HP Diagnostics

Windows® および UNIX® オペレーティング・システム用

ソフトウェア・バージョン 8.00

インストールおよび設定ガイド

ドキュメント番号：T6227-99004
ドキュメント発行日：2009 年 1 月（英語版）
ソフトウェア・リリース日：2009 年 1 月（英語版）

利用規約

保証

HP の製品およびサービスの保証は，かかる製品およびサービスに付属する明示的な保証の声明において定められている保証に限ります。本ドキュメントの内容は，追加の保証を構成するものではありません。HP は，本ドキュメントに技術的な間違いまたは編集上の間違い，あるいは欠落があった場合でも責任を負わないものとします。

本ドキュメントに含まれる情報は，事前の予告なく変更されることがあります。

制限事項

本コンピュータ・ソフトウェアは，機密性があります。これらを所有，使用，または複製するには，HP からの有効なライセンスが必要です。FAR 12.211 および 12.212 に従って，商用コンピュータ ソフトウェア，コンピュータ ソフトウェアのドキュメント，および商用アイテムの技術データは，HP の標準商用ライセンス条件に基づいて米国政府にライセンスされています。

サードパーティ Web サイト

HP は，補足情報の検索に役立つ外部サードパーティ Web サイトへのリンクを提供します。サイトの内容と利用の可否は予告なしに変更される場合があります。HP は，サイトの内容または利用の可否について，いかなる表明も保証も行いません。

著作権

商標

Java™ は，Sun Microsystems, Inc. の米国商標です。

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

Oracle® は，米国カリフォルニア州 Redwood City に所在する Oracle Corporation の米国登録商標です。

UNIX® は，The Open Group の登録商標です。

文書の更新

本書のタイトル・ページには，次の識別情報が含まれています。

- ソフトウェアのバージョンを示すソフトウェア・バージョン番号
- ドキュメントが更新されるたびに更新されるドキュメント発行日
- 本バージョンのソフトウェアをリリースした日付を示す，ソフトウェア・リリース日付

最新のアップデートまたはドキュメントの最新版を使用していることを確認するには，**http://h20230.www2.hp.com/selfsolve/manuals** を参照します。

このサイトでは，HP Passport に登録してサインインする必要があります。HP Passport ID の登録は，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

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てください。

適切な製品サポート・サービスに登録すると，更新情報や最新情報も入手できます。詳細については HP の営業担当にお問い合わせください。

サポート

HP ソフトウェアのサポート Web サイト **http://support.openview.hp.com** をご利用いただけます。

HP ソフトウェアのオンライン・サポートでは，インタラクティブなテクニカル・サポート・ツールを効率的にご利用いただけます。有償サポートをご利用のお客様は，サポート・サイトの以下の機能をご利用いただけます。

- 関心のある情報やドキュメントの検索
- サポート相談や改善依頼の送信および追跡
- 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 サポート連絡先の検索
- 使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申し込み

ほとんどのサポート・エリアでは，HP Passport ユーザとして登録し，ログインする必要があります。また，多くの場合，サポート契約も必要です。アクセス・レベルに関する詳細は，以下の Web サイトをご覧ください。
http://h20230.www2.hp.com/new_access_levels.jsp

HP Passport ID の登録は，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

目次

ようこそ

『**HP Diagnostics インストールおよび設定ガイド**』にようこそ。本ガイドでは，HP Diagnostics コンポーネントのインストール方法と設定方法について説明します。また，HP Diagnostics と統合するために他の HP ソフトウェア製品をセットアップする方法についても説明します。

本ガイドの構成

本ガイドは，次の各章で構成されます。

第 I 部　はじめに

Diagnostics コンポーネントをインストールおよび設定するための計画と準備に関する情報や手順を紹介します。

第 II 部　Diagnostics Server および Collector のインストール

HP Diagnostics Server のインストール方法と設定方法を説明します。

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

Diagnostics Probe のインストール方法と設定方法を説明します。

第 IV 部　他の HP ソフトウェア製品との統合のセットアップ

LoadRunner，Performance Center，および Business Availability Center を HP Diagnostics と統合するのに必要なセットアップ手順を説明します。

第 V 部　Java および .NET アプリケーションの監視のカスタム・インストゥルメンテーション

HP Diagnostics が監視対象のアプリケーションのクラスとメソッドに適用してパフォーマンス測定値の収集を可能にする測定の制御方法を説明します。

第 VI 部　Diagnostics Server と Java および .NET Agent の詳細設定

Diagnostics Server，Diagnostics .NET および Java Probe の高度な設定方法を説明します。

第 VII 部　プロキシおよびファイアウォールの通信設定

さまざまな通信チャネルを使って，Diagnostics プラットフォームをセットアップする方法を説明します。

第 VIII 部 Diagnostics 測定値コレクタの設定

測定値のキャプチャと，測定値コレクタの設定方法を説明します。

第 IX 部　付録

特定の項について詳しく説明します。

HP Diagnostics オンライン・ドキュメント

HP Diagnostics アプリケーションには，次のドキュメントが付属します。

- 『**Diagnostics User's Guide**』（英語版）: 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を説明します。このガイドにオンラインでアクセスするには，Diagnostics の［**ヘルプ**］ボタンか，または統合されている HP ソフトウェア製品の［ヘルプ］メニューを使用します。Windows 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，または Diagnostics Server インストール・ディレクトリ）からこのガイドの PDF 版にアクセスします。
- 『**Diagnostics インストールおよび設定ガイド**』: HP Diagnostics コンポーネントのインストール方法と設定方法を説明します。このガイドにオンラインでアクセスするには，Diagnostics の［**ヘルプ**］ボタンか，または統合されている HP ソフトウェア製品の［ヘルプ］メニューを使用します。Windows スタート・メニュー（［**スタート**］＞［**プログラム**］＞［**HP Diagnostics Server**］＞［**Installation Guide**］），HP Diagnostics インストール・ディスクの **Documentation** ディレクトリ，または Diagnostics Server インストール・ディレクトリ）からこのガイドの PDF 版にアクセスします。
- 『**最初にお読みください**』: HP Diagnostics に関する最新の技術情報とトラブルシューティング情報を提供します。このファイルは，HP Diagnostics インストール・ディスクのルート・ディレクトリにあります。
- 『**HP Diagnostics Probe for Java Installation Quick Start**』（英語版）: Diagnostics Probe for Java をインストールするための基本手順を説明します。HP Diagnostics インストール・ディスクの **Documentation** ディレクトリか，または Windows スタート・メニュー（［**スタート**］＞［**プログラム**］＞［**HP Java Agent**］＞［**QuickStart**］）からアクセスします。
- 『**Diagnostics Profiler for Java Installation and User's Guide**』（英語版）: Diagnostics Profiler for Java のインストール方法，設定方法，および使用方法について説明します。このガイドは，Diagnostics Profiler for Java の［**ヘルプ**］をクリックしてオンラインでアクセスします。
- 『**Diagnostics Profiler for .NET Installation and User's Guide**』（英語版）: Diagnostics Profiler for .NET のインストール方法，設定方法，および使用方法について説明します。このガイドは，Diagnostics Profiler for .NET の［ヘルプ］をクリックしてオンラインでアクセスします。

注：Diagnostics Profiler ガイド内の情報は，『**Diagnostics インストールおよび設定ガイド**』および『**Diagnostics User's Guide**』（英語版）にも記載されています。

その他のオンライン・リソース

［**トラブルシューティング & ナレッジ ベース**］: HP ソフトウェア・サポート Web サイトのトラブルシューティングのページにアクセスします。このページで，セルフ・ソルブ技術情報を検索できます。［**ヘルプ**］＞［**トラブルシューティング & ナレッジ ベース**］を選択します。この Web サイトの URL は http://h20230.www2.hp.com/troubleshooting.jsp です。

［**HP ソフトウェア Web サイト**］: HP のホームページにアクセスします。このサイトでは，HP ソフトウェア製品に関する最新情報をご覧になれます。たとえば，新しいソフトウェアのリリース，セミナー，展示会，カスタマー・サポートなどの情報が含まれます。［**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 http://welcome.hp.com/country/jp/ja/prodserv/software.html です。

［**HP ソフトウェア・サポート**］: HP ソフトウェア・サポートの Web サイトにアクセスします。このサイトで，セルフ・ソルブ技術情報を参照できます。また，ユーザ・ディスカッション・フォーラムへの書き込みや検索，サポート要求の送信，パッチや更新されたドキュメントのダウンロードなどを行うこともできます。［**ヘルプ**］＞［**HP ソフトウェア・サポート**］を選択します。Web サイトの URL は http://support.openview.hp.com です。

サポート・エリアの大部分において，HP パスポート・ユーザとしての登録と，サインインが必要になります。また，多くの場合サポート契約も必要になります。

アクセス・レベルの詳細については，下記の URL にアクセスしてください：
http://h20230.www2.hp.com/new_access_levels.jsp

HP パスポート・ユーザ ID を登録するには，下記の URL にアクセスしてください：
http://h20229.www2.hp.com/passport-registration.html

第 I 部

はじめに

第 1 章

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以下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

HP Diagnostics は，次のコンポーネントで構成されます。

➤ **Diagnostics Probe**：アプリケーションからイベントをキャプチャし，測定値を集計して，パフォーマンス測定値を Diagnostics Server に送信します。この Probe は，メソッドの起動，ビジネス・トランザクションの開始と終了，およびサーバ・トランザクションなどのイベントをキャプチャします。

注：Java Probe の機能は，HP Diagnostics/ TransactionVision Java Agent（Java Agent）によって提供されます。.NET Probe の機能は，HP Diagnostics/ TransactionVision .NET Agent（.NET Agent）によって提供されます。これらの Agent は，Diagnostics Probe と各 TransactionVision Sensor の機能を 1 つのコンポーネントに組み合わせたものです。

これらの Agent は，Diagnostics 環境ではプローブとして，TransactionVision 環境と複合環境では Sensor として機能するように設定され，プローブと Sensor の両方として同時に機能します。

