかった場合，その原因は LoadRunner Controller の Diagnostics タブにデータを表示する Diagnostics アプレットに起因するメモリ制限である可能性があります。

この問題を修正するには，LoadRunner システムの OS のシステム環境変数を APPCRITIC_MAX_MEM=256m および JAVA_TOOL_OPTIONS=-Xmx256m と定義することで，Diagnostics アプレットのヒープ・メモリを増加します（念のため，256MB の最大 VM ヒープも必要です)。

24

Diagnostics を使用するための Performance Center のセットアップ

HP Diagnostics を負荷テストで使用できるように Performance Center を設定する方法について，一般的な情報を示します。

本章の内容

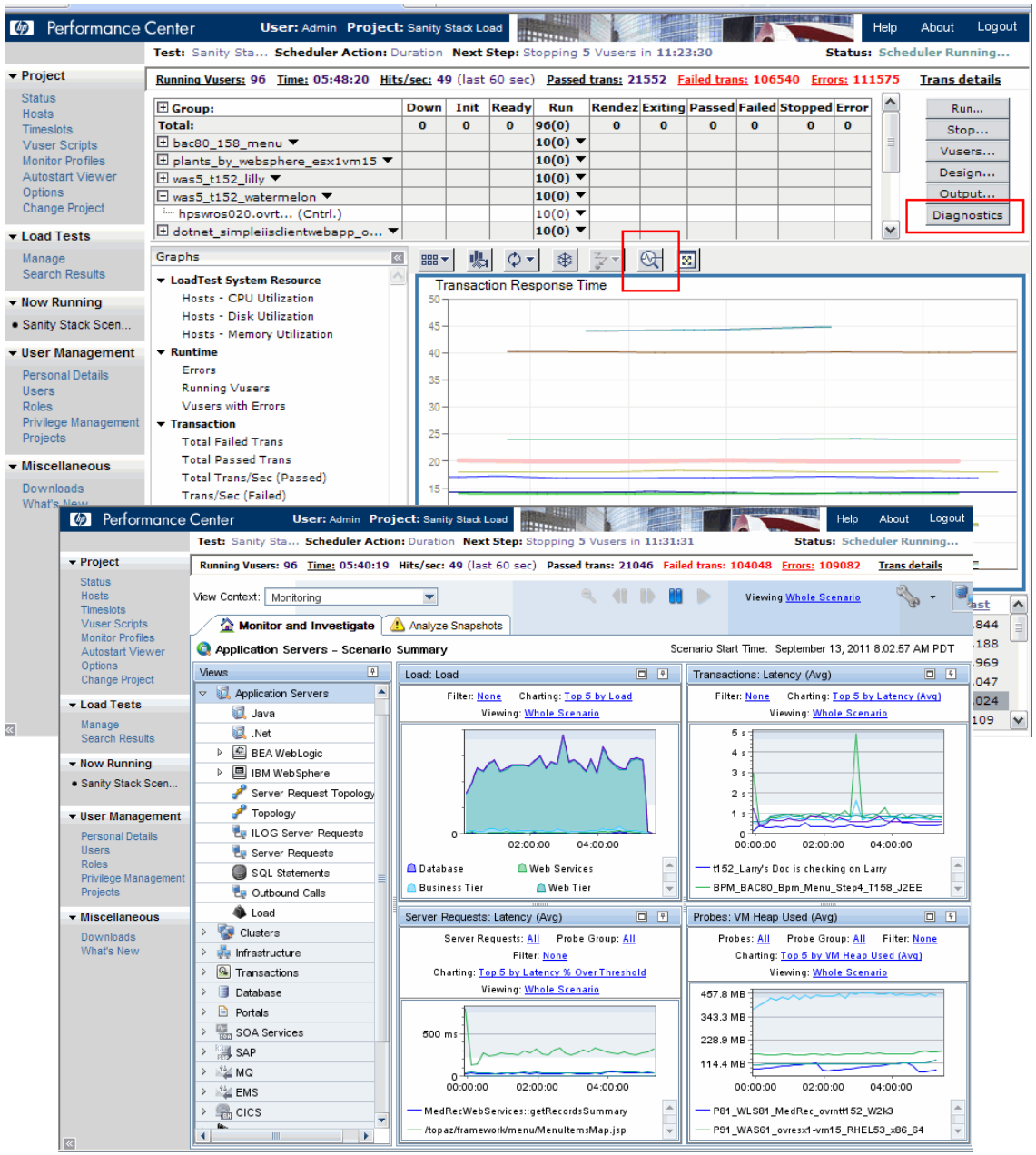
Performance Center で HP Diagnostics を使用する方法

Performance Center 診断モジュールと HP Diagnostics との統合により，Performance Center で詳細なパフォーマンス情報が得られ，Siebel，Oracle，SAP，J2EE，.NET 環境のパフォーマンスの問題を迅速に特定できます。

Performance Center では，HP Diagnostics によって **J2EE/.NET 診断**機能が提供され，J2EE および .NET アプリケーションのテスト環境の複雑なパフォーマンス問題を監視，分析，解決できます。

Performance Center と HP Diagnostics の統合をセットアップしたら，Performance Center から HP Diagnostics のデータを表示できます。

Performance Center の負荷テストを実行する場合，HP Diagnostics UI にドリルダウンして，負荷テスト全体または特定のトランザクションの詳細なパフォーマンス・データを取得できます。負荷テストの実行後に，HP LoadRunner Analysis を使用して，負荷テスト時に生成されたオフライン診断データを分析できます。


Performance Center
User: Admin Project: Sanity Stack Load
Help About Logout
Test: Sanity Sta... Scheduler Action: Duration Next Step: Stopping 5 Vusers in 11:23:30
Status: Scheduler Running...
Running Vusers: 96 Time: 05:48:20 Hits/sec: 49 (last 60 sec) Passed trans: 21552 Failed trans: 106540 Errors: 111575 Trans details
Group: Down Init Ready Run Rendez Exiting Passed Failed Stopped Error
Total: 0 0 0 96(0) 0 0 0 0 0 0
bac80_158_menu 10(0)
plants_by_websphere_esx1vm15 10(0)
was5_t152_lilly 10(0)
was5_t152_watermelon 10(0)
hpswros020.ovrt... (Cntrl.) 10(0)
dotnet_simpleiisclientwebapp_o... 10(0)
Run...
Stop...
Vusers...
Design...
Output...
Diagnostics
Project
Status
Hosts
Timeslots
Vuser Scripts
Monitor Profiles
Autostart Viewer
Options
Change Project
Load Tests
Manage
Search Results
Now Running
Sanity Stack Scen...
User Management
Personal Details
Users
Roles
Privilege Management
Projects
Miscellaneous
Downloads
What's New
Graphs
LoadTest System Resource
Hosts - CPU Utilization
Hosts - Disk Utilization
Hosts - Memory Utilization
Runtime
Errors
Running Vusers
Vusers with Errors
Transaction
Total Failed Trans
Total Passed Trans
Total Trans/Sec (Passed)
Trans/Sec (Failed)
Transaction Response Time
50
45
40
35
30
25
20
15
Performance Center
User: Admin Project: Sanity Stack Load
Help About Logout
Test: Sanity Sta... Scheduler Action: Duration Next Step: Stopping 5 Vusers in 11:31:31
Status: Scheduler Running...
Running Vusers: 96 Time: 05:40:19 Hits/sec: 49 (last 60 sec) Passed trans: 21046 Failed trans: 104048 Errors: 109082 Trans details
View Context: Monitoring
Viewing Whole Scenario
Monitor and Investigate
Analyze Snapshots
Application Servers - Scenario Summary
Scenario Start Time: September 13, 2011 8:02:57 AM PDT
Views
Application Servers
Java
.Net
BEA WebLogic
IBM WebSphere
Server Request Topology
Topology
ILOG Server Requests
Server Requests
SQL Statements
Outbound Calls
Load
Clusters
Infrastructure
Transactions
Database
Portals
SOA Services
SAP
MQ
EMS
CICS
Load: Load
Filter: None Charting: Top 5 by Load
Viewing: Whole Scenario
02:00:00
04:00:00
Database
Web Services
Business Tier
Web Tier
Transactions: Latency (Avg)
Filter: None Charting: Top 5 by Latency (Avg)
Viewing: Whole Scenario
00:00:00
02:00:00
04:00:00
t152_Larry's Doc is checking on Larry
BPM_BAC80_Bpm_Menu_Step4_T158_J2EE
Server Requests: Latency (Avg)
Server Requests: All Probe Group: All
Filter: None
Charting: Top 5 by Latency % Over Threshold
Viewing: Whole Scenario
500 ms
MedRecWebServices::getRecordsSummary
/topaz/framework/menu/MenuItemsMap.jsp
Probes: VM Heap Used (Avg)
Probes: All Probe Group: All Filter: None
Charting: Top 5 by VM Heap Used (Avg)
Viewing: Whole Scenario
457.8 MB
343.3 MB
228.9 MB
114.4 MB
P81_WLS81_MedRec_ovmtt152_W2k3
P91_WAS61_ovresx1-vm15_RHEL53_x86_64

Diagnostics を使用するための Performance Center のセットアップについて

Performance Center と Diagnostics は，連携して動作する，アプリケーションのパフォーマンスの把握と向上に役立つ情報を提供するように設計された統合製品です。

Performance Center から Diagnostics にアクセスできるようにするには，Performance Center で次の Diagnostics サーバの詳細情報を指定します。

- **サーバ名 :** Diagnostics コマンド・サーバのホスト・マシンの名前。
- **ポート番号 :** Diagnostics コマンド・サーバで使用するポート番号。デフォルトのポート番号は **2006** です。
- **ログイン名 :** HP Diagnostics にログオンするときに使うユーザ名。デフォルトのユーザ名は admin です。

  指定したユーザ名には，**表示**，**変更**および**実行**権限が必要です。ユーザ権限の詳細は，757 ページ「ユーザ権限について」を参照してください。
- **パスワード :** HP Diagnostics にログオンするときに使うパスワードを入力します。デフォルトのパスワードは admin です。
- **通信 :** Performance Center が Diagnostics サーバへのアクセスに使用する通信プロトコル。

  通信プロトコルに **HTTPS** を選択した場合，追加の設定手順が必要になります。必要な手順の詳細は，付録 C,「コンポーネント間での HTTPS 有効化」を参照してください。

注 : これらの情報は，Diagnostics で Performance Center を初めて使用するときだけ入力する必要があります。この情報は, Performance Center Performance Center 管理サイトの Diagnostics ページで入力します。

Diagnostics を使用するための Performance Center 負荷テストの設定

負荷テスト・シナリオで Diagnostics メトリックスをキャプチャするたびに，負荷テストに Diagnostics パラメータを設定し，負荷テストに含まれるプローブを選択する必要があります。

Performance Center を Diagnostics と統合するための設定方法の詳細については，『HP Performance Center ユーザーズ・ガイド』の HP Diagnostics と Performance Center の統合に関するセクションを参照してください。

Performance Center Controller と負荷テストに関係のある Diagnostics サーバの間にファイアウォールがある場合，Controller と Diagnostics サーバを設定し，MI Listener ユーティリティを使ってオフライン分析ファイルの転送を有効にする必要があります。MI Listener マシンの IP アドレスも Performance Center に指定する必要があります。

また，Diagnostics サーバ（Mediator モード）をファイアウォール経由で動作するように設定する必要があります。詳細については，631 ページ「ファイアウォール環境で動作するための Diagnostics の設定」を参照してください。

この統合で「サーバ要求をモニタする」機能を有効にする利点は，次の場合でもバックエンド VM の呼び出しをキャプチャできることです。

- プローブが RMI 呼び出しをキャプチャしていない。
- RMI 呼び出しをキャプチャできない（サポートされていないアプリケーション・コンテナが使われている場合など）。
- アプリケーションで，複数の VM 間の通信に別のメカニズムを使用している。

注 : サーバ要求を監視するように統合を設定すると，この機能によってプローブ上のオーバーヘッドが増加します。

Diagnostics コンポーネント間の接続の問題を調べるには，Performance Center からアクセスできるシステムの状況モニタを使用します。

Performance Center オフライン・ファイルの管理

HP Performance Center オフライン・ファイルはデフォルトで保存されます。オフライン・ファイルを管理するには，これらのファイルを削除できるように Diagnostics サーバ（Mediator モード）を設定する必要があります。

これを行うには，**＜Diagnostics サーバのインストール・ディレクトリ＞/etc/server.properties** でプロパティ **distributor.offlinedelivery.preserveFiles** を true に設定します。このプロパティを true に設定すると，サーバのデータ・ディレクトリに格納されている実行時固有の「オフライン」ファイルは，サーバの **webserver.properties** ファイルで指定した **facade.run_delete_delay** プロパティの期間保存されます（標準設定は 5 日間）。

この保存期間中，実行を正常に照合できます。保持期間終了後しばらくして，関連するオフライン・ファイルがシステムから削除されます。

第 IX 部

付録

本項の内容

- Diagnostics 管理 UI
- ユーザの認証と承認
- コンポーネント間での HTTPS 有効化
- 管理者用のシステム・ビューの使用
- Diagnostics のデータ管理
- Diagnostics の技術的な図
- アップグレードとパッチ・インストールの手順
- HP Diagnostics のトラブルシューティング
- 全般的な参照情報
- データのエクスポート

A

Diagnostics 管理 UI

Diagnostics プロパティの設定や Diagnostics ソフトウェアの管理を行うために，Diagnostics サーバの管理 UI にアクセスして使用する方法について説明します。

本章の内容

Diagnostics 管理 UI へのアクセス

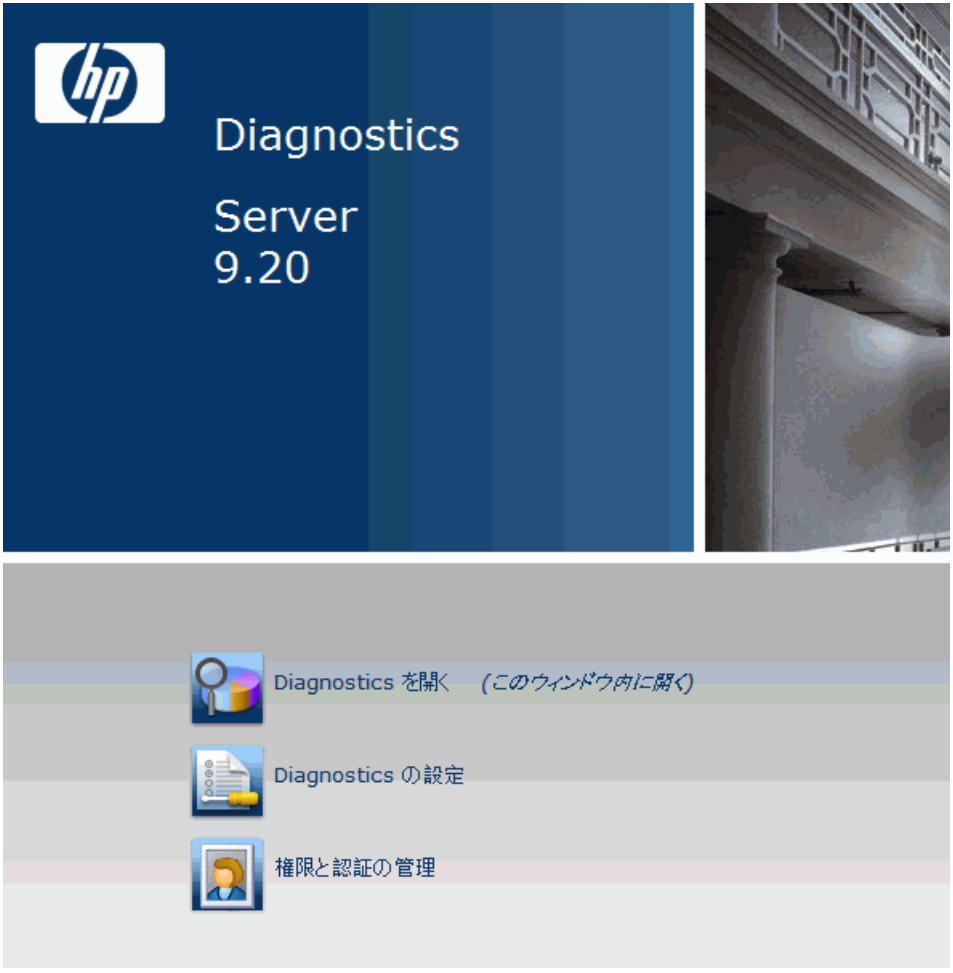
Diagnostics の設定情報の表示，ユーザ権限の設定，Diagnostics の設定値の設定や Diagnostics ソフトウェアの管理は，Diagnostics のメイン UI から直接実行できます。

Diagnostics 管理 UI にアクセスするには，次の手順を実行します。

1 ブラウザで
http://<Diagnostics サーバのホスト>:2006 にアクセスするか，または［**スタート**］>［**すべてのプログラム**］>［**Diagnostics Server**］>［**Administration**］を選択して，メイン Diagnostics UI を開きます。URL のポート番号 **2006** は，Diagnostics サーバの標準設定のポートです。ほかのポートを使うように Diagnostics サーバを設定している場合は，URL にそのポート番号を使ってください。

まだ Diagnostics サーバにログオンしていない場合，ユーザ名とパスワードの入力を求められます。これは，有効なユーザ名である必要があり，「**表示**」権限と「**変更**」権限の両方が必要です。有効なユーザ名と権限については，付録 B，「ユーザの認証と承認」を参照してください。

Diagnostics のメイン UI がブラウザ内に開きます。



次のような 3 つのオプションがあります。

- **Diagnostics を開く**：Diagnostics UI を開いて，Diagnostics サーバに報告を行っているエージェントで収集されたパフォーマンス・メトリックスを表示できます。パフォーマンス・メトリックスは，Diagnostics ビューに表示されます。

  Diagnostics ビューの詳細については，オンライン・ヘルプまたは『HP Diagnostics ユーザー・ガイド』を参照してください。

➤ **Diagnostics の設定**：Diagnostics サーバ［構成］ページへのリンクがある［コンポーネント］管理ページを開きます。

Diagnostics プロパティの設定に関する詳細は，751 ページ「サーバ設定の変更」を参照してください。

➤ **権限と認証の管理**：［ユーザ管理］ページを開き，セキュリティ情報および特定のユーザのユーザ権限を追加および保守することができます。セキュリティとユーザ権限の詳細は，755 ページ「ユーザの認証と承認」を参照してください。

2 Diagnostics のメイン UI で，［**Diagnostics の設定**］を選択します。詳細については，748 ページ「Diagnostics 管理 UI の使用」を参照してください。

注：

➤ 有効な資格情報が入力されるまで，Diagnostics ではユーザ名とパスワードを要求します。

➤ ［**キャンセル**］をクリックした場合，ブラウザに次のエラー・メッセージが表示されます。**アクセスが拒否されました。有効なユーザ名とパスワードを入力してください。**

➤ 有効なユーザ名とパスワードを入力したものの，適切な権限がない場合は，ブラウザに次のエラー・メッセージが表示されます。**アクセスが拒否されました。この画面を表示するのに必要な権限がありません。**

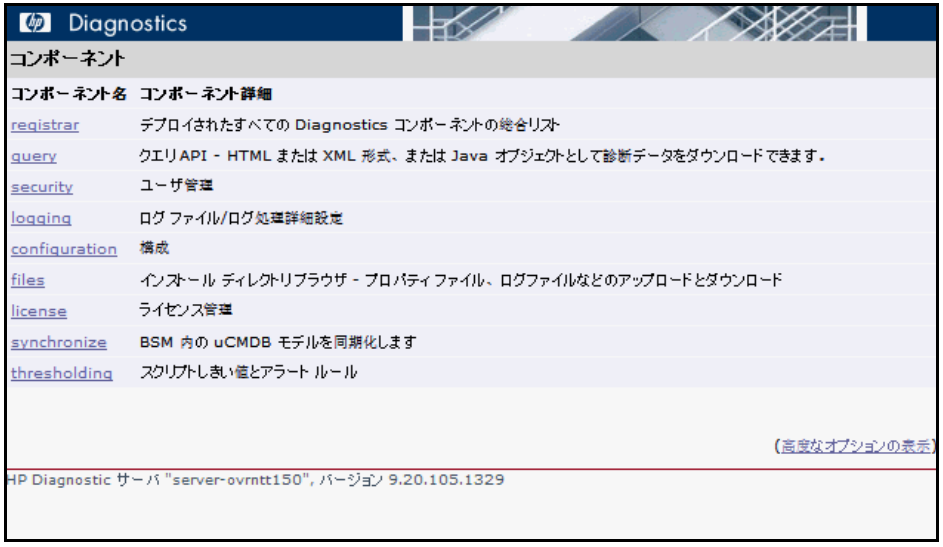
現在ログオンしている場所に別のユーザとしてログオンするには，ブラウザを一度閉じて，開きなおす必要があります。

Diagnostics 管理 UI の使用

Diagnostics 管理 UI では，Diagnostics の設定情報を表示したり，Diagnostics サーバとほかの Diagnostics コンポーネントの通信や，プローブから受信したデータを Diagnostics サーバでどのように処理するのかを制御するプロパティ値を設定したりします。

有効な値を確実に入力するために，プロパティ・ファイルを直接編集するのではなく，設定ページを使って Diagnostics サーバのプロパティを変更することをお勧めします。

Diagnostics のメイン UI で，［**Diagnostics の設定**］を選択して，［**コンポーネント**］ページを表示します。



この［コンポーネント］ページには，任意の Diagnostics ビューの［メンテナンス］リンクを選択してアクセスすることもできます。

次のいずれかのリンクを選択すると，Diagnostics のさまざまな管理ページに移動できます。情報ページに移動するリンクや，設定を変更できるリンクがあります。

- **Registrar :** すべての Diagnostics コンポーネント・デプロイメントのセントラル・リスト。
- **Query :** HTML または XML 形式，または Java オブジェクトとして診断データをダウンロードできるクエリ API。/contrib ディレクトリに，Diagnostics クエリ API を使ってカスタム・ダッシュボードを作成する例があります。オンライン・ヘルプのホーム・ページまたはドキュメント・ディレクトリからアクセスできる『HP Diagnsotics Data Model and Query API Guide』(PDF) も参照してください。

  最初のクエリ・ページの下部にある［**Active Users**］リンクを選択すると，最近の 60 秒間に Diagnostics サーバで認識されたアクティブ・ユーザのリストが表示されます。および，サマリまたはトレンド・クエリでユーザが生成する負荷の量がクエリ / 秒で表示されます。
- **Security :** ビルトイン・ユーザ管理。詳細については，760 ページ「Diagnostics サーバの［権限］ページについて」を参照してください。
- **Logging :** ログ・ファイルとログ詳細の設定。
- **Configuration :** Diagnostics サーバの設定。その他の設定ページの詳細については，751 ページ「サーバ設定の変更」を参照してください。
- **Files :** プロパティ・ファイル，ログ・ファイルなどのファイルのアップロードとダウンロードに使用するためのインストール・ディレクトリ・ブラウザ。
- **License :** ライセンス管理。詳細については，77 ページ「HP Diagnostics ライセンスの有効化」を参照してください。
- **Synchronize :** CI と Business Service Management の同期化。ハード同期化（Business Service Management と完全に同期化する）またはソフト同期化（新しい CI のみ Business Service Management と同期化する）を強制的に実行できます。
- **Thresholding :** しきい値および警告を設定するためのスクリプト・ステートメント。

［コンポーネント］ページに表示されるコンポーネントは，一般的に使用されるコンポーネントです。デフォルトでは，高度なコンポーネントは表示されません。

重要 : HP ソフトウェア・カスタマ・サポート担当者からアドバイスを受けずに，詳細設定オプションを操作しないでください。

詳細設定オプションを表示するには，次の手順を実行します。

➤ ページの一番下にある［**高度なオプションの表示**］をクリックします。

ページのオプション・リストが更新され，詳細設定オプションが表示されます。また，リンクが［**高度なオプションの非表示**］に変わります。

追加の詳細構成オプションが表示されます。

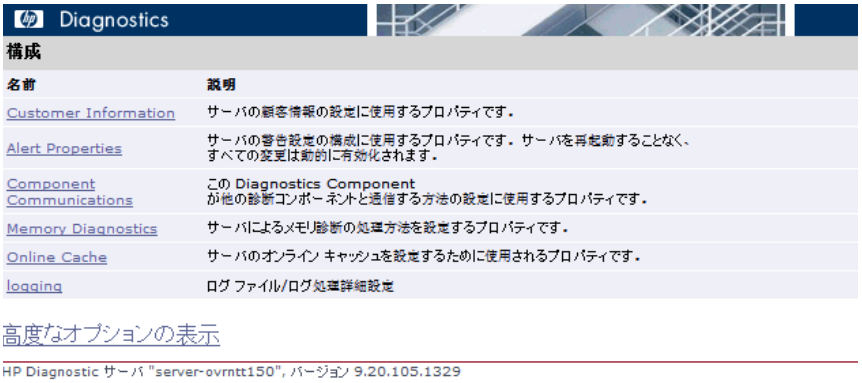
詳細設定オプションを隠すには

➤ ページの一番下にある［**高度なオプションの非表示**］をクリックします。

ページのオプション・リストが更新され，詳細設定オプションが表示されなくなります。また，リンクが［**高度なオプションの表示**］に変わります。

サーバ設定の変更

Diagnostics のメイン UI で［**Diagnostics の設定**］を選択して，［**configuration**］リンクを選択して次のような［構成］ページにアクセスします。



1 プロパティを更新するページへのリンクをクリックします。Diagnostics サーバに関する次の設定を実行できます。

- Customer Information
- Alert Properties
- Component Communications
- Memory Diagnostics
- Online Cache
- Logging

このページに表示される設定オプションは，一般に設定されるオプションです。デフォルトで，詳細構成オプションは隠れています。さらに多くの構成オプションを表示するには，［**高度なオプションの表示**］を選択します。

重要：HP ソフトウェア・カスタマ・サポート担当者からアドバイスを受けずに，詳細設定オプションを操作しないでください。

---

2 たとえは，［Customer Information］を選択すると，次のページが表示されます。表示されたプロパティを見直し，更新します。

3 変更に問題がなければ，［**送信**］をクリックして変更を保存します。［**すべてリセット**］をクリックして値をすべてデフォルト設定に戻すか，または変更しない場合はダイアログを閉じます。

変更が保存された旨を示すメッセージがページ上部に表示されます。

注 :

- 更新を行うと，ほとんどのプロパティで Diagnostics サーバの再起動を求めるメッセージが表示されます。Diagnostics サーバを再起動しないとプロパティの変更が反映されません。

  Diagnostics サーバのプロパティにほかの変更を加える場合，すべて変更してから Diagnostics サーバを再起動してください。

  サーバを再起動すると，データのロスが生じます (最大で 6 分間)。したがって，都合に合わせて一度に再起動する必要があります。

- ログ処理レベルの詳細の変更では，Diagnostics サーバを再起動する必要がありません。ただし，変更の適用に最大で 1 分かかることがあります。

B

ユーザの認証と承認

Diagnostics の認証および承認プロセスと，ユーザ・セキュリティ権限を作成および保守する方法について説明します。

本章の内容

ユーザの認証と承認について

Diagnostics コンポーネントのユーザの認証および承認設定はすべて, Diagnostics コマンダ で行われます。

認証は，人の身元を確認するプロセスです。承認は，既知の人に特定のアクションを実行するための権限（権利）があることを確認するプロセスです。また，役割は，ユーザに割り当てられている一連の権限です。

ユーザ名を作成，編集し，ユーザがアプリケーション内で自ら責任を負う機能を実行できるように，ユーザ権限を与えることで，認証と承認を管理します。

また，特定の Diagnostics サーバに接続しているプローブの Profiler（.NET Diagnostics Profiler または Java Diagnostics Profiler）のユーザ権限と特権も，Diagnostics コマンダ で定義されます。ユーザに，特定のプローブ・グループの Profiler にアクセスするための権限を 1 セット, Diagnostics サーバにアクセスするための権限をもう 1 セット割り当てることができます。

重要 :

- Agent を Profiler 専用でインストールしている（Diagnostics サーバに接続していない）場合，Agent 自体で Profiler のユーザの権限と承認を管理します。
- Profiler 専用でインストールしている Java Agent の認証および承認の管理については，474 ページ「Diagnostics Java Profiler での権限と認証」を参照してください。
- Profiler 専用でインストールしている .NET Agent の認証および承認の管理については, 614 ページ「.NET Profiler での権限と認証」を参照してください。

Diagnostics データを表示したり，Diagnostics 設定またはユーザ権限に変更を加える前に，適切な権限でユーザ・セキュリティ・アクセスが可能なユーザ名を使って Diagnostics コマンダにログオンする必要があります。

特定のブラウザ・セッションで Diagnostics サーバにログインすると，ブラウザのセッションが終了するまでユーザ名は有効になります。Diagnostics での操作が完了したら，ブラウザを閉じて，他者があなたの権限を使って Diagnostics にアクセスしないようにしてください。

ユーザ権限について

Diagnostics ユーザに割り当てられる権限レベルは次のとおりです。

権限	説明
表示	ユーザは，UI から Diagnostics のデータを表示できます。
実行	ユーザは, しきい値の変更やコメントの追加といった設定の変更を UI で行うことができます。Profiler では，この権限は，ガベージ・コレクションを実行したり，Profiler が保持するパフォーマンス・データをクリアする権利を与えます。
変更	ユーザは，[**Diagnostics の設定**] メニューにアクセスして，コンポーネント設定を変更したり，ユーザ情報を保守できます。Profiler では，これは，ヒープダンプの取得やインストルメンテーションの変更といった，潜在的に危険を伴う操作を実行する権利を与えます。

注 :

- 権限レベルの **rhttpout** と **system** は，内部使用のみを目的としています。**rhttpout** は，rhttp/out URL にアクセスして，分散サーバのリモート管理を行うためのユーザ・アクセス権を与えます。
- **system** は，通常，**HP** 特別ユーザだけに与えられる内部権限です。この権限は，Diagnostics コンポーネントの相互通信を可能にします (Diagnostics サーバに登録するのにプローブが必要とする権限など)。システム状況を表示するのに **system** 権限も必要です。

各権限レベルは独立しています。ある権限レベルから次の権限レベルへの継承はありません。1 人のユーザに, 実行に必要なすべての権限レベルを付与できます。

たとえば，しきい値を変更できるようにするには，ユーザに「**表示**」と「**実行**」の両方の権限を与える必要があります。「**実行**」権限だけを与えられたユーザ名は，変更を許可された UI を表示できないため有用でありません。

ユーザへの権限の割り当てについては，770 ページ「Diagnostics デプロイメント全体への権限の割り当て」を参照してください。

役割について

ユーザ / 権限の割り当てのほかに，役割に権限を割り当てて，それらの割り当てをユーザに割り当てることもできます。これにより，複数のユーザの管理が容易になります。新しいユーザを Diagnostics に追加した場合，ユーザ / 役割の割り当てだけを実行する必要があります。これは，特に Diagnostics サーバおよび特定のプローブ・グループの Profiler にアクセスするためのさまざまな権限をユーザにセットアップしているときに便利です。