➤ **Diagnostics Collector**：Oracle 10g Database，SQL Server，IBM WebSphere MQ，SAP NetWeaver - ABAP システムなど，外部の ERP/CRM 環境からデータを収集するには，Diagnostics Collector をインストールして監視対象とするこれらのシステムのインスタンスを定義します。Collector の各インスタンスは，Diagnostics の UI に Probe として表示されます。

➤ **Diagnostics Server**：Probe やほかの HP ソフトウェア製品と連携して機能して，アプリケーションのパフォーマンス測定値をキャプチャ，処理および表示します。

Diagnostics Server は，各 Probe から受信したデータを処理および集計し，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 Server（Commander モード）は，命令を行い，さまざまな Diagnostics コンポーネントと，Diagnostics が作動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。

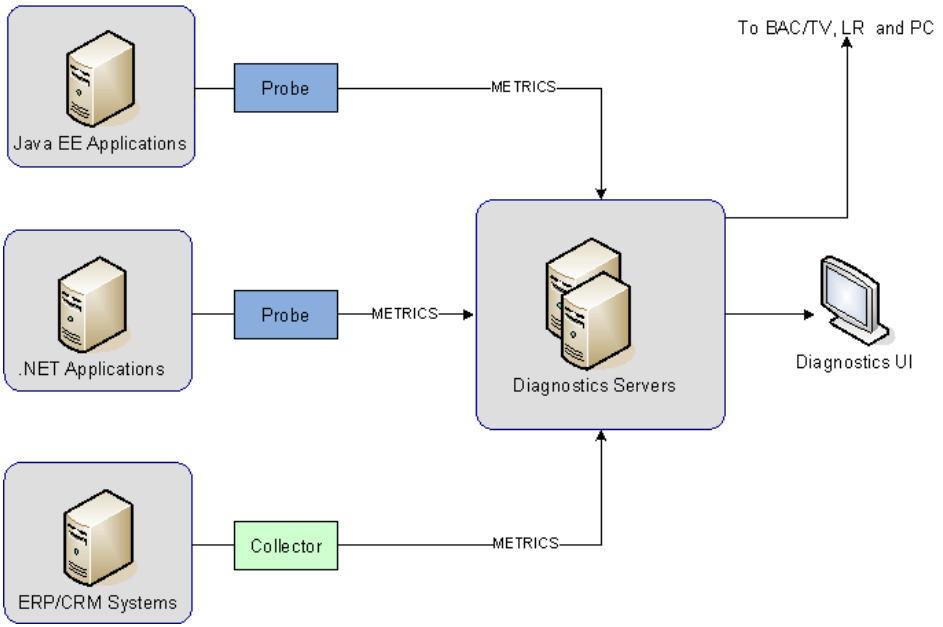
Diagnostics Server（Commander モード）は，ほかの Diagnostics コンポーネントの場所と状態を追跡し，ほかのコンポーネント間の通信ハブの役割を果たします。

また，Diagnostics Server（Commander モード）は，監視しているアプリケーションのパフォーマンス情報を Diagnostics ビューの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 ビューにアクセスできます。

Diagnostics のデプロイメントは，1 台または多数の Diagnostics Server で構成されることがあります。デプロイメントに Diagnostics Server 1 台しかない場合は，Commander モードに設定され，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 Server がある場合，1 台は Commander モードに設定され，その他はすべて Mediator モードに設定されます。

Diagnostics のデータ・フロー

次の図は，Diagnostics のコンポーネント間のデータ・フローを示します。



前記の図には，1 台以上の Mediator モードの Diagnostics Server に接続された Diagnostics Server（Commander モード）が 1 台あります。各 Diagnostics Server（Mediator モード）は，さまざまな Probe や Collector に接続されています。Diagnostics Probe と Diagnostics Collector は，監視対象のアプリケーションからイベントをキャプチャします。

キャプチャしたパフォーマンス測定値は，Probe と Collector によって Diagnostics Server（Mediator モード）に送信され，イベントがフィルタおよび集計されます。この情報は，Diagnostics Server（Commander モード）に送信され，カスタマイズ可能なビューに処理された測定値が表示されます。

アプリケーションをリアルタイムで監視している場合，Diagnostics の UI には Business Availability Center から，または Diagnostics Server から直接アクセスします。負荷テストの際，LoadRunner または Performance Center から Diagnostics の UI にアクセスします。

Java Diagnostics Profiler と .NET Diagnostics Profiler には，表示する Profiler のある Probe から直接，または Diagnostics の UI を通じてアクセスすることもできます。

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 10g Database**
- **SQL Server データベース**
- **IBM WebSphere MQ**

サポート対象環境の最新情報については，Diagnostics の製品可用性マトリックス（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

注：HP Diagnostics では，アプリケーション・サーバによって呼び出され，Probe によって測定されるメソッドだけを監視します。インストゥルメントおよび追加メソッドのインストゥルメント方法については，第 9 章「Java アプリーションのカスタム・インストゥルメンテーション」および第 10 章「.NET アプリーションのカスタム・インストゥルメンテーション」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，このセクション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。

Diagnostics Server の要件

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

Java 1.6 JVM は，次のオペレーティング・システムでサポートされ，以下に示すパッチを必要とします。

サポート対象環境の最新情報については，Diagnostics の製品可用性マトリックス（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

サポートされる Windows のバージョン：

- 32 ビット：Windows XP，Windows 2003，Windows 2003 R2，Windows 2008
- 64 ビット：Windows 2003，Windows 2003 R2，Windows 2008

サポートされる HP-UX のバージョン：

- HP PA-RISC：HP-UX 11i v1（11.11），HP-UX 11i v2（11.23），HP-UX 11i v3（11.31）
- HP Itanium：HP-UX 11i v2（11.23），HP-UX v3（11.31）

必要な HP-UX のパッチ：

- HP-UX システムで Java を実行するには（Diagnostics Server 用または Java Probe 用のいずれか），Java で必要なすべてのパッチがインストールされている必要があります。Java で必要なパッチは，Java のバージョンと HP-UX のバージョンによって異なります。HP-UX 11.11 の場合は PHCO_29903 パッチが必要です。このバージョン，およびほかのバージョンの HP-UX オペレーティング・システムで必要になる可能性のある追加のパッチに関する詳細については，次の Web サイトを参照してください。
  http://h18012.www.1.hp.com/java/patches/index.html

- Java 5.0 Quality Pack
- リンカー・パッチが必要です。パッチ ID は，HP-UX 11i v1（11.11）システムでは PHSS_35385，HP-UX 11i v2（11.23）システムでは PHSS_37201，HP-UX 11i v3（11.31）システムでは PHSS_37202 です。
- HP PA-RISC HP-UX 11i v2（11.23）の Diagnostics Server では，以下のパッチ（およびパッチの依存関係）が必要です。PHKL_37121，PHSS_37947

サポートされる Linux のバージョン：

- Red Hat 2.1，Red Hat Enterprise Linux 3.0，4.0，5.0

サポートされる Solaris のバージョン：

- Solaris 10，9，8

必要な Solaris のパッチ：

- http://sunsolve.sun.com/show.do?target=patches/JavaSE を参照してください。

Diagnostics クライアント・ホストの要件

Diagnostics 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 を表示するマシンに JRE 1.5 以上をインストールする必要があります。

Diagnostics Server ホストの要件

Diagnostics Server のホストのシステム設定要件は，Probe と，Probe に報告している Diagnostics Server（Mediator モード）の数に応じて異なります。Diagnostics Server が Diagnostics Server（Commander モード）に指定されている場合，システム設定要件を決定する際，報告する各 Diagnostics Server（Mediator モード）は，Probe の半分として数えます。

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 Probe で実行されたテストに基づいた目安です。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），トレンド付けメソッドの数，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，Diagnostics Server と Java Probe 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 Probe 50 台以内	Java Probe 100 台以内	Java Probe 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	3x Ultra Sparc 2	4x Ultra Sparc 2	4x Ultra Sparc 2
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
HP-UX	CPU	PA-RISC 2x 650 MHz	PA-RISC 2x 699 MHz	PA-RISC 2x 750 MHz
HP-UX	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	Probe 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
- Probe あたりの一意のサーバ要求数：23

次の表は，Diagnostics Server と .NET Probe 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 Probe 10 台以内	.NET Probe 20 台以内	.NET Probe 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	3x Ultra Sparc 2
Solaris	メモリ	1 GB	1.5 GB	3 GB

プラットフォーム	項目	.NET Probe 10 台以内	.NET Probe 20 台以内	.NET Probe 50 台以内
Linux	CPU	1x 1.0 GHz	1x 2.0 GHz	2x 2.4 GHz
Linux	メモリ	768 MB	1 GB	3 GB
HP-UX	CPU	1x 1.0 GHz	1x 2.0 GHz	2x 2.4 GHz
HP-UX	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	Probe あたり 3 GB		

次のページに示すスケーラビリティの数値は，以下のリファレンス・ハードウェア構成に基づいて算出されています。

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： 8GB

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

Java ヒープ： 5.9GB（-Xmx6096m），64 ビット JVM

ディスク容量： プローブあたり 2 ～ 4 GB（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： メモリの拡張性が高い 64 ビット OS と 64 ビット JVM を強くお勧めします（Windows の 32 ビット JVM には 1.2 GB の制限があります）。また，64 ビット JVM では参照が倍増するため，32 ビット JVM より多くのメモリが必要です。

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 Probe 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（デフォルト）

Java Probe 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（デフォルト）

Java Probe 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 Probe 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 Probe 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（デフォルト）

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。

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

注：

- Probe が多数存在する環境では，1 つのホスト上に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 Probe 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 Server（Commander モード）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，659 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 Server を最適化して処理する Probe の数を増やす方法については，375 ページ「実稼動環境の Diagnostics Server の処理 Probe 数増加の最適化」を参照してください。

Diagnostics Probe for Java ホストの要件

監視しているシステムで Diagnostics Probe for Java（Java Probe）によって生じるオーバーヘッドは，極めて低いものです。以下は，Probe 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：	200 MB の追加容量

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。

Java Probe の最大ヒープの設定については，184 ページ「アプリケーションの起動スクリプトを使用した Java Probe のヒープ・サイズの調整」を参照してください。

Java Diagnostics Profiler ユーザ・インタフェース・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 を表示するマシンに JRE 1.5 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，Probe は 35000 で始まる最初の空きポートに割り当てられます。

Diagnostics Probe for .NET 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。以下は，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	1.1 以降

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

.NET Diagnostics Profiler のユーザ・インタフェースは，IE6 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://<probehost>:<probeport>/profiler）にアクセス可能である必要があります。Probe は，Probe 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。デフォルトのポート範囲は 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 または SQL Server アプリケーションのホスト・マシンとやり取り可能なマシンにインストールできます。

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を開始する前に，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提供する必要のある情報を収集するために役立ててください。

注：Diagnostics に Business Availability Center 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 Server

必要な情報	調べる場所	値
Diagnostics Server（Commander モード）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 Server 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 Server（Mediator モード）の場合，Diagnostics Server（Commander モード）の URL	Diagnostics Server（Commander モード）をインストールした後，システムの状況モニタから確認できます（詳細については，付録 D「System Health Monitor の使用」を参照してください）。	
Diagnostics Server を SaaS 環境または Business Availability Center 環境のどちらで使用するか		

Java Probe

➤ **HP ソフトウェア製品および Diagnostics Server 情報**

必要な情報	調べる場所	値
Probe を使用する HP ソフトウェア製品の名前	製品ライセンスに従って選択します。 ➤ **Performance Center / LoadRunner** ➤ **Business Availability Center**	
Diagnostics Server（Commander モード）の URL	システムの状況モニタ（付録 D「System Health Monitor の使用」参照）	
Diagnostics Server のイベント・ホスト	システムの状況モニタ（付録 D「System Health Monitor の使用」参照）	
Diagnostics Server のイベント・ポート	システムの状況モニタ（付録 D「System Health Monitor の使用」参照）	**デフォルト値**：2006

➤ Probe およびアプリケーション・サーバ情報

必要な情報	調べる場所	値
Probe 名	**一意の**文字列。 ユーザが作成したもの。 **注**：Probe の名前は，Probe タイプを示す必要があり，さまざまな Probe のタイプを区別するのに役立ちます。	例：JavaProbe1
Probe グループ	測定値が論理グループとして報告されるように Probe を関連付けるのに使います。 これは，Probe のインストール時にユーザが定義したものです。	**デフォルト値：** DefaultProbeGroup
Java Probe が監視す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。	
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，使用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。 設定してい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。 137 ページ「JRE Instrumenter の実行」を参照してください。	

.NET Probe

➤ **Diagnostics Server 情報**

必要な情報	調べる場所	値
Diagnostics Server（Commander モード）の URL	システムの状況モニタ。 Probe が Diagnostics Server への登録に使用する URL。 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。	
Diagnostics Server のイベント・ホスト	システムの状況モニタ。 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。	
Diagnostics Server のイベント・ポート	システムの状況モニタ。 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。	**デフォルト値**：2612

➤ Probe およびポート情報

必要な情報	調べる場所	値
Probe グループ	測定値がグループとして報告されるように Probe を関連付けるのに使います。 これは，Probe のインストール時にユーザが定義したものです。	**デフォルト値：** Default
最小 Web ポート	システム管理者。 Probe に割り当てることができる Probe ホストのポート範囲の最小ポート番号。	**デフォルト値**：35000
最大 Web ポート	システム管理者。 Probe に割り当てることができる Probe ホストのポート範囲の最大ポート番号。	**デフォルト値**：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，［**スタート**］＞［**設定**］＞［**コントロール パネル**］＞［**管理ツール**］＞［**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。

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なければなりません。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。

Diagnostics Server

- Diagnostics Server（Commander モード）に有効なライセンスが与えられるまで，HP Diagnostics のパフォーマンス測定値は表示できません。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 以上になっていることを確認してください。

Diagnostics Probe for Java

- テスト時に，Java Probe 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 Server（Commander モード）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・モードで作動します。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 Probe for .NET

- テスト時に，.NET Probe 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 Server（Commander モード）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・モードで作動します。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。

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，慎重に計画および準備すると，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。

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。

注：ここに記載されているインストール順序は，製品とコンポーネントをインストールする際の推奨される順序です。このインストール順序を守らなかった場合，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。

---

1. **システム要件とインストール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，25 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 Server をインストールします。**

   詳細については，第 2 章「Diagnostics Server のインストール」を参照してください。

3 Diagnostics Probe と Collector の両方，またはいずれかをインストールおよび設定します。

- Java EE 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（Probe）のインストール」を参照してください。
- Oracle，SAP NetWeaver-ABAP，および SQL Server 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 Probe の場合，アプリケーション・サーバが Probe と連動するように設定します。

詳細については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」を参照してください。

5 HP Diagnostics が LoadRunner, Performance Center または Business Availability Center と統合されている場合，HP Diagnostics を使うようにセットアップする必要があります。

- Business Availability Center については，第 23 章「Diagnostics を使用するための Business Availability Center 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 24 章「LoadRunner Diagnostics Add-in のインストール」を参照），HP Diagnostics を使うように LoadRunner をセットアップします（第 25 章「HP Diagnostics を使用するための LoadRunner のセットアップ」参照）。
- Performance Center については，第 26 章「HP 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 Server（Commander モード）の有効なライセンスを取得する必要があります。ライセンスは，ノード固定型のライセンスで，Diagnostics Server（Commander モード）で表示される Diagnostics ビューのロックを解除し，Diagnostics Probe からアクセスされる Profiler ビューの読み込み制限を解除します。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。

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「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」の手順に従います。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 Server および Collector のインストール

第 2 章

Diagnostics Server のインストール

Windows および UNIX マシンに Diagnostics Server 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

Windows および UNIX への Diagnostics Server のインストール

以下のセクションでは，Diagnostics Server をインストールするための詳細な手順について説明します。これらの手順は，次の環境に適用されます。

- Windows 環境
- グラフィック・インストールまたはコンソール・モード・インストールを使った，ほとんどの UNIX 環境

  UNIX の場合は，Solaris マシンにコンポーネントをインストールするための手順です。ほかの認定 UNIX プラットフォームでも同じ手順でインストールできます。

注：マシンに Diagnostics Server の以前のバージョンをインストールしている場合は，付録 G「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」を参照してください。

重要：以下の手順は，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 の画面とコマンドを使った作業については，付録 I「UNIX コマンドの使用」を参照してください。

本項の内容

- Diagnostics Server インストーラの起動
- インストールの実行

Diagnostics Server インストーラの起動

環境によって，インストーラの起動方法は異なります。

Windows インストーラを起動するには

1 **HP Diagnostics インストール・ディスクのルート・ディレクトリにある setup.exe ファイルを実行します。**Diagnostics セットアップ・プログラムが起動し，インストール・メニュー・ページが表示されます。

インストーラに，HP Diagnostics メイン・インストール・メニューが表示されます。

2 **Diagnostics Server をクリックしてインストーラを起動します。**これによって，32 ビット Windows バージョンの Diagnostics Server がインストールされます。64 ビット・バージョンの Diagnostics Server をインストールする場合は，**DVD を参照して Diagnostics_Servers** ディレクトリを見つけ，**DiagnosticsServerSetupWinx64_ ＜バージョン＞ .exe** ファイルをダブルクリックする必要があります。

注：別の場所からインストーラを起動するには，実行ファイルである **DiagnosticsServerSetupWin_ ＜バージョン＞ .exe**（32 ビット用）または **DiagnosticsServerSetupWinx64_ ＜バージョン＞ .exe**（64 ビット JVM で実行される 64 ビット用）を **＜ HP Diagnostics インストール・ディスク＞ ¥ Diagnostics_Servers** ディレクトリから新しい場所にコピーし，それをダブルクリックします。

---

46 ページ「インストールの実行」に進みます。

UNIX インストーラを起動するには

1 **＜ HP Diagnostics インストール・ディスク＞ /Diagnostics_Servers** ディレクトリから，Diagnostics Server がインストールされるマシンにインストーラ **DiagnosticsServerSetup ＜プラットフォーム＞ _ ＜バージョン＞ .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，UNIX コマンド・プロンプトでインストーラのファイル名 **DiagnosticsServerSetup ＜プラットフォーム＞ _ ＜バージョン＞ .bin** を入力します。以下は例です。

```
./DiagnosticsServerSetupSolaris_8_00.bin
```

➤ コンソール・モードでインストーラを実行するには，UNIX コマンド・プロンプトでインストーラのファイル名 **DiagnosticsServerSetup ＜プラットフォーム＞ _ ＜バージョン＞ .bin** を入力し，**-console** オプションを指定します。以下は例です。

```
./DiagnosticsServerSetupSolaris_8_00.bin -console
```

46 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

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，インストールを実行できます。

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。

［**次へ**］を選択してインストールを続行します。

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 Server 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，［**ブラウズ**］をクリックして別のディレクトリを指定できます。

［**次へ**］を選択してインストールを続行します。

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 Server の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 台以上の Diagnostics Server で構成されます。デプロイメントの Diagnostics Server が 1 台だけの場合は，Commander モードに設定され，Commander と Mediator 両方の役割を実行できます。デプロイメントに複数の Diagnostics Server がある場合，1 台は Commander モードに，その他はすべて Mediator モードに設定されます。

- デプロイメントに Diagnostics Server 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 Server 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。

［**次へ**］を選択してインストールを続行します。

注：ここで，Commander モードと Mediator モードのどちらで Diagnostics Server 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 Server（Commander モード）をインストールするには，47 ページ「Diagnostics Server（Commander モード）のインストール」に進みます。
- Diagnostics Server（Mediator モード）をインストールするには，50 ページ「Diagnostics Server（Mediator モード）のインストール」に進みます。

Diagnostics Server（Commander モード）のインストール

Diagnostics Server（Commander モード）をインストールする場合は，次の手順に進みます。

1 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 Server 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。
- **登録済み Business Availability Center コア・サーバと同期**：Diagnostics Server を Business Availability Center 環境で作動させる場合，このオプションを選択して，Business Availability Center コア・サーバと同期させます。
- **システム時刻と同期**：Diagnostics Server を Business Availability Center 以外の環境で作動させ，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。