次の例を検討します。

所有している Agent システムの Profiler に対して，すべての権限（表示，実行，変更）を必要とし，所有していない Agent システムで表示権限を必要とする 2 つの開発チーム（Dev1 と Dev2）があります。両方のチームは，UI に対して表示権限と実行権限を持つ必要があります。

作成する必要がある役割は次のとおりです。

役割	権限
Enterprise（UI へのアクセス）	[DevUI] = 表示，実行
Dev1 プローブ・グループ	[Dev1All] = 表示，実行，変更 [Dev2View] = 表示
Dev2 プローブ・グループ	[Dev2All] = 表示，実行，変更 [Dev1View] = 表示

役割とユーザを区別するために，役割は括弧で囲む必要があります。たとえば，Dev1 チームの新しいユーザを Diagnostics に追加する場合，役割 [DevUI]，[Dev1All]，[Dev1View] の一部である必要があります。

標準設定のユーザ名を使用した Diagnostics へのアクセス

Diagnostics には，次の標準設定のユーザ名が定義されています。

標準設定のユーザ名	権限	説明
user	表示	UI からのデータの表示のみ可能です。
superuser	表示，実行	UI からデータの表示，しきい値の変更，警告とコメントの作成が可能です。
admin	表示，変更，実行，システム	UI からデータの表示，しきい値の変更，警告とコメントの作成が可能です。コンポーネントの設定，ユーザ情報の保守が可能です。

これらの標準設定のユーザ名を使って，Diagnostics の機能にアクセスできます。

標準設定のユーザ名のパスワードは，ユーザ名と同じです。たとえば，ユーザ名が admin の場合，パスワードは admin です。

標準設定のユーザ名のパスワードや権限は，必要に応じて変更できます。また，Diagnostics へのユーザ・アクセスを制御するために新しいユーザ名を定義することもできます。

重要：内部使用を目的とする 2 つの標準設定のユーザ **HP** および **bac** は変更できません。これらのユーザは，コンポーネント間の内部通信のためのものです。

Diagnostics サーバの［権限］ページについて

［権限］ページで，ユーザを管理したり，ユーザ権限を割り当てたりします。

本項の内容

［権限］ページへのアクセス

［権限］ページへは，［**権限と認証の管理**］を選択して，Diagnostics のメイン UI からアクセスします。

この［権限］ページへは，Diagnostics ビューにある［メンテナンス］リンクを選択してから，［セキュリティ］リンクを選択します。

Diagnostics データを表示したり，Diagnostics 設定またはユーザ権限に変更を加える前に，適切な権限でユーザ・セキュリティ・アクセスが可能なユーザ名を使って Diagnostics コマンダにログオンする必要があります。

［権限］ページが開いたときに，Diagnostics サーバにまだログオンしていない場合，ユーザ名とパスワードの入力が求められます。権限を表示してパスワードを変更するには，少なくとも「**表示**」権限が必要です。ユーザを追加または削除したりユーザ権限を更新するには，「**表示**」権限と「**変更**」権限の両方が必要です。

注 :

- 有効な情報が入力されるまで，Diagnostics ではユーザ名とパスワードを要求します。
- ［**キャンセル**］をクリックした場合，ブラウザに次のエラー・メッセージが表示されます。**アクセスが拒否されました。有効なユーザ名とパスワードを入力してください。**
- 有効なユーザ名とパスワードを入力したものの，適切な権限がない場合は，ブラウザに次のエラー・メッセージが表示されます。**アクセスが拒否されました。この画面を表示するのに必要な権限がありません。**

［権限］ページの概要

次の画面は，Diagnostics サーバ［権限］ページの例です。

［権限］ページは，次の 3 つのセクションに分かれています。

- **組織の診断権限：**このセクションでは，Diagnostics ユーザを管理したり，Diagnostics サーバおよび Agent を含めた Diagnostics デプロイメント全体に権限を割り当てたりします。

  標準設定で，特定の Diagnostics サーバへのアクセスの認証をユーザに与えている場合，そのサーバに接続しているすべてのプローブへのアクセスも同様に認証（および権限）が与えられます。

注 : Diagnostics には中央管理された許可システムがあり，これで，権限をユーザに設定でき，権限は Diagnostics システムに接続されたすべての分散サーバとプローブに適用されます。ただし，権限は 5 分に 1 回だけ分散コンポーネントに与えられるので，権限の変更がただちに有効になるわけではありません。

- **次に接続されているプローブを制御 : ＜ Diagnostics コマンダ・サーバ＞** : このセクションでは，プローブの Profiler にアクセスするユーザに権限を割り当てます。ユーザに，特定のプローブ・グループの Profiler にアクセスするための権限を 1 セット，Diagnostics コマンダ・サーバにアクセスするための権限をもう 1 セット割り当てることができます。
- **内部 Diagnostics パスワードの暗号化。** EncryptPassword ユーティリティにアクセスしてパスワードを暗号化できます。

エンタープライズおよびアプリケーション権限

［権限］ページで設定したエンタープライズおよびプローブ・レベルの権限のほかに，アプリケーション・レベルの権限を設定することもできます。アプリケーション権限は，最初の［アプリケーション］ウィンドウの Diagnostics UI で設定されます。アプリケーション権限の設定に関する詳細は，『HP Diagnostics ユーザー・ガイド』を参照してください。

権限の 3 つのグループは次のとおりです。

- エンタープライズ
  - **表示 :** ユーザは，Diagnostics UI にパフォーマンス・データを表示できます。
  - **実行 :** ユーザは，しきい値を変更してコメントを追加し，アプリケーションを作成できます。
  - **変更 :** ユーザは，システムのすべての管理アクセス権を持ちます（ユーザの作成など）。

- プローブ・グループあたり（Profiler に適用）
  - **表示 :** ユーザは, Profiler で収集したパフォーマンス･データを表示できます。
  - **実行 :** ユーザは，ガベージ・コレクションを実行して，Profiler で保持しているパフォーマンス・データをクリアできます。
  - **変更 :** ユーザは, ヒープダンプの取得やインストルメンテーションの変更といった操作を実行できます。
- アプリケーション
  - **表示 :** ユーザは，アプリケーションを表示してエンティティのプロパティを編集します（エンタープライズ権限を「変更」に設定する必要があります)。
  - **変更 :** ユーザは，アプリケーションの削除，名前の変更，変更，アプリケーションからのエンティティの追加または削除を実行できます。
  - **画面の編集 :** ユーザは，［アプリケーション］画面を使って編集できます。

注 : 権限は包含的でありません（実行に表示は含まれません)。

エリアとアクション	エンタープライズ権限			アプリケーション権限		
	表示	実行	変更	表示	変更する	画面の編集
Diagnostics UI						
UI への Diagnostics データの表示	X					
カスタム属性の変更		X				
UI でのしきい値の設定		X				
UI でのコメントの作成/変更/削除		X				
UI での警告ルールの作成		X				
［診断を設定］ページ			X			
ビジネス・トランザクションの設定		X				

エリアとアクション	エンタープライズ権限			アプリケーション権限		
	表示	実行	変更	表示	変更する	画面の編集
システム状況の表示 **注：**システム状況を表示するには，**システム・**エンタープライズ権限も必要です。			X			
ほかのユーザの認証と承認の管理			X			
［メンテナンス］ページへのアクセス			X			
インシデントの作業	X					
カスタム・ビューの作業	X					
Profiler UI						
Profiler でのガーベッジ・コレクションの実行		X				
Profiler でのパフォーマンス・データのクリア		X				
Profiler への Diagnostics データの表示	X					
ヒープダンプの実行（メモリおよびアロケーション分析）			X			
設定の変更			X			
ユーザ定義のアプリケーション						
アプリケーションの作成		X				
アプリケーションの削除		X			X	
アプリケーション名の変更		X			X	
アプリケーション・パスの変更		X			X	

エリアとアクション	エンタープライズ権限			アプリケーション権限		
	表示	実行	変更	表示	変更する	画面の編集
アプリケーション権限の変更		X			X	
カスタム・アプリケーション画面の編集	X					X
エンティティのアプリケーションへの追加	X				X	
アプリケーションからのエンティティの削除	X				X	
エンティティのプロパティの編集(しきい値, コメントなど)		X		X		
アプリケーションの表示	X			X		
自動検出されたアプリケーション，トランザクション・アプリケーションまたはエンタープライズ全体						
アプリケーション・パスの作成，削除，変更は許可されません						
アプリケーション権限の変更		X			X	
アプリケーション画面の編集	X					X
エンティティのアプリケーションへの追加	X				X	
アプリケーションからのエンティティの削除	X				X	
エンティティのプロパティの編集		X		X		
アプリケーションの表示	X			X		

ユーザの作成，編集，削除

「**表示**」権限と「**変更**」権限の両方があるユーザは，新しいユーザを作成したり，既存のユーザのパスワードを変更したり，ユーザを削除できます。「**表示**」権限しかないユーザは，自分のパスワードだけを変更できます。

新しいユーザを作成するには，次の手順を実行します。

1 761 ページ「［権限］ページへのアクセス」の説明に従って，Diagnostics サーバの［権限］ページにアクセスします。

2 ［権限］ページで，［**ユーザ管理**］をクリックして［ユーザ管理］ページを開きます。

3 ［ユーザ管理］ページで，［**ユーザの作成**］をクリックします。

4 ［**新規ユーザ名**］ボックスに，新しいユーザのユーザ名を入力して［**OK**］をクリックします。ユーザ名のリストに新しいユーザが表示されます。

注：基本的な認証を処理する際にブラウザの制限があるため，ユーザ名とパスワードには英字を使う必要があります。

---

5 ［**パスワードの変更**］で，［**パスワード**］ボックスに新しいユーザのパスワードを入力し，［**パスワードの確認**］ボックスに再入力して確認します。

6 ［**＜現在のユーザ＞のパスワード**］ボックスに，現在ログオンしているユーザのパスワードを入力します。

7 ［**変更を保存**］をクリックします。

標準設定で，新しいユーザに「**表示**」権限が割り当てられています。ユーザに割り当てる権限の変更については，770 ページ「Diagnostics デプロイメント全体への権限の割り当て」を参照してください。

役割を割り当てるには，次の手順を実行します。

1 761 ページ「［権限］ページへのアクセス」の説明に従って，Diagnostics サーバの［権限］ページにアクセスします。

2 ［ユーザ管理］ページで，役割をユーザに割り当てます。役割は括弧で囲みます（[aRole]）。役割はカンマで区切ることができます（[Role1],[Role2]）。

注：［Enterprise］や［Probe グループごと］ダイアログで役割に権限をセットアップする必要があります（770 ページ「Diagnostics デプロイメント全体への権限の割り当て」と771 ページ「プローブ・グループへの権限の割り当て」を参照）。

ユーザを削除するには，次の手順を実行します。

1 761 ページ「［権限］ページへのアクセス」の説明に従って，Diagnostics サーバの［権限］ページにアクセスします。

2 ［権限］ページで，［**ユーザ管理**］をクリックして［ユーザ管理］ページを開きます。

3 ［ユーザ管理］ページの［**＜現在のユーザ＞のパスワード**］ボックスに，現在ログオンしているユーザのパスワードを入力します。

4 削除するユーザに対応する赤の X（［**ユーザの削除**］）ボタン をクリックします。

5 選択したユーザを削除するかどうかを尋ねるメッセージ・ボックスが表示されます。

［**OK**］をクリックしてユーザを削除します。

「表示」権限と「変更」権限がある場合にユーザのパスワードを変更するには，次の手順を実行します。

1 761 ページ「［権限］ページへのアクセス」の説明に従って，Diagnostics サーバの［権限］ページにアクセスします。

2 ［権限］ページで，［**ユーザ管理**］をクリックして［ユーザ管理］ページを開きます。

3 ［ユーザ管理］ページの関連するユーザを示す行で，［**パスワード**］および［**パスワードの確認**］ボックスに新しいパスワードを入力します。

4 ［**<現在のユーザ>のパスワード**］ボックスに，現在ログオンしているユーザのパスワードを入力します。

5 ［**変更を保存**］をクリックして，複数のユーザ名に加えた変更をすべて保存します。

「表示」権限だけがある場合に自分のパスワードを変更するには，次の手順を実行します。

1 761 ページ「［権限］ページへのアクセス」の説明に従って，Diagnostics サーバの［権限］ページにアクセスします。

2 ［権限］ページで，［**ユーザ管理**］をクリックして［ユーザ管理］ページを開きます。

3 ［ユーザ管理］ページで，［**パスワード**］および［**パスワードの確認**］ボックスに新しいパスワードを入力します。

4 ［**旧 パスワード**］ボックスに，古いパスワードを入力します。

5 ［**変更を保存**］をクリックします。

Diagnostics デプロイメント全体への権限の割り当て

「表示」権限と**「変更」**権限の両方を持つユーザは，Diagnostics デプロイメント全体にユーザ権限を割り当てることができます。

注 : Diagnostics ユーザに割り当てられるユーザ権限については，757 ページ「ユーザ権限について」を参照してください。

Diagnostics 全体にユーザ権限を割り当てるには

1 761 ページ「［権限］ページへのアクセス」の説明に従って，Diagnostics サーバの［権限］ページにアクセスします。

2 ［権限］ページで，［**組織権限の編集**］をクリックして［組織の診断権限の編集］ページを開きます。

［組織の診断権限の編集］ページは編集可能なページで，ユーザ権限を変更できます。

3 変更する権限を持つユーザの名前を見つけます。

重要 : 768 ページ「ユーザの作成，編集，削除」の説明に従って，［ユーザ管理］ページでユーザを追加できます。

---

4 カンマ区切り値として，そのユーザ名に権限を追加します。

たとえば，**newuser** という名前でユーザを定義していて，そのユーザに「表示」権限と「実行」権限を割り当てる場合は，**newuser** を見つけて次のように行を編集する必要があります。

```
newuser = view,execute
```

［Editing Enterprise Permissions］ページには，一連の標準設定ユーザも含まれます。これらのユーザについては，759 ページ「標準設定のユーザ名を使用した Diagnostics へのアクセス」を参照してください。これらの標準設定ユーザの権限を変更できます。

プローブ・グループへの権限の割り当て

「表示」権限と**「変更」**権限の両方があるユーザは，特定のプローブ・グループに属すプローブの Profiler にアクセスする権限をユーザに割り当てることができます。

標準設定で，特定の Diagnostics サーバへのアクセスの認証をユーザに与えてる場合，そのサーバに接続しているすべてのプローブの Profiler へのアクセスも同様に認証（および権限）が与えられます。

ただし, ユーザが Diagnostics サーバに対して持っているものとは異なるプローブ・グループの権限セットを割り当てることができます。

注 : Diagnostics ユーザに割り当てられるユーザ権限については，757 ページ「ユーザ権限について」を参照してください。

プローブ・グループごとにユーザ権限を変更したり，システムに追加される将来のプローブ・グループにユーザ権限設定を定義する Permissions テンプレートを変更できます。

注 : ユーザおよび権限の設定を有効にするには，変更を保存してから約 1 分かかることがあります。

各プローブ・グループで，特定の権限を持つユーザの標準設定のユーザ・グループが 3 つあります。これらのグループをコメントアウトすることも，それらの権限を変更することもできます。標準設定で，次のユーザ・グループは，すべてのプローブ・グループに定義されています。

ユーザ・グループ	権限
any_diagnostics_admin	このグループは，Diagnostics サーバの administration（変更）権限を持つユーザを参照します。標準設定で，このカテゴリに該当し，ほかの定義済みの権限設定を持たないすべてのユーザは，そのサーバに接続されているすべてのプローブの管理権限を持ちます。
any_diagnostics_superuser	このグループは，Diagnostics サーバの superuser（実行）権限を持つユーザを参照します。標準設定で，このカテゴリに該当し，ほかの定義済みの権限設定を持たないすべてのユーザは，そのサーバに接続されているすべてのプローブの実行権限を持ちます。
any_diagnostics_user	このグループは，Diagnostics サーバの user（表示）権限を持つユーザを参照します。標準設定で，このカテゴリに該当し，ほかの定義済みの権限設定を持たないすべてのユーザは，そのサーバに接続されているすべてのプローブの表示権限を持ちます。

特定のプローブ・グループにアクセスするためのユーザ権限を割り当てるには

1 761 ページ「[権限] ページへのアクセス」の説明に従って，Diagnostics サーバの［権限］ページにアクセスします。

2 ［権限］ページの［**次に接続されているプローブを制御：< Diagnostics コマンダ・サーバ>**］セクションで，［**<プローブ・グループ名>の編集**］をクリックします。

［Editing Permissions］ページが開きます。これは編集可能なページであり，ユーザ権限を変更できます。

3 一意の権限を割り当てるユーザ名を入力し，その権限をカンマ区切り値としてユーザ名に追加します。

たとえば，**newuser** という名前でユーザを定義していて，このユーザに特定のプローブ・グループの「表示」権限と「実行」権限を割り当てる場合，次の行を入力します。

```
newuser = view,execute
```

権限テンプレートを使ってユーザ権限を割り当てるには

1 761 ページ「[権限] ページへのアクセス」の説明に従って，Diagnostics サーバの［権限］ページにアクセスします。

2 ［権限］ページの［**次に接続されているプローブを制御：< Diagnostics コマンダ・サーバ>**］セクションで，［**権限のテンプレートの編集**］をクリックします。

［Editing Template Permissions］ページが開きます。これは編集可能なページであり，ユーザ権限を変更できます。

3 一意の権限を割り当てるユーザ名を入力し，その権限をカンマ区切り値としてユーザ名に追加します。

たとえば，**newuser** という名前でユーザを定義していて，このユーザに特定のプローブ・グループの「表示」権限と「実行」権限を割り当てる場合，次の行を入力します。

```
newuser = view,execute
```

また，テンプレートに定義されているいずれかのユーザ・グループ設定を変更したり，コメントアウトすることもできます。

重要 : Diagnostics サーバに接続されている今後のプローブ・グループはすべて，Permissions テンプレートのユーザ権限設定を継承します。

統合された HP ソフトウェア製品のユーザの認証と承認

Diagnostics は，ほかの HP ソフトウェア・アプリケーションと統合できます（Business Service Management，Performance Center，または LoadRunner）。このセクションでは，これらの統合製品のユーザで認証と承認がどのように機能するのかについて説明します。このセクションの構成は次のとおりです。

- 774 ページ「Business Service Management ユーザの認証と承認」
- 775 ページ「Performance Center および LoadRunner ユーザの認証と承認」

Business Service Management ユーザの認証と承認

Business Service Management で，Diagnostics のユーザ権限を定義できます。詳細については，704 ページ「Business Service Management での Diagnostics ユーザへの権限の割り当て」を参照してください。

既存または新しい Business Service Management ユーザが Business Service Management から Diagnostics を開くと，Business Service Management セッションからユーザの権限が取得され，Diagnostics 権限システム（関係がある場合，SaaS カスタマの下）にコピーされます。

ユーザが Diagnostics を開く場合にのみ，Business Service Management ユーザ権限への更新が取得されます（Diagnostics がすでに開いている場合，閉じた後に再び開くまで検出されません）。

Business Service Management のパスワードは，Diagnostics に送信されません。Diagnostics は，正常に完了した Business Service Management へのログインを信用します。

Business Service Management ユーザの権限が変わると，そのユーザが Diagnostics を再度開くまで変更は適用されません。

Business Service Management のユーザが削除された場合, Diagnostics からそれらの権限を手動で削除することをお勧めします。詳細については，768 ページ「ユーザの作成，編集，削除」を参照してください。

注 : Diagnostics サーバは，ユーザに加えられた権限の変更を検出するのに最高で 5 分かかります。

Performance Center および LoadRunner ユーザの認証と承認

Diagnostics と統合するために LoadRunner または Performance Center の両方をセットアップするとき, LoadRunner / Performance Center 内の Diagnostics サーバ詳細を指定します。これらの詳細には，HP Diagnostics へのログオンに使用するユーザ名とパスワードが含まれます。

LoadRunner または Performance Center から Diagnostics にアクセスする際, 統合セットアップ時に指定したものと同じユーザ名とパスワードを使って Diagnostics にログインします。

したがって, LoadRunner または Performance Center から Diagnostics にアクセスするユーザは，統合セットアップ時に指定されたユーザ名に関連付けられている権限を持ちます。

ユーザ管理活動の追跡

ユーザが Diagnostics サーバの［ユーザ管理］ページに入るたびに，行われたすべての活動が次のログ・ファイルに記録されます。**＜Diagnostics サーバのインストール・ディレクトリ＞¥log¥useradmin.log**.

ファイルに記録されるデータには，アクションを実行した日時，アクションの説明，アクションを実行するユーザの名前が含まれます。

ログ・ファイルを表示するには，次の手順を実行します。

1 次のいずれかの方法で Diagnostics サーバの管理ページを開きます。

- ［**スタート**］>［**すべてのプログラム**］>［**HP Diagnostics Server**］>［**Administration**］を選択します。
- ブラウザで http://＜Diagnostics サーバのホスト＞:2006 にアクセスします。URL のポート番号 **2006** は，Diagnostics サーバの標準設定のポートです。ほかのポートを使うように Diagnostics サーバを設定している場合は，URL にそのポート番号を使ってください。

Diagnostics UI のメイン・ページが開きます。

2［**Diagnostics の設定**］をクリックします。

3 まだ Diagnostics サーバにログオンしていない場合，ユーザ名とパスワードの入力を求められます。これは，有効なユーザ名である必要があり，「**表示**」権限と「**変更**」権限の両方が必要です。有効なユーザ名と権限については，757 ページ「ユーザ権限について」を参照してください。

注：

- 有効な資格情報が入力されるまで，Diagnostics ではユーザ名とパスワードを要求します。
- ［**キャンセル**］をクリックした場合，ブラウザに次のエラー・メッセージが表示されます。**アクセスが拒否されました。有効なユーザ名とパスワードを入力してください。**
- 有効なユーザ名とパスワードを入力したものの，適切な権限がない場合は，ブラウザに次のエラー・メッセージが表示されます。**アクセスが拒否されました。この画面を表示するのに必要な権限がありません。**

Diagnostics サーバの［コンポーネント］ページが開きます。

4 ［**logging**］をクリックします。［ログ］ページが開きます。

5 ［**ログ ファイルの表示**］をクリックします。ログ・ファイルのリストが表示されます。

6 **＜Diagnostics サーバのインストール・ディレクトリ＞¥log¥useradmin.log** リンクをクリックします。

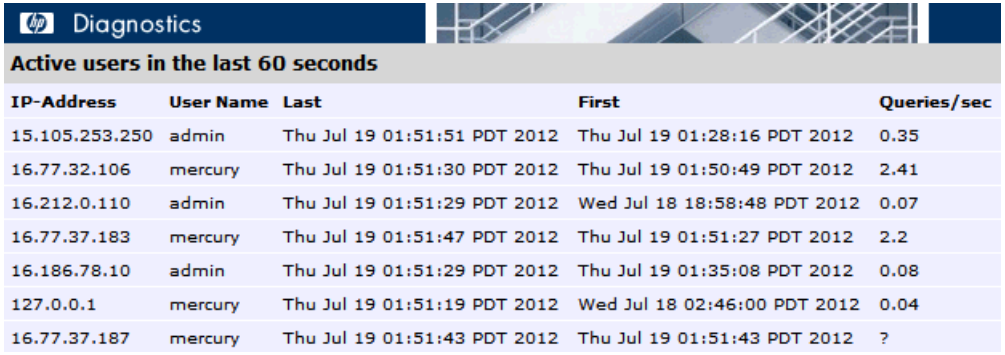
ページの下部にログ・ファイルが表示されます。

アクティブなユーザのリスト

Diagnostics サーバに表示されるアクティブなユーザのリストを最後の 60 秒間に取得できます。および，サマリまたはトレンド・クエリでユーザが生成する負荷の量がクエリ / 秒で表示されます。

メイン Diagnostics UI から，［**Diagnostics の設定**］を選択すると，［コンポーネント］ページが表示されます（この［コンポーネント］ページへは，Diagnostics ビューにある［メンテナンス］リンクを選択してアクセスすることもできます）。

［**クエリ**］リンクを選択し，そのページの一番下にある［**アクティブなユーザ**］リンクを選択してアクティブなユーザのリストを表示します。


Diagnostics

Active users in the last 60 seconds

IP-Address	User Name	Last	First	Queries/sec
15.105.253.250	admin	Thu Jul 19 01:51:51 PDT 2012	Thu Jul 19 01:28:16 PDT 2012	0.35
16.77.32.106	mercury	Thu Jul 19 01:51:30 PDT 2012	Thu Jul 19 01:50:49 PDT 2012	2.41
16.212.0.110	admin	Thu Jul 19 01:51:29 PDT 2012	Wed Jul 18 18:58:48 PDT 2012	0.07
16.77.37.183	mercury	Thu Jul 19 01:51:47 PDT 2012	Thu Jul 19 01:51:27 PDT 2012	2.2
16.186.78.10	admin	Thu Jul 19 01:51:29 PDT 2012	Thu Jul 19 01:35:08 PDT 2012	0.08
127.0.0.1	mercury	Thu Jul 19 01:51:19 PDT 2012	Wed Jul 18 02:46:00 PDT 2012	0.04
16.77.37.187	mercury	Thu Jul 19 01:51:43 PDT 2012	Thu Jul 19 01:51:43 PDT 2012	?

クエリの負荷がかなり大きいことがわかった場合は，UI が実行中のクエリの制限を設定できます。サーバの **ui.properties** ファイルを使用して，サーバへの UI クエリの更新頻度をスロットルするプロパティを設定できます。

JAAS を使用するための Diagnostics の設定

ユーザの認証に JAAS（Java 認証および認可サービス）を使うように Diagnostics を設定できます。JAAS が有効になっていると，UI にアクセスしたときに［ログイン］ダイアログに入力したユーザ名とパスワードが，設定した JAAS プラグ可能認証モジュール（LoginModule）によって認証されます。

注 : JAAS のサポートは，Diagnostics コマンダ・サーバでのみ使用できます。

JAAS は，次の 2 行をコメントアウトして **<インストール・ディレクトリ>/etc/server.properties** ファイルで有効にする必要があります。

```
authentication.jaas.config.file=jaas.configuration
authentication.jaas.realm=Diagnostics
```

authentication.jaas.config.file プロパティは，LoginModule を定義する（etc ディレクトリに関連する）設定ファイルを指定し，**authentication.jaas.realm** は，設定ファイルで使う必要のあるエントリを指定します。

jaas.configuration の例 :

```
Diagnostics
{
 com.mercury.diagnostics.server.jaas.spi.SiteMinderLoginModule sufficient
        ip="1.2.3.4";

 com.mercury.diagnostics.server.jaas.spi.LDAPLoginModule sufficient
   useSSL="true"
   serverCertificate="etc/ldap.keystore"
   providerURL="ldap://ldap.yourdomain.com:636"
   baseDN="ou=People,o=yourdomain.com";

};
```

JAAS 設定ファイルの詳細については, Oracle のドキュメントの JAAS や Oracle の javadoc の javax.security.auth.login.Configuration を参照してください。

注 :［権限と認証の管理］Web ページで，Diagnostics によって作成されたユーザは，ユーザ名とパスワードの認証時に*最初に*使用されます。認証が失敗した場合のみ，JAAS 認証が実行されます。

Diagnostics には，次の LoginModule が搭載されています。

- **LDAP :**（com.mercury.diagnostics.server.jaas.spi.LDAPLoginModule）LDAP サーバの認証を可能にします。
- **SiteMinder** :（com.mercury.diagnostics.server.jaas.spi.SiteMinderLoginModule）SiteMinder 環境の認証を可能します。

注 :

- server.properties や jaas.configuration ファイルに変更を加えた後, Diagnostics コマンダ・サーバを再起動する必要があります。
- ほかのアプリケーション（LDAP など）でも使用される JAAS 認証プロバイダを使っている場合，Diagnostics UI へのアクセスに HTTPS をオンにすることをお勧めします。
- JAAS 認証プロバイダを使用している場合，ユーザ・アカウントは認証ソースによって保守されます。ユーザ名の構文については，管理者にお問い合わせください。
- 認証受けたユーザの権限のその後の諸運は，［permissions］ページを使って保守維持されます。また，適切な LoginModule が役割を使うように設定されている場合は，役割も適用できます。この場合，既存の役割を使うことも，新しい役割を作成することもできます。

LDAP 認証の設定

Diagnostics で LDAP 認証を設定するには，まず JAAS を使うように Diagnostics を設定し（778 ページ「JAAS を使用するための Diagnostics の設定」を参照），Diagnostics コマンダ・サーバで LDAPLoginModule を設定する必要があります。

ここでは，Diagnostics による権限の処理方法と LDAP 認証の処理方法（次のページのフロー図を参照）について概説します。

1. ユーザからのユーザ名およびパスワードを受け入れます。
2. キャッシュされているユーザ名およびパスワードとこのユーザ名およびパスワードを比較します。
   - **a** ユーザ名がキャッシュにあり，パスワードが一致する場合，同じ資格情報を使用した最後のログイン試行に基づいてログインが成功または失敗します。
   - **b** それ以外の場合，次の手順に進みます。

3 組織の診断権限（EDP）に対してユーザ名およびパスワードを比較します。

   a ユーザ名が EDP にあり，パスワードが一致する場合，手順 6 に進みます。

   b それ以外の場合，次の手順に進みます。

4 jaas.configuration で設定されたすべての LDAP ログイン・モジュールをループします。

   a LoginModule でユーザ名 / パスワードが認証されたら，手順 6 に進みます。

   b それ以外の場合，次のログイン・モジュール進みます。

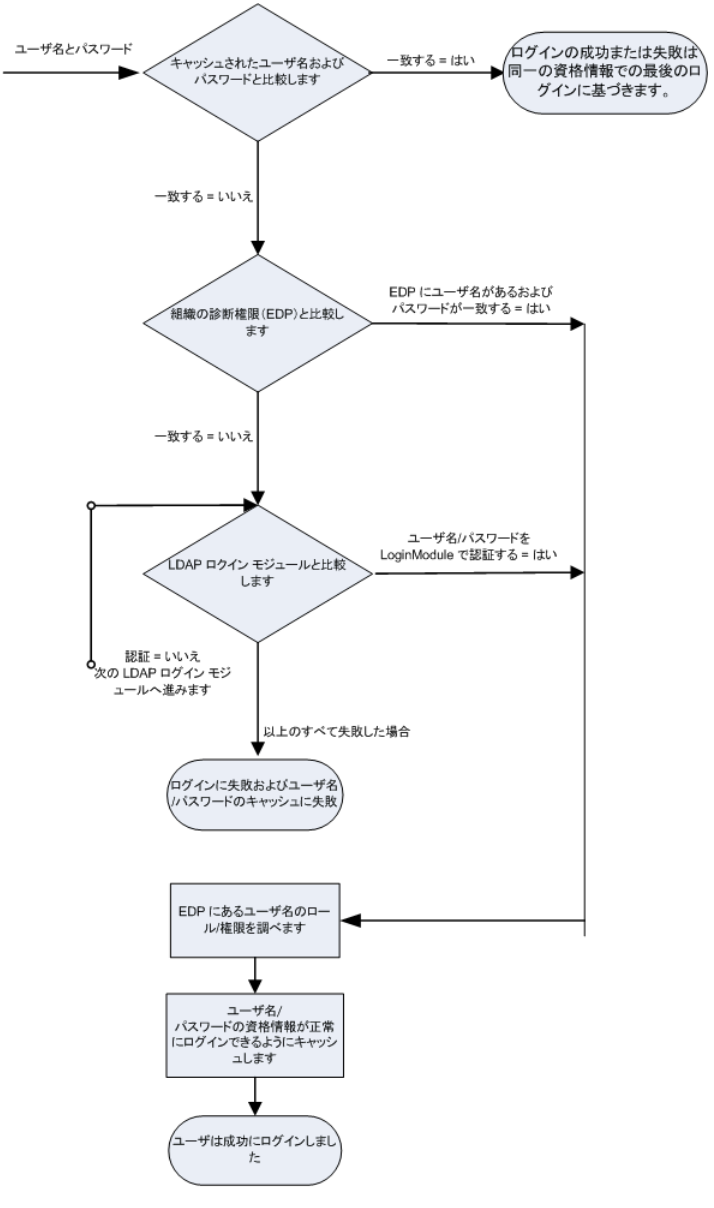
5 上記のすべてに失敗した場合，ログインに失敗し，このユーザ名 / パスワードが失敗したことがキャッシュされます。

6 EDP でユーザ名の役割 / 権限を検索します。

7 ユーザ名 / パスワードの資格情報でログインに成功したことをキャッシュします。

8 ユーザが正常にログインしたことを返します。

Diagnostics LDAP 認証のフロー



LDAP サーバ特定のオプション値を使って，**<インストール・ディレクトリ>/etc/jaas.configuration** で Diagnostics JAAS Realm（アプリケーション）を編集します。

LDAPLoginModule は，簡易モードまたは詳細モードで使用できます。

- 両方のモード：
  - SSL およびサーバの証明書を設定できる。
  - 役割を設定できる。
  - デバッグ情報を要求できる。
- 簡易モード：
  - 匿名ディレクトリ検索ができる。
  - 定義済みの検索フィルタが使用される。
  - 1 つのベース DN（識別名）のみ設定できる。
  - 委任は使用できない。
- 詳細モード：
  - ディレクトリ検索で資格情報を提供する必要がある。
  - RFC 2254 に準拠した検索フィルタを使用できる。
  - 複数のベース DN を設定できる。
  - 委任を使用できる。

  Active Directory システムで **ldp.exe** ユーティリティを起動して設定をテストできます。

次の表に，（すべてのモードで）**共通の LDAPLoginModule の属性**を示します。

属性	説明と例	値
authType	ユーザの認証時に使用するセキュリティ・レベルを指定します。	"simple"（標準設定） "none" "strong"
debug	デバッグ情報を server.log に書き込むかどうかを指定します。	"false"（標準設定） "true"
defaultRoles	認証された各ユーザを割り当てる役割のカンマ区切りリスト 例 : "SuperUsers"	
roleAttributes	ユーザの役割として値が使われるユーザの DN 属性のカンマ区切りリスト。**defaultRoles** も設定されている場合, 結果の役割は **defaultRoles** と **roleAttributes** の結合になります。 例 : "employeeType,hpJobFunction"	"roles"（標準設定）
serverCertificate	LDAP サーバの証明書が含まれる truststore ファイルへのパス。サーバのインストール・ディレクトリへの絶対パスまたは相対パス。キーストアの生成については，付録 C,「コンポーネント間での HTTPS 有効化」を参照してください。 例 : "etc/jssecacerts"	
useSSL	**true** に設定すると，SSL を使用して LDAP サーバに接続します。	"false"（標準設定） "true"

次の表に，**LDAPLoginModule の簡易モードの属性**を示します。

属性	説明と例	値
allowAnonymous	**true** に設定すると，LDAP サーバの匿名検索でユーザのプリンシパル DN を取得できます。有効にするには，**searchFirst** 属性も **true** に設定する必要があります。	"false"（標準設定） "true"
baseDN	プリンシパルの DN を作成するために使用します。匿名検索が許可されている場合，これはユーザを検索するベース DN を指定するためにも使用されます（必須）。 例： "OU=Users,DC=your,DC=ldap,DC=domain,DC=com"	
providerURL	LDAP サーバへの URL。認証に使用されます。匿名検索が許可されている場合，これはユーザの検索にも使用されます （必須）。 例 : "ldap://your.ldap.domain.com:389" SSL の例： "ldaps://yourldap.domain.com:636"	
searchFirst	**true** に設定した場合，**allowAnonymous** も **true** に設定されていると，ユーザの匿名検索が実行されます。それ以外の場合，**uidAttribute**，ユーザのログイン名，**baseDN** 属性からユーザのプリンシパル DN を作成します。	"false"（標準設定） "true"
uidAttribute	ユーザのプリンシパル DN の作成に使用されます。匿名検索が許可されている場合，これは検索フィルタの作成にも使用されます。	"uid"（標準設定） 共通値 : "uid"，"CN"

ユーザのプリンシパル DN の作成例：

uidAttribute="UID"，ユーザ・ログイン名が jsmith，baseDN="OU=Users, DC=your,DC=ldap,DC=domain,DC=com" の場合，ユーザのプリンシパル DN は次のようになります。

"UID=jsmith,OU=Users,DC=your,DC=ldap,DC=domain,DC=com"

次の表に，**LDAPLoginModule の詳細モードの属性**を示します。

属性	説明と例	値
providerURL	認証に使用される LDAP サーバへの URL。ユーザの認証に使用されます。 例 : "ldap://yourldap.domain.com:389" SSL の例 : "ldaps://your.ldap.domain.com:636"	標準設定は, searchProviderURL 属性の値。
searchBaseDNs	検索フィルタに適用されるベース DN のセミコロン区切りリスト。(必須)。 例 : "DN=America,DN=ns,DN=root,DN=com; DN=asia,DN=ns,DN=root,DN=com; DN=europe,DN=ns,DN=root,DN=com" 委任の例 : "DN=ns,DN=root,DN=com"	

属性	説明と例	値
searchDN	認証するユーザ・プリンシパルの検索に使用されるプリンシパルの DN （必須）。指定していない場合でも searchFirst が true に設定されていると想定されます。 例： "CN=SearchAdmin,OU=Administrators,DC=americas,DC=ns,DC=root,DC=com"	
searchFilter	RFC 2254 に準拠した検索フィルタ（http://www.ief.org/rfc/rfc2254.txt（英語サイト）を参照）。フィルタの "{USERNAME}" 文字列は，ディレクトリの検索前にユーザのログイン名に置き換えられます。Active Directory に接続する場合，**jaas.configuration** ファイルに配置する前に **ldp.exe** を使用して検索フィルタをテストするのに便利です。（必須）。 例 1："(uid={USERNAME})" 例 2： "(&(CN={USERNAME})(objectClass=user))" 例 3： "(sAMAccountName={USERNAME})"	
searchFirst	**true** に設定されます。	"true"

属性	説明と例	値
searchPassword	**searchDN** 属性のパスワード。プレーン・テキストになるか, 暗号化されます（必須）。パスワードの暗号化については, 付録 C,「コンポーネント間での HTTPS 有効化」を参照してください。 例 : "Secret123" 暗号化の例 : "OBF:1fof1j1u1igh1ym51t331ym91idp1iz01fmn"	
searchProviderURL	ユーザのプリンシパル DN の検索に使用される LDAP サーバへの URL。これは，ユーザの検索に使用されます。 例 : "ldap://america.ns.root.com:389" SSL の例 : "ldaps://america.ns.root.com:636" 委任の例 : "ldaps://ns-root.com:636"	標準設定は，**providerURL** 属性の値。
searchReferral	**follow** に設定すると，LDAP サーバは検索または認証要求を解決できない場合に検索要求をほかの LDAP サーバに委任します。**follow** に設定した場合，フォレストのプリンシパル DN のみが **searchBaseDNs** にリストされる必要があります。	"ignore"（標準設定） "follow" "throw" （http://download.oracle.com/javase/1.5.0/docs/guide/jndi/jndi-ldap-gl.html#referral（英語サイト）を参照）

注 : server.properties や **jaas.configuration** ファイルに変更を加えた後, Diagnostics コマンダ・サーバを再起動する必要があります。

次の例は，すべてのユーザに "CN" で始まる同じベース DN があるため，検索することなくそのプリンシパル DB を決定できるという設定です。

```
Diagnostics {
  baseDN="OU=Users,DC=simple,DC=domain,DC=com"
  providerURL="ldap://simple.domain.com:389"
  uidAttribute="CN"
  ;
};
```

「larry」がログインする場合，そのプリンシパル DN は "CN=larry,OU=Users, DC=simple,DC=domain,DC=com" のようになります。

次の例は，すべてのユーザに "CN" で始まる同じベース DN があるため，検索することなくプリンシパル DB を決定できるが，どうしても検索したいという設定です。

```
Diagnostics {
  allowAnonymous="true"
  baseDN="OU=Users,DC=simple,DC=domain,DC=com"
  providerURL="ldap://simple.domain.com:389"
  searchFirst="true"
  uidAttribute="CN"
  ;
};
```

「sally」がログインする場合，そのプリンシパル DN は "CN=sally,OU=Users, DC=simple,DC=domain,DC=com" のようになります。

次の例は，世界中のユーザのうち 3 つの地域の IT 社員にのみ興味があるという設定です。

```
Diagnostics {
  searchFirst="true"
  searchReferral="follow"
  useSSL="true"
  serverCertificate="etc/key.store"
  searchProviderURL="ldaps://america.ns.root.com:636"
  searchDN="CN=Searcher,OU=Admins,DC=america,DC=ns,DC=root,DC=com"
  searchPassword="OBF:1fof1j1u1igh1ym51t331ym91idp1iz01fmn"
  searchFilter="(&(CN={USERNAME})(objectClass=IT))"
  searchBaseDNs="DC=america,DC=ns,DC=root,DC=com; ¥
                 DC=africa,DC=ns,DC=root,DC=com; ¥
                 DC=russia,DC=ns,DC=root,DC=com"
  ;
};
```

アフリカの「ororro」がログインする場合，そのプリンシパル DN は "CN=ororro, OU=IT,DC=africa,DC=ns,DC=root,DC=com" のようになります。

次の例は，世界中のユーザが異なる LDAP サーバでホストされている設定です。また，ユーザの CN は，ローカライズされる場合があるが，その sAMAccountNames は ISO 8859-1 であることが保証されています。

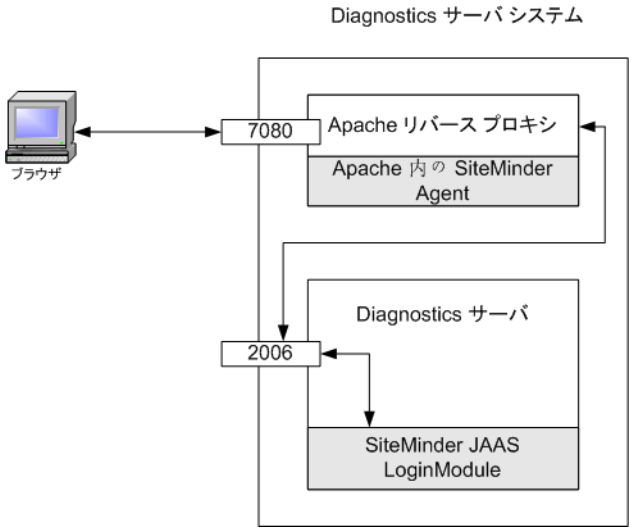
```
Diagnostics {
  searchFirst="true"
  searchReferral="follow"
  useSSL="true"
  serverCertificate="etc/key.store"
  searchProviderURL="ldaps://ns.root.com:636"
  searchDN="CN=Searcher,OU=Admins,DC=america,DC=ns,DC=root,DC=com"
  searchPassword="OBF:1fof1j1u1igh1ym51t331ym91idp1iz01fmn"
  searchFilter="(sAMAccountName={USERNAME})"
  searchBaseDNs="DC=ns,DC=root,DC=com"
  ;
};
```

ギリシャの Σαλοθ が sAMAccountName「Saloth」としてログインする場合，認証に使用されるそのプリンシパル DN は "CN=Σαλοθ,OU=Υσερζ,DC=greece, DC=ns,DC=root,DC=com" のようになります。

SiteMinder JAAS LoginModule でリバース・プロキシを使う

SiteMinder JAAS LoginModule では，SiteMinder Web エージェントがインストールされているリバース・プロキシ・サーバを必要とします。プロキシ・サーバは，HTTP/S 要求を単に Diagnostics サーバに転送します。

セットアップ例：



上図で，ポート 7080 の要求をリスンしている Apache Web サーバは，リバース・プロキシに設定されています。これには，SiteMinder Web エージェントが含まれます。このエージェントは，認証を実行して成功した場合に，Diagnostics サーバがリスンしているポート 2006（または，Diagnostics サーバが接続されているポート）を通じて要求を渡すことを Apache に許可します。

注 : リバース・プロキシが実行するリダイレクト，および SiteMinder モジュールが実行するリダイレクトとの競合を避けるために，［ログイン］ページをほかの Web サーバ（リバース・プロキシと異なる Web サーバ）に置くことをお勧めします。
または，Apache 2.2 で，ProxyPass の指示を使って特定の URL（ProxyPass / loginpage ! など）のプロキシを停止することもできます。詳細については, Apache 2.2 のドキュメントを参照してください。

Diagnostics サーバは，SiteMinder LoginModule を通じて SiteMinder からの要求を検出します。

注 : SiteMinder JAAS 認証を使うには，ユーザはリバース・プロキシのポートに移動する必要があります（上の例では, ポート 2006 ではなく 7080）。Diagnostics サーバと同じシステムにプロキシ・サーバがインストールされていない場合, URL には，Diagnostics Sever のコンピュータ名やローカルホストの代わりに，プロキシ・サーバのコンピュータ名を使う必要があります。

HP-UX への Apache リバース・プロキシのセットアップ例 : Apache 設定ファイル **httpd.conf** を編集して，次のプロパティを追加します。

- ProxyPass /siteminderagent !
- ProxyPass / http://＜Diagnostics サーバの IP アドレス＞:2006/

  （2006 は，Diagnostics サーバの標準設定ポート。Diagnostics サーバに設定したポートを使用します。）
- ProxyPassReverse / http://＜Diagnostics サーバの IP アドレス＞:2006/

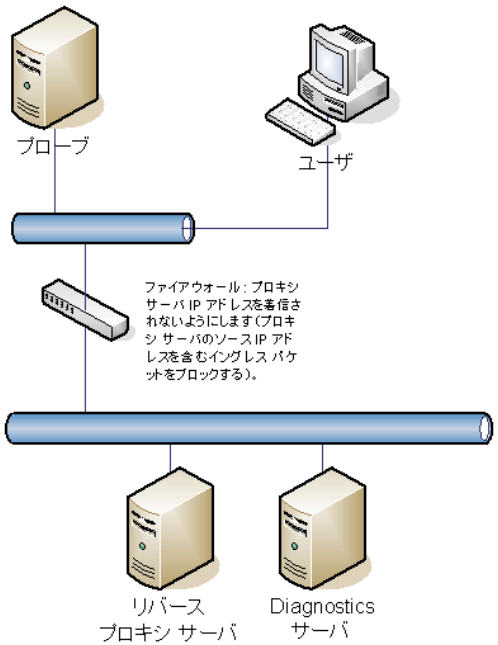
  （2006 は，Diagnostics サーバの標準設定ポート。Diagnostics サーバに設定したポートを使用します。）

注 : httpd.conf ファイルに変更を加えた後, Apache サーバを再起動する必要があります（apachectl を一度停止してから起動します）。

スプーフィングの心配がある場合は，任意で次の手順を行なってセキュリティを高めることができます。

➤ Diagnostics サーバと同じシステムにプロキシ・サーバがインストールされていない場合，同じサブネットに Diagnostics サーバとプロキシ・サーバを置いて，スイッチ / ルータのプロキシの IP アドレスにイングレス・フィルタを設定し，サブネット外からのリバース・プロキシの IP アドレスのスプーフィングを回避できます。

下の図を参照してください。



SiteMinder JAAS 認証の設定

Diagnostics に SiteMinder 認証を設定するには，Diagnostics コマンダ・サーバで次を設定する必要があります。

1 JAAS を使うように Diagnostics を設定します（778 ページ「JAAS を使用するための Diagnostics の設定」参照）。

2 **<インストール・ディレクトリ>/etc/webserver.properties** ファイルを編集します。

   a **authentication.header.filter.username** プロパティをコメントアウトします。ユーザ名の取得に使用する HTTP 要求ヘッダのフィールドに **authentication. header.filter.username** プロパティを設定します。標準設定で，これは **SM_UNIVERSALID** に設定されています。これは，SiteMinder がユーザ ID を含む HTTP 要求に作成するフィールドです。

   b Diagnostics の役割を使用するには，**authentication.header. filter.roles** プロパティのコメントアウトを解除します（この手順は任意です）。役割情報の取得に使用する HTTP ヘッダのフィールドに **authentication.header. filter.roles** プロパティを設定します。このフィールドには，カンマで区切って 1 つ以上の役割を含めることができます。defaultRoles も設定されている場合，結果の役割は defaultRoles とこれらの役割の結合になります。

3 **<インストール･ディレクトリ> /etc/jaas.configuration** ファイルの Diagnostics JAAS Realm（アプリケーション）ブロックを編集します。次に例を示します。

```
Diagnostics
{
    com.mercury.diagnostics.server.jaas.spi.SiteMinderLoginModule sufficient
    defaultRoles="Role1,Role2"
     ip="16.228.25.40";
};
```

SiteMinder LoginModule のオプション

次は, JAAS 設定ファイルで JAAS LoginModule に指定可能な全オプションのリストです。

オプション名	説明	必須 / 任意	標準設定値	例
IP	リバース・プロキシ・サーバの IP アドレス	必須		ip="16.228.25.40"
defaultRoles	認証された各ユーザを割り当てる役割のカンマ区切りリスト	任意		defaultRoles= "SuperUsers"

注 : server.properties，webserver.properties，jaas.configuration ファイルに変更を加えた後，Diagnostics コマンダ・サーバを再起動する必要があります。

C

コンポーネント間での HTTPS 有効化

Business Service Management を使用している場合に，HP Diagnostics コンポーネント間で HTTPS 通信を有効にするための設定手順について説明します。

本章の内容

注： これらの設定手順は，HP Diagnostics に関する深い知識を有する熟練ユーザを対象にしています。Diagnostics コンポーネントの構成設定を変更する際は十分に注意してください。

HTTPS 通信の設定について

各タイプのコンポーネントを設定するための説明には，次の主な手順の詳細が含まれます。

- コンポーネントにキーストアを作成する。
- キーストアから証明書をエクスポートする。
- キーストアへのアクセスを提供するパスワードを難読化する。
- 通信を開始する Diagnostics コンポーネントにコンポーネントの証明書をコピーする。
- コンポーネントのセキュリティ・プロパティを設定して SSL を有効にし，HTTPS 通信の実行に必要なパスワードを指定する。

注 : この情報を見直す際，付録 F,「Diagnostics の技術的な図」の「コンポーネント通信図」が参考になります。

暗号化の暗号スイートのフィルタ

暗号スイートは，HTTPS/SSL 暗号化に使用するセキュリティ・アルゴリズムおよびキー・サイズを定義します。サポートされる暗号スイートは，使用する Java のバージョンに基づきます。セキュリティが不十分であるか許可されていない特定の暗号を除外するポリシーが組織で設定されている可能性があります。**webserver.properties** ファイルで設定できる，暗号スイートに関連するプロパティが 2 つあります。

- **log.cipher.suites** : 暗号スイートに関するメッセージを **server.log** ファイルに記録します。これらのメッセージには，サポートされている暗号スイートおよび cipher.suites.filters の適用後に有効にされた暗号化スイートが含まれます。
- **cipher.suites.filters** : 暗号スイートをフィルタするために使用する，除外および包含正規表現のカンマ区切り値リスト。

たとえば，インストールで次の暗号スイートがリストされているとします。

```
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_DH_anon_WITH_RC4_128_MD5
```

このリストから 40 ビット，匿名，DES ベースの暗号スイートを除外し，RC4 ベースの暗号化のみを含めるとします。この場合，フィルタを次のように指定します。

```
cipher.suite.filters=\
  exclude:.*[0-9]?40[0-9]?.*, \
  exclude:.*_anon_.*, \
  exclude:.*_DES_.*, \
  exclude:.*_3DES_.*, \
  INCLUDE:.*_WITH_RC4_.*, \
  exclude:.*
```

Diagnostics コンポーネント別の HTTPS チェックリスト

次の表は，各 Diagnostics コンポーネントに対して HTTPS 通信を有効にするための設定手順のまとめです。

設定手順	コマンダ・サーバ	メディエータ・サーバ	Java プローブ	Collector	.NET プローブ
キーを作成して証明書をエクスポートする	**はい**	**はい**	**はい**	**はい**	いいえ
パスワードを難読化する	**はい**	**はい**	**はい**	**はい**	いいえ
コマンダ・サーバの証明書をコピーする	いいえ	**はい**	いいえ	いいえ	いいえ
メディエータ・サーバの証明書をコピーする	**はい**	いいえ	**はい**	**はい**	**はい**
Java プローブの証明書をコピーする	**はい**	**はい**	いいえ	いいえ	いいえ

設定手順	コマンダ・サーバ	メディエータ・サーバ	Java プローブ	Collector	.NET プローブ
Collector の証明書をコピーする	いいえ	**はい**	いいえ	いいえ	いいえ
security.properties を編集して，enablessl=true，keystorepassword，keypassword	**はい**	**はい**	**はい**	**はい**	いいえ
security.properties を編集して，コマンダ・サーバの証明書を trusted.certificates に追加する	いいえ	**はい**	いいえ	いいえ	いいえ
security.properties を編集して，メディエータ・サーバの証明書を trusted.certificates に追加する	**はい**	いいえ	**はい**	**はい**	いいえ
security.properties を編集して，Java プローブの証明書を trusted.certificates に追加する	**はい**	**はい**	いいえ	いいえ	いいえ
security.properties を編集して，Collector の証明書を trusted.certificates に追加する	いいえ	**はい**	いいえ	いいえ	いいえ
メディエータ・サーバの証明書を信頼されたルート証明機関にインポートする	いいえ	いいえ	いいえ	いいえ	**はい**
server.properties を編集して，commander.url を設定する	**はい**	**はい**	いいえ	いいえ	いいえ
dispatcher.properties を編集して，registrar.url を設定する	いいえ	いいえ	**はい**	いいえ	いいえ
collector.properties を編集して，registrar.url を設定する	いいえ	いいえ	いいえ	**はい**	いいえ
probe_configuration.xml を編集して，diagnosticsserver url，Mediator ホスト，metricport，および ssl を設定する	いいえ	いいえ	いいえ	いいえ	**はい**
metric.config を編集して，metrics.server.uri を設定する	いいえ	いいえ	いいえ	いいえ	**はい**

Diagnostics コンポーネントでの受信 HTTPS 通信の有効化

このセクションでは，受信 HTTPS 通信を受信するように Diagnostics サーバ，Java Agent，および Collector を設定するための手順を紹介します。HTTPS 通信は，Web ブラウザを使って Diagnostics コンポーネントにアクセスしたときから，またはほかの外部アプリケーションを使ってコンポーネントにアクセスしたときから，ほかの Diagnostics コンポーネントより受信できます。

本項の内容

クライアント証明書の生成

証明書サービスの詳細設定を使用してクライアント証明書を生成し，FQDN を指定します。キーをエクスポート可能にマークして，Friendly Name に CLIENT を指定します。

Diagnostics サーバの受信 HTTPS 接続用の設定

注 :

- DNS とホスト名の解決に関する問題を回避するために，Diagnostics コマンダ・サーバの Commander URL は **localhost** として設定する必要があります。これは，**＜ Server のインストール・ディレクトリ＞ /etc/server.properties** の commander.url プロパティを http://localhost:2006（または該当するポート番号）に設定することによってできます。
- Diagnostics コマンダ・サーバで HTTPS 通信を有効にする場合，次のようにポート 8443 を使って Enterprise UI を実行する必要があります。
  https:// ＜コマンダ・サーバ＞ :8443

Diagnostics サーバを受信 HTTPS 接続用に設定するには，次の手順を実行します。

1 **＜Diagnostics サーバのインストール・ディレクトリ＞/etc** ディレクトリにキーストアを作成します。次に，コマンド例を示します。

```
＜ Diagnostics サーバのインストール・ディレクトリ＞ /_jvm/bin/keytool -genkey
-keystore ＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/keystore
-storepass ＜パスワード＞ -alias SERVER -keyalg RSA -keypass ＜パスワード＞
-dname "CN= ＜ Diagnostics サーバのホスト名＞ , OU=Diagnostics,
O=Hewlett-Packard, L=Palo Alto, S=CA, C=USA" -validity 3650
```

このコマンド例を使用するには，次の手順を実行します。

- **＜ Diagnostics サーバのインストール・ディレクトリ＞**を Diagnostics サーバのインストール・ディレクトリのパスに置き換えます。
- **＜ Diagnostics サーバのホスト名＞** を Diagnostics サーバ のホストのマシン名に置き換えます（証明書のサブジェクト（CN）の完全修飾ドメイン名を使用する必要があります）。
- **＜パスワード＞** を，それぞれ同じパスワード文字列に置き換えます。**storepass** と **keypass** に異なるパスワードを割り当てることもできます。

このコマンドを実行すると，**＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/keystore** にキーストアと，Diagnostics サーバ・ホストに対応する **SERVER** というエントリが作成されます。

2 次のコマンドを使って，キーストアに SERVER エントリの証明書をエクスポートします。

```
＜ Diagnostics サーバのインストール・ディレクトリ＞ /_jvm/bin/keytool -export
-keystore ＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/keystore
-storepass ＜パスワード＞ -alias SERVER -rfc -file ＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/ ＜ Server の証明書名＞ .cer
```

このコマンドを使用するには，次の手順を実行します。

- **＜ Diagnostics サーバのインストール・ディレクトリ＞**を Diagnostics サーバのインストール・ディレクトリのパスに置き換えます。
- **＜パスワード＞** を，キーストアを作成したときに **storepass** のパスワードとして割り当てた文字列に置き換えます。
- **＜Server の証明書名＞** を証明書ファイルに割り当てる名前に置き換えます。証明書が作成されたコンポーネントを特定しやすい証明書名を割り当てることをお勧めします。

diag_server_commander.cer または **diag_server_mediator.cer** を使用します。

このコマンドの実行後，**＜ Server の証明書名＞**で割り当てた名前を持つ証明書ファイルが，Diagnostics サーバの **＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc** ディレクトリに作成されます。たとえば，**diag_server_commander.cer** のようになります。

注 : 証明書ファイルは，Diagnostics サーバとの通信を始めることが予想される各 Diagnostics コンポーネントのホスト・マシンにインポートする必要があります。各 Diagnostics コンポーネントに証明書ファイルをインポートする方法は，後で説明します。

---

3 次の例でコマンドを使用して，キーストアを作成したときに割り当てた **storepass** パスワードと **keypass** パスワードについて，難読化されたパスワードを作成します。

a **＜ Diagnostics サーバのインストール・ディレクトリ＞**を Diagnostics サーバのインストール・ディレクトリのパスに置き換えます。

b **＜パスワード＞** を，キーストアを作成したときにパスワードとして割り当てた文字列に置き換えます。

```
＜ Diagnostics サーバのインストール・ディレクトリ＞ /_jvm/bin/java -cp
＜ Diagnostics サーバのインストール・ディレクトリ＞ /lib/ThirdPartyLibs.jar
org.mortbay.util.Password ＜パスワード＞
```

難読化されたパスワードは，次の例のようになります。この例のパスワード文字列は testpass でした。この出力は 3 行で構成されます。難読化した元の文字列と，難読化されたパスワードを表す 2 行です。このプロセスの次の手順でプロパティを設定する際，OBF で始まる行だけが使用されます。

```
testpass
OBF:1ytc1vu91v2p1y831y7v1v1p1vv11yta
MD5:179ad45c6ce2cb97cf1029e212046e81
```

注 : keypass と **storepass** に同じパスワードを使わなかった場合，このコマンドを 2 回実行して，それぞれの難読化されたパスワードを作成する必要があります。

---

4 Diagnostics コマンダ・サーバでは，**＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/security.properties** ファイルにある次のプロパティを変更します。

a **enableSSL=true** に設定します。

b **keyStorePassword= ＜暗号化されたパスワード＞** を設定します。

c **keyPassword= ＜暗号化されたパスワード＞** を設定します。

注 : ＜暗号化されたパスワード＞に入力した値には，前の手順のコマンドから得た OBF 行全体を含める必要があります。たとえば，次のようになります。

keyStorePassword=OBF:1ytc1vu91v2p1y831y7v1v1p1vv11yta

Java Agent の受信 HTTPS 接続用の設定

注 :

- Java Agent に対する SSL および HTTPS 通信の有効化は，Sun，IBM および JRockit JVM を搭載した SUT でサポートされています。ただし，JVM 1.4 以前を使用している場合, SSL を有効にする前に SUT サーバに Sun JSSE Optional Package をダウンロードおよびインストールする必要があります。

  IBM などのほかの JSSE 実装はサポートされていません。

- Java Agent システムで HTTPS 通信を有効にする場合，次のようにポート 45000 を使って Profiler UI を実行する必要があります。
  https:// ＜プローブ・システム＞ :45000

注 : Agent がインストールされる場所が Diagnostics ＜プローブのインストール・ディレクトリ＞になります。標準設定では，Windows の場合は C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent,
UNIX の場合は /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent です。

Java Agent を着信 HTTPS 接続用に設定するには，次の手順を実行します。

1 次のコマンドを使って，**＜プローブのインストール・ディレクトリ＞ /etc** ディレクトリにキーストアを作成します。

```
/opt/MercuryDiagnostics/JavaAgent/_jvm/bin/keytool -genkey -keystore ＜プローブのインストール・ディレクトリ＞ /etc/keystore -storepass ＜パスワード＞ -alias PROBE -keyalg RSA -keypass ＜パスワード＞ -dname "CN=＜プローブのホスト名＞, OU=Diagnostics, O=Hewlett-Packard, L=Palo Alto, S=CA, C=USA" -validity 3650
```