［**次へ**］を選択してインストールを続行します。

2 Diagnostics Server を HP Software-as-a-Service (SaaS) 環境で使うのか，Business Availability Center で使うのかを指定します。

この Diagnostics Server に適用するオプションを選択し，［**次へ**］を選択してインストールを続行します。

注：［**HP Software-as-a-Service (SaaS)**］オプションだけを選択した場合は，手順 4 にスキップしてください。

---

3 Business Availability Center 環境のみ：Agent および Collector インストーラへのパスを指定します。

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

Business Availability Center 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**］を選択してインストールを続行します。

インストーラが Diagnostics Server インストール・ディレクトリに自動的にコピーされ，Business Availability Center 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ように指定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 Availability Center がアクセスできる Diagnostics Server インストール・ディレクトリ（**＜ Diagnostics Server 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。

---

4 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。情報が正しいことを確認します。

注：Diagnostics Server（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，Business Availability Center で使用できる場合に，Probe インストーラのサイズは含まれません。

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［**Previous**］を選択します）。

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

5 インストール・ウィザードを終了します。

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 Server は自動的に起動を試みます。ほかのアプリケーションがデフォルトの Diagnostics Server ポートを使用している場合，Diagnostics Server は起動しません。Diagnostics Server を手動で起動する方法については，52 ページ「Diagnostics Server の起動および停止」を参照してください。

---

6 Diagnostics ライセンス・ファイルをアップロードします。

パフォーマンス測定値を表示するには，Diagnostics Server（Commander モード）をインストールした後に，有効な HP Diagnostics ライセンス・ファイルをアップロードする必要があります。

有効なライセンス・ファイルの請求およびアップロードについては，第 3 章「HP Diagnostics ライセンスの有効化」を参照してください。

Diagnostics Server（Mediator モード）のインストール

Diagnostics Server（Mediator モード）をインストールする場合は，次の手順に進みます。

1 Diagnostics Server（Commander モード）の場所を指定します。

Diagnostics Server（Mediator モード）を Diagnostics Server（Commander モード）に接続できるようにするための情報を入力します。

a Diagnostics Server（Commander モード）のホスト名を入力します。

b Diagnostics Server（Commander モード）のポートを入力します。

Diagnostics Server（Commander モード）のデフォルトのポートは **2006** です。Diagnostics Server をインストールした後にポートを変更した場合は，デフォルト・ポートではなく，変更後のポート番号を指定する必要があります。Diagnostics Server のポートの変更については，363 ページ「デフォルトの Diagnostics Server ポートの変更」を参照してください。

c 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。

［**次へ**］を選択してインストールを続行します。

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。結果が芳しくない場合は，次のインストール手順に進む前に報告されます。

2 Diagnostics Server を HP Software-as-a-Service (SaaS) 環境で使うのか，Business Availability Center で使うのかを指定します。

この Diagnostics Server に適用するオプションを選択し，［**次へ**］を選択してインストールを続行します。

注 :［**Business Availability Center**］オプションだけを選択した場合は，手順 4 にスキップしてください。

---

3 Diagnostics Server を HP Software-as-a-Service (SaaS) 環境で使用するように指定した場合は，HP Software-as-a-Service (SaaS) 環境のカスタマ名を入力します。

HP Software-as-a-Service (SaaS) 環境のカスタマの一意の名前を入力してください。

［**次へ**］を選択してインストールを続行します。

4 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。情報が正しいことを確認します。

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［**Previous**］を選択します）。

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

5 インストール・ウィザードを終了します。

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 Server は自動的に起動を試みます。ほかのアプリケーションがデフォルトの Diagnostics Server ポートを使用している場合，Diagnostics Server は起動しません。Diagnostics Server を手動で起動する方法については，52 ページ「Diagnostics Server の起動および停止」を参照してください。

Diagnostics Server の起動および停止

Windows マシンの場合

Windows マシンで Diagnostics Server を起動するには

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

Windows マシンで Diagnostics Server を停止するには

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

UNIX 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 Server 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 Server を起動および停止します。

UNIX または Linux マシンで Diagnostics Server を起動するには

1 **M_LROOT** 環境変数が，Diagnostics Server 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 Server のインストール・ディレクトリ＞
/nanny/ ＜プラットフォーム＞
```

＜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_LROOTUnable 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である可能性があります。

UNIX または Linux マシンで Diagnostics Server を停止するには

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
```

UNIX または Linux マシンの場合（nanny を使用しない）

次の手順では，nanny を使わずに Diagnostics Server を起動および停止します。

UNIX または Linux マシンで Diagnostics Server を起動するには

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

UNIX または Linux マシンで Diagnostics Server を停止するには

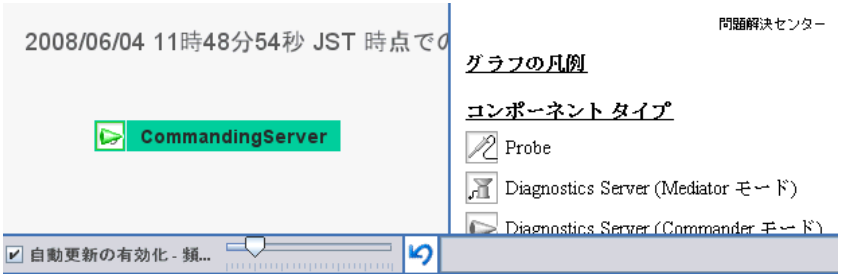
kill などのユーティリティを使ってプロセスを終了します。

Diagnostics Server のインストールの確認

Diagnostics Server が正常にインストールされていること，および正しく起動していることを確認するには，システムの状況モニタを使用します（システムの状況モニタの起動については，付録 D「System Health Monitor の使用」を参照してください）。

推奨インストール手順に従って，Diagnostics Server のインストール後，システムの状況モニタを使って Diagnostics Server がインストールおよび起動したかどうかを確認できます。

Diagnostics Server（Commander モード）は，次の例のようにシステムの状況モニタに表示されるはずです。



Diagnostics Server が Commander モードでデプロイされている場合，命令役と仲介役の両方の役割を持ちます。Diagnostics Server（Commander モード）は，「**Commanding Server**」というラベルの付いたアイコンで，System Health Monitor component map に表示されます。

Diagnostics Server が Mediator モードでデプロイされている場合，「**server- ＜ホスト名＞**」というラベルの付いた 1 つのアイコンで示されます。

システムの状況モニタは，Diagnostics Server コンポーネントの一部です。Diagnostics Server をインストールした後にシステムの状況モニタにアクセスできない場合，間違った URL を入力したか，または Diagnostics Server が起動していないことが原因です。Diagnostics Server の起動については，52 ページ「Diagnostics Server の起動および停止」を参照してください。

システムの状況モニタをブラウザに表示した状態で，コンポーネントのインストール状況を確認し，コンポーネントの残りのインストールで直面する問題を特定および解決できます。

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

Diagnostics Server をインストールした後は，有効なライセンス・ファイルを提供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 Server の設定

Diagnostics Server は，ほとんどの状況で有効に動作するデフォルトの設定でインストールされています。設定を変更することで Diagnostics Server のパフォーマンスが向上したり，通常でない状況で作動できるようになることがあります。

Diagnostics Server の設定については，第 14 章「Diagnostics Server の詳細設定」を参照してください。

Diagnostics Server のバージョン確認

サポートを依頼する際，Diagnostics Server のバージョンを把握しておくと便利です。

Diagnostics Server のバージョンを確認するには

Diagnostics ツールバーから［**ヘルプの表示**］ボタン をクリックして，メニューの［**HP Diagnostics のバージョン情報**］を選択します。

［HP Diagnostics のバージョン情報］ダイアログ・ボックスに，Diagnostics Server のバージョンが表示されます。

Diagnostics Server のアンインストール

次のセクションでは，Diagnostics Server をアンインストールする方法について説明します。

Windows マシンからの Diagnostics Server のアンインストール

➤［**スタート**］＞［**プログラム**］＞［**HP Diagnostics Server**］＞［**Uninstall HP Diagnostics Server**］を選択して，Diagnostics Server をアンインストールします。

➤ または，＜ **Diagnostics Server 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。

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。次のように応答してください。

➤ Diagnostics Server 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 Server を再インストールする予定で，アンインストールしている Diagnostics Server のカスタム・プロパティ設定を保持する場合は，**etc** にあるプロパティ・ファイルを新しい場所に戻す必要があります。

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 マシンから Diagnostics Server をアンインストールする