このコマンド例を使用するには，次の手順を実行します。

- **＜プローブのインストール・ディレクトリ＞**を Java Agent のインストール・ディレクトリのパスに置き換えます。

- **＜プローブのホスト名＞** を Java Agent のホストのマシン名に置き換えます。この値をサーバの IP アドレスにすることはできません。証明書のサブジェクト（CN）の完全修飾ドメイン名を使用する必要があります。
- **＜パスワード＞** を，それぞれ同じパスワード文字列に置き換えます。**storepass** と **keypass** に異なるパスワードを割り当てることもできます。

このコマンドを実行すると，**＜プローブのインストール・ディレクトリ＞ /etc/keystore** にキーストアと，Java Agent のホストに対応する **PROBE** というエントリが作成されます。

2 次のコマンドを使って，キーストアに PROBE エントリの証明書をエクスポートします。

```
/opt/MercuryDiagnostics/JavaAgent/_jvm/bin/keytool -export -keystore ＜プローブのインストール・ディレクトリ＞ /etc/keystore -storepass ＜パスワード＞ -alias PROBE -rfc -file ＜プローブのインストール・ディレクトリ＞ /etc/ ＜プローブの証明署名＞ .cer
```

このコマンドを使用するには，次の手順を実行します。

- **＜プローブのインストール・ディレクトリ＞**を Java Agent のインストール・ディレクトリのパスに置き換えます。
- **＜パスワード＞** を，キーストアを作成したときに **storepass** のパスワードとして割り当てた文字列に置き換えます。
- **＜プローブの証明書名＞**を証明書ファイルに割り当てる名前に置き換えます。証明書が作成されたコンポーネントを特定しやすい証明書名を割り当てることをお勧めします。

  プローブのタイプとプローブのホスト名を含めて，証明書が作成されたコンポーネントを特定しやすくします。たとえば，次のようになります。**Java_probe_ ＜プローブのホスト名＞**

このコマンドの実行後，Java Agent の **＜プローブのインストール・ディレクトリ＞/etc** ディレクトリに **Java_probe_＜プローブのホスト名＞.cer** という証明書ファイルが作成されます。

注：証明書ファイルは，Java Agent との通信を始めることが予想される各 Diagnostics コンポーネントのホスト・マシンにインポートする必要があります。各 Diagnostics コンポーネントに証明書ファイルをインポートする方法は，後で説明します。

3 次の例でコマンドを使用して，キーストアを作成したときに割り当てた **storepass** パスワードと **keypass** パスワードについて，難読化されたパスワードを作成します。

a **＜プローブのインストール・ディレクトリ＞**を Java Agent のインストール・ディレクトリのパスに置き換えます。

b **＜パスワード＞** を，キーストアを作成したときにパスワードとして割り当てた文字列に置き換えます。

```
/opt/MercuryDiagnostics/JavaAgent/_jvm/bin/java -cp
＜プローブのインストール・ディレクトリ＞ /lib/ThirdPartyLibs.jar
org.mortbay.util.Password ＜パスワード＞
```

難読化されたパスワードは，次の例のようになります。この例のパスワード文字列は testpass でした。この出力は 3 行で構成されます。難読化した元の文字列と，難読化されたパスワードを表す 2 行です。このプロセスの次の手順でプロパティを設定する際，OBF で始まる行だけが使用されます。

```
testpass
OBF:1ytc1vu91v2p1y831y7v1v1p1vv11yta
MD5:179ad45c6ce2cb97cf1029e212046e81
```

注：keypass と **storepass** に同じパスワードを使わなかった場合，このコマンドを 2 回実行して，それぞれの難読化されたパスワードを作成する必要があります。

4 **＜プローブのインストール・ディレクトリ＞ /etc/security.properties** ファイルにある以下のプロパティを変更します。

a **enableSSL=true** に設定します。

b **keyStorePassword= ＜暗号化されたパスワード＞** を設定します。

c **keyPassword= ＜暗号化されたパスワード＞** を設定します。

注：＜暗号化されたパスワード＞に入力した値には，前の手順のコマンドから得た OBF 行全体を含める必要があります。たとえば，次のようになります。

keyStorePassword=OBF:1ytc1vu91v2p1y831y7v1v1p1vv11yta

Collector の受信 HTTPS 接続用の設定

このセクションでは，受信 HTTPS 接続を受信するように Collector を設定するための手順を紹介します。

Collector を受信 HTTPS 接続用に設定するには，次の手順を実行します。

1 次のコマンドを使って，**＜ Collector のインストール・ディレクトリ＞ /etc** ディレクトリにキーストアを作成します。

```
＜ Collector のインストール・ディレクトリ＞ /_jvm/bin/keytool -genkey -keystore
＜ Collector のインストール・ディレクトリ＞ /etc/keystore -storepass ＜パスワード＞
-alias COLLECTOR -keyalg RSA -keypass ＜パスワード＞ -dname "CN= ＜ Collector
のホスト名＞ , OU=Diagnostics, O=Hewlett-Packard, L=Palo Alto, S=CA, C=USA"
-validity 3650
```

このコマンド例を使用するには，次の手順を実行します。

- **＜ Collector のインストール・ディレクトリ＞** を Collector のインストール・ディレクトリのパスに置き換えます。
- **＜ Collector のホスト名＞** を Collector のホストのマシン名に置き換えます。この値をサーバの IP アドレスにすることはできません。証明書のサブジェクト（CN）の完全修飾ドメイン名を使用する必要があります。

➤ **＜パスワード＞** を，それぞれ同じパスワード文字列に置き換えます。**storepass** と **keypass** に異なるパスワードを割り当てることもできます。

このコマンドを実行すると，**＜ Collector のインストール・ディレクトリ＞ /etc/keystore** にキーストアと，Collector のホストに対応する **COLLECTOR** というエントリが作成されます。

2 次のコマンドを使って，キーストアに COLLECTOR エントリの証明書をエクスポートします。

```
＜ Collector のインストール・ディレクトリ＞ /_jvm/bin/keytool -export -keystore
＜ Collector のインストール・ディレクトリ＞ /etc/keystore -storepass ＜パスワード＞ -alias COLLECTOR -rfc -file ＜ Collector のインストール・ディレクトリ＞ /etc/ ＜ Collector の証明書名＞ .cer
```

このコマンドを使用するには，次の手順を実行します。

➤ **＜ Collector のインストール・ディレクトリ＞** を Collector のインストール・ディレクトリのパスに置き換えます。

➤ **＜パスワード＞** を，キーストアを作成したときに **storepass** のパスワードとして割り当てた文字列に置き換えます。

➤ **＜Collector の証明書名＞** を証明書ファイルに割り当てる名前に置き換えます。証明書が作成されたコンポーネントを特定しやすい証明書名を割り当てることをお勧めします。

Collector のタイプと Collector のホスト名を含めて，証明書が作成されたコンポーネントを特定しやすくします。たとえば，次のようになります。**collector_ ＜ Collector のホスト名＞**

このコマンドの実行後，Collector の **＜ Collector のインストール・ディレクトリ＞ /etc** ディレクトリに **collector_ ＜ Collector のホスト名＞ .cer** という証明書ファイルが作成されます。

注 : 証明書ファイルは，Collector との通信を始めることが予想される各 Diagnostics コンポーネントのホスト・マシンにインポートする必要があります。各 Diagnostics コンポーネントに証明書ファイルをインポートする方法は，後で説明します。

---

3 次の例でコマンドを使用して，キーストアを作成したときに割り当てた **storepass** パスワードと **keypass** パスワードについて，難読化されたパスワードを作成します。

a **＜Collector のインストール・ディレクトリ＞** を Collector のインストール・ディレクトリのパスに置き換えます。

b **＜パスワード＞** を，キーストアを作成したときにパスワードとして割り当てた文字列に置き換えます。

```
＜Collector のインストール・ディレクトリ＞/_jvm/bin/java -cp
＜Collector のインストール・ディレクトリ＞/lib/ThirdPartyLibs.jar
org.mortbay.util.Password ＜パスワード＞
```

難読化されたパスワードは，次の例のようになります。この例のパスワード文字列は testpass でした。この出力は 3 行で構成されます。難読化した元の文字列と，難読化されたパスワードを表す 2 行です。このプロセスの次の手順でプロパティを設定する際，OBF で始まる行だけが使用されます。

```
testpass
OBF:1ytc1vu91v2p1y831y7v1v1p1vv11yta
MD5:179ad45c6ce2cb97cf1029e212046e81
```

注 : keypass と **storepass** に同じパスワードを使わなかった場合，このコマンドを 2 回実行して，それぞれの難読化されたパスワードを作成する必要があります。

---

4 **＜Collector のインストール・ディレクトリ＞/etc/security.properties** ファイルにある以下のプロパティを変更します。

a **enableSSL=true** に設定します。

b **keyStorePassword= ＜暗号化されたパスワード＞** を設定します。

c keyPassword= ＜暗号化されたパスワード＞ を設定します。

注 : ＜暗号化されたパスワード＞に入力した値には, 前の手順のコマンドから得た OBF 行全体を含める必要があります。たとえば，次のようになります。

keyStorePassword=OBF:1ytc1vu91v2p1y831y7v1v1p1vv11yta

Diagnostics コンポーネントからの発信 HTTPS 通信の有効化

ここでは，Diagnostics コンポーネントを設定して発信 HTTPS 通信をほかの Diagnostics コンポーネントに送信するための手順を紹介します。

注 : Agent がインストールされる場所が Diagnostics ＜プローブのインストール・ディレクトリ＞になります。標準設定では，Windows の場合は C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent, UNIX の場合は /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent です。

Diagnostics コマンダ・サーバで，HTTPS を使用した Diagnostic メディエータ・サーバへの発信通信を有効にするには，次の手順を実行します。

1 Diagnostics サーバの **＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** から，Diagnostics コマンダ・サーバの **＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** に証明書ファイルをコピーします。

2 Diagnostics コマンダ・サーバの **＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/security.properties** ファイルにある **trusted.certificate** プロパティの値を変更します。

3 **trusted.certificate=diag_server_mediator.cer** を設定します。このプロパティの値に，すでにほかの証明書ファイルがある場合は，リストの末尾に，前の値とカンマで区切って証明書ファイルを追加します。

4 受信 Diagnostics サーバ通信の場合，Diagnostics コマンダ・サーバの **＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/server.properties** ファイルにある次のプロパティを更新して，Diagnostics サーバの URL を指定します。

commander.url を **https://＜Diagnostics コマンダ・サーバのホスト名＞:8443** に設定します。

Diagnostics Mediator Server で，HTTPS を使用した Diagnostics コマンダ・サーバへの発信通信を有効にするには，次の手順を実行します。

1 Diagnostics コマンダ・サーバの **＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_commander.cer** から，Diagnostic メディエータ・サーバの **＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_commander.cer** に証明書ファイルをコピーします。

2 Diagnostic メディエータ・サーバの **＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/security.properties** ファイルにある **trusted.certificate** プロパティの値を変更します。

3 **trusted.certificate=diag_server_commander.cer** を設定します。このプロパティの値に，すでにほかの証明書ファイルがある場合は，リストの末尾に，前の値とカンマで区切って証明書ファイルを追加します。

4 受信 Diagnostics サーバ通信の場合，Diagnostics サーバ（Mediator モード）の **＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/server.properties** ファイルにある次のプロパティを更新して，Diagnostics サーバの URL を指定します。

commander.url を **https://＜Diagnostics コマンダ・サーバのホスト名＞:8443** に設定します。

注 : Server で HTTPS を有効にする場合，URL でポート 8443 を使って Diagnostics UI を実行する必要があります。

Diagnostics サーバ（Commander または Mediator モード）で，HTTPS を使用したプローブへの発信通信を有効にするには，次の手順を実行します。

1 各プローブの **＜プローブのインストール・ディレクトリ＞ /etc/java_probe_ ＜プローブのホスト＞ .cer** から Diagnostics サーバ の **＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/Java_probe_ ＜プローブのホスト＞ .cer** に証明書ファイルをコピーします。

2 Diagnostics サーバ の **＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/security.properties** ファイルにある **trusted.certificate** プロパティの値を変更します。

trusted.certificate=Java_probe_ ＜ probe_host ＞ .cer を設定します。このプロパティの値に，すでにほかの証明書ファイルがある場合は，リストの末尾に，前の値とカンマで区切って証明書ファイルを追加します。

Diagnostics サーバ（Mediator モード）で，HTTPS を使用した Collector への発信通信を有効にするには，次の手順を実行します。

1 各プローブの **＜ Collector のインストール・ディレクトリ＞ /etc/collector_ ＜ Collector のホスト＞ .cer** から，Diagnostics サーバの**＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/collector_ ＜ Collector のホスト＞ .cer** に証明書ファイルをコピーします。

2 Diagnostics サーバの ＜ **Diagnostics サーバのインストール・ディレクトリ＞ /etc/security.properties** ファイルにある trusted.certificate プロパティの値を変更します。

trusted.certificate=collector_ ＜ Collector のホスト＞ .cer を設定します。このプロパティの値に，すでにほかの証明書ファイルがある場合は，リストの末尾に，前の値とカンマで区切って証明書ファイルを追加します。

Java Agent で，HTTPS を使用した Diagnostic メディエータ・サーバへの発信通信を有効にするには，次の手順を実行します。

1 Diagnostic メディエータ・サーバの **＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** から，Java Agent の **＜プローブのインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** に証明書ファイルをコピーします。

2 Java Agent の **＜プローブのインストール・ディレクトリ＞ /etc/security.properties** ファイルにある **trusted.certificate** プロパティの値を変更します。

trusted.certificate=diag_server_mediator.cer を設定します。このプロパティの値に，すでにほかの証明書ファイルがある場合は，リストの末尾に，前の値とカンマで区切って証明書ファイルを追加します。

3 着信 Java Agent 通信の場合，**＜プローブのインストール・ディレクトリ＞ /etc/dispatcher.properties** ファイルの以下のプロパティを更新して，Diagnostic メディエータ・サーバの URL を指定します。

registrar.url を **https://＜Diagnostics サーバ（Mediator モード）のホスト名＞:8443/commander/registrar/** に設定します。

注 : Server で HTTPS を有効にする場合，URL でポート 8443 を使って Diagnostics UI を実行する必要があります。

サーバ /Collector の埋め込み Java プローブで，HTTPS を使用した Diagnostics サーバ（Mediator モード）への発信通信を有効にするには，次の手順を実行します。

1 Diagnostics サーバ（Mediator モード）の **＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** から，埋め込み Java プローブの **＜サーバ /Collector のインストール・ディレクトリ＞ /probe/etc/diag_server_mediator.cer** に証明書ファイルをコピーします。

2 埋め込み Java プローブの **＜サーバ /Collector のインストール・ディレクトリ＞ /probe/etc/security.properties** ファイルにある **trusted.certificate** プロパティの値を変更します。**trusted.certificate=diag_server_mediator.cer** を設定します。このプロパティの値に，すでにほかの証明書ファイルがある場合は，リストの末尾に，前の値とカンマで区切って証明書ファイルを追加します。

3 受信の埋め込み Java プローブ通信の場合は，**＜サーバ /Collector のインストール・ディレクトリ＞/probe/etc/collector.properties** ファイルの次のプロパティを更新して，Diagnostics サーバ（Mediator モード）の URL を指定します。**registrar.url** を **https:// ＜ Diagnostics サーバ（Mediator モード）のホスト名＞:8443/commander/registrar/** に設定します。

Collector で，HTTPS を使用した Diagnostic メディエータ・サーバへの発信通信を有効にするには，次の手順を実行します。

1 Diagnostic メディエータ・サーバの **＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** から，Collector の **＜Collector のインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** に証明書ファイルをコピーします。

2 Collector の **＜Collector のインストール・ディレクトリ＞ /etc/security.properties** ファイルにある **trusted.certificate** プロパティの値を変更します。

trusted.certificate=diag_server_mediator.cer を設定します。このプロパティの値に，すでにほかの証明書ファイルがある場合は，リストの末尾に，前の値とカンマで区切って証明書ファイルを追加します。

3 着信 Collector 通信の場合，**＜Collector のインストール・ディレクトリ＞ /etc/collector.properties** ファイルの以下のプロパティを更新して，Diagnostic メディエータ・サーバの URL を指定します。

registrar.url を **https://＜Diagnostics サーバ(Mediator モード)のホスト名＞:8443/commander/registrar/** に設定します。

注 : Server で HTTPS を有効にする場合，URL でポート 8443 を使って Diagnostics UI を実行する必要があります。

.NET Agent で，HTTPS を使用した Diagnostics コマンダ・サーバへの発信通信を有効にするには，次の手順を実行します。

1 Diagnostic メディエータ・サーバの証明書を .NET Agent のホストにコピーします。証明書は，Diagnostic メディエータ・サーバで HTTPS を受信するように設定したときに作成されています。Diagnostic メディエータ・サーバ で HTTPS を受信するように設定する方法については，801 ページ「Diagnostics コンポーネントでの受信 HTTPS 通信の有効化」を参照してください。その指示に従って設定した場合，証明書は **＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** で見つけることができます。

2 Windows タスクバーで，［**スタート**］＞［**ファイル名を指定して実行**］を選択します。

3 mmc と入力し，［**OK**］をクリックして Microsoft 管理コンソールを実行します。

4 Micorosoft 管理コンソールのメニューで，［**ファイル**］＞［**スナップインの追加と削除**］をクリックして［スナップインの追加と削除］ダイアログを表示します。

5 ［スナップインの追加と削除］ダイアログで［**追加**］をクリックします。

6 ［利用できるスタンドアロン スナップイン］リストから［**証明書**］を選択して，［**追加**］をクリックします。

7 ［証明書スナップイン］ダイアログ・ボックスで［**コンピュータ アカウント**］を選択して，［**次へ**］をクリックします。

8 ［コンピュータの選択］ダイアログ・ボックスで，［**ローカル コンピュータ：（このコンソールを実行しているコンピュータ）**］を選択して，［**完了**］をクリックします。

9 ［スタンドアロン スナップインの追加］ダイアログで［**閉じる**］をクリックします。

10 ［スナップインの追加と削除］ダイアログで［**OK**］をクリックします。

11 Microsoft 管理コンソールでは，［コンソール ルート］ダイアログの左側のペインに，証明書（ローカル・コンピュータ）のリストが展開されます。

12 ［証明書（ローカル コンピュータ）］で，［信頼されたルート証明機関］を展開します。

13 ［信頼されたルート証明機関］で［証明書］を右クリックし，［**すべてのタスク**］＞［**インポート**］を選択して［証明書のインポート ウィザード］を開きます。

14 ［**次へ**］をクリックして，［証明書のインポート ウィザード］の次の画面に進みます。

15 ［**参照**］をクリックして，Diagnostic メディエータ・サーバの公開キーストアに移動します。

   a ［ファイルの種類］で［すべてのファイル (*.*)］を選択します。

   b 手順 1 で Diagnostics コマンダ・サーバのキーストアをコピーしたディレクトリに移動して［開く］をクリックします。このディレクトリは次のようになります。**＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/diag_server_mediator.cer**

16 ［**次へ**］をクリックしてファイルをインポートします。

17 ［**次へ**］をクリックして，「信頼されたルート証明機関」のデフォルトの証明書の保存場所を受け入れます。

18 ［証明書のインポート ウィザードの完了］で［**完了**］をクリックします。

19 ［証明書のインポート ウィザード］の確認ダイアログで［**OK**］をクリックします。

20 ［信頼されたルート証明機関］で［**証明書**］を選択し，追加した証明書を見つけます（メディエータ・サーバのホスト名になっているはずです）。［**Issued to**］列の値を書き留めます。この値は，プローブ設定ファイルを変更する際に使用します。

21 **＜プローブのインストール・ディレクトリ＞ /etc/probe_config.xml** を編集して，**diagnosticsserver url** プロパティを HTTPS URL: **＜ diagnosticsserver url="https:// ＜ Diagnostics Mediator Server のホスト名＞ : 8443/commander" / ＞**を使用するように変更します。

22 Mediator ホストおよびポートを変更して, ssl="true" を追加します。**＜mediator host=" ＜ Diagnostics Mediator Server のホスト名＞ " port="2612" metricport="8443" ssl="true"/ ＞**

23 **＜プローブのインストール･ディレクトリ＞ /etc/metrics.config** を編集します。**metrics.server.uri** の値を変更して，HTTPS URL: **metrics.server.uri = https:// ＜ Diagnostics Mediator Server のホスト名＞ :8443/metricdata/** を指定します。

注 : probe_config.xml および **metrics.config** ファイルの両方で, **＜Diagnostics Mediator Server のホスト名＞** 値は，証明書に表示される名前に一致する必要があります。たとえば，証明書のホスト名が完全修飾の場合，設定ファイルのホスト名も完全修飾のはずです。

---

24 IIS を再起動します。IIS の再起動については，281 ページ「検出および標準インストルメンテーション」を参照してください。

Diagnostics コマンダ・サーバとの HTTPS 通信が .NET プローブに正常に設定されたことを確認するには，次の手順を実行します。

1 .NET アプリケーションを参照して .NET Agent をアクティブにします。

2 Diagnostics UI の［システムの状況］ビューをチェックして，.NET Agent が利用可能であることを確認します。

Business Service Management サーバの HTTPS 通信の有効化

次に，Business Service Management に Diagnostics との HTTPS 通信を設定するための手順を示します。

Diagnostics コマンダ・サーバと Business Service Management の間で HTTPS 通信を有効にするには，次の手順を実行します。

1 Diagnostics コマンダ・サーバのインストール・ディレクトリ**＜Diagnostics サーバのインストール・ディレクトリ＞ /etc/** から Business Service Management のホストに，Diagnostics 証明書ファイル **diag_server_commander.cer** をコピーします。

2 Business Service Management のホストで次のコマンドを実行して，コピーした証明書 **diag_server_commander.cer** を Business Service Management サーバ cacart キーストアにインポートします。

```
＜BAC サーバのインストール・ディレクトリ＞/_jvm/bin/keytool -import -file
＜コピーした Diagnostics 証明書のディレクトリ＞/diag_server_commander.cer
-keystore ＜BAC サーバのインストール・ディレクトリ＞/jre/lib/security/cacerts
-alias SERVER
```

- **＜BAC サーバのインストール・ディレクトリ＞** を Business Service Management のインストール・ディレクトリのパスに置き換えます。
- **＜コピーした Diagnostics 証明書のディレクトリ＞** をコピーした Diagnostics 証明書ファイルのパスに置き換えます。

キーストアのパスワードを入力するプロンプトが表示されたら changeit と入力します。

証明書が信頼できるかどうかを尋ねられたら，デフォルトの no の代わりに yes と入力します。

3 Business Service Management の証明書ファイル **BAC_certificate_file.cer** を Diagnostics サーバのホストにコピーします。

4 Diagnostics サーバのホストで次のコマンドを実行して，コピーした証明書を Diagnostics サーバの cacert キーストアにインポートします。

```
＜Diagnostics サーバのインストール・ディレクトリ＞/_jvm/bin/keytool -import -file
＜コピーした Bac 証明書のディレクトリ＞/＜BAC_certificate_file.cer＞ -keystore
＜Diagnostics サーバのインストール・ディレクトリ＞/JRE/lib/security/cacerts
```

- **＜Diagnostics サーバのインストール・ディレクトリ＞** を Diagnostics コマンダ・サーバのインストール・ディレクトリのパスに置き換えます。
- **＜コピーした BAC 証明書のディレクトリ＞** をコピーした Business Service Management 証明書ファイルのパスに置き換えます。

キーストア・パスワードの入力を求められたら，キーストアの作成時に **storepass** パスワードとして割り当てた文字列を入力します。

証明書が信頼できるかどうかを尋ねられたら，デフォルトの no の代わりに yes と入力します。

5 Business Service Management サーバを Diagnostics コマンダ・サーバの HTTPS ポートに指定します。

a ［**管理**］>［**Diagnostics**］を選択して，Business Service Management で Diagnostics の［管理］を開きます。

b ［**登録**］タブをクリックします。

c Diagnostics サーバの情報セクションを特定します。

d 適切なフィールドに以下の情報を入力します。

- Diagnostics コマンダ・サーバのホスト名に，Diagnostics コマンダ・サーバのキーストアを作成する際に **CN** パラメータに入力したものと同じ値を入力します。証明書のサブジェクト（CN）の**完全修飾ドメイン名**が使用されている必要があります。詳細については，801 ページ「Diagnostics コンポーネントでの受信 HTTPS 通信の有効化」を参照してください。
- プロトコルに HTTPS と入力します。
- Diagnostics サーバの Web ポートに 8443 と入力します。

e ［**Submit Configuration**］をクリックします。

D

管理者用のシステム・ビューの使用

Diagnostics システム・ビューを使って，Diagnostics コンポーネントの状況を監視したり，それらのコンポーネントが正常に機能していることを確認したりできます。

本章の内容

Diagnostics の管理者用のシステム・ビュー

大規模な Diagnostics デプロイメントでは，システムの状況モニタの代わりにシステム・ビューを使用できます。専用のシステム・ビューでは，さまざまなシステム属性に基づいてシステムまたはシステム・グループをすばやく特定できます。システム状況を監視して，システムが限界に近くなるタイミングを特定することが容易になっています。

多くの場合，システム・ビューは，Diagnostics デプロイメントのコンポーネントおよびそれらをホストするマシンの情報を把握する必要があるときに，最初にするべき，唯一の手段となります。ユーザは，問題のあるコンポーネントを一目で確認できます。

システム・ビューにアクセスするには，次の手順を実行します。

1 http://＜Diagnostics コマンド･サーバ名＞:2006/query/ から Mercury System カスタマとして Diagnostics UI を開きます。

2 クエリ･ページで, リスト内から Mercury System カスタマを見つけて, Diagnostics を開くリンクを選択します。

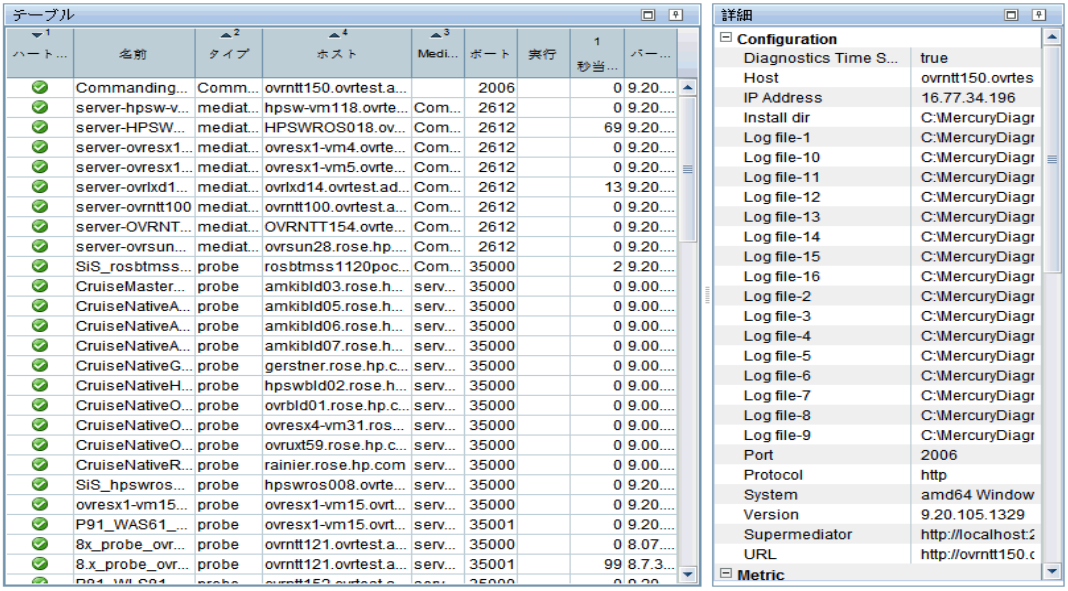
3 Diagnostics にログインして，［アプリケーション］ウィンドウで［全組織］を選択し，Diagnostics ビューを開くためのリンクを選択します。

4 ［ビュー］表示枠にシステム・ビューのビュー・グループが表示されます。ビュー・グループを開き，システムの状況ビューまたはシステム・キャパシティ・ビューを選択します。

注： システムの状況モニタは，互換性を確保するために引き続き使用でき，Performance Center，LoadRunner Controller，Business Service Management からアクセスできます。システムの状況モニタにアクセスするには，システム権限が必要です。Diagnostics では，http:// ＜ Diagnostics コマンド・サーバのホスト＞ : ＜ Diagnostics コマンド・サーバのポート＞ /registrar/health からアクセスできます。

［システムの状況］ビューの説明

Diagnostics UI の［システムの状況］ビューには，Diagnostics 環境にインストールしたコンポーネントの全体的な状況の情報があります。システムの状況モニタで提供される情報と類似していますが，目的のメトリックスや属性の整理，並べ替え，選択を行うことができます。



［システム キャパシティ］ビューの説明

Diagnostics UI の［システム キャパシティ］ビューには，Diagnostics 環境のキャパシティを管理するための情報があります。このビューには，各 Diagnostics Mediator に割り当てられているプローブ・グループやプローブの数が表示されます。

名前	ホスト	ポート	プローブ グループの数量	プローブの数量
⊟ server-ovresx1-vm5	ovresx1-vm5.ovrtest.adapps.hp.com	2612	4	8

プローブ グループ名	プローブの数量
Sanity_LR_9_1_ovresx1-vm5	3
Sanity_CallChain_WebService_ovresx1-vm5	3
Sanity_Collectors_ovresx1-vm5	1
Sanity_RUM_Engines	1

⊞ server-HPSWROS018	HPSWROS018.ovrtest.adapps.hp.com	2612	4	7
⊞ server-ovrntt100	ovrntt100.ovrtest.adapps.hp.com	2612	3	10
⊞ server-ovrsun28.rose.hp.com	ovrsun28.rose.hp.com	2612	2	10
⊞ CommandingServer	ovrntt150.ovrtest.adapps.hp.com	2006	2	2
⊞ server-ovrlxd14.ovrtest.adapps.hp.c...	ovrlxd14.ovrtest.adapps.hp.com	2612	1	8
⊞ server-OVRNTT154	OVRNTT154.ovrtest.adapps.hp.com	2612	2	4
⊞ server-hpsw-vm118.ovrtest.adapps....	hpsw-vm118.ovrtest.adapps.hp.com	2612	1	10
⊞ server-ovresx1-vm4.ovrtest.adapps....	ovresx1-vm4.ovrtest.adapps.hp.com	2612	5	13

E

Diagnostics のデータ管理

Diagnostics データの管理および保存方法について詳しく説明します。

本章の内容

Diagnostics データについて

Diagnostics データは，主に 2 種類あります。

- 各ユーザが作成したカスタム・ビュー。
- プローブによって収集され，Diagnostics サーバによって集計されるデータ。このデータは，Diagnostics サーバの時系列のデータベースに保存されます。

各 Diagnostics では，報告を行うプローブによって収集されるデータを保存します。また，Diagnostics コマンド・サーバは，LoadRunner/Performance Center の実行と Business Service Management やアプリケーション・メトリックスの両方の仮想トランザクションのデータを保存します。この章では，データベースを構成するデータ・ファイルの構成およびメンテナンスについて説明します。

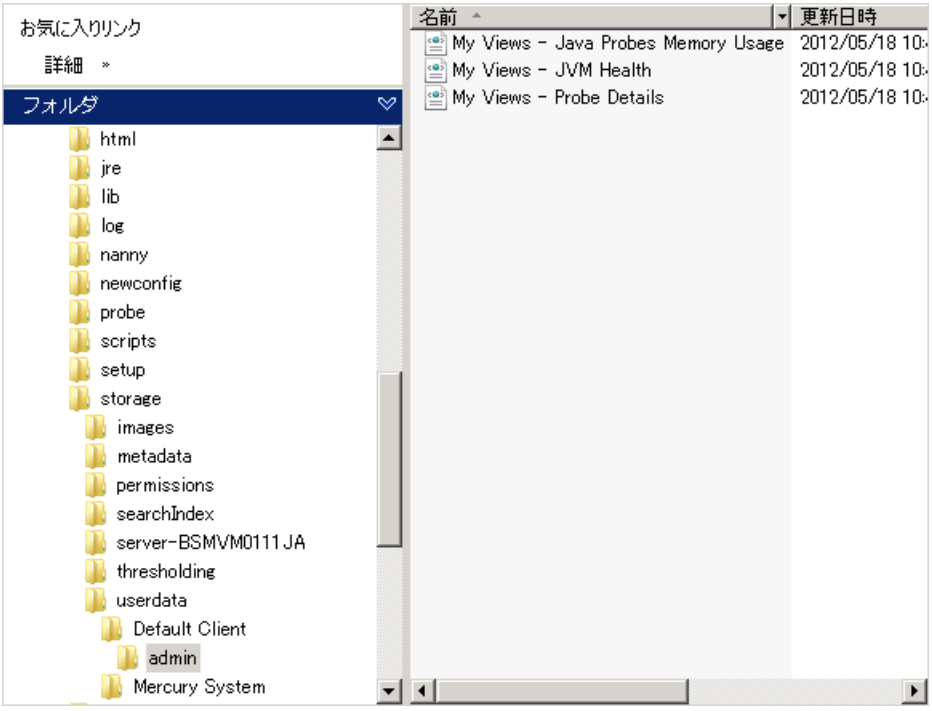
カスタム・ビュー・データ

Diagnostics ユーザは，『HP Diagnostics ユーザー・ガイド』の Diagnostics のビューのカスタマイズに関する章の説明に従って，カスタマイズ・ビューを作成および保存できます。Diagnostics では，カスタマイズ・ビューを Diagnostics コマンド・サーバのホストに XML ファイルとして保存します。

カスタム・ビュー・データの構成

ユーザ定義のカスタム・ビューは，Diagnostics コマンド・サーバのホストの **＜Diagnostics サーバのインストール・ディレクトリ＞ /storage/userdata** ディレクトリに XML ファイルとして保存されます。カスタム・ビュー・ファイルは，比較的小さなファイルです。

カスタム・ビューを定義した各ユーザには，**userdata** ディレクトリに独自のカスタム・ビュー・サブディレクトリがあります。たとえば，**admin** ユーザが Sales Status と Host Status の 2 つのカスタム・ビューを作成した場合，2 つのビューは Diagnostics コマンド・サーバの **＜ Diagnostics サーバのインストール・ディレクトリ＞ /storage/userdata/Default Client/admin** ディレクトリに独立した .xml ファイルとして保存されます（次の例を参照）。



パフォーマンス履歴データ

Diagnostics は，Diagnostic メディエータ・サーバの時系列のデータベース（TSDB）にパフォーマンス履歴データを保存します。Diagnostics サーバに，報告を行う数多くのプローブがある場合，保存されたパフォーマンス履歴データは数ギガバイトに及ぶ可能性があります。アプリケーションごとに収集されるデータ量はサイズが異なりますが, 監視している仮想マシンごとに約 3 GB を計画しておくことをお勧めします。詳細については，834 ページ「データの保存」を参照してください。

本項の内容

パフォーマンス履歴データの構成

Diagnostics パフォーマンス履歴データは，Diagnostic メディエータ・サーバの **＜Diagnostics サーバのインストール・ディレクトリ＞ /archive/mediator-＜ホスト名＞ /persistence/ ＜カスタマ名＞** _ ディレクトリにあります。

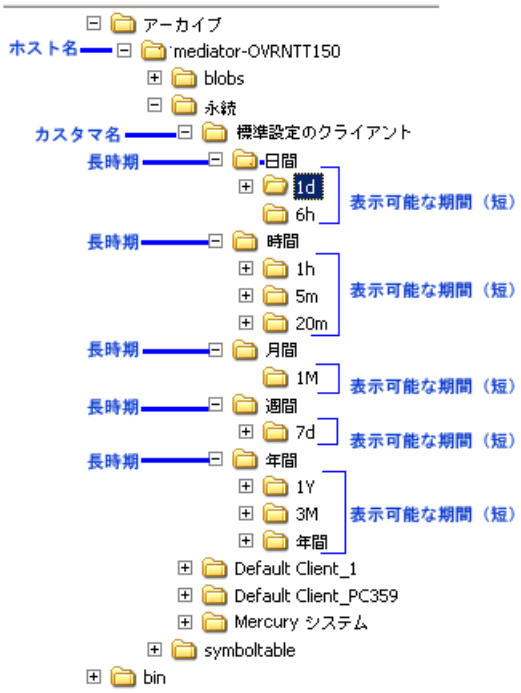
- **＜ホスト名＞**は，Diagnostic メディエータ・サーバのホストの名前です。
- **＜カスタマ名＞** は，Diagnostic メディエータ・サーバをインストールしたときに入力したカスタマ名です。このディレクトリの名前は，アンダーラインが付いたカスタマ名です。

etc/server.properties のアーカイブ **archive.dirname** プロパティでは，アーカイブの保存先を指定します（< Server のインストール・ディレクトリ >の絶対パスまたは相対パス)。アーカイブを NAS ドライブに移動または保存する場合（// < ホスト名 > / < 共有名 > など），**archive.dirname** で設定する共有に Diagnostics を実行するユーザの読み取り / 書き込み権限が必要になります。

注 :

- HP Software-as-a-Service（SaaS）カスタマでないかぎり，カスタマ名は必ず**標準設定のクライアント**になります。
- Performance Center または LoadRunner の実行に関する Diagnostics Performance 履歴データは，**../persistence/ ＜カスタマ名＞ _ ＜ run identifier ＞ ディレクトリ**に保存されます。

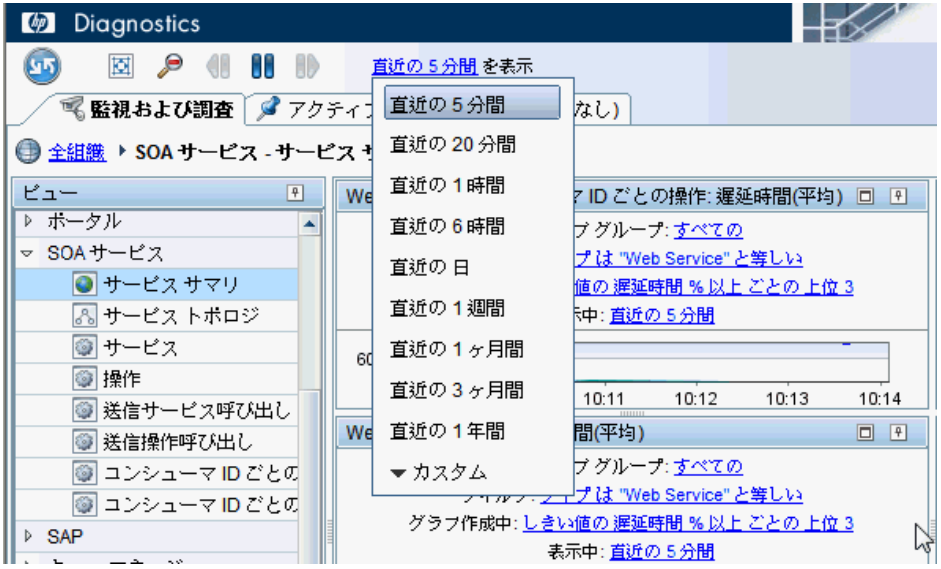
/persistence ディレクトリでは，パフォーマンス・データは次のようなディレクトリに編成されます。



ディレクトリ・レベルは，**長期間**（日，時間，月，週，年）を表し，サブディレクトリは**短期間**（1 日，6 時間，1 時間，20 分，5 分，1 か月，7 日，1 年，3 か月，数年）を表します。これらの期間は，データの粒度も表します。

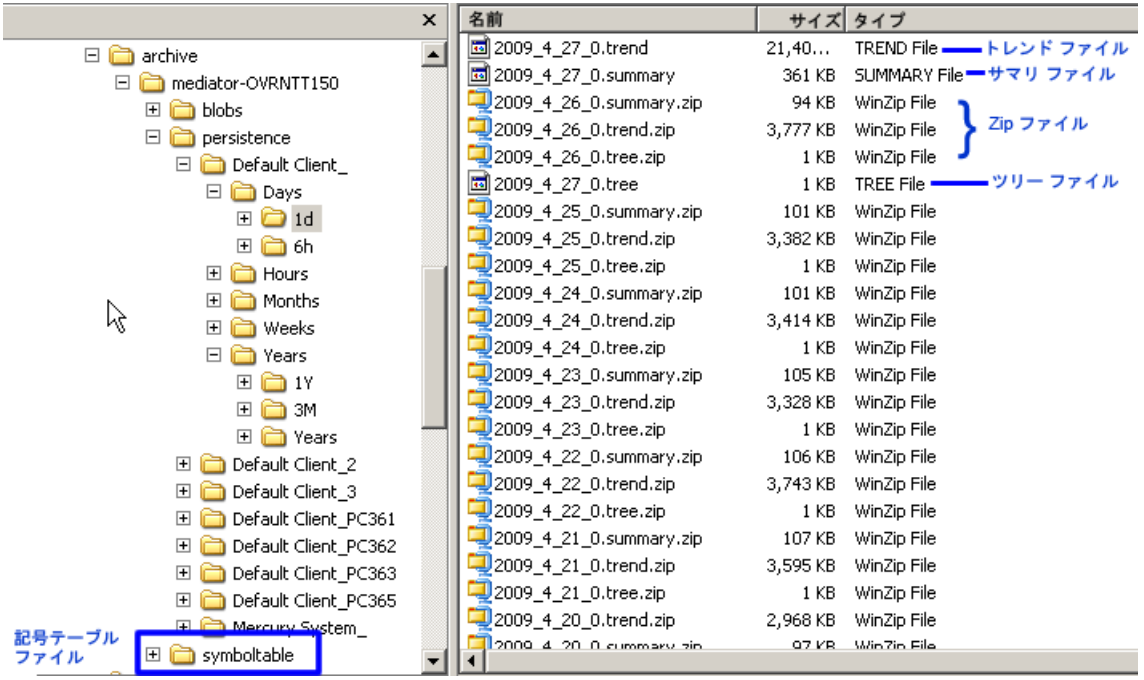
上記のディレクトリの例にあるように，Hours ディレクトリは長期間を表し，その下に短期間（5 分，20 分，1 時間）を表す 3 つのサブディレクトリがあります。

これらと同じ短期間は，UI の表示期間として機能します。次に，Diagnostics の表示フィルタの例を示します。



パフォーマンス履歴データ・ファイル・タイプ

Diagnostics パフォーマンス履歴データは，次の 7 種類のファイルに保存されます。次に，これらのファイルのディレクトリ例を示します。



記号テーブル・ファイル

記号テーブルには，ほかのデータ・ファイルに小型で高速なデータ・エンコードを行うための文字列から整数へのマッピングが含まれます。たとえば，/login.do は 1347854 にエンコードされます。

記号テーブルは，**＜Diagnostics サーバのインストール・ディレクトリ＞/archive/mediator-＜host_name＞/symboltable** ディレクトリに保存されます。

サマリ・ファイル

サマリ・ファイルは，Diagnostics ビューのエンティティ・テーブルおよび詳細表示枠にデータを表示する場合にアクセスします。Diagnostics ビューに表示されるステータスは，サマリ・ファイルに基づいています。表示可能な期間は，個別の *サマリ・ファイル*に保存されます。

各サマリ・ファイルには，Diagnostics がサマリ・データの保存を開始した時刻（GMT）に応じて名前が付けられます。たとえば，2009 年 4 月 24 日の深夜（GMT）に始まるデータを含むサマリ・ファイルには，**2009_4_27_0.summary** という名前が付けられます。

トレンド・ファイル

トレンド・ファイルは，Diagnostics ビューにグラフ（図表）データを表示する場合にアクセスします。5 分などの短期間のトレンドを取得するために，長期間トレンド・ファイルのデータの一部（この場合は 1 時間）だけが読み取られます。Diagnostics は，各長期間のグラフ化されたトレンド・データを *トレンド・ファイル*に保存します。

トレンド・ファイルには，トレンド・データがキャプチャされた時刻（GMT）に基づいて名前が付けられます。たとえば，2009 年 4 月 24 日の深夜に開始する毎時トレンド・データが含まれるファイルには，**2009_4_27_0.trend** という名前が付けられます。

Diagnostics ビューにトレンド測定値が表示されると，Diagnostics は，表示可能な各期間に対して約 60 のデータ・ポイントを表示して，表示されたトレンドが重要でわかりやすいものになるようにします。表示可能な各期間で必要なデータ・ポイントに到達するために，Diagnostics サーバは，データ・ファイルの適切な層からデータを統合します。たとえば，最後の 1 時間のトレンド・データを確認している場合，1 分あたり 1 つのデータ・ポイントがグラフに表示されます。これらのデータ・ポイントは，12 個の 5 秒間の生データ・ポイントを統合して作成されたものです。

インスタンス・ツリー・ファイル

インスタンス・ツリー・ファイルは，ユーザが Diagnostics サーバの［要求］ビューでインスタンス呼び出しツリーをドリルダウンしたときにアクセスされます。

*インスタンス・ツリー・ファイル*は，トレンド・ファイルと似ています。それぞれの長期間に，収集されたインスタンス呼び出しツリーのダンプが対応しています（**2009_4_27_0.tree** など）。

圧縮 zip ファイル

インスタンス・ツリー，トレンド・ファイル，およびサマリ・ファイルのデータは，現在の期間のものです。これらのファイル内のデータは圧縮されていません。データ・ファイルは極めて大きいため，ディスク容量の節約のために各期間が完了するたびに自動的に圧縮されます。圧縮されたファイルには，未圧縮ファイルと同じ名前が付けられますが，**2009_4_26_0.summary.zip** のように .zip 拡張子が付いています。

データの保存

Diagnostics では，データ保存計画を使って，ディスク容量の使用方法を最適化します。データ保存の標準設定はすぐに使用できるように設定されています。システムを監視して空きディスク容量を確認し，それに応じてデータ保存設定を変更する必要があります。

データ保存設定は削除のタイミングを決定するときに考慮されます。そのため，予想よりも多くのディスク容量が使用されている場合は，データ保存設定を確認して変更する必要があるかどうかを判断する必要があります。詳細については，837 ページ「記号テーブルの削除」を参照してください。

本項の内容

Mediator のデータ保存

ディスク容量を最適に使用するため，履歴パフォーマンス・データは，長期間と短期間に分けて保存され，各期間のデータは診断データ保存ポリシーに基づいて保存されます。

データ・ポイントの解像度が低い層のデータは，キャパシティ・プラニングなどの活動を支援するために長期間保存されます。データ・ポイントの解像度が高い期間のデータは，パフォーマンス診断などのアクティビティを支援するために短期間保存されます。この保存ポリシーに従って測定した場合，調査した仮想マシンごとに約 3 GB が使用されます。

次の表に，長期間（828 ページ「パフォーマンス履歴データの構成」に記載されたディレクトリ），および短期間（表示可能な期間またはサブディレクトリ）を示します。各ディレクトリの下にある表示可能期間は，解像度が同じであるためグループ化されています。たとえば，Days ディレクトリにある 1 日と 6 時間のデータは，両方とも解像度が 5 分です。

次の図に，一般的な Diagnostics データ保存ポリシーを示します。

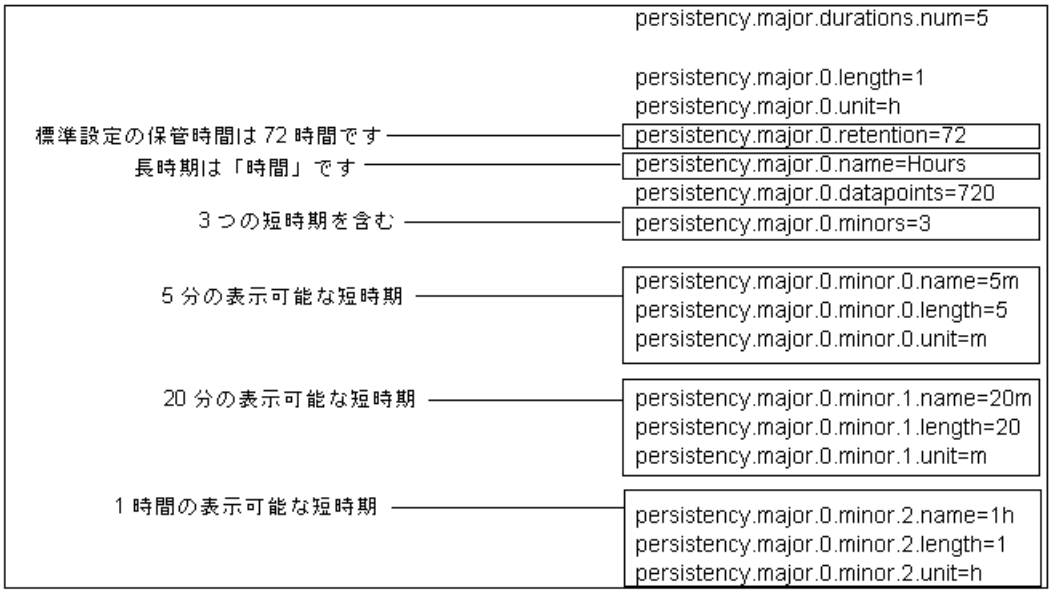
上記の図は，指定された期間中にエンティティが変わらず，常に利用可能な場合にのみ適用されます。次の「記号テーブルの削除」を参照してください。

上記の表が示すように，最近の 1 時間，20 分，および 5 分間の表示期間内のデータは 72 時間保存されますが，最近の四半期のデータは 5 年間保存されます。

データ保存の設定

データ保存を設定するには，各メディエータ・サーバにある **server.properties** ファイルを使用します。データ保存の標準設定はすぐに使用できるように設定されています。システムを監視して空きディスク容量を確認し，それに応じてデータ保存設定を変更する必要があります。

次に，server.properties ファイルの永続セクションの表示例を示します。



長期間の保存レベルは，短期間に継承されます。

短期間の保存レベルを別の値に設定するには，次のような 1 行を追加します (1 時間の短期間の場合)。このようにすると，1 時間ごとのデータ 90 時間保存されます。

```
persistency.major.0.minor.2.name=1h
persistency.major.0.minor.2.length=1
persistency.major.0.minor.2.unit=h
persistency.major.0.minor.2.retention=90
```

1 時間ごとのデータを 1 週間保存するには，保存レベルを 168 時間に設定します。単位は時間（unit=h）であるため，1 週間を時間数で計算します［7（日）x 24（時間）= 168］。

```
persistency.major.0.minor.2.name=1h
persistency.major.0.minor.2.length=1
persistency.major.0.minor.2.unit=h
persistency.major.0.minor.2.retention=168
```

1 日ごとのデータを 6 か月間保存するには，保存レベルを 186 日に設定します。単位は日（unit=d）であるため，6 か月を日数で計算します［6（月）x 31（日 / 月）= 186］。

```
persistency.major.1.minor.1.name=1d
persistency.major.1.minor.1.length=24
persistency.major.1.minor.1.unit=d
persistency.major.1.minor.1.retention=186
```

記号テーブルの削除

削除メカニズムでは，データを削除するタイミングと方法を決定するときにいくつかの設定やその他の要素を考慮する必要があります。

標準設定では，6 か月（4320 時間）ごとに削除が実行されます。削除間隔は，**server.properties** の **persistency.major.4.total.length** パラメータで時間数を変更することによって変更できます（サーバの再起動が必要です）。

重要 : 削除間隔を増やすと，より多くのサーバ・メモリが必要になります。

スナップショットの一部であるデータを削除すると，スナップショットが役に立たなくなるため，削除されません。

プローブの名前が変更され，6 か月間（デフォルト）プローブからデータが受信されなければ，Diagnostics は自動的にプローブとそのデータをデータベースから削除します。

TSDB で使用する容量の大きさに対してプロパティを指定でき，通常，指定したしきい値よりも小さなサイズに保持するためにデータが削除されます。**＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/server.properties** ファイルで **persistence.purging.threshold** プロパティが設定されます。ただし，ほかのいくつかの設定ではこのしきい値よりも高い優先度を指定できるため，保持されるデータが多くなり過ぎる可能性があります。

ディスク容量がなくなっている場合でも，削除が機能していないというわけではありません。次のいずれかの要素が削除メカニズムに影響している可能性があります。たとえば，Diagnostics サーバで 10GB のディスク容量を割り当てたのにアーカイブが 20GB になっていて，システムのディスク容量がなくなる危険性があることがあります。これは次のいずれかの原因による可能性があります。

- 削除間隔にまだ達していない。間隔が短くなるように調整できます。
- 意図的に削除しないようになっているスナップショットがシステムに多数存在する。
- データ保存設定で保持するように指定されているデータが多すぎる。ディスク容量を節約するために必要に応じてデータ保存を調整してください（834 ページ「データの保存」を参照）。

スナップショットのデータが含まれるデータ・ファイルは削除されません。TSDB が配布され，スナップショットだけが Commanding Server に残るため，どの期間を保存するのかを特定するためのメカニズムが必要です。

スナップショットが作成されると，Commanding Server は保存されている時間のグローバル・リストにインシデントの期間を追加します。スナップショットを削除すると，その期間も削除されます。同時に複数のスナップショットを作成できるように，スナップショットごとに新しい保存エントリが作成されます。削除処理が機能しなくなるため，同一のエントリがマージされることはありません。UI は，サーバにスナップショットの期間を個別に保存する必要があることを伝えます。これにより，スナップショット以外の目的でのデータの保存が可能になります。

サーバが削除処理を開始すると，Commanding Server から保存されている期間を取得します。リストを取得できなかった場合，削除プロセスが中止され，後で再実行されます。Commanding Server が永久にオフラインになっていた場合に，分散サーバのデータが無限に増加する状況を避けるため，1 週間が経過しても Commanding Server に接続されなかったときは，保存リストを考慮しないで削除されます。

アーカイブのサイズが削除しきい値を超えるまでデータは削除されません（**persistence.purging.threshold** プロパティ）。その後，アーカイブがそのしきい値を下回るまで，後で定義するポリシーを使ってファイルが削除されます（ほかの要素が削除メカニズムに影響しない場合）。デフォルトで，しきい値は 5G に設定されますが，**server.properties** ファイルの **persistence.purging.threshold** で値を変更することができます。

削除プロセスでは，データ・ファイルをスキャンし，すべての削除候補を特定します。このプロセスは，ファイル・セットで実行されます。トレンド・ファイルは，ディスク容量の最大のコンシューマであり，サマリ・ファイルを必要とするため，データの削除はトレンド・ファイルに基づいて行われます。存在するにもかかわらず，保存期間が格納されていないトレンド・ファイルがある場合は，そのトレンドに関連付けられているすべてのファイルを含むファイル・セットが作成されます。これにより，トレンド・ファイルを含む長期間のサマリ（長期間のサマリだけにトレンド・ファイルが含まれます），およびトレンド・ファイルにインデックス化する短期間のサマリにサマリとツリー・ファイルが含まれます。

各ファイルセットには，削除対象を特定するための複数の値が関連付けられています。ファイル・セットの「終了時間」は，セットに含まれる最後のデータ・ポイントの時間です。「削除サイズ」は，削除可能なサマリのすべてのファイルのサイズです。

サーバのディスク容量の問題

サーバのディスク容量の問題がある場合に Diagnostics 管理者に電子メールで警告が送信されるようにセットアップできます。管理者の電子メール・アドレスは，サーバのインストール時や後で［Alert Properties］ページを使用してセットアップするときに入力できます。

サーバのアーカイブ･ディレクトリの空きディスク容量が 100MB を下回ると警告が発行されます。サーバの容量が 50MB を下回ると，サーバはデータの収集を停止して終了します。また，空きディスク容量が 50MB を下回っている場合，サーバは起動しません。これらの予防策により，ディスク容量がなくなる前にサーバがデータの収集を停止してサーバの安定性を維持できます。

Diagnostics 管理者のためのこれらの警告タイプを決定するしきい値は，出荷時にサーバの **server.properties** ファイルで設定されています。詳細については，このファイルの各種 **watchdog** プロパティのコメントを参照してください。

インストール前のデータ管理に関する留意点

大規模な Diagnostics サーバのインストールおよび設定を準備する際，パフォーマンス・チューニングについて次の点に留意する必要があります。

- パフォーマンスを最大化するために，Diagnostics サーバを空のディスクまたは最近デフラグしたディスクにインストールする必要があります。アーカイブ･ディレクトリは，同じディスクに保存する必要があります。また，アーカイブ・ディスクを空のディスクまたは最近デフラグしたディスクにマウントすることもできます。

  注：＜ Diagnostics サーバのインストール・ディレクトリ＞ /archive に保存される時系列の診断データベースで使われるディスクをほかのディスク・アクティビティに使用しないことをお勧めします（つまり，システム・ファイル，一時ファイルなどで使うのと同じディスクに **＜ Diagnostics サーバのインストール･ディレクトリ＞ /archive** ディレクトリをマウントしないでください）。

- 時間経過に伴う断片化を低減し，システム･パフォーマンスを高めるために，アーカイブ・ディレクトリ専用の個別のディスク（またはパーティション）を使用することをお勧めします。

- アーカイブ・ディレクトリが保存されているディスクでは，集中的なバックグラウンド・ディスク処理（ディスクのデフラグやウイルス・スキャンなど）を無効にしてください。
- NFS や Samba などのネットワーク・ファイル・システムは使用しないでください。

注 : Diagnostics サーバのアーカイブ・ディレクトリ専用に使うディスクの生パフォーマンスが高いほど，Diagnostics サーバはより多くの負荷を処理できます。アーカイブ・ディレクトリにマウントされているディスクが高パフォーマンス・ディレクトリまたはアレイであることをシステム管理者に確認してください。

Diagnostics データのバックアップ

ディスクやシステムに障害が発生したときに復元できるように，Diagnostics データを定期的にバックアップすることをお勧めします。

独自のバックアップ方法を使用する場合，サーバをシャットダウンする必要があります（PathSymbolTable.pst がロックされているため）。ただし，Diagnostics で提供されるバックアップ・スクリプトを使用することをお勧めします。

注 : Diagnostics デプロイメントで，Diagnostics サーバに高い可用性が必要な場合は，各 Diagnostics サーバにスタンバイ Diagnostics サーバを作成できます。スタンバイは，Diagnostics サーバのホストにハードウェア障害などの問題が生じたときにいつでも使用できます。445 ページ「可用性の高い Diagnostics サーバの準備」を参照してください。

本項の内容

データのリモート・バックアップの実行

リモート・バックアップを行うには，HTTP を通じて Diagnostics データ・ファイルをローカル・ディレクトリにダウンロードし，通常のバックアップ手順でディレクトリのバックアップを作成します。

Diagnostics サーバでは，HTTP If-Modified-Since および Request-Range ヘッダ（「re-get」）をサポートしており，標準の HTTP ミラーリング・ソフトウェアでこれらのファイルをダウンロードしたり，インクリメンタル更新を行うこともできます。自分の HTTP ミラーリング・ソフトウェアを使うように選択した場合，HP のサポート・スタッフと協力して適切な順序でファイルをバックアップし，データの整合性を保つようにします。

Diagnostics は，**＜ Diagnostics のインストール・ディレクトリ＞ /server/bin/remote-backup.sh** に保存されているリモート・バックアップ・スクリプトと一緒にインストールされています。UNIX スクリプトは wget ユーティリティを使用して（http://www.gnu.org/software/wget/wget.html）（英語サイト），HTTP を通じてインクリメンタルにダウンロードします。Windows の場合は，Cygwin（http://www.cygwin.com/）（英語サイト）を使って，このスクリプトを実行できます。

Windows 対応の **remote-backup.cmd** もあります。.cmd スクリプトを使用するには，**＜ Diagnostics のインストール・ディレクトリ＞ /server/bin/wget** ディレクトリ内に **wget.exe** が必要です（.sh スクリプトを使用するには，パス上に wget を配置する必要があります）。

バックアップ・スクリプトを使用すると，リモートでデータをバックアップし，必要に応じてそのディレクトリから従来のバックアップを行うことができます。また，スクリプトでローカル・ディレクトリ（理想的には同じホストの別のドライブ）にデータをバックアップすることもできます。

重要 : Diagnostics サーバに付属のバックアップ・スクリプトは，特定の順序でデータをバックアップします。正しい順序でファイルをバックアップしないと，復元されたバックアップが使用できなくなります。したがって，データのバックアップを作成する際は，必ず付属のスクリプトを使うことをお勧めします。

次の表は，**remote-backup.sh** パラメータの一覧です。

パラメータ	説明
-h	ダウンロード元のホスト（または IP アドレス）
-o	バックアップの保存先ディレクトリ
-u	使用する HTTP ユーザ名 デフォルト：admin
-p	使用する HTTP パスワード デフォルト：admin
-P	使用する HTTP ポート番号（省略可能） デフォルト：2006
-r	（RHTTP バックアップの）読み取り元の Diagnostics サーバ（Mediator モード）の ID（省略可能）。たとえば，「commander」と「mediator」の 2 台のサーバがある場合は，-h commander -r mediatorId を使用して，RHTTP を介して mediator をバックアップできる。
-c	消去オプション。指定すると，出力ディレクトリ内にあり，サーバ上には存在しないファイルが削除される（その他のファイルは，ダウンロード時にタイムスタンプ機能によってパフォーマンスが向上するように保存しておく必要がある）。
-v	詳細出力を行うように指定する。

たとえば，Dragonfly マシンで実行している Diagnostics サーバのバックアップを **dragonfly-backup** ディレクトリに作成する場合は，次のようになります。

```
% mkdir dragonfly-backup
% bin/remote-backup.sh -u admin -p secret -h dragonfly -o dragonfly-backup
```

データは次のディレクトリにバックアップされます。

データ	バックアップ・ディレクトリ
サーバ設定	**etc/**
ユーザ・カスタム・ビュー	**storage/userdata**
生パフォーマンス履歴データ	**archive/.../persistence/**
記号テーブル	**archive/.../symboltable/**