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

Diagnostics Server をアンインストールするには

1 Diagnostics Server を停止します。その方法については，52 ページ「Diagnostics Server 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 Server 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 Server のインストール・ディレクトリ＞
/Server/_uninst/uninstaller.bin
```

第 3 章

HP Diagnostics ライセンスの有効化

HP Diagnostics では，Diagnostics Server（Commander モード）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 Server（Commander モード）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。ライセンスでは，Diagnostics Server のホストの MAC アドレスに基づくノード固定型のライセンスを使用します。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。

プローブと Diagnostics Server（Mediator モード）が Diagnostics Server（Commander モード）に初めて接続すると，Diagnostics Server（Commander モード）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。

ライセンスの種類

インストール時には，30 日間の試用ライセンスが付与されます。30 日の試用期間（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測定値の処理を開始できます。この 30 日の期間内に，永久ライセンスまたはカスタム・ライセンスを取得する必要があります。

- **永久**ライセンスには，有効期限がありません。
- **カスタム**・ライセンスには，有効期限があります。有効期限が設定されているライセンスは，一定期間だけ有効になります。このライセンスは，永久ライセンスを購入する前の試用期間として利用できます。

30 日の試用期間が終了するまでに永久ライセンスまたはカスタム・ライセンスを取得しないと，Diagnostics Server は新しいデータを受け付けなくなります。

注：完全な Enterprise 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。

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

HP ソフトウェア・サポートから Diagnostics ライセンスを取得し，それを Diagnostics Server（Commander モード）にアップロードします。

Diagnostics デプロイメントにライセンスを与えるには

1 Diagnostics UI のメイン・ウィンドウで［**Diagnostics の設定**］＞［**License**］を選択して，Diagnostics Server（Commander モード）の［ライセンス管理］ページにアクセスします。

［ライセンス管理］ページが開き，次の情報が表示されます。

- Diagnostics Server の MAC ライセンス（ライセンスを取得するために必要）
- 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
- 監視対象環境内の合計論理プロセッサ / コアに関する情報

2 ［ライセンス管理］ページの［システム情報］セクションの値を見て，Diagnostics Server（Commander モード）のホストの MAC アドレスを特定します。［**更新**］をクリックして，Diagnostics Server で現在の MAC アドレス情報を検出できるようにします。

MAC アドレス値が**不明**な場合は，ライセンス・ファイルを要求するときに，HP ソフトウェア・サポートに伝えます。ほかの方法で MAC アドレスを調べないでください。

注：ライセンスを要求するときに，［ライセンス管理］ページの［システム情報］セクションに表示される MAC アドレスを使用する必要があります。Diagnostics Server は，［ライセンス管理］ページに表示する MAC アドレスを特定できない場合，ライセンスに基づいて MAC アドレスを検証しようとしたときに，MAC アドレスを見つけることができません。

---

3 HP ソフトウェアのカスタマ・サポート担当者にライセンスを要求します。

注：Diagnostics Server（Commander モード）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。ファイルの名前は，末尾に拡張子 **.lic** が必ず付きます。

［ライセンス管理］ページには，［**現在接続されている Probe の合計論理プロセッサ / コア**］または［**現在ホストに接続されている論理プロセッサ / コア**］に関する情報が表示されます。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。これらの情報は，Diagnostics 8.00 のプローブでのみ有効です。

4 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［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。

ライセンス・ファイルの格納場所へのパスを入力するか，［**参照**］をクリックして，ライセンス・ファイルの場所を指定します。［**アップロード**］をクリックして，Diagnostics Server にライセンス・ファイルを適用します。

ファイルは，アップロード・プロセスによって名前が変更され，Diagnostics Server（Commander モード）のインストール・ディレクトリの適切な場所に保存されます。

注：ライセンス・ファイルを Diagnostics Server 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［ライセンスのアップロード］セクションを使用します。

---

5 ライセンス情報を参照します。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），およびそのライセンスで扱われるプローブの最大数が表示されます。

他の Diagnostics コンポーネントのライセンスの有効化

Diagnostics Server（Mediator モード）および Diagnostics Probe に，個別のライセンスはありません。それらのライセンスは，Diagnostics Server（Commander モード）のライセンスに基づきます。それらが初めてライセンスのある Diagnostics Server（Commander モード）に接続したときに，Diagnostics Probe と Diagnostics Server（Mediator モード）は自動的にライセンスが設定されます。

Java または .NET Probe 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。Profiler は，Profiler のビルトイン UI または HP Diagnostics の UI から直接アクセス可能な独立した診断コンポーネントです。

Diagnostics Profiler は，Probe が正しくライセンスが与えられた Diagnostics Server（Commander モード）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・モードで作動します。非ライセンス・モードで，Profiler には，5 つの同時実行スレッドからデータをキャプチャする上で制限があります。

第 4 章

Diagnostics Collector のインストール

Windows および UNIX マシンに Diagnostics Collector 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 Collector のインストールについて

Diagnostics Collector は，外部のアクティブ・システムからデータを収集します。Collector は，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設定できます。

- SAP NetWeaver - ABAP システム
- Oracle 10g Database
- IBM WebSphere MQ
- SQL サーバ

Collector のインストール時に，これらのアクティブ・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 10g，SAP NetWeaver - ABAP，CSQL Server システムおよび IBM WebSphere MQ 環境のインスタンスを定義します。

Windows マシンへの Diagnostics Collector のインストール

Windows マシンに 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。また，これらの手順は，グラフィック・インストーラを使って Collector を UNIX マシンにインストールする際にも使用します。

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 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，31 ページ「Diagnostics Collector ホストの要件」を参照してください。

インストーラの起動

インストーラは，Diagnostics インストール・ディスクまたは Business Availability Center の［Diagnostics Downloads］ページから直接起動できます。

注：ホスト・マシンにすでに Collector がインストールされている場合は，付録 G「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」を参照してください。

製品インストール・ディスクからインストーラを起動するには

1 インストール・ディスクのルート・ディレクトリにある **setup.exe** ファイルを実行します。Diagnostics セットアップ・プログラムが起動し，インストール・メニュー・ページが表示されます。

注：ほかの場所からインストーラを起動するには，実行ファイル **CollectorSetupWin_ ＜バージョン＞ .exe** を **＜ HP Diagnostics インストール・ディスク＞ ¥Diagnostics_Installers** ディレクトリから新しい場所にコピーして実行します。

---

2 インストール・メニュー・ページで，［**Diagnostics Collector**］を選択してインストーラを起動します。

3 68 ページ「インストールの実行」に進みます。

Business Availability Center の［Diagnostics downloads］ページからインストーラを起動するには

注：Collector インストーラは Business Availability Center からアクセスするために必要なディレクトリに置かれていれば，Business Availability Center で利用できます。

Diagnostics Server 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，**¥Diagnostics_Installers¥CollectorSetupWin_ ＜バージョン＞ .exe** ファイルをインストール・ディスクから Diagnostics Server インストール・ディレクトリの **＜ Diagnostics Server のインストール・ディレクトリ＞ /html/opal/downloads** に手動でコピーすることにより，Business Availability Center から Collector インストーラをダウンロードできます。第 2 章「Diagnostics Server のインストール」の手順 3（48 ページ）を参照してください。

---

1 **Business Availability Center** 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 Windows 用 Collector インストーラを起動します。

「インストールの実行」に進みます。

インストールの実行

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

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。

続行するには［**次へ**］をクリックします。

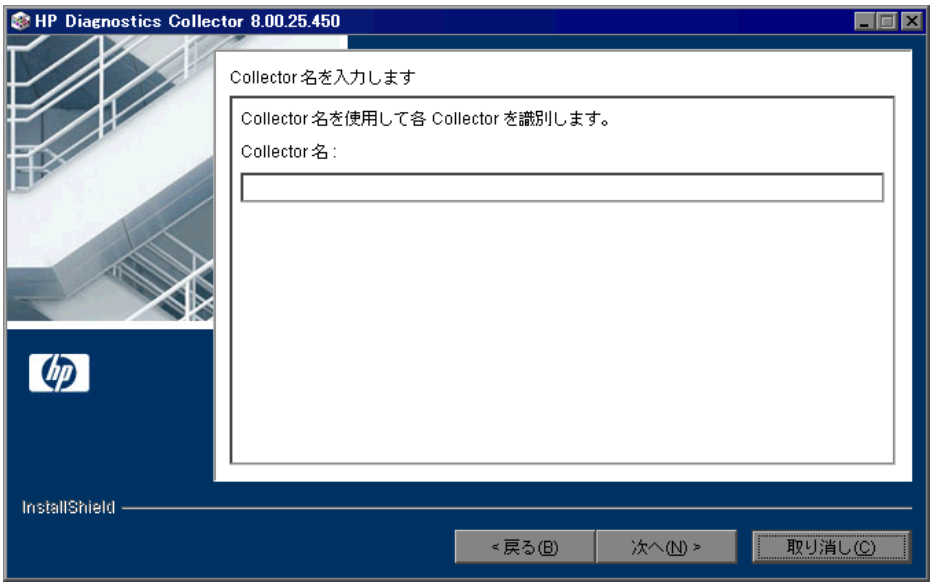
2 **Collector のインストール先を指定します。**

［**インストール先ディレクトリ名**］ボックスに，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。または，デフォルトのディレクトリである **C:¥MercuryDiagnostics¥Collector** を受け入れるか，［**ブラウズ**］をクリックしてほかのディレクトリを指定します。

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「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」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**］をクリックします。

3 Collector に一意の名前を割り当てます。



システムの状況モニタのこの特定の Collector を一意に識別する名前を Collector に割り当てます。システムの状況モニタを使って，Collector のインストールと設定を確認します。詳細については，94 ページ「Diagnostics Collector のインストールの確認」を参照してください。

名前に使用できる文字は，**-**，**_** およびすべての英数字です。

続行するには［**次へ**］をクリックします。

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。いずれかのオプション，または両方のオプションを選択できます。

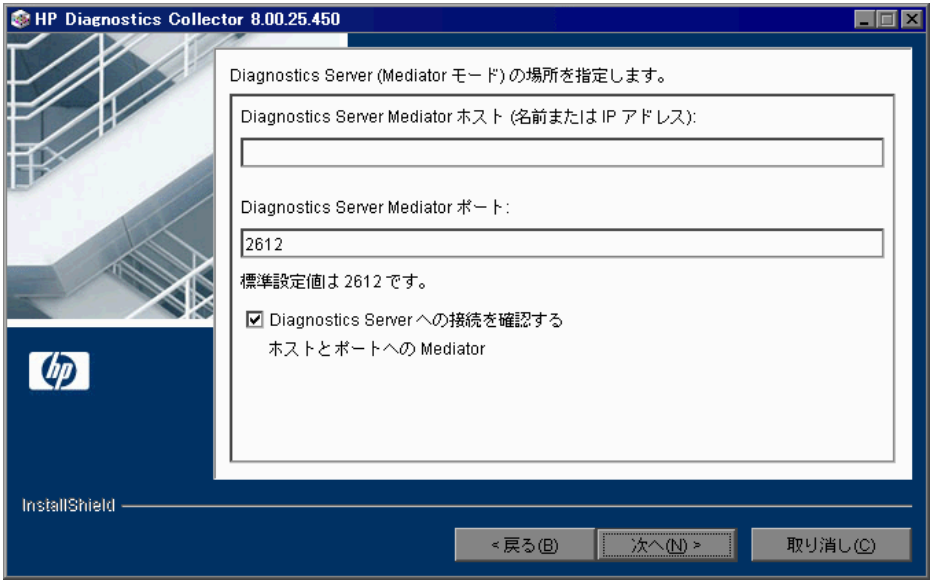
- SAP NetWeaver - ABAP 環境でデータを収集するには，［**SAP NetWeaver - ABAP**］を選択します。
- Oracle 10g Database Server でデータを収集するには，［**Oracle**］を選択します。
- MQ Series 環境でデータを収集するには，［**MQ**］を選択します。
- SQL Server データベースでデータを収集するには，［**SQL Server**］を選択します。

重要：インストール後，監視する SAP NetWeaver - ABAP，Oracle，MQ および SQL Server 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，82 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**］をクリックします。