記号テーブルのバックアップの設定

多数の記号ファイルが存在する場合は，symboltable の下にある jdb ファイルのバックアップ・フォルダで大量のディスク領域が使用されることがあります。したがって，記号テーブルのバックアップは次のように設定してください。

- 記号テーブルのバックアップを有効または無効にするには，**server.properties** ファイルの **symboltable.backup** プロパティを true または false に設定します。
- 記号テーブルのバックアップ頻度を設定するには，**server.properties** ファイルの **symboltable.backup.majors** プロパティを設定します。

  カンマ区切りリストを使用して，symboltable.backup.majors プロパティを目的のバックアップ頻度に設定します（日，週，月）。頻度値は，**persistency.major.<n>.name** プロパティの定義と同じです（835 ページ「データ保存の設定」を参照）。たとえば，記号テーブルを毎週バックアップするには，**persistency.major.2.name**（2 は週を表す）を使用します。

  server.properties で定義されている記号テーブルのバックアップの標準設定は，次のとおりです。

```
 # Should the server backup the symboltable?
symboltable.backup = true
# Which majors should be backed up?
symboltable.backup.majors = Days,Weeks,Months
```

障害発生時にデータを復元する

バックアップ・ディレクトリのファイルは，Diagnostics サーバで使われる構造に保存されます。

バックアップから時系列データベースを復元するには，次の手順を実行します。

1 クリーンな Diagnostics サーバをインストールします。インストールが完了すると，Diagnostics サーバは自動的に起動します。

2 Diagnostics サーバをシャットダウンします。

3 プロセス・リストに java/javaw プロセスがないことを確かめて，Diagnostics サーバがシャットダウンしたことを確認します。この場合，Windows システムではタスク・マネージャを，UNIX システムでは ps をそれぞれ使用します。

4 Diagnostics サーバから **＜Diagnostics サーバのインストール・ディレクトリ＞/archive** ディレクトリを削除します。

5 データベースのバックアップをコピーして，**＜Diagnostics サーバのインストール・ディレクトリ＞/archive** を置き換えます。

6 バックアップを取った後に Diagnostics サーバのホスト名が変更されている場合は，Diagnostics サーバのホスト名に基づいてディレクトリ名を更新して，新しいホスト名を反映させる必要があります。

＜Diagnostics サーバのインストール・ディレクトリ＞/archive/mediator-＜ホスト名＞の名前を変更して，**＜ホスト名＞**に新しい Diagnostics サーバのホスト名を反映させます。たとえば，バックアップのホスト名が oldhost で，新しいホスト名が newhost の場合，**＜Diagnostics サーバのインストール・ディレクトリ＞/archive/mediator-oldhost** を **＜Diagnostics サーバのインストール・ディレクトリ＞/archive/mediator-newhost** に変更します。

インデックスの再生成

復元した Diagnostics サーバを先に起動した場合，バックアップしていないインデックス・データを生成し直す必要があります。インデックスの再生成はバックグラウンドで自動的に開始し，完了までに数時間かかることがあります。インデックスの再生成中，Diagnostics サーバはプローブからイベントを受信できますが，復元が完了するまで Diagnostics ビューに一部の履歴データを表示できません。

既知の制限

Diagnostics サーバでは，バイナリ・データはネイティブ・バイト・オーダで書き込まれます。つまり，Big Endian マシンの Diagnostics データ・バックアップは復元できず，Little Endian マシンで使用できません。

Diagnostics アップグレード時の Diagnostics データの取り扱い

アップグレード時の Diagnostics データの扱い方については，付録 G,「アップグレードとパッチ・インストールの手順」を参照してください。

F

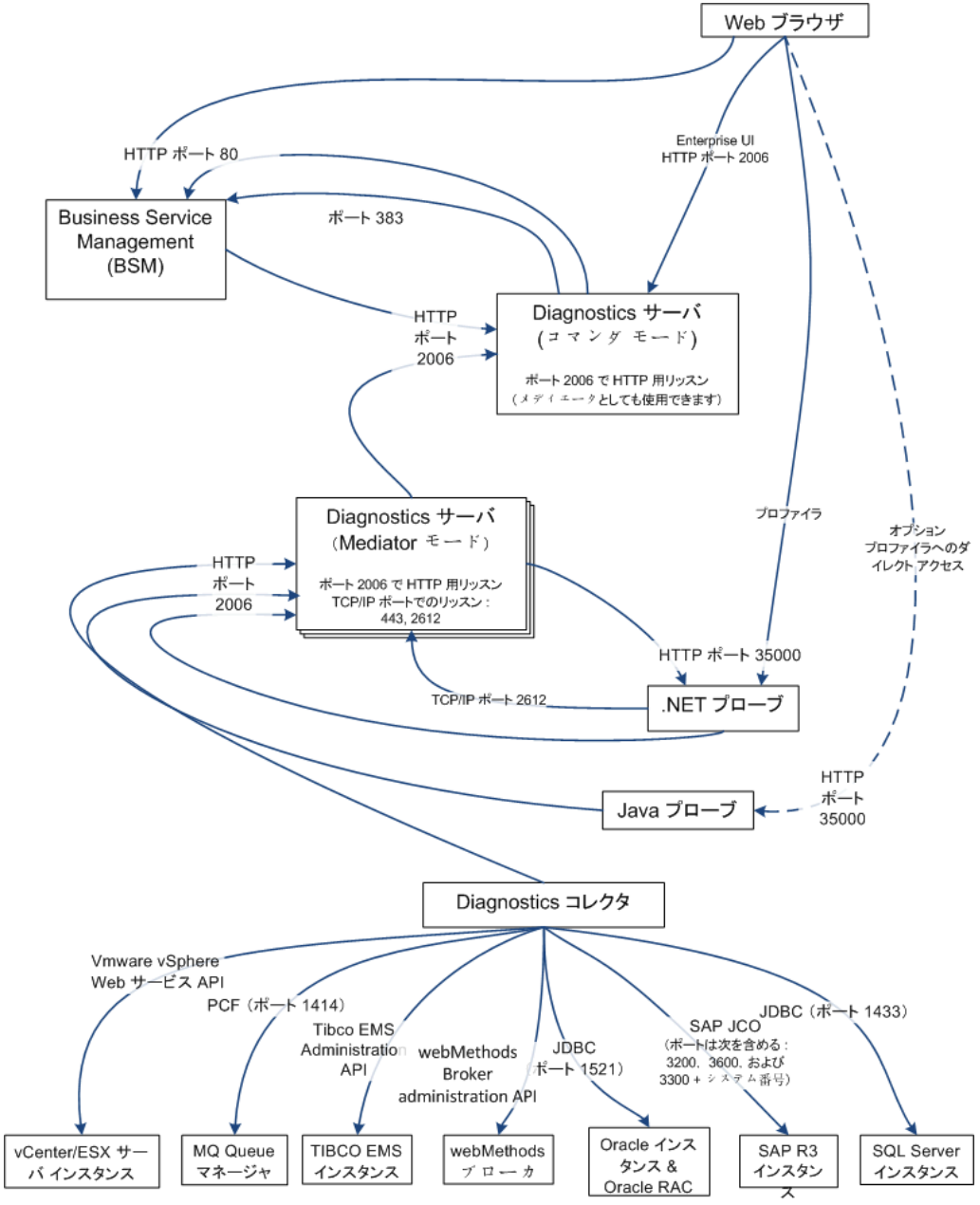
Diagnostics の技術的な図

Diagnostics コンポーネントを配備し，Diagnostics とほかの HP ソフトウェア製品を統合するときに役立つデータ・フローおよび通信図を示します。

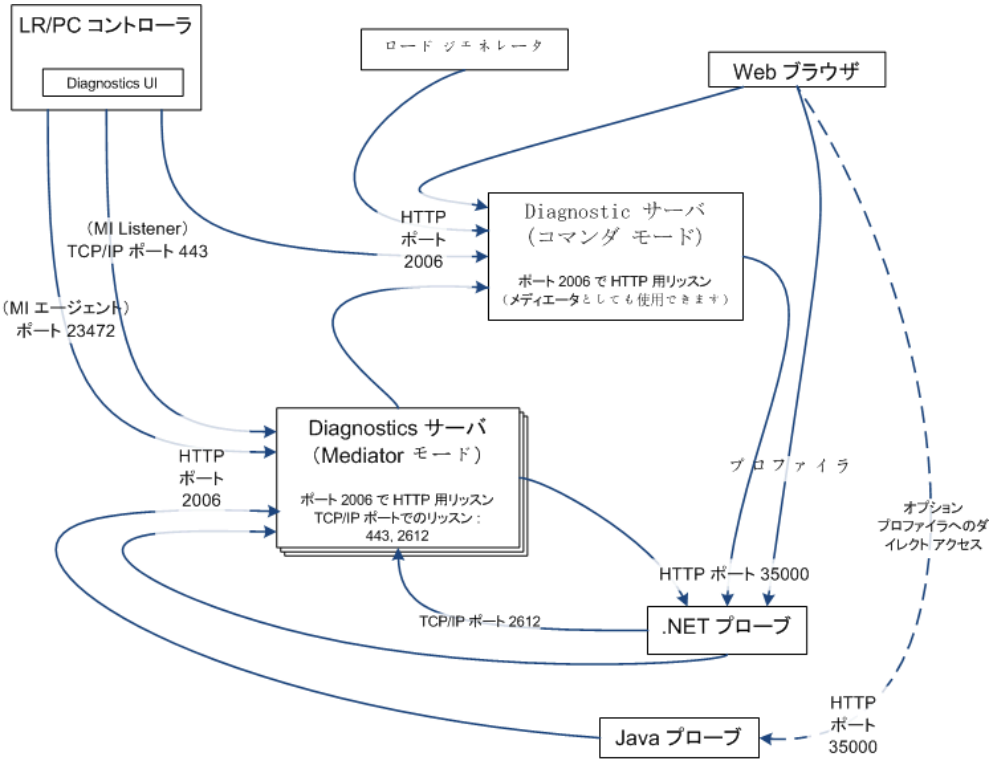
本章の内容

注 : この図は，コンポーネントの作業に関する深い知識を提供するのではなく，高度な考え方を提供することを目的としています。

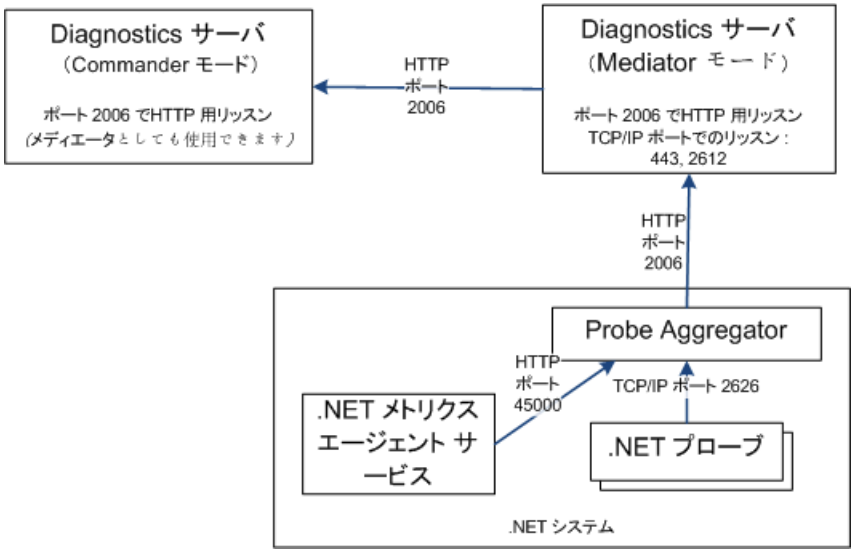
Business Service Management との通信



LoadRunner および Performance Center との通信



.NET Probe Aggregator のデータ・フロー



G

アップグレードとパッチ・インストールの手順

Diagnostics の主要リリースの間でのアップグレード（たとえば，8.0 から 9.0）の手順を示します。パッチ・リリースをインストールするときも同じ手順を行なってください。パッチ・リリースは Diagnostics 製品コンポーネントを完全に置き換えるものなので，アップグレードと同じ手順を行う必要があります。

注 : Diagnostics Agent および TransactionVision Agent をアップグレードするとき，またはそのパッチ・リリースをインストールするときは，Java Agent および .NET Agent 手順が適用されます。

本章の内容

始める前に

次の推奨事項は，通常，Diagnostics の旧バージョンからアップグレードするとき，またはパッチ・リリースをインストールする際に適用されます。

- 旧バージョンのコンポーネントが使われていたのと同じホストで新バージョンのコンポーネントにアップグレードする前に，新バージョンのコンポーネントのシステム要件をホストが満たすことを確認してください。システム要件については,『HP Diagnostics インストールおよび設定ガイド』(英語版) の Readme ファイルを参照してください。
- Agent をアップグレードする前に Diagnostics サーバをアップグレードする必要があります。
- 以前のバージョンの Business Service Management または Performance Center をアップグレードする必要がある場合は，HP ソフトウェア・カスタマ・サポートにお問い合わせください。Diagnostics の統合に関する重要な説明については，これらの製品のアップグレード・ドキュメントを参照してください。

Diagnostics と以前の Diagnostics バージョンとの互換性

Diagnostics サーバは, 次の旧バージョンの Agent および Collector と連携して動作できます。

- Java Agent 8.x，9.0x
- .NET Agent 8.x，9.0x
- Collector 8.x，9.0x

Diagnostics コンポーネントのアップグレードまたはパッチ・インストールの手順

既存の Diagnostics コンポーネントをアップグレードするか，またはパッチ・リリースからコンポーネントをインストールするには，次の手順を実行します。

本項では，次の手順について説明します。

- 853 ページ「Diagnostics サーバ」
- 857 ページ「Java Agent」

➤ 861 ページ「.NET Agent」

➤ 862 ページ「Diagnostics Collector」

Diagnostics サーバ

本項では，Diagnostics サーバを旧バージョンからアップグレードする手順について説明します。同じ手順がパッチ・リリースのインストールにも適用されます。

注：

➤ Diagnostics サーバをアップグレードする場合，デプロイメントのすべての Diagnostics サーバをアップグレードする必要があります。デプロイメントのすべての Diagnostics サーバでは，同じ Diagnostics バージョンを実行する必要があります。

➤ HP Software-as-a-Service（SaaS）カスタマの場合は，アップグレード方法について SaaS サポートにお問い合わせください。

インストール中，キーストアは JRE と一緒に上書きされます。その結果，アップグレード後，信頼された証明書が使用できなくなります。

Diagnostics の新しいリリースごとに，複数のマシン上でサイレント・インストールを実行する前に，Diagnostics サーバのサイレント・インストール応答ファイルを再度記録する必要があります。

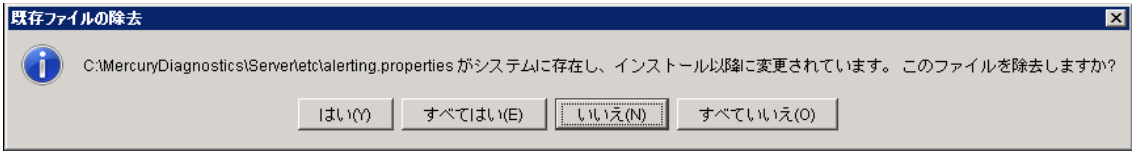
Diagnostics サーバをアップグレードするには，次の手順を実行します。

1 現在の Diagnostics サーバをシャットダウンします。

2 現在の Diagnostics サーバ・ディレクトリの**バックアップ・コピー**を作成します。標準設定では，これは Windows では C:¥MercuryDiagnostics¥Server，UNIX では /opt/MercuryDiagnostics/Server ですが，サーバのインストール時に別のディレクトリが指定されていることもあります。

アップグレード手順では現在の Diagnostics サーバをアンインストールする必要があるため，やり直す必要がある場合にはバックアップ・コピーを使用できます。

3 現在の Diagnostics サーバをアンインストールしますが，**プロンプトが表示されたら変更したファイルを保存してください**。たとえば，Windows では次のプロンプトで［**すべていいえ**］をクリックします。

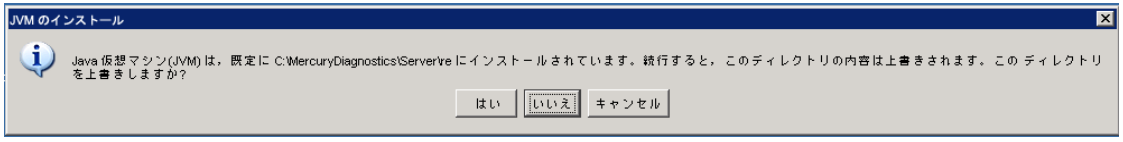


Diagnostics サーバのアンインストールと削除の詳細については，第 2 章，「Diagnostics サーバのインストール」を参照してください。

アップグレード中に既存の **etc** ディレクトリが **etc.old_ <タイムスタンプ>**という名前に変更されます。

4 新しい Diagnostics サーバを，旧バージョンの Diagnostics サーバで使っていたものと**同じインストール・ディレクトリ**にインストールします。新しい Diagnostics サーバに対して，以前の Diagnostics サーバで使われていたものと**同じホストとポート**を指定します。

次のメッセージが表示されたら，［はい］をクリックします。



5 Windows でインストールする場合は，Diagnostics サーバを停止します。

Windows で，インストーラが完了すると，Diagnostics サーバは自動的に起動します。UNIX では，サーバが自動的に起動しないため，停止する必要はありません。

6 **etc** ディレクトリと **etc.old_ <タイムスタンプ>**ディレクトリを比較し，2 つのディレクトリの違いを確認できます。この場合，diff/merge ツールを使うと便利です。

重要：9.20 以前のバージョンからアップグレードする場合，以前にカスタマイズしていたかどうかに関係なく，次の 2 つのプロパティを変更する必要がある場合があります。

- server.properties のコピーが etc.old_ ＜タイムスタンプ＞に存在する場合，**thresholding.evaluation.status.red.for.availability** プロパティとその値を古いファイルから新しいファイルにコピーします。
- thresholds.configuration のコピーが etc.old_ ＜タイムスタンプ＞に存在する場合，**com.mercury.diagnostics.common.data.graph.node.ProbeData.Availability** プロパティとその値を古いファイルから新しいファイルにコピーします。それ以外の場合は，新しいファイルでこの値を ",-95" ではなく "-95" に設定します。

こうした変更を行っても，以前に設定した可用性のしきい値の動作は保持されます。最初のアップグレード中にこうした変更を忘れてしまった場合，後から変更を行うこともできます。

（**etc.old_ ＜タイムスタンプ＞**ディレクトリに）行ったカスタマイズによって生じた相違を **etc** ディレクトリに適用して，それらのカスタマイズが失われないようにします。次の変更を探す必要があります。

プロパティ・ファイル	新しい Diagnostics サーバにコピーする設定プロパティ
alerting.properties	SNMP および SMTP サーバ，メール・アドレス。
security.properties	システムが SSL モードにセットアップされている場合，new /etc フォルダにコピーするすべてのパラメータと証明書を手動で更新する必要がある。
server.properties	タイムアウト / トリミング設定，Commander の URL 上記の thresholding.evaluation.status.red.for.availability プロパティに関する重要事項を参照してください。
thresholds.configuration	以前に行ったすべてのカスタマイズをコピーします。 上記の com.mercury.diagnostics.common.data.graph.node.ProbeData.Availability プロパティに関する重要事項を参照してください。

プロパティ・ファイル	新しい Diagnostics サーバにコピーする設定プロパティ
webserver.properties	デフォルトのポートの情報。
.htaccess	.htaccess ファイルはセキュリティのために使用され，ユーザの役割の元の設定を保持するために新しい /etc フォルダにコピーする必要があります。

7 システムが LoadRunner または Performance Center と統合されている場合，**etc.old_ ＜タイムスタンプ＞**ディレクトリから新しい **etc** ディレクトリに **run_id.xml** をコピーして，実行 ID が今後の実行のために正しくインクリメントするようにする必要があります。

8 Diagnostics コマンド・サーバをアップグレードしている場合，**DiagnosticsLicFile.txt** ファイルまたは **DiagnosticsServer.lic** ファイルを **etc.old_ ＜タイムスタンプ＞**ディレクトリから新しい **etc** ディレクトリにコピーします。

9 Diagnostics サーバを起動します。

10 Diagnostics UI にアクセスする前に，ブラウザのキャッシュをクリアして，ブラウザを再起動します。

11 Diagnostics UI の［システムの状況］ビューでバージョンをチェックして，アップグレードした Diagnostics サーバが実行していることを確認できます。アップグレードが正常に完了し，Diagnostics サーバが再起動すると，バージョンが最新バージョンになります。［システムの状況］ビューにアクセスするには，http:// ＜ Diagnostics コマンド・サーバ名＞ :2006/query/ から Mercury System カスタマとして Diagnostics UI にアクセスする必要があります。こうすると［ビュー］ペインで［システム ビュー］ビュー・グループを選択できます。

12 Diagnostics と **Business Service Management** を統合している場合，次の手順を実行する必要があります。

a Diagnostics と BSM 9.01/9.00 を統合する場合，Diagnostics Readme に記載されているインストール時の注意事項を確認してください。

b Diagnostics コマンダ・サーバのアップグレード後に，**RegistrarPersistence.xml** ファイルを **etc.old_＜タイムスタンプ＞**フォルダから新しい **etc** フォルダにコピーする必要があります。［BSM］>［管理］>［Diagnostics］ページで Diagnostics の統合を確認し，Diagnostics サーバが適切に機能していない場合は BSM で登録しなおします。第 21 章，「Business Service Management と Diagnostics 間の統合のセットアップ」を参照してください。

13 Diagnostics サーバが正しくアップグレードされたことを確認したら，手順 2 で作成したバックアップ・コピーを削除します。

注：旧バージョンの Diagnostics で作成したカスタム・ビューを新しいバージョンで初めて開いたときに，Diagnostics の機能に変更が加えられているため，Diagnostics はビューをアップグレードして必要な変更を適用します。
Diagnostics がカスタム・ビューを変更すると，カスタム・ビューが変更された旨を伝えるメッセージが表示されます。

Java Agent

本項では，Diagnostics Java Agent を旧バージョンからアップグレードする手順について説明します。同じ手順がパッチ・リリースのインストールにも適用されます。

注：Diagnostics サーバは下位互換性がないため，接続されている Agent をアップグレードする前に Diagnostics サーバをアップグレードする必要があります。

注意 : Diagnostics の新しいリリースごとに，複数のマシン上でサイレント・インストールを実行する前に，Java Agent のサイレント・インストール応答ファイルを再度記録する必要があります。

Java Agent をアップグレードするには，次の手順を実行します。

注 : 指示に従って起動スクリプトを更新して新しい Agent を起動し，アプリケーションを再起動するまで，新しい Agent のインストールはアプリケーションの監視を開始しません。

1 現在の Agent のインストール・ディレクトリとは別のディレクトリに Diagnostics Agent for Java **をインストールします。**

インストール時には，Diagnostics について必ず次の作業を行ってください。これにより，アプリケーションの保持するデータが新しい Agent によってキャプチャされたメトリックスと確実に適合します。

- Java Agent を Diagnostics サーバと連携して動作する Java Agent として設定するか，またはスタンドアロンの Diagnostics Profiler として設定します。必要であれば，Java Agent を TransactionVision Server と連携するように設定することもできます。
- Agent 名には，以前の Agent によって使われたプローブ名と同じ名前を使用します。
- Agent グループ名には，以前の Agent によって使われたグループ名と同じ名前を使用します。
- メディエータ・サーバ名とポートには，以前の Agent によって使われた情報と同じ情報を使用します。

2 インストーラは，新しいインストール・ディレクトリに**＜プローブのインストール・ディレクトリ＞ ¥etc** ディレクトリを作成します。

7.50 以降のリリースでは，＜プローブのインストール・ディレクトリ＞の標準設定のディレクトリは変更されています。標準設定の場所は，Windows の場合は C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent，
UNIX の場合は /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent です。

3 インストールが完了したら, 新しい Agent の **¥etc** ディレクトリと以前の Agent の **¥etc** ディレクトリを比較して，2 つのディレクトリの違いを確認します。

2 つの異なる Java Agent インストールの違い（プロパティとポイント）を示す Java Agent の **Property Scanner** ユーティリティを実行することをお勧めします。Property Scanner ユーティリティを実行するには，カレント・ディレクトリを **＜プローブのインストール・ディレクトリ＞ /contrib/JASMUtilities/Snapins** に変更し，次のように **runPropertyScanner.cmd -console**（UNIX の場合は **.sh**）コマンドを実行します。

runPropertyScanner -console -diffOnly yes -Source1 ..¥..¥..¥etc -Source2 OtherEtc

入力例 :

C:¥MercuryDiagnostics¥JavaAgent8¥DiagnosticsAgent¥contrib¥JASMUtilities¥Snapins>runPropertyScanner -console -diffOnly yes -Source1 C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent¥etc -Source2 C:¥MercuryDiagnostics¥JavaAgent8¥DiagnosticsAgent¥etc

出力例 :

****** Property dispatcher.properties:stack.trace.method.calls.max
PropertyFile=dispatcher.properties
Property=stack.trace.method.calls.max
Source1=
Source2=1000

以前の Agent の **¥etc** ディレクトリに行ったカスタマイズによって生じた相違を新しい Agent の **¥etc** ディレクトリに適用して, それらのカスタマイズが失われないようにします。次の変更を探す必要があります。

プロパティ・ファイル	新しい Diagnostics Agent にコピーする設定プロパティ
auto_detect.points	作成したカスタム・ポイントと，変更したポイントを古い **etc** ディレクトリの **auto_detect.points** ファイルから新しい **etc** ディレクトリにコピーします。変更したポイントを探す際，RMI，LWMD，args_by_class のポイントを必ず確認してください。
capture.properties	深さとレイテンシのトリミング。

プロパティ・ファイル	新しい Diagnostics Agent にコピーする設定プロパティ
inst.properties	define.pre.process
dispatcher.properties	minimum.sql.latency sql.parsing.mode
dynamic.properties	cpu.timestamp.collection.method
metrics.config	以前の測定値を引き続き使用できるように，前バージョンでコメントアウトしなかった測定値が，新しいバージョンでもコメントアウトされていないことを確認します。
security.properties	システムが SSL モードでセットアップされている場合，すべてのプロパティを設定して，古いプロパティ・ファイルからこのプロパティ・ファイルに証明書をコピーします。

4 『HP Diagnostics インストールおよび設定ガイド』の「Java Agent で監視するためのアプリケーション・サーバの準備」の章で説明されているように，JRE のインストルメンテーション方法を使用してアプリケーション・サーバを監視できるように準備します。特に，アプリケーションの起動スクリプトまたは JVM パラメータをアップグレードした Agent インストールを参照するように更新する必要があります。パラメータには，-javaagent または -Xbootclasspath，あるいはその両方が含まれます。

5 承認時に，古い Agent によって監視されていたアプリケーションをシャットダウンします。

6 アプリケーションを再起動し，新しいバージョンの Agent がアプリケーションの監視を開始できるようにします。

7 Java Diagnostics Profiler ユーザ・インタフェースにアクセスする前に，ブラウザのキャッシュをクリアして，ブラウザを再起動します。これを行わないと，サイズが一致しないというエラー・メッセージが表示されることがあります。

8 Diagnostics UI の［システムの状況］ビューでバージョンをチェックして，アップグレードした Diagnostics Agent が実行していることを確認できます。アップグレードが正常に完了すると，バージョンが最新バージョンになります。［システムの状況］ビューにアクセスするには，http:// ＜ Diagnostics コマンド・サーバ名＞ :2006/query/ から Mercury System カスタマとして Diagnostics UI にアクセスする必要があります。こうすると［ビュー］ペインで［システム ビュー］ビュー・グループを選択できます。

9 すべてのアプリケーションが最新バージョンに移行され，すべて正しく作動していれば，旧ディレクトリを削除できます。新バージョンがアンインストールされてしまうため，旧バージョンはアンインストールしようとしないでください。

.NET Agent

本項では，Diagnostics .NET Agent を旧バージョンからアップグレードする手順について説明します。同じ手順がパッチ・リリースのインストールにも適用されます。

注 : Diagnostics サーバは下位互換性がないため，接続されている .NET Agent をアップグレードする前に Diagnostics サーバをアップグレードする必要があります。

.NET Agent をアップグレードするには，次の手順を実行します。

1 新しい Diagnostics Agent for .NET をインストールします（アップグレードを選択）。

調査対象のアプリケーションを再起動すると，アップグレードが有効になります。

アップグレードを強制的に有効にするには，次の手順を実行します。

a 現在の .NET プローブによって監視されているすべてのアプリケーションをシャットダウンします。

b IIS を再起動します。

c 古いプローブによって監視されていたアプリケーションを再起動します。

2 Diagnostics UI の［システムの状況］ビューでバージョンをチェックして，アップグレードした Diagnostics Agent が実行していることを確認できます。アップグレードが正常に完了すると，バージョンが最新バージョンになります。［システムの状況］ビューにアクセスするには，http:// ＜ Diagnostics コマンド・サーバ名＞ :2006/query/ から Mercury System カスタマとして Diagnostics UI にアクセスする必要があります。こうすると［ビュー］ペインで［システム ビュー］ビュー・グループを選択できます。

Diagnostics Collector

本項では，Diagnostics Collector を旧バージョンからアップグレードする手順について説明します。同じ手順がパッチ・リリースのインストールにも適用されます。

注 : Diagnostics サーバは下位互換性がないため，接続されている Collector をアップグレードする前に Diagnostics サーバをアップグレードする必要があります。

重要 : Diagnostics の新しいリリースごとに，複数のマシン上でサイレント・インストールを実行する前に，Diagnostics Collector のサイレント・インストール応答ファイルを再度記録する必要があります。

Diagnostics Collector をアップグレードするには，次の手順を実行します。

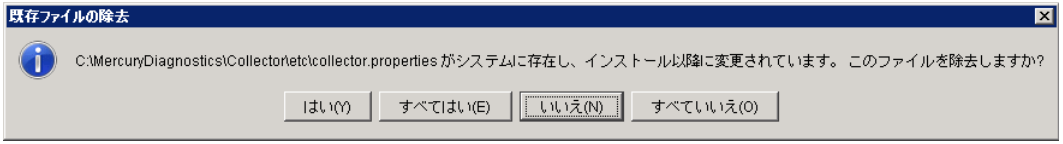
1 アップグレードする Diagnostics Collector を停止またはシャットダウンします。

2 現在の Collector のインストール・ディレクトリをバックアップします。

標準設定では，Windows の場合は **C:¥MercuryDiagnostics¥Collector**，UNIX の場合は **/opt/MercuryDiagnostics/Collector** です。

アップグレード手順では現在の Diagnostics Collector をアンインストールする必要があるため，やり直す必要がある場合にはバックアップ・コピーを使用できます。

3 現在の Diagnostics Collector をアンインストールしますが，プロンプトが表示されたら**変更したファイルを保存してください**。たとえば，Windows では次のプロンプトで［**すべていいえ**］をクリックします。