5 Diagnostics Server（Mediator モード）の情報を入力します。

Diagnostics Server（Mediator モード）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 Server が 1 つしかない場合，その Diagnostics Server 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 Server が複数ある場合は，Collector からイベントを受信する Diagnostics Server（Mediator モード）の情報を入力します。

a ［**Diagnostics Server Mediator ホスト**］ボックスに，Diagnostics Server（Mediator モード）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 Server Mediator ポート**］ボックスに，Diagnostics Server が Collector 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。デフォルトのポート番号は **2612** です。Diagnostics Server をインストールした後にポートを変更した場合は，デフォルトのポートではなく，変更後のポート番号を指定します。

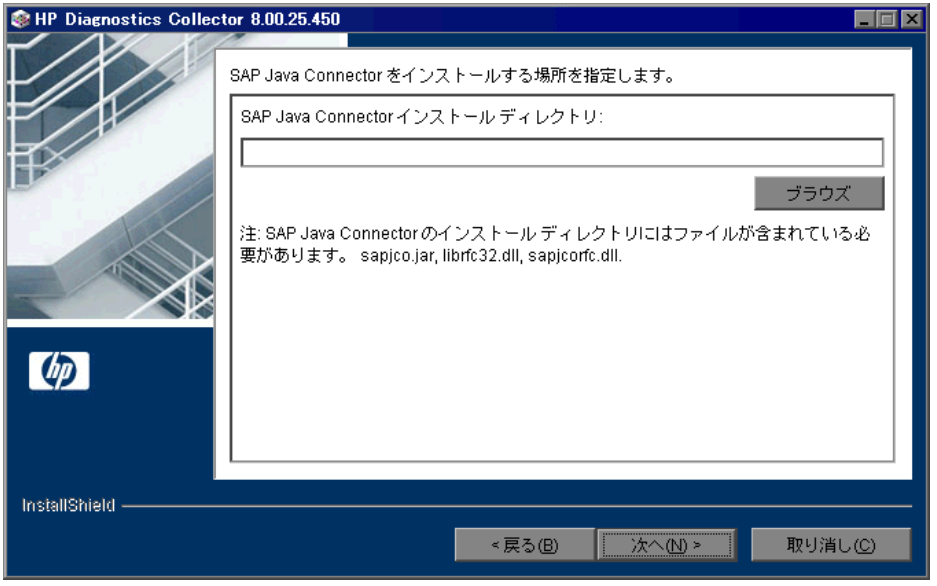
c Diagnostics Server が実行中でインストール・ホストからアクセス可能なことを確認するには，［**Diagnostics Server への接続を確認する Mediator のホストとポート**］を選択します。

続行するには［**次へ**］をクリックします。

注：［**Diagnostics Server への接続を確認する Mediator のホストとポート**］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性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 Server への接続を確認する Mediator のホストとポート**］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の名前を入力します。

このディレクトリには，次のファイルが含まれていなければなりません。

- sapjco.jar
- librfc.dll
- sapjcorfc.dll

注：

- SAP Java Connector のディレクトリ名がわからない場合は，SAP 窓口に問い合わせてください。
- このディレクトリにいずれかのファイルがない場合は，SAP 窓口に問い合わせてください。

---

7 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読み取り専用ウィンドウに表示されます。情報が正しいことを確認します。

インストール設定を変更する場合は，［**戻る**］をクリックします。

インストールを開始するには，［**次へ**］をクリックします。

8 インストール・ウィザードを終了します。

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。インストールを終了するには，［**終了**］をクリックしてください。

9 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，82 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。

10 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。

システムの状況モニタを使って，Collector が正常に実行していることを確認できます。詳細については，94 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 の UI には，Collector の各インスタンスは次のシステム・タイプの Probe として表示されます。Oracle Probe，SAP Probe，MQ Probe または SQL Server Probe

UNIX マシンへの Diagnostics Collector のインストール

ここでは，グラフィック・インストールまたはコンソール・モード・インストールを使って，UNIX 環境に Collector をインストールするのに必要な手順を紹介します。

グラフィック・インストールに表示されるインストール画面は，66 ページ「Windows マシンへの Diagnostics Collector のインストール」で Windows へのインストールに使用されているものと同じです。

インストーラの起動

インストーラは，Diagnostics インストール・ディスクまたは Business Availability Center の［Diagnostics Downloads］ページから直接起動できます。

インストール・ディスクからインストーラを起動するには

1 **＜ HP Diagnostics インストール・ディスク＞ /Diagnostics_Installers** ディレクトリから，Collector がインストールされるマシンにインストーラ **CollectorSetup ＜プラットフォーム＞ _ ＜バージョン＞ .bin** をコピーします。たとえば，**CollectorSetupLinux_8_00.bin** などです。

2 75 ページ「インストールの実行」に進みます。

Business Availability Center の［Diagnostics downloads］ページからインストーラを起動するには

1 Business Availability Center 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，環境に適したインストーラへのリンクをクリックします。

3 Collector をインストールするマシンにインストーラを保存します。

注：Diagnostics Server（Commander モード）をインストールした Probe インストーラ・ディレクトリへのパスを指定した場合のみ，Business Availability Center で Collector インストーラを使用できます。詳細については，47 ページ「Diagnostics Server（Commander モード）のインストール」を参照してください。

---

75 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

Collector をインストールするマシンにインストーラをコピーしたら，インストーラを実行できます。

注 ：

- ホスト・マシンにすでに **Collector** がインストールされている場合は，付録 G「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」を参照してください。
- 以下の手順は，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，付録 I「UNIX コマンドの使用」を参照してください。

Collector をインストールするには

1 インストーラを実行します。

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。

- インストーラをグラフィック・モードで実行するには，UNIX コマンド・プロンプトで ＜インストーラ＞ executable と入力します。つまり，＜インストーラ＞ の場所に，**CollectorSetupSolaris_ ＜リリース番号＞ .bin**，**CollectorSetupHP11x_ ＜リリース番号＞ .bin** または **CollectorSetupLinux_ ＜リリース番号＞ .bin** のように入力します。

  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。66 ページ「Windows マシンへの Diagnostics Collector のインストール」に進みます。

- インストーラをコンソール・モードで実行するには，UNIX コマンド・プロンプトで ＜ installer ＞ -console と入力します。つまり，＜ installer ＞ の場所に，**CollectorSetupSolaris_ ＜リリース番号＞ .bin**，**CollectorSetupHP11x_ ＜リリース番号＞ .bin** または **CollectorSetupLinux_ ＜リリース番号＞ .bin** のように入力します。

  インストーラがコンソール・モードになり，ソフトウェア使用許諾契約書が表示されます。

2 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，規約に同意します。

使用許諾契約を続けるには **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。

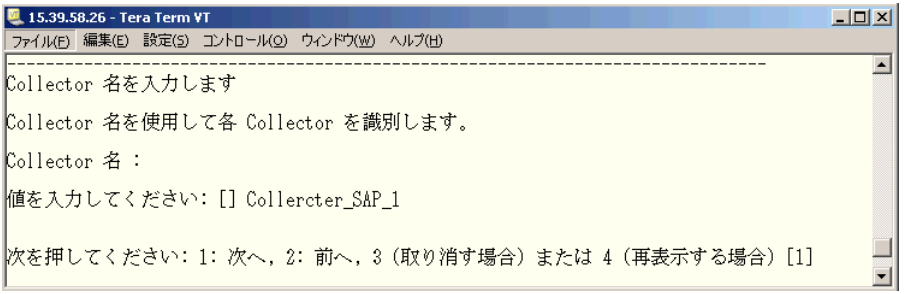
［**次へ**］を選択してインストールを続行します。

3 Collector のインストール先を指定します。

［**ディレクトリ名**］プロンプトで，括弧内に表示されるデフォルトのインストール先を受け入れるか，またはほかのインストール先のパスを入力します。

［**次へ**］を選択してインストールを続行します。

4 Collector に一意の名前を割り当てます。

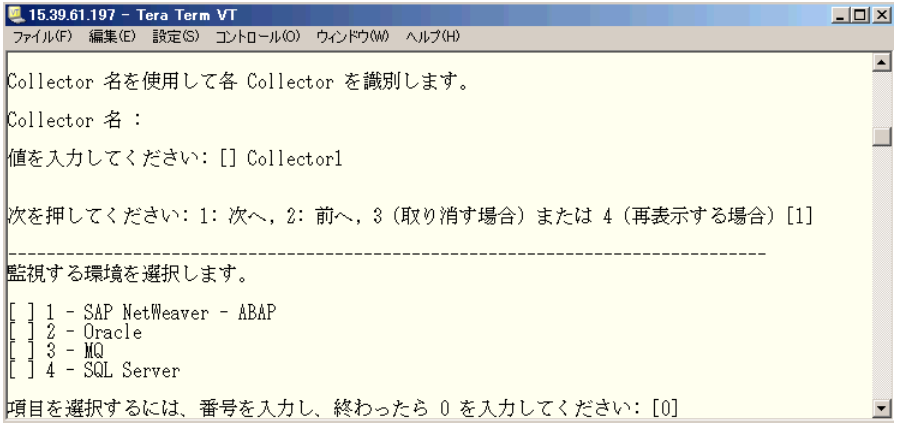


システムの状況モニタのこの特定の Collector を一意に識別する名前を Collector に割り当てます。システムの状況モニタを使って，Collector のインストールと設定を確認します。詳細については，94 ページ「Diagnostics Collector のインストールの確認」を参照してください。

名前に使用できる文字は，**-**，**_** およびすべての英数字です。

続行するには［**次へ**］を選択します。

5 Collector が監視するアクティブ・システムを指定します。



この Collector に適用するアクティブ・システムを選択します。

➤ 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。

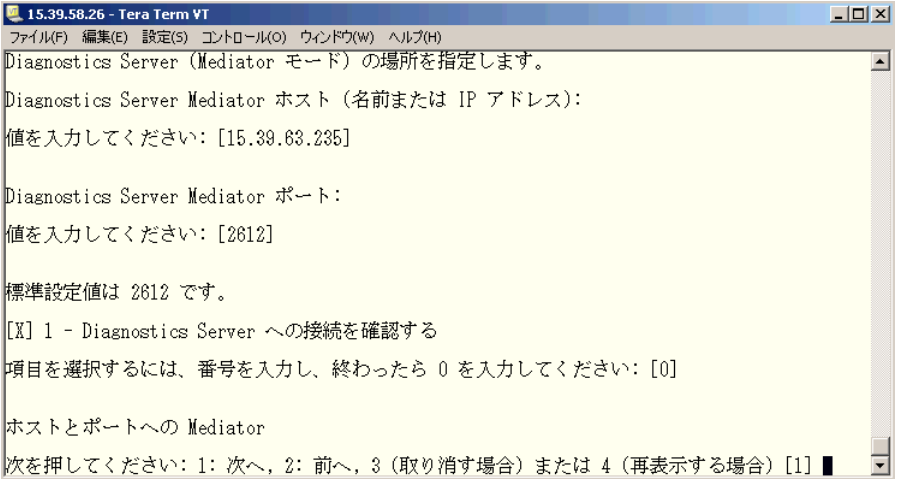
- ➤ Oracle 10g Database Server でデータを収集するには，［**Oracle**］を選択します。
- ➤ MQ Series 環境でデータを収集するには，［**MQ**］を選択します。
- ➤ SQL Server データベースでデータを収集するには，［**SQL Server**］を選択します。

重要：インストール後，監視する SAP NetWeaver - ABAP，Oracle，MQ および SQL Server 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，82 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。

［**次へ**］を選択してインストールを続行します。

6 Diagnostics Server（Mediator モード）の詳細情報を入力します。

Diagnostics Server（Mediator モード）との通信を可能にする情報を入力します。



Collector が実行する Diagnostics デプロイメントに Diagnostics Server が 1 つしかない場合，その Diagnostics Server 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 Server が複数ある場合は，Collector からイベントを受信する Diagnostics Server（Mediator モード）の情報を入力します。

a Diagnostics Server（Mediator モード）のホストのホスト名または 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 Server が Collector の通信のためにリスンしているポートの番号を入力してください。デフォルトのポートは **2612** です。Diagnostics Server をインストールした後にポートを変更した場合は，デフォルトのポートではなく，変更後のポート番号を指定します。

c Diagnostics Server 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 Server への接続を確認する**］を選択します。

［**次へ**］を選択してインストールを続行します。

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。結果が芳しくない場合は，次のインストール手順に進む前に報告されます。

ここでこの問題を解決しない場合は，［**Diagnostics Server への接続を確認する**］オプションをクリアします。

7 SAP Java Connector の場所を指定します。

プロンプトに，SAP Java Connector がインストールされているディレクトリの名前を入力します。

このディレクトリには，次のファイルが含まれていなければなりません。

- sapjco.jar
- librfc.dll
- sapjcorfc.dll

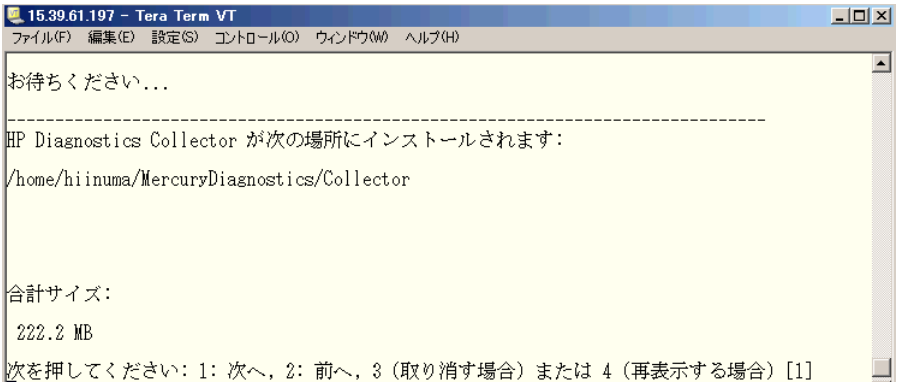
注：

- SAP Java Connector のディレクトリ名がわからない場合は，SAP 窓口に問い合わせてください。
- このディレクトリにいずれかのファイルがない場合は，SAP 窓口に問い合わせてください。

［**次へ**］を選択してインストールを続行します。

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。情報が正しいことを確認します。



設定を変更するには，［**前へ**］を選択して前のプロンプトに戻ります。

Collector の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・ウィザードを終了します。

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。

10 アクティブ・システムの XML ファイルを設定します。

手順 5 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。

アクティブ・システム・プロパティの設定については，82 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 のインストールを確認します。

システムの状況モニタを使って，Collector が正常に実行していることを確認できます。詳細については，94 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 Collector のサイレント・インストール

サイレント・インストールとは，ユーザが操作することなく自動的に実行されるインストールです。ユーザの入力の代わりに，サイレント・インストールでは，各インストール手順の応答ファイルからの入力を受け入れます。

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。

応答ファイルには拡張子 **.rsp** が付いています。必要な場合は標準的なテキスト・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

➤ 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 ＞
```

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，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，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。XML ファイルのインスタンス定義は，アクティブ・システムのインスタンス上の Probe のように動作します。

本項の内容

SAP NetWeaver - ABAP Probe 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・インスタンスには，直接アクセスできる場合もあれば，SAP Message Server を使用した接続を要求されることもあります。定義する各 SAP NetWeaver - ABAP Probe について，使用している接続オプションを把握しておく必要があります。

＜ Collector のインストール・ディレクトリ＞ ¥etc¥r3config.xml ファイルで SAP NetWeaver - ABAP Probe を定義および設定します。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 Probe を設定するには

1 **＜ Collector のインストール・ディレクトリ＞ ¥etc¥r3config.xml** を開きます。

2 SAP NetWeaver - ABAP インスタンスが SAP Message Server を通る SAP NetWeaver - ABAP Probe を定義している場合，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 インスタンスが直接アクセスする SAP NetWeaver - ABAP Probe を定義している場合，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・コードにコメントを挿入します。

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 を表示するための Probe 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。
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 - ABA システムに接続しているユーザの名前。 このユーザは，少なくとも S_RFC 認証オブジェクトが必要になりますが，R/3 4.7 以前のシステムでは不十分です。そのようなシステムについては，R/3 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 のインストール・ディレクトリ > ¥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する方法で対応できます。	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）。	EncryptPassword.jsp ユーティリティを使って（92 ページ「パスワードの難読化」参照），パスワードを暗号化します。

プロパティ	説明	値
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。
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。

Oracle Probe の設定

＜ Collector のインストール・ディレクトリ＞ ¥etc¥oracle-config.xml で Oracle Probe を定義および設定します。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 Probe を設定するには

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 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・コードにコメントを挿入します。

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	Oracle データベース・サーバのホスト・マシンの名前。	Oracle 管理者から取得可能。
portNumber	Oracle データベース・サーバが要求をリスンするポート。	デフォルト値：**1521**

プロパティ	説明	値
instanceName	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。	デフォルト値：**Orcl** 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**：パフォーマンス測定値を収集するには，少なくてもユーザには，少なくても **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（92 ページ「パスワードの難読化」を参照），パスワードを暗号化します。
probeName	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。	ユーザ定義。この値が定義されていない場合は，**instanceName** と同じ値が使われます。
probeGroupName	Diagnostics UI にこの Probe を表示するための Probe グループの論理名。既存の Probe グループを使うことも，または新しい Probe グループを定義することもできます。	ユーザ定義。 例： 既存：**Default** 新規：**Oracle**

MQ Probe の設定

＜ Collector のインストール・ディレクトリ＞ etc¥mq-config.xml で MQ Probe を定義および設定します。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（Collector）には，以下の権限が必要です。

```
setmqaut -m QM_ovrbat5 -n ** -t queue -g testMQGroup +dsp +get
setmqaut -m QM_ovrbat5 -n SYSTEM.ADMIN.COMMAND.QUEUE -t queue -g
testMQGroup +dsp +get +put
setmqaut -m QM_ovrbat5 -n ** -t channel -g testMQGroup +dsp
setmqaut -m QM_ovrbat5 -t qmgr -g testMQGroup +connect +dsp +inq
```

MQ Probe が測定値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測定値を取り出すことができます。デフォルトでは，MQ Probe はすべてのキューのタイプから測定値を収集します。
＜ 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，無視するキューのタイプを指定します。

MQ Probe を定義して設定するには

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・コードにコメントを挿入します。
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。	デフォルト値 : **1414**
queueManagerName	接続先する MQ Manager。	MQ 管理者から取得可能。

プロパティ	説明	値
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	Security Exit は，プラグ可能なセキュリティ・プロバイダに関する IBM の用語です（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 Security Exit 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 Security Exit 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 Probe 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。 この値が定義されていない場合，デフォルトでキュー・マネージャの名前が使用されます。
probeGroupName	Diagnostics UI にこの Probe を表示するための Probe グループの論理名。 既存の Probe グループを使うことも，または新しい Probe グループを定義することもできます。	ユーザ定義。 例： 既存：**Default** 新規：**MQ**

収集する測定値のキューを限定するには

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 で測定値を収集しない 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.queue	MQQDT_PERMANENT_DYNAMIC
collect.temporary.dynamic.queues	MQQDT_TEMPORARY_DYNAMIC
collect.shared.dynamic.queues	MQQDT_SHARED_DYNAMIC