Collector のアンインストールと削除の詳細については, 128 ページ「Diagnostics Collector のアンインストール」を参照してください。

アップグレード中に既存の **etc** ディレクトリが **etc.old_ <タイムスタンプ>** という名前に変更されます。

4 古いバージョンの Collector で使われていたインストール・ディレクトリと同じディレクトリに，新しい Collector をインストールします。

アプリケーションの保持するデータが新しい Collector によってキャプチャされたメトリックスと適合するように，**同じ Collector 名と Mediator ホスト**を使用します。

バックアップした **< Collector のインストール ディレクトリ> ¥etc¥collector.properties** ファイルを確認することで，古い Collector 名がわかります。

5 Windows でインストールする場合は，Collector を停止します。

インストーラが完了すると，Collector は自動的に起動します。

UNIX では，Collector が自動的に起動しないため，停止する必要はありません。

6 新しい **etc** ディレクトリと **etc.old_ <タイムスタンプ>** ディレクトリを比較し，2 つのディレクトリの違いを確認します （この場合，diff/merge ツールを使うと便利です)。

（**etc.old_ <タイムスタンプ>**ディレクトリに）行ったカスタマイズによって生じた相違を新しい **etc** ディレクトリに適用して，それらの相違が失われないようにします。

プロパティ・ファイル	新しい Diagnostics サーバにコピーする設定プロパティ
mq-config.xml	Collector が MQ 環境を監視している場合。
oracle-config.xml	Collector が Oracle 環境を監視している場合。
sqlserver-config.xml	Collector が SQL Server 環境を監視している場合。 今後の Collector のバージョンでは databaseName が自動的に検出されるため，7.x からアップグレードする場合は，sqlserver-config.xml ファイルから databaseName 属性を削除する必要があります。
r3config.xml	Collector が SAP ABAP 環境を監視している場合。
vmware-config.xml	Collector が VMware 環境を監視している場合。
tibco-ems-config.xml	Collector が TIBCO EMS 環境を監視している場合。
wm-broker-config.xml	Collector が webMethods Broker 環境を監視している場合。
security.properties	システムが SSL モードでセットアップされている場合，すべてのプロパティを設定して，古いプロパティ・ファイルからこのプロパティ・ファイルに証明書をコピーします。

7 Diagnostics Collector を起動します。

8 バージョンをチェックして，アップグレードした Collector を確認できます。

9 Diagnostics Collector が正しくアップグレードされたことを確認したら，手順 2 で作成したバックアップ・コピーを削除します。

Diagnostics とほかの HP ソフトウェア製品の互換性

バージョン互換性の最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

H

HP Diagnostics のトラブルシューティング

HP Diagnostics の使用中に生じた問題を解決するためのヒントを提供します。

本章の内容

Solaris マシンにおけるコンポーネント・インストールの中断

Solaris マシンでコンポーネント・インストーラがコンポーネントのインストールを完了する前に中断した場合，コンポーネントを自動的にアンインストールまたは再インストールする方法はほかにありません。再度インストールを開始する前に，一部だけインストールされたコンポーネントを手動で削除する必要があります。

インストールの中断後に，手動で削除するには，次の手順を実行します。

1. インストール・ディレクトリを削除します。
2. **~/vpd.properties** および **~/vpd.patches** を削除します。
3. Solaris ディレクトリを削除します。**/var/sadm/pkg/IS*** および **/var/sadm/pkg/MERQ**

Java Agent が正常に作動しない

Java Agent が正常に作動しない場合，インストール・プロセス中に，**＜プローブのインストール・ディレクトリ＞ ¥classes¥boot¥java¥lang¥** フォルダに **ClassLoader.class** ファイルが作成されたかどうかを確認します。

ファイルが作成されなかった場合，ファイルを作成する JRE がインストルメントされたことを確認します。第 6 章,「Java Agent で監視するためのアプリケーション・サーバの準備」を参照してください。

Diagnostics Profiler for Java での WAS 起動時のエラー

症状：

Diagnostics Profiler for Java で WAS を起動するときに，Class Loader エラーが発生する。

原因：

追加クラスをインストルメンテーションから除外する必要がある。

解決方法：

1 プロパティ・ファイル **＜プローブのインストール・ディレクトリ＞¥etc¥inst.properties** を開きます。

2 既存の値の末尾にクラスを追加することで，**classes.to.exclude** プロパティを更新し，**!com¥.ibm¥..*** を除外します。

```
classes.to.exclude=!iaik¥.security¥..*,!c8e¥..*,!org¥.jboss¥.net¥.protocol¥.file¥.Handler,!org¥.jboss¥.net¥.protocol¥.file¥.FileURLConnection,!.*ByCGLIB.*,!com¥.ibm¥..*
```

サーバ側のトランザクションが表示されない

症状 :

Diagnostics に，各プローブのサーバ要求は表示されるが，サーバ要求に関連付けられている BPM トランザクションが表示されない。

server.log ファイルで特定する症状が 2 つあります。

タイムアウトした 1 つ以上のトランザクションでドロップしていない – これは，一定期間（標準設定で 10 分）にトランザクションがデータを受信しなかったこと，および ELT を受信しなかったことを示します。この警告は，トランザクションがタイムアウトしたときのみ，不定期に発行されます。この警告の後，UI にトランザクション・データが表示されます。ELT の詳細については，440 ページ「Diagnostics サーバのメモリ使用量の削減」を参照してください。

長時間経ってからデータを受信しました。データを調整しています ... – これは，サーバが ELT を非常に遅く，ただし，トランザクションがタイムアウトになる前に受信したことを示します。データは報告されますが，BSM または SaaS に報告されたものとは異なるタイミングとなります。

原因 :

上記のログ・メッセージのいずれかが表示されず，トランザクション・データがない場合，ほとんどの原因は BPM がスクリプトを実行していないことです。

解決方法 :

1. Business Process Monitor が Business Service Management または HP Software-as-a-Service（SaaS）で作動していること，およびモニタが機能していることを確認します。
2. Business Process Monitor コンソールのプロファイルのステータスを確認します。

イベント・キャプチャ・バッファ・フルの警告

症状 :

一部の Diagnostics データが消失し，次のエラーがプローブ・ログ・ファイルに記録されます。

「The event capture buffer is full, at least one event dropped.」

原因：

このログ・エントリは，アプリケーションの負荷が高すぎるか，アプリケーションが過剰にインストルメントされていることを示します。

解決方法：

場合によっては，etc/capture.properties ファイル内の **event_buffer.size** プロパティ値を大きくすると，イベントのドロップを防止できます。ただし，通常はアプリケーション・インストルメンテーションを軽減する必要があります。

Java Agent サポート・コレクタ

runSupportSnapshot ユーティリティを実行すると，Diagnostics または TransactionVision デプロイメント環境内の 1 つ以上の Java Agent インスタンスのトラブルシューティングに関連するファイル・セット全体を格納した .zip ファイルが作成されます。

この .zip ファイルには，以下が格納されています。

- **＜Diagnostics プローブのインストール・ディレクトリ＞¥etc** ディレクトリのファイル
- **＜Diagnostics プローブのインストール・ディレクトリ＞¥log** ディレクトリのファイル
- **＜TransactionVision Agent のインストール・ディレクトリ＞¥config** ディレクトリのファイル
- **＜TransactionVision Agent のインストール・ディレクトリ＞¥logs** ディレクトリのファイル
- 2 つの Agent のディレクトリを比較し，プロパティ・ファイル，ポイント・ファイル，および XML ファイル（TransactionVision Agent のみ）間の違いを報告する Property Scanner レポート。
- プローブ・インスタンスの情報（プロパティ設定を含む）。JVM 1.5 で動作する Agent の場合は，環境変数，スタック・ダンプ，およびクラス・ローダーの情報も含まれます。

runSupportSnapshot を実行するには，次の手順を実行します。

1 **＜Diagnostics プローブのインストール・ディレクトリ＞¥contrib¥JASMUtilities¥Snapins** に移動します。

このユーティリティは＜プローブのインストール・ディレクトリ＞¥bin ディレクトリでも利用可能です。

2 Windows の場合は **.¥runSupportSnapshot.cmd -console** を，UNIX または Linux の場合は **./runSupportSnapshot.sh -console** を実行します。

3 .zip ファイルが作成されます。zip ファイルの標準設定の保存場所は .../DiagnosticsAgent/ フォルダです。

イベント・ベース状況インジケータ・ステータスのトラブルシューティング・フロー

Business Service Management 9.0 以降と統合された Diagnostics は，Business Service Management ゲートウェイ・サーバに状況インジケータ・ステータス・イベントを送信します。

重要 : イベント・チャネル統合が行われた場合の BSM ゲートウェイ・サーバと BSM 処理サーバ間の通信では，マシン間に信頼関係が必要です。この設定方法については，716 ページ「DPS とゲートウェイに対する個別の BSM サーバの設定」を参照してください。

次に，Business Service Management に送信される状況インジケータ・ステータス・イベントに問題があった場合に，使用できるトラブルシューティング・フローを示します。

状況インジケータ・ステータス・イベントに問題があった場合のトラブルシューティング方法 :

1 プローブ測定値に関する状況インジケータ・ステータスが変更された場合に，Diagnostics が **bachi_data.log** に書き込むかどうかを確認します。

エントリ :

```
C|latency|jbossas|bsavm12.ovrtest.adapps.hp.com|17b87476ac6de3938ff9898cd19c8bd8|mercury|Default Client|1|J2EE|PROBE|2010-05-17 13:40:51|latency [BpmTxJpaImpl.getBamNodeStatusKey()  (144.5s > 122.6s)]
```

2 **ovc -status** を使用して，**opcle** が起動しているかどうかを確認します。

```
opcle       OVO Logfile Encapsulator    AGENT,EA    (5528)   Running
opcmsga     OVO Message Agent           AGENT,EA    (5460)   Running
```

3 エージェント・ログ・ファイル内でエラーを検索します（BSM サーバとの通信不能や，証明書エラーなど）。

C:¥Documents and Settings¥All Users¥Application Data¥HP¥HP BTO Software¥log¥System.txt

4 必要に応じて，追跡を有効にします。

```
ovconfchg -ns eaagt -set OPC_TRACE TRUE -set OPC_TRC_PROCS opcle -set
OPC_TRACE_AREA ALL
```

トレースは **C:¥Documents and Settings/All Users/Application Data/HP/HP BTO Software/tmp/OpC and /var/opt/OV/log/tmp** に書き込まれます。

Business Service Management ゲートウェイ・サーバのイベント・チャネルをテストするには，次の手順を実行します。

1 OPR（hpbsm_opr-backend）がイベントの手動送信によって作動しているかどうかを確認します。

a 最初に，Diagnostics が設定した CI の CMDBID を取得します（BSM Diagnostics プローブおよび［インフラ］ビュー内で検索）。

b 次に, **opr¥support** に移動して, 次のコマンドを実行します（太字の CMDBID は上記手順の CMDBID で置き換えます）。

```
sendEvent.bat -s critical -t foo -eh CPU Critical -rch
UCMDB:7b75a57ee89fe6c076ce8d258be4a971
```

2 **log¥opr-backend¥opr-backend.log** 内のフローを確認します。

```
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'PipelineEntry': got 1 events
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'CIResolver': got 1 events
2010-05-11 06:17:18,981 [Thread-37] ERROR
EventChannelCiResolver.resolveHost(172) - No host CI found for event
com.hp.opr.common.model.Event@4ef05e84[865b7200-5cff-71df-00eb-0f2bf8e70000,
Back to normal: Threshold violation(s) for latency on
mercury:bsavm12.ovrtest.adapps.hp.com,<null>,OPEN,NORMAL,NONE,J2EE,<null>,
<null>,UCMDB:17b87476ac6de3938ff9898cd19c8bd8,<null>,<null>,<null>,com.hp.opr.
common.model.ResolutionHints@6cd5499[<null>,ROS84604HAE.ovrtest.adapps.hp.c
om,15.43.248.231,ad2c79b2-9af7-7543-002d-ceeb548960bc],com.hp.opr.common.mo
del.ResolutionHints@126d0c4c[Diagnostics:mercury,ROS84604HAE.ovrtest.adapps.h
p.com,15.43.248.231,ad2c79b2-9af7-7543-002d-ceeb548960bc],<null>,<null>,false,-1,
-1,[],{},Tue May 11 06:17:18 PDT 2010,Tue May 11 06:17:18 PDT 2010,Tue May 11
06:17:18 PDT
2010,0,latency:Normal,<null>,<null>,Diagnostics,latency,N:17b87476ac6de3938ff9898
cd19c8bd8:latency,^<*>:17b87476ac6de3938ff9898cd19c8bd8:latency$,N|latency|jbos
sas|bsavm12.ovrtest.adapps.hp.com|17b87476ac6de3938ff9898cd19c8bd8|mercury|D
efault Client|1|J2EE|PROBE|2010-05-11
06:16:48|latency,false,com.hp.opr.common.model.MatchInfo@35425b07,<null>,<null>]
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'CiVariableReplacer': got 1 events
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'DowntimeProvider': got 1 events
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'EtiResolverByHint': got 1 events
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'EtiResolverByRule': got 1 events
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'HIUpdater': got 1 events
2010-05-11 06:17:18,981 [Thread-37] INFO  MarbleHealthSubmitter.submit(129) -
submitting 1 health updates for customer 1
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'ResolutionCompleted': got 1 events
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'IMDBStore': got 1 events
2010-05-11 06:17:18,981 [Thread-37] INFO  Step.process(45) - Pipeline Step
'PairwiseCorrelation': got 1 events
```

3 必要に応じて，さらにトレースを有効にします。

```
HPBSM¥conf¥core¥Tools¥log4j¥opr-backend¥opr.backend.properties
```

その他のトラブルシューティング手順については，次の項の OM エージェントのトラブルシューティングを参照してください。また，716 ページ「DPS とゲートウェイに対する個別の BSM サーバの設定」も確認できます。

OM エージェントのトラブルシューティング

Diagnostics と Business Service Management 9.0 以降を統合する場合は，OM エージェントおよび IAPA コンポーネントを Diagnostics コマンダ・サーバにインストールする必要があります。これらのコンポーネントは Diagnostics で使用されて，Business Service Management ゲートウェイ・サーバに状況インジケータ・ステータス・イベントを送信します。

- **OM エージェントのインストールの検証済み :** OM エージェントのコンポーネントがインストールされていることを確認します。インストールに問題がある場合は，**＜ Diagnostics のインストール・ディレクトリ＞ /server/log.txt** ファイルにエラーが報告されます。
- **ルート・アクセス要件 :** ルート・アクセスは，OM エージェントと IAPA コンポーネントのインストールに必要です。ルート・アクセスなしで Diagnostics サーバをインストールする必要がある場合は，これら 2 つのコンポーネントをインストールしないで後から手動でインストールしてもかまいません。OM エージェントおよび IAPA コンポーネントのインストール・ダイアログ・ボックスでチェックボックスをオフのまま残して，後からインストールします（72 ページ「OM エージェントおよび IAPA コンポーネントの手動インストール」を参照）。
- **ゲートウェイ・サーバでの証明書の付与 :** OM エージェントおよび IAPA コンポーネントをインストールしたら，Diagnostics を Business Service Management 9.0 以降に登録するときに実行される追加設定を実行する必要があります。登録時に必要な証明書が要求されたら，ゲートウェイ・サーバに証明書を設定するための追加手順を実行します。（「証明書を手動で設定するには」の項の 696 ページ「Business Service Management での Diagnostics サーバの登録」を参照してください。）

Diagnostics コマンダ・サーバから Business Service Management ゲートウェイ・サーバに ping を送信できることを確認するには，次の手順を実行します。

1 Diagnostics コマンダ・サーバで次のコマンドを実行します。

```
bbcutil –ping ＜ゲートウェイ・サーバのホスト名＞
```

予測される出力は，次のとおりです。

```
bsavm12.rose.hp.com: status=eServiceOK
coreID=6c852d02-9ae6-7543-1d6e-b6fab24428f0
bbcV=06.20.101 appN=ovbbccb appV=06.20.101 conn=2
time=218 ms
```

2 eSSLError が表示される場合は，証明書を削除します（Diagnostics コマンダ・サーバの OM エージェントが別の OM サーバで使用されていない場合）。

```
C:\ovcert -list
+------------------------------------------------------------+
| Keystore Content                                           |
+------------------------------------------------------------+
| Certificates:                                              |
|      ad2c79b2-9af7-7543-002d-ceeb548960bc (*)              |
+------------------------------------------------------------+
| Trusted Certificates:                                      |
|      CA_6c852d02-9ae6-7543-1d6e-b6fab24428f0               |
+------------------------------------------------------------+
C:\ovcert -remove ad2c79b2-9af7-7543-002d-ceeb548960bc
C:\ovcert -remove CA_6c852d02-9ae6-7543-1d6e-b6fab24428f0
```

3 **＜Server のインストール・ディレクトリ＞¥server¥bin¥switch_ovo_agent.vbs** を再実行します。

```
cscript switch_ovo_agent.vbs –server ＜ゲートウェイ・サーバ＞ -cert_srv ＜データ処理サーバ＞
```

エージェントがサーバと通信できないかどうか確認するには，次の手順を実行します。

1 **C:¥Documents and Settings¥All Users¥Application Data¥HP¥HP BTO Software¥log¥System.txt** ファイルに, エージェントがサーバと通信できないことを示すエラーが記述されている場合は，**com.hp.ov.opc.msgr** が Business Service Management ゲートウェイ・サーバ上で作動していることを確認します。

```
bbcutil –reg
BasePath=/com.hp.ov.ctrl.ovcd/
        Protocol=HTTPS
        BindAddress=localhost
        Port=1057
        Authentication=REMOTE
    BasePath=/com.hp.ov.opc.msgr/
        Protocol=https
        BindAddress=ANY
        Port=2506
        Authentication=REMOTE
```

OM エージェントが原因で登録がタイムアウトする問題を修正するには，次の手順を実行します。

1 次のようなエラーが表示される場合：

```
Timed Out: cscript //NoLogo C:¥MercuryDiagnostics¥Server¥bin¥switch_ovo_agent.vbs
-server hpswros055.ovrtest.adapps.hp.com -cert_srv
hpswros055.ovrtest.adapps.hp.com
64-bit OS
Server is currently set to '' need to register 'hpswros055.ovrtest.adapps.hp.com'
Certificate server is currently set to '' need to register
'hpswros055.ovrtest.adapps.hp.com'
```

2 Diagnostics の Business Service Management の［登録］ページで，［登録の保存］を再選択します。

BSM ゲートウェイ・サーバとデータ処理サーバ間での OMi 登録に関するトラブルシューティング

Diagnostics 9.x イベントを BSM で表示するために，別のシステムにある BSM ゲートウェイ・サーバとデータ処理サーバ間で OMi を登録するには，次の手順を完了します。

トラブルシューティングを行うには，次の手順を実行します。

1 まず，BSM ゲートウェイ・サーバで証明書を確認します。ゲートウェイ・サーバのコマンド・プロンプトで，コマンド **ovcert --check** を実行します。

問題がある場合は，次のようなエラーが表示されます。

```
C:¥Documents and Settings¥Admin>ovcert -check
OvCoreId set                    : OK
Private key installed           : FAILED
Certificate installed           : FAILED
Certificate valid               : FAILED
Trusted certificates installed  : FAILED
Trusted certificates valid      : FAILED
```

問題がない場合は，次のようなメッセージが表示されます。

```
C:¥Documents and Settings¥Admin>ovcert -check
OvCoreId set                    : OK
Private key installed           : OK
Certificate installed           : OK
Certificate valid               : OK
Trusted certificates installed  : OK
Trusted certificates valid      : OK
Check succeeded.
```

2 BSM データ処理サーバで OMi BSM ゲートウェイ・サーバが正しく登録されていることを確認します。BSM ゲートウェイ・サーバのコマンド・プロンプトで，コマンド **bbcutil -ping ＜データ処理サーバ＞**を実行します。

問題がある場合は，次のようなエラーが表示されます。

(bbc-289) status=eServiceError coreID= bbcV= appN= appV=conn=0 time=109 ms

問題がない場合は，次のようなメッセージが表示されます。

status=eServiceOK coreID=09139942-991a-7549-1eae-ee2cbe62289a

bbcV=11.00.044 appN=ovbbccb appV=11.00.044 conn=2 time=156 ms

3 BSM ゲートウェイ・サーバで OMi BSM データ処理サーバが正しく登録されていることを確認します。BSM データ処理サーバのコマンド・プロンプトで，コマンド **bbcutil -ping <ゲートウェイ・サーバ>**を実行します。

問題がある場合は，次のようなエラーが表示されます。

(bbc-288) status=eServiceError coreID= bbcV= appN= appV=
conn=0 time=109 ms

問題がない場合は，次のようなメッセージが表示されます。

status=eServiceOK

coreID=c3475f92-a584-7546-1cfb-b2612a96538f

bbcV=11.00.044 appN=ovbbccb appV=11.00.044

conn=2 time=94 ms

4 上記のいずれかの問題がある場合は，次の手順を使用して，別の BSM ゲートウェイ・サーバおよびデータ処理サーバで証明書を設定します（『BSM デプロイメント・ガイド』を参照）。

a BSM ゲートウェイ・サーバのコマンド・プロンプトで，次のコマンドを実行します。

ovconfchg -ns sec.cm.client -set CERTIFICATE_SERVER <データ処理サーバ>

ovcert -certreq

INFO: Certificate request has been successfully triggered

b BSM データ処理サーバのコマンド・プロンプトで，次のコマンドを実行します。

ovcm -listpending -l

RequestID: e0609222-7ea6-754d-0a03-e1a09dc5dd43
Context:
CN: c3475f92-a584-7546-1cfb-b2612a96538f
Nodename: bsavm9.ovrtest.adapps.hp.com
IPAddress: 16.93.25.199
PeerAddress: 16.93.25.199
Platform: Windows 5.2, CPU: x64
InstallType: Manual
TimeReceived: 1/19/2011 1:49:18 PM Pacific Standard Time

c 要求 ID を付与します。

ovcm -grant e0609222-7ea6-754d-0a03-e1a09dc5dd43

d BSM ゲートウェイ・サーバのコマンド・プロンプトで，次のコマンドを実行します。

ovcert --list

次の例に示すような証明書と信頼された証明書が表示されます。

```
+-------------------------------------------------------+
| Keystore Content                                      |
+-------------------------------------------------------+
| Certificates:                                         |
|    c3475f92-a584-7546-1cfb-b2612a96538f (*)           |
+-------------------------------------------------------+
| Trusted Certificates:                                 |
|    CA_09139942-991a-7549-1eae-ee2cbe62289a            |
+-------------------------------------------------------+
```

e BSM ゲートウェイ・サーバのコマンド・プロンプトで，次のコマンドを実行します。

bbcutil -ping ＜データ処理サーバ＞

これで Ping が動作するようになります。

status=eServiceOK coreID=09139942-991a-7549-1eae-ee2cbe62289a

bbcV=11.00.044 appN=ovbbccb appV=11.00.044 conn=2 time=156 ms

5 これで動作するようになります。

全般的な参照情報

このセクションには一般的な参照トピックが含まれています。

本章の内容

UNIX コマンドの使用

UNIX プラットフォームでインストールを実行する場合，通常，画面に表示される指示に従って操作することができます。ときどき，画面の指示が混乱することがあります。

不明点があるときは，次のガイドラインを使ってください。

- オプションのリストからオプションを選択するには，オプションに対応する番号を入力して **Enter** キーを押します。0 を入力し，もう一度 **Enter** キーを押して選択内容を確定します。
- 複数のオプションを選択する場合，選択ごとに対応する番号を入力して **Enter** キーを押します。すべてのオプションを選択し終えたら，0 を入力し，**Enter** キーを再度押して，選択内容を確認します。
- 選択したオプションを削除する場合は，対応する番号を再入力するか，ほかのオプションの番号を入力して **Enter** キーを押します。0 を入力し，もう一度 **Enter** キーを押して選択内容を確定します。
- プロンプトを情報を入力する場合

- プロンプトに表示されたデフォルト値を受け入れるには，**Enter** キーを押します。
- 情報を入力し，**Enter** キーを押して進みます。

- インストールの次の手順に進むには，1 を入力して［**次へ**］を選択し，**Enter** キーを押します。
- 前のプロンプトに戻って変更する場合は，2 を入力して［**前へ**］を選択し，**Enter** キーを押します。
- インストールを中止するには，3 を入力して［**キャンセル**］を選択し，**Enter** キーを押します。
- プロンプトを再表示するには，4 を入力して［**Redisplay**］を選択し，**Enter** キーを押します。

正規表現の使用

キャプチャ・ポイント・ファイルで各プローブのインストルメンテーション定義を指定する際，ポイントのほとんどの引数に正規表現を使用できます。

正規表現は，複雑な検索語句を指定する文字列です。ピリオド（.），アスタリスク（*），脱字記号（^）および角括弧（[]）などの特殊文字を使うと，検索条件を定義できます。

注 : Diagnostics の正規表現は，感嘆符を先頭に付ける必要があります。

標準設定で，Diagnostics では，ピリオド（.），ハイフン（-）アスタリスク（*），脱字記号（^），角括弧（[ ]），丸括弧（()），ドル記号（$），縦線（|），プラス符号（+），疑問符（?）および円マーク（¥）を除いて，正規表現のすべての文字をそのまま処理します。これらのいずれかの特殊文字の前に円マーク（¥）が付いている場合，Diagnostics は，そのままの文字として扱います。

一般的な正規表現の演算子

このセクションでは，正規表現の作成に使うことができるいくつかの一般的な演算子について説明します。

注：サポートされている正規表現の文字の完全なリストおよび説明については，『Microsoft VBScritp マニュアル』の「正規表現」セクションを参照してください。

演算子	用途
(**¥**)	特殊文字リテラルを表現する リテラル文字から特殊文字を作成する
(**.**)	1 文字の検索
(**[xy]**)	リストにある 1 文字の検索
(**[^xy]**)	リストにない 1 文字の検索
(**[x-y]**)	範囲内の 1 文字を検索する
(*****)	ゼロ個以上の特定の文字を検索する
(**+**)	1 個以上の文字を検索する
(**?**)	ゼロまたは特定の 1 文字を検索する
(**( )**)	正規表現のグループ化
(**|**)	複数の正規表現のうち 1 つの検索
(**^**)	行の先頭の検索
(**$**)	行の末尾の検索
(**¥w**)	下線を含む英数字の検索
(**¥W**)	非英数字の検索

円マークの使用

円マーク（¥）は 2 つの役割を果たすことができます。特殊文字と一緒に使って，次の文字をリテラル文字として扱うように指定できます。たとえば，¥. はワイルドカードの代わりにピリオド（.）として処理されます（885 ページ「1 文字の検索」を参照）。

円マーク（¥）を n，t，w または d 文字など，リテラル文字として処理される文字と一緒に使うと，その組み合わせで特殊文字を表現します。たとえば，¥n は改行文字を表します。

次に例を示します。

- w は，文字 w に対応します。
- ¥w は，下線を含む文字に対応する特殊文字です。
- ¥¥ は，リテラル文字 ¥ に対応します。
- ¥(は，リテラル文字 (に対応します。

たとえば，次の名前のファイルを探している場合。

filename.ext

ピリオドは，正規表現の記号と誤解されることがあります。ピリオドが正規表現の一部でないことを示すには，次のように入力します。

filename¥.ext

注 : 特別な意味のない文字の前に円マークが使われている場合, 円マークは無視されます。たとえば，¥z は z と解釈されます。

---

1 文字の検索

ピリオド (.) は，Diagnostics に任意の 1 文字を検索するように指示します (¥n を除く)。次のようになります。

```
welcome.
```

この場合，後ろにスペースや 1 文字が付く **welcomes**，**welcomed** または **welcome** を一致とみなします。ピリオドの数は，未指定の文字の数を表します。

¥n を含む 1 文字を検索するには，次のように入力します。

```
(.|¥n)
```

( ) 正規表現文字については，887 ページ「正規表現のグループ化」を参照してください。**|** 正規表現文字については，888 ページ「複数の正規表現のうち 1 つの検索」を参照してください。

リストにある 1 文字の検索

角括弧は，Diagnostics に文字のリストの中の 1 文字を検索するように指示します。たとえば，日付の 1967，1968 または 1969 を検索する場合は次のように入力します。

```
196[789]
```

リストにない 1 文字の検索

角括弧内で脱字記号（^）が先頭にある場合は，Diagnostics に，文字列で指定したもの以外のリストの文字を検索するように指示します。次のようになります。

[^ab]

この場合，**a** または **b** 以外の文字を一致とみなします。

注 : 脱字記号は，括弧内の先頭にある場合のみ特別な意味を持ちます。

範囲内の 1 文字を検索する

範囲内の 1 文字を検索する場合，角括弧（[ ]）とハイフン（-）を使用できます。たとえば，1960 年代の特定の年度を検索する場合は次のように入力します。

196[0-9]

ハイフンは，括弧内の先頭または末尾にある場合，または脱字記号（^）の後にある場合は範囲を表しません。

たとえば，[-a-z] は，ハイフンまたは小文字を一致とみなします。

注 : 括弧内で，文字「.」，「*」，「[」，および「¥」はリテラルです。たとえば，**[.*]** は，. または * を一致とみなします。範囲内の先頭の文字が開く括弧の場合もリテラルになります。

ゼロ個以上の特定の文字を検索する

アスタリスク（*）は，Diagnostics に，直前の文字がゼロ個以上のものを検索するように指示します。次のようになります。

```
ca*r
```

この場合，**car**，**caaaaaar** および **cr** を一致とみなします。

1 個以上の文字を検索する

プラス符号（+）は，Diagnostics に，直前の文字が 1 つ以上あるものを検索するように指示します。次のようになります。

```
ca+r
```

この場合，**car** および **caaaaaar** を一致とみなしますが，**cr** は一致とみなしません。

ゼロまたは特定の 1 文字を検索する

疑問符（?）は，Diagnostics に，直前の文字が 0 個または 1 個のものを検索するように指示します。次のようになります。

```
ca?r
```

この場合，**car** および **cr** だけを一致とみなします。

正規表現のグループ化

丸括弧（()）は，Diagnostics に，含まれる数列を数学やプログラミング言語のように一塊として処理するように指示します。

グループを使うと，引数を選択演算子（ | ）または反復演算子（*，+，?，{ }）に区切る場合に特に便利です。

複数の正規表現のうち 1 つの検索

縦線（|）は，Diagnostics に式の選択肢の中からいずれかを検索するように指示します。次のようになります。

```
foo|bar
```

この場合，Diagnostics は **foo** または **bar** を検索します。

```
fo(o|b)ar
```

この場合，Diagnostics は **fooar** または **fobar** を検索します。

行の先頭の検索

脱字記号（^）は，行頭または改行文字の直後にある場合のみ検索するように Diagnostics に指定します。

次に例を示します。

```
book
```

は行 **book**，**my book**，**book list** の中の **book** を検索しますが，これに対して

```
^book
```

は **lines book** と **book list** でのみ **book** を検索します。

行の末尾の検索

ドル記号（$）は，行末または改行文字の直前にある場合のみ検索するように Diagnostics に指定します。次のようになります。

```
book
```

は，行 **my book** と **book list** の中で **book** を検索しますが，直後に（$）がある場合はその文字列で終了している行のみ検索します。

```
book$
```

は，行 **my book** でのみ **book** を検索します。

下線を含む英数字の検索

¥w は，Diagnostics に，英数字と下線（A-Z，a-z，0-9，_）を検索するように指示します。

次に例を示します。

¥w* は，Diagnostics に，**A-Z**，**a-z**，**0-9** および下線（_）など，英数字がゼロ個以上のものを検索するように指示します。この場合，**Ab**，**r9Cj**，または **12_uYLgeu_435** を一致とみなします。

次に例を示します。

¥w{3} は，Diagnostics に，**A-Z**，**a-z**，**0-9** および下線（_）など，英数字が 3 つあるものを検索するように指示します。この場合，**Ab4**，**r9_**，または **z_M** を一致とみなします。

非英数字の検索

¥W は，Diagnostics に，英数字と下線以外の任意の文字を検索するように指示します。

次に例を示します。

¥W は，**&**，*****，^，**%**，**$** および **#** を一致とみなします。

正規表現の演算子の結合

1 つの式で正規表現演算子を結合して，正確な検索条件を指定することができます。

たとえば，‘.’と ‘*’の文字を組み合わせて，0 個以上の任意の文字を検索することができます（¥n を除く）。

次に例を示します。

```
start.*
```

この場合，**start**，**started**，**starting**，**starter** などを一致とみなします。

括弧とアスタリスクの組み合わせを使って，非英数字の組み合わせに検索対象を絞ることができます。次のようになります。

```
[a-zA-Z]*
```

0 ～ 1200 の数値を検索する場合，1000 ～ 1200 の 1 桁，2 桁，3 桁または 4 桁を検索する必要があります。

次の正規表現は，0 ～ 1200 の数値を一致とみなします。

```
([0-9]?[0-9]?[0-9]|1[01][0-9][0-9]|1200)
```

多言語ユーザ・インタフェースのサポート

Diagnostics ユーザ・インタフェース（UI）は，複数の言語で Web ブラウザに表示できます。これは，Diagnostics が Business Service Management と統合されているか，またはスタンドアロン・モード（統合なし）で実行されている Windows 環境に適用されます。

Diagnostics が LoadRunner または Performance Center と統合されている場合，UI の表示言語は，クライアントのロケール設定（お使いのオペレーティング・システムの地域設定で定義）によって決まります。

注意 : Diagnostics では，Agent 名のローカリゼーションはサポートされていません。

本付録では，Diagnostics ユーザ・インタフェースを特定の言語で表示する方法について説明します。Diagnostics UI は，Web ブラウザで次の言語で表示できます。

言語	Web ブラウザの言語設定
英語	英語
簡体中国語	中国語（中国）[zh-cn], 中国語（シンガポール）[zh-sg]
韓国語	韓国語 [ko]
日本語	日本語 [ja]

Diagnostics の表示方法を選択するには，ブラウザの言語設定オプションを使用します。言語設定の選択は，ユーザのローカル・マシンだけに影響があり，Diagnostics サーバや，同じ Diagnostics サーバにアクセスしているほかのユーザには影響しません。

Diagnostics UI を特定の言語で表示するには，次の手順を実行します。

1 ローカル・マシンに適切な言語のフォントがインストールされていない場合は，インストールします。フォントがインストールされていない言語を Web ブラウザで選択すると，Diagnostics ユーザ・インタフェースでは，ローカル・マシンの標準設定の言語が使用されます。

たとえば，ローカル・マシンの標準言語が英語で，Web ブラウザが日本語を使うように設定されていると仮定します。ローカル・マシンに日本語フォントがインストールされていない場合，Diagnostics ユーザ・インタフェースは英語で表示されます。

2 Internet Explorer を使用している場合，ローカル・マシンの Web ブラウザを設定して，Diagnostics ユーザ・インタフェースを表示する言語を指定します。詳細については，http://support.microsoft.com/kb/306872/ja-jp を参照してください。

手順 4 に進みます。

3 FireFox を使用している場合，ローカル・マシンの Web ブラウザを次のように設定します。

a ［**ツール**］＞［**オプション**］＞［**詳細**］を選択します。［**言語設定**］をクリックします。［言語］ダイアログ・ボックスが開きます。

b Diagnostics を表示する言語を強調表示します。

使用する言語がダイアログ・ボックスにない場合は，［**言語を選択して追加**］リストを拡張して言語を選択し，［**追加**］をクリックします。

c ［**上へ**］をクリックして，選択した言語を先頭の行に移動します。

d ［**OK**］をクリックして設定を保存します。［**OK**］をクリックして［言語］ダイアログ・ボックスを閉じます。

4 既存のブラウザを閉じて，新しいブラウザに Diagnostics を開きます。Diagnostics ユーザ・インタフェースが選択した言語で表示されます。

J

データのエクスポート

Diagnostics で収集したメトリックス・データを，サード・パーティのデータベースへ直接アーカイブして保存したり，データベースによってサポートされているレポート形式に書式化したりできます。

このデータ・エクスポートは，Diagnostics のすべての永続データ用のリポジトリである Diagnostics Time Series データベース（TSDB）から必要な測定値を取得する，XPath に似たクエリによって行われます。TSDB の詳細については，825 ページ「Diagnostics のデータ管理」を参照してください。

本章の内容

タスク 1：ターゲット・データベースの準備

エクスポートするデータ用のターゲット・データベースとして，Diagnostics Commanding Server からアクセスできる SQL Server または Oracle データベースを使用できます。サポートされている環境の最新情報については，Diagnostics のサポート早見表（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

Diagnostics サーバがデータ・エクスポートを実行すると，ターゲット・データベース内に自動的にスキーマとテーブルが作成されます。ターゲット・データベースには 1 GB 以上の空き容量が必要です。最初に何回かエクスポートする際に，データベースのサイズを観察して容量を増やす必要があるかを確認する必要があります。

ターゲット・データベースに接続するには，データベースへの読み取り / 書き込み権限があり，テーブル定義権限のあるユーザのログイン資格情報を指定する必要があります。

注： Diagnostics 9.10 以降にアップグレードして 9.10 より前の旧データベース・コンテンツを保持する場合，9.10 以降の新機能（最小値と最大値で整数の代わりに倍精度を使用することで，エクスポートするデータに小数位を表示可能）を利用するようにデータベースを変更できます。アップグレード後にデータベースを次のように変更します。

Oracle :

```
ALTER TABLE RECORD MODIFY (
  REC_COUNT   NUMBER(38),
  TOTAL       FLOAT,
  MINIMUM     FLOAT,
  MAXIMUM     FLOAT)
```

SQL Server :

```
ALTER TABLE RECORD ALTER COLUMN REC_COUNT   DECIMAL(19)
ALTER TABLE RECORD ALTER COLUMN TOTAL       FLOAT
ALTER TABLE RECORD ALTER COLUMN MINIMUM     FLOAT
ALTER TABLE RECORD ALTER COLUMN MAXIMUM     FLOAT
```

タスク 2 : エクスポートするメトリックスの決定

いくつかの方法で，エクスポートするメトリックスを制御できます。特定のエンティティ・タイプのすべての測定値を取得するように指定できます。そのグループからメトリックスを除外したり, 特定のメトリックスだけを含むように指定したりできます。Diagnostics データ・モデルの詳細については，DVD およびヘルプ内の『Diagnostics Data Model and Query API』ドキュメントを参照してください。

重要 : データ・エクスポート操作では，測定データのみエクスポートされます。つまり回数，遅延時間，平均値です。インスタンス・データやステータス・データはエクスポートされません。また，呼び出しプロファイル・データもエクスポートできません。

測定値は，適用するエンティティ・タイプ別やほかの条件によってグループ化されます。次のエンティティ・タイプ・グループが最もよく使われます。

- /probegroup/probe

  すべてのプローブ・グループのすべてのプローブの測定値。
- /probegroup/probe/fragment

  すべてのプローブ・グループとプローブのすべてのサーバ要求の測定値。
- /probegroup/index[name='rollup_fragment']/fragment

  すべてのプローブ・グループのプローブによってロールアップされたサーバ要求のメトリックス。
- /probegroup/probe/index[name='services']/service

  すべてのプローブ・グループとプローブの Web サービス（操作以外）の測定値。
- /index [equals(name,'apps')]/app/app_metrics

  特定のアプリケーションの測定値。
- /probegroup/probe[equals(probeType, 'Oracle')]

  すべての Oracle Collector の測定値。
- /probegroup/probe[equals(probeType, 'SqlServer')]

  すべての SqlServer Collector の測定値。
- /host -- すべてのホストのメトリックス（さまざまなシステム・メトリックス）。
- /txn -- すべての BPM トランザクションのメトリックス。

次の表は，メトリックスのタイプとそのカテゴリの例を示します。

カテゴリ	メトリックス
Classes	現在ロードされているクラス ロードされているクラス アンロードされているクラス
Dynamic Caching	現在のキャッシュ・サイズをキャッシュ 最大キャッシュ・サイズをキャッシュ
EJB	EJB のアクティブ化 EJB アクティブ化時間 EJB のコミットされたトランザクション / 秒 EJB 同時実行アクティブ・メソッド EJB 同時実行ライブ・ビーン EJB 作成時間 作成される EJB EJB ドレイン・サイズ プールからの EJB ドレイン フリー EJB 見つかった EJB プールから取得される EJB 初期化される EJB EJB ロード時間 ロードされる EJB EJB の非アクティブ化 EJB 非アクティブ化時間 EJB 非アクティブ化ビーン EJB プール・サイズ EJB 準備完了ビーン EJB リモート時間 削除される EJB EJB 応答時間 返される破棄された EJB プールに返される EJB EJB のロールバックされたトランザクション / 秒 EJB 格納時間 格納される EJB

カテゴリ	メトリックス
EJB （続き）	EJB のタイムアウトしたトランザクション / 秒 EJB 合計メソッド呼び出し EJB キャッシュ・アクセス / 秒 EJB のキャッシュされたキャッシュ・ビーン EJB キャッシュ取得の失敗 / 秒 EJB プール・アクセス / 秒 EJB プールの使用可能なビーン 使用中の EJB プール・ビーン EJB プールの現在の待機 EJB プール取得の失敗 / 秒 EJB プール取得タイムアウト / 秒
Execute Queues	スレッドがアイドル状態の実行キュー 要求が保留中の実行キュー 実行キュー要求 / 秒 実行キューの合計スレッド
GC	GC コレクション / 秒 コレクションに費やされた GC 時間
Http Status	5xx-6xx
J2C Connections	処理される J2C 接続 リリースされる J2C 接続 割り当てられる J2C 接続 閉じられる J2C 接続 作成される J2C 接続
JDBC	作成される JDBC 接続 / 秒 JDBC 作成された接続の遅延 JDBC 現在のキャパシティ JDBC の実行ステートメント JDBC のリークされた接続 JDBC 再接続失敗 接続を待機中の JDBC 要求 JDBC ステートメント・キャッシュ・アクセス / 秒 JDBC ステートメント・キャッシュ・ヒット / 秒 JDBC ステートメント・キャッシュ・サイズ JDBC の開いている合計接続数 JDBC 待機秒数（高）

カテゴリ	メトリックス
Latency	latency total_cpu exception_count timeout_count スループット

タスク 4 の説明に従って，エクスポートするグループまたは個別のメトリックスを指定します。

タスク 3：頻度と回復期間の決定

エクスポート操作ごとに頻度が指定され，それによってエクスポートが発生する頻度と返される測定値の粒度が制御されます。推奨される頻度は 1h（毎時）です。つまり 1 時間ごとにエクスポート操作が実行されます。それ以外の頻度のオプションは 5m と 1d です。

注：データ・エクスポート操作は必要な頻度で実行できますが，この操作は Diagnostics サーバのパフォーマンスに影響します。頻度が高くなるほど，サーバへの負荷は高くなります。Commanding Server の負荷を軽減するために，Mediator のエクスポート処理を一括で行うように設定できます（サーバの **etc/data-export-config.xml** ファイルで **servers-per-query** 属性を設定）。

また，頻度の回復期間も指定できます。回復期間は，Diagnostics コマンド・サーバがシャットダウンされた場合か，または回復期間を使わないと Diagnostics コマンド・サーバが利用できなくなる場合にかぎり使われます。この値は，Diagnostics コマンド・サーバの動作が再開した場合に，データ・エクスポート操作をどのくらい過去に遡って実行するかを Diagnostics コマンド・サーバに指定します。

頻度回復期間の数式は次のとおりです。

（現在の時間）-（回復期間 * 頻度）

たとえば，Diagnostics コマンド・サーバが 24 時間アクティブでなかったとします。頻度が 1 時間ごとのデータ・エクスポート操作は，少なくとも 23 回実行されていません。デフォルトでは，データ・エクスポート操作は障害が起きた時点でクエリを開始します（過去 24 時間）。1 時間ごとのデータのメトリックスはより大きなバケットに集計されているため，返されるメトリックスには意味がありません。

ただし，回復期間が 6h と指定されている場合，1 時間ごとのタスクは TSDB へのクエリを開始するために（24 時間ではなく）6 時間遡ります。この測定値は意味があります。

タスク 4 の説明に従って，<frequency> 要素と <recovery-periods> 要素を設定します。

タスク 4：データ・エクスポート設定ファイルの変更

Diagnostics データをエクスポートするクエリは，Diagnostic コマンダ・サーバの**＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/ data-export-config.xml** ファイルで定義されます。

このファイルをセットアップするするには，次の手順を実行します。

1. 必要に応じて，＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/ data-export-config.xml ファイルのバックアップ・コピーを作成します。
2. ＜ Diagnostics サーバのインストール・ディレクトリ＞ /etc/ data-export-config.xml ファイルを編集用に開きます。
3. <enabled> 要素を探して true に設定します。

```
<enabled>true</enabled>
```

この要素を使ってデータ・エクスポート操作のオンとオフを切り替えます。データ・エクスポート操作は，不要なシステム・オーバーヘッドを回避する必要がない場合は無効にする必要があります。デフォルトではデータ・エクスポート操作は無効になっています。

4. <customer name> 要素を探してカスタマ名に設定します。

SaaS カスタマでない限り，カスタマ名は常に Default Client にする必要があります。

```
<customer name=’Default Client">
```

5 <db-target> 要素を探し，ターゲット・データベースのドライブ名，接続 URL，ユーザ名およびパスワード（暗号化またはプレーン・テキスト）を入力します。

たとえば，暗号化されたパスワードを使用した SQL Server の場合は次のようになります。

```
<db-target>
   <driver>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver>
   <connection-url>jdbc:sqlserver://testapps.hp.com:1433;databaseName=DIAG</
connection-url>
   <username>sa</username>
   <encrypted-password>OBF:1ym51y0s1uo71z0f1unr1y0y1ym9</encrypted-password>
   <batchsize>200</batchsize>
</db-target>
```

たとえば，暗号化されていないパスワードを使った Oracle の場合は次のようになります。

```
<db-target>
   <driver>oracle.jdbc.driver.OracleDriver</driver>
   <connection-url>jdbc:oracle:thin:@testapps.hp.com:1521:ORCL</connection-url>
   <username>diagfan</username>
   <password>tiger2</password>
   <connection-property  name="oracle.jdbc.defaultNChar" value="true"/>
</db-target>
```

Oracle データベースでは，UTF8 / UTF16 文字サポートが必要な場合は **oracle.jdbc.defaultNChar** プロパティを true に設定する必要があります。

データベース・パスワードを暗号化するには，Diagnostics パスワード・エンクリプタを使います。詳細については，123 ページ「パスワードの暗号化」を参照してください。

<batchsize> 要素には，最適な JDBC PreparedStatement の実行のためのバッチ・サイズをユニット数で指定します。標準設定では 100 に設定されています。ペイロードが大きい大規模な実装では，標準設定値に調節する必要があります。

6 エクスポートするメトリックス・セットごとに，< query >内の次の項目を指定します。

id= 定義するクエリを特定する名前。この data-export-config.xml ファイルに対して一意である必要があります。id の値にスペースは使用できません。

frequency= クエリを実行する頻度を指定する文字列値。選択肢は，1h，5m，1d です。

recovery-periods= 障害発生後，クエリを開始するために遡る時間を指定します。

< entity-path >：タスク 2 で説明したエンティティ・パス・グループの 1 つ。

< init-query-time >または< init-query-periods >：クエリを開始する過去の時刻。< init-query-time >は，標準 XSD 時刻書式で指定される時刻の値です。< init-query-periods >は，クエリの時刻を決定するために，頻度を乗じる整数です。省略した場合，クエリは次の頻度の区切りで実行されます。

たとえば，次のエントリでは，1 時間ごとに実行しすべてのプローブ・グループ内のすべてのプローブの測定値をすべて返すクエリが作成されます。障害が発生した場合，現在時刻から 2 時間遡って回復します。

```
<query id="Probes" frequency="1h" recovery-periods="2">
   <entity-path>/probegroup/probe</entity-path>
</query>
```

次のエントリでは，1 時間ごとに実行し，すべてのプローブ・グループの 1 時間ごとのロールアップ・フラグメントの遅延メトリックスを返すクエリが作成されます。

```
<query id="Aggregate-SRs" frequency="1h" recovery-periods="2">
   <entity-path>/probegroup/index[name='rollup_fragment']/fragment</entity-path>
</query>
```

次のエントリでは，1 時間ごとに実行し，4 月 22 日午後 3 時以降のすべてのプローブ・グループの 1 時間ごとの Web サービスの遅延メトリックスを返すクエリが作成されます。

```
<query id="Web-Services" frequency="1h" recovery-periods="2">
   <entity-path>/probegroup/probe/index[name='services']/service</entity-path>
   <init-query-time>2009-04-22T15:00:00</init-query-time>
</query>
```

7 必要に応じて，サーバの負荷を軽減する方法としてクエリで servers-per-query 属性を使用できます。

servers-per-query は，エクスポート・クエリの一括処理を指定します。

```
<query id="Probes" frequency="1h" recovery-periods="2" servers-per-query="10">
   <entity-path>/probegroup/probe</entity-path>
</query>
```

たとえば，30 個のメディエータ・サーバがあり，servers-per-query を 10 に設定した場合，エクスポートで 10 個のメディエータの結果が取得され，これらの結果をエクスポートしてから次の 10 個のメディエータが処理されます。

この属性は，10 個以上のメディエータを使用する場合のみに設定します。

8 任意で，各クエリには，<entity-path>　要素で指定したクエリに適用する包含フィルタまたは除外フィルタを追加できます。包含フィルタ要素は，除外フィルタ要素よりも先に指定する必要があります。

どちらのフィルタに対しても，次の項目を指定する必要があります。

name= フィルタリングする測定値名を照合する正規表現。"" はすべての測定値と一致します。

category= フィルタリングするカテゴリ名を照合する正規表現。"" はすべてのカテゴリと一致します。

<order> : 複数の包含または除外フィルタの場合，フィルタを処理する順番。

たとえば，次のエントリは Database 測定値の測定値を返します。

```
<query id="Probes" frequency="5m" recovery-periods="2">
   <entity-path>/probegroup/probe</entity-path>
   <metric-include-filter order="1" name="" category="Database" />
   <metric-exclude-filter order="1" category="" />
</query>
```

次の例では，EJB-Poll 測定値を除き，EJB のすべての測定値を返します。

```
<query id="EJBStats" frequency="5m" recovery-periods="2">
  <entity-path>/probegroup/probe</entity-path>
  <metric-include-filter order="1" name="" category="EJB" />
  <metric-exclude-filter order="1" name="EJB-Pool" />
</query>
```

正規表現の詳細については, 882 ページ「正規表現の使用」を参照してください。

9 任意で, <purge> 要素を指定することによって, 抽出したデータのデータ保存ルールを指定します。これにより，データベースの容量不足を回避します。

たとえば，次のエントリの場合，24 時間を過ぎたデータ（retention="1d"）はターゲット・データベースから削除されます。削除操作は 1 時間ごとに開始され（frequency="1h"），対象となるすべての削除操作は 1 時間の増分（purgeInterval="1h"）でデータを削除します。これによりシステム全体の負荷を軽減しています。

```
<purge id="Default.Client.Purger" frequency="1h" retention="1d" purgeInterval="1h"/>
```

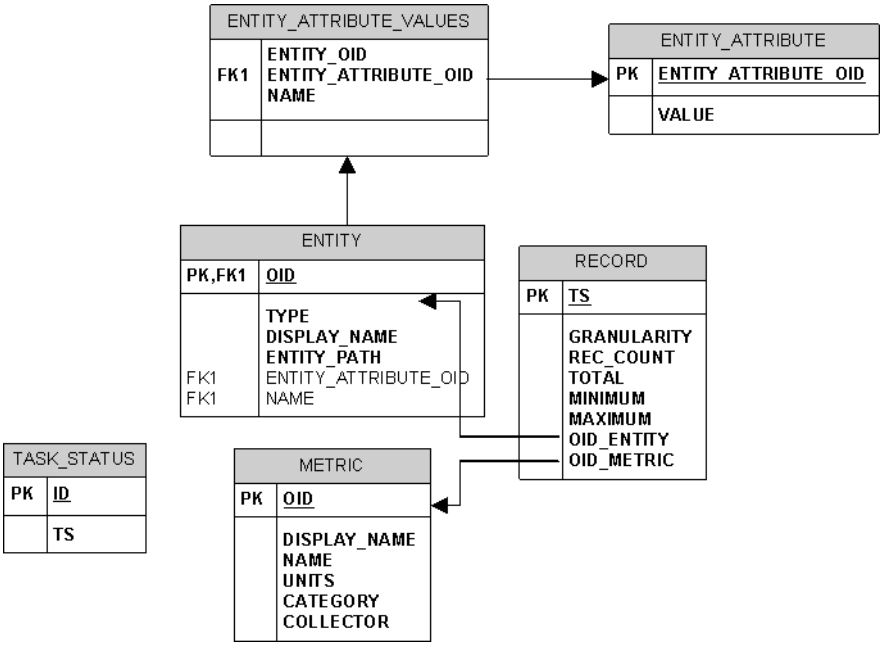
10 **data-export-config.xml** ファイルへの変更を保存します。

タスク 5 : データ・エクスポート操作の監視

対応する Commanding Diagnostics サーバが開始すると, 保存されている設定ファイルのエントリがすぐに有効になります。Diagnostics コマンド・サーバを再起動する必要はありません。

data-export-config.xml は次のように解析され，検証されます。

1 XML で指定されている各 [db-target] は，接続してデータベース・テーブルが利用可能であることを確認することによって検証されます。データベース・テーブルが利用できない場合，次のデータ関係を使って作成されます。



2 Diagnostics サーバは，指定された <frequency> に基づいて各クエリをいつ実行するかをスケジューリングします。たとえば，日次レポート（frequency = 1d）は，1 日の最後の 1 時間が日次サマリに集計された後，1 日に 1 回実行されます。これは特に，クエリが自動的に既存の粒度の区切りで揃えられていることを意味します。

3 予定されているクエリが実行すると，クエリの結果は次のようにデータベース・テーブルに保存されます。

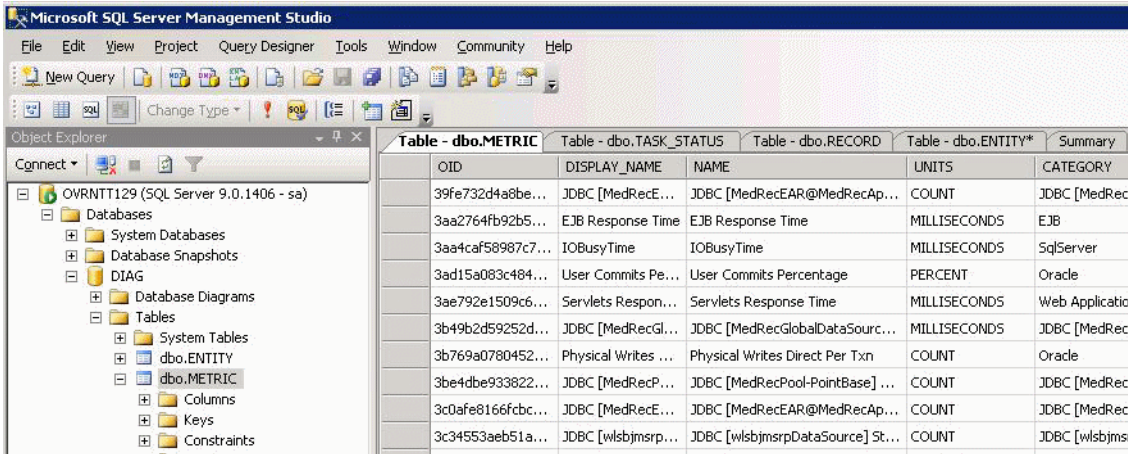
- プローブ，フラグメント，ホストなどのエンティティの詳細は ENTITY テーブルに保存され，一意のキーは，連携している環境内で一意にするために，TSDB エンティティ・キーと分散ソース値の MD5 ハッシュです。

- 測定値の詳細は METRIC テーブルに保存され，一意のキーは，連携している環境内で一意にするために，測定値データ名，測定値データ・コレクタ名，および分散ソースの MD5 ハッシュです。
- 測定値は RECORD テーブルに保存され，ENTITY および METRIC テーブルの一意キーを指す外部キー値を持ちます。これはエンティティと測定値の詳細が RECORD テーブルの各行で重複するため，データ・ストレージの全体サイズを縮小するために行われます。
- ENTITY_ATTRIBUTE_VALUES と ENTITY_ATTRIBUTE という 2 つのテーブルは，ENTITY のディメンションを詳細に描写するためのルックアップ・テーブルの役割を果たします。

タスク 6 : 結果の検証

ターゲット・データベースのデータベース・ツールを使って，期待される結果を検証します。たとえば，以下の図は，SQL Server データベースの METRIC テーブルに保存されている結果を示します。この測定値セットは，次のクエリ・ステートメントに基づいています。

```
<query id="Probes" frequency="1h" recovery-periods="2">
   <entity-path>/probegroup/probe</entity-path>
</query>
```



タスク 7 : ターゲット・データベースからのデータの選択

エクスポートしたデータをターゲット・データベースに保存したら，必要に応じてそれをクエリできます。ただし，データを有効なレポート書式で取得するには変換処理が必要です。変換処理では，外部キー参照を使って，ファクト・テーブルと連結した詳細データを納めたフラットな行にディメンション・テーブル・データを結合します。

サンプルの SQL スクリプト，クエリ，レポートが次のように < Server のインストール・ディレクトリ > /contrib/dataexport/ に含まれています。

- **sql_server_sample_view.sql :** エクスポートしたデータを利用しやすいレポート書式に非正規化して変換するために使われる SQL Server ビュー。
- **oracle_server_sample_view.sql** : エクスポートしたデータを利用しやすいレポート書式に非正規化して変換するために使われる Oracle DB Server ビュー。
- **oracle_view_query_samples.sql :** Oracle ビューのクエリのさまざまな例。
- **sql_server_reports directory** : SQL Server Reports プロジェクトとさまざまなサンプル・レポートが含まれているディレクトリ。SQL Server Reporting ツールを使って，sql_server_reports ディレクトリを開き，ファイル「Diagnostic Fragments.sln」を開きます。

サンプル・クエリ

このセクションでは，時刻の処理，合計のクエリ，および平均のクエリを行うクエリの例をいくつか示します。

時刻の処理

午前 8 時から午後 5 時までのデータをクエリするには，開始時刻を 8:00 に，終了時刻を 16:59 に指定する必要があります。クエリの終了時刻に 17:00 を指定した場合は，次回のバケット（17:00 ～ 18:00 など）のデータが含まれます。これは，次の例にも反映されています。

合計のクエリ

メトリックスの合計を計算するには，sum() 関数を使用します。

例：

このクエリは，次のエンティティ・パスを含むサーバ要求に対して午前 6 時から午後 8 時までのしきい値違反の合計数を計算します。
Default Client / Default / ROS54770TST_Diag80_JDK_15

```
select entity_display_name as Server_Request, sum(total) as
Avg_Latency,sum(total) as Tot_Latency, sum(rec_count) as count, metric_name,
units, name, entity_path
from DIAG.DBO.REG_FRAGMENT_TYPE_METRICS_VIEW
where metric_name = 'threshold_violations'
  and ts between '2009-06-11 18:00:00.000' and '2009-06-11 19:59:00.000'
  and entity_path = 'Default Client / Default / ROS54770TST_Diag80_JDK_15 / '
  group by entity_path, entity_display_name , metric_name,units, name
  order by entity_path, entity_display_name
```

平均のクエリ

測定値の平均を計算するには，測定値の合計をカウントで割ります。カウントは rec_count フィールドに格納されます。

注 : SOAP 障害測定値のカウントは合計数に設定されているため，この測定値の平均値は計算できません。この測定値に実行できるのは，合計の計算のみです。Diagnostics バージョン 7.5 では，しきい値違反測定値にもこの注意事項が適用されます。Diagnostics 8.0 以降ではしきい値違反にカウントを使用できるため，この測定値の平均を計算できます。

例：

このクエリは，次のエンティティ・パスを含むサーバ要求に対して午前 6 時から午後 8 時までの平均レイテンシを計算します。
Default Client / Default / ROS54770TST_Diag80_JDK_15

```
select entity_display_name as Server_Request, sum(total)/sum(rec_count) as
Avg_Latency,sum(total) as Tot_Latency, sum(rec_count) as count, metric_name,
units, name, entity_path
from DIAG.DBO.REG_FRAGMENT_TYPE_METRICS_VIEW
where metric_name = 'latency'
  and ts between '2009-06-11 18:00:00.000' and '2009-06-11 19:59:00.000'
  and entity_path = 'Default Client / Default / ROS54770TST_Diag80_JDK_15 / '
  group by entity_path, entity_display_name , metric_name,units, name
  order by entity_path, entity_display_name
```

例：

このクエリは，次のエンティティ・パスを含むサーバ要求に対して午前 6 時から午後 8 時までのしきい値違反の平均数を計算します。
Default Client / Default / ROS54770TST_Diag80_JDK_15

```
select entity_display_name as Server_Request, sum(total)/sum(rec_count) as
Avg_Latency,sum(total) as Tot_Latency, sum(rec_count) as count, metric_name,
units, name, entity_path
from DIAG.DBO.REG_FRAGMENT_TYPE_METRICS_VIEW
where metric_name = 'threshold_violations'
  and ts between '2009-06-11 18:00:00.000' and '2009-06-11 19:59:00.000'
  and entity_path = 'Default Client / Default / ROS54770TST_Diag80_JDK_15 / '
  group by entity_path, entity_display_name , metric_name,units, name
  order by entity_path, entity_display_name
```

索引

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

へ

ほ

ま

め

も

ゆ

よ

ら

り

れ

ろ