3 Collector で測定値を収集したくないタイプに対して「true」の代わりに「false」を指定し，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。

SQL Server Probe の設定

＜ Collector のインストール・ディレクトリ＞ ¥etc¥sqlserver-config.xml で SQL Server Probe を定義および設定します。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 Probe を設定するには

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・コードにコメントを挿入します。

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・ホスト・マシンの名前。	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スンするポートの番号。	デフォルト値：**1433**
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できます。	デフォルト値：**Default** SQL Server 管理者から取得可能。
userId	SQL Server データベースに接続しているユーザの ID。 **注**：ユーザには，パフォーマンス測定値を収集するために少なくとも **VIEW SERVER STATE** が必要です。	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。

プロパティ	説明	値
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（92 ページ「パスワードの難読化」を参照），パスワードを暗号化します。
probeName	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。 インスタンス内に ***n*** 個のデータベースがある場合は，実際には ***n+1*** 個のプローブが使用されます。追加のプローブは，待機イベントなどの測定値を含むインスタンスの合計用です。 追加のプローブは UI に probeName で表示され，各データベース用のプローブは probeName_databaseName で表示されます。	ユーザ定義。 この値が定義されていない場合は，**instanceName** と同じ値が使われます。
probeGroupName	Diagnostics UI にこの Probe を表示するための Probe グループの論理名。 既存の Probe グループを使うことも，または新しい Probe グループを定義することもできます。	ユーザ定義。 例： 既存：**Default** 新規：**SQL Server**

パスワードの難読化

Diagnostics : **http:// ＜ホスト名＞ :2006/security/EncryptPassword.jsp** に含まれる Web アプリケーションを使って難読化パスワードを作成します。＜ホスト名＞ は，Diagnostics Server がインストールされているコンピュータの名前にします。

作成した難読化パスワードは，SAP NetWeaver - ABAP Collector の設定に使われる **r3config.xml** ファイル，Oracle Collector の設定に使われる **oracle-config.xml** ファイル，SQL Server Collectorin の設定に使われる **sqlserver-config.xml** ファイルで使用できます。

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。難読化パスワードが表示されます。OBF を含めて，このページから難読化パスワード全体をコピーします。まず，適切なプロパティ・ファイルに貼り付けます（**r3config.xml** ファイル，**oracle-config.xml** ファイル，**sqlserver-config.xml** ファイル）。

注：プレーンテキストのパスワード・プロパティも同様に使用できます。

---

security.encrypted-password プロパティは，**collector.properties**，**dispatcher.properties** および **server.properties** プロパティ・ファイルで，**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。以下は，影響のあるプロパティ・ファイルのセクションからのコピーです。

```
##############################################################################
# リモート・サーバ認証プロパティ
##############################################################################

#
# このユーザ名とパスワードは，Diagnostics コンポーネント（Probe およびサーバ）
# 間の通信に使われます。場合によっては，このパスワードを頻繁に変更して
# 企業内のシステムの安全性を保持する必要があります。このパスワードを
# 変更しない場合は，まず
# http:// ＜ホスト名＞ :2006/security/EncryptPassword.jsp でパスワードを暗号化する
必要があります。
# security.encrypted-password を security.password に置き換えることで，
# プレーンテキスト・パスワードを使用できます。また，
# ＜インストール・ディレクトリ＞ /etc/.htaccess ファイルおよび企業内で互いに通信
する Diagnostics Probe
# およびサーバの暗号化されたパスワードも変更する必要があります。
#
security.username=mercury
security.encrypted-
password=OBF:1c431jg81hv41k1d1l161wu81z0d1pyl1wmt1n6h1ym71n511wnd1pw11
z0h1wu61kxw1jyl1hse1jd21c2z
```

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。システムの状況モニタを使って，Collector のインストールを確認できます。システムの状況モニタの詳しい使い方については，付録 D「System Health Monitor の使用」を参照してください。

推奨インストール手順に従った場合，Collector のインストール後，以下を確認できます。

- Diagnostics Server（Commander モード）が正常にインストールされている。
- 追加の Diagnostics Server（Mediator モード）が正常にインストールされ，Diagnostics Server（Commander モード）と通信している。
- Collector が正常にインストールされていて，関連する Diagnostics Server との接続を確立している。

  デプロイメントに応じて，Collector は，システムの状況モニタに Diagnostics Server（Commander モード）または Diagnostics Server（Mediator モード）の子として表示されます。Collector は，**＜Collector 名＞というアイコンで示されます**。Collector 名は，インストール時に割り当てられます。

Collector が停止すると，システムの状況モニタの Collector ノードが灰色になります。Collector が開始すると，ノードは緑色になります。

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

➤ ［**スタート**］＞［**プログラム**］＞［**HP Diagnostics Collector**］＞［**Start HP Diagnostics Collector**］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

➤ ［**スタート**］＞［**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

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_LROOTUnable 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

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 を使わずに 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

➤ **＜ 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

➤ **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，次の場所で確認できます。

➤ **＜ Collector のインストール・ディレクトリ＞ ¥version.txt** ファイルの中

➤ システムの状況モニタ上の Collector の詳細情報の中

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

- 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 と .NET Agent（Probe）のインストールおよびセットアップ

第 5 章

Java Agent のインストール

Windows マシン，UNIX マシン，および z/OS メインフレーム・マシンに Java Agent をインストールする方法について説明します。また，標準のインストーラを使ってほかのプラットフォームに Java Agent をインストールする方法についても説明します。

本章の内容

Java Agent インストーラについて

Java Agent の機能は，HP Diagnostics/TransactionVision Java Agent（Java Agent）によって提供されます。Java Agent は，Diagnostics Java Agent と TransactionVision Java Sensors（JMS，Servlet，JDBC および EJB）の機能を 1 つのコンポーネントに組み合わせたものです。

Java Agent は，Diagnostics 環境では Java Agent として，TransactionVision 環境と複合環境では Java Sensor として機能するように設定され，Probe と Sensor の両方として同時に機能します。

Java Agent を使用して HP Diagnostics でアプリケーションを監視するには，1）Java Agent をインストールし，2）JRE をインストゥルメントし，3）アプリケーション・サーバの起動スクリプトを Java Agent と連動するように設定します。JRE のインストゥルメントについては，第 6 章「JRE Instrumenter の実行」を参照してください。アプリケーションの起動スクリプトの変更に関する詳細については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」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。

注：Agent がインストールされる場所が Diagnostics ＜ Probe のインストール・ディレクトリ＞ になります。デフォルトでは，Windows の場合は C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent，UNIX の場合は /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent です。

注：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，30 ページ「Diagnostics Probe for Java ホストの要件」を参照してください。

Java Agent のインストールの詳細については，オペレーティング・システムとインストール方法に応じて，次のいずれかのセクションを参照してください。

- ➤ 103 ページ「Windows での Java Agent のインストールと設定」
- ➤ 113 ページ「UNIX での Java Agent のインストールと設定」
- ➤ 126 ページ「z/OS メインフレームへの Java Agent のインストール」

➤ 128 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」

➤ 130 ページ「Java Agent のサイレント・インストール」

Windows での Java Agent のインストールと設定

Windows マシンに Java Agent 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。また，これらの手順は，グラフィック・インストーラを使って Java Agent を UNIX マシンにインストールする際にも使用します。

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は，667 ページ「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」を参照してください。

本項の内容

Windows での Java Agent インストーラの起動

Java Agent インストーラは，Diagnostics インストール・ディスク，別の場所，または Business Availability Center の［Diagnostics Downloads］ページから起動できます。

製品インストール・ディスクからインストーラを起動するには

1 インストール・ディスクのルート・ディレクトリにある **setup.exe** ファイルを実行します。HP Diagnostics のメイン・インストール・メニューが表示されます。

2 インストール・メニュー・ページで，［**Diagnostics Agent for Java**］を選択してインストーラを起動します。

3 104 ページ「インストールの実行」に進みます。

別の場所からインストーラを起動するには

1 **＜ HP Diagnostics インストール・ディスク＞ ¥Diagnostics_Installers** ディレクトリから，実行ファイル **JavaAgentSetup_win_ ＜バージョン＞ .exe** を新しい場所にコピーして，実行します。

2 104 ページ「インストールの実行」に進みます。

Business Availability Center の［Diagnostics downloads］ページからインストーラを起動するには

1 Business Availability Center 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 Java Agent installer for Windows をダウンロードします。

注：Java Agent インストーラは Business Availability Center からアクセスするために必要なディレクトリに置かれている場合にかぎり，Business Availability Center で利用できます。Diagnostics Server 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。

---

104 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

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，インストールを実行できます。

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は，付録 G「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」を参照してください。

Windows マシンに Java Agent をインストールするには

1 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。

続行するには［**次へ**］をクリックします。

2 Agent のインストール先を指定します。

デフォルトのディレクトリを受け入れるか，ほかの場所を選択します。ほかの場所を選択するには，［**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［**ブラウズ**］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。

注：この場所が ＜ Probe のインストール・ディレクトリ ＞ になります。デフォルトでは，Windows の場合は C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent，UNIX の場合は /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent です。

続行するには［**次へ**］をクリックします。

3 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。

続行するには［**次へ**］をクリックします。

4 インストールのサマリ情報を確認します。サマリ情報パネルにエラーが表示されていなければ，［次へ］をクリックして進みます。

Java Agent Setup Module が起動します。

Java Agent の Profiler 専用の設定

Java Agent を Diagnostics Profiler と連携して動作するスタンドアロンの Java Agent として，あるいは Diagnostics Server と連携して動作する Java Agent として設定できます。Probe を Profiler 専用で設定すると，今後 Diagnostics Server と連携して動作するように Probe を再設定できます。

Java Agent を Diagnostics Profiler 専用として設定するには，Java Agent Setup Module を使用します。

Java Agent のインストールが終わると，Java Agent Setup Module は自動的に起動します。［**スタート**］＞［**プログラム**］＞［**HP Java Agent**］＞［**Setup Module**］を選択することにより，いつでも Java Agent Setup Module を起動できます。

Java Agent を Diagnostics Profiler として設定するには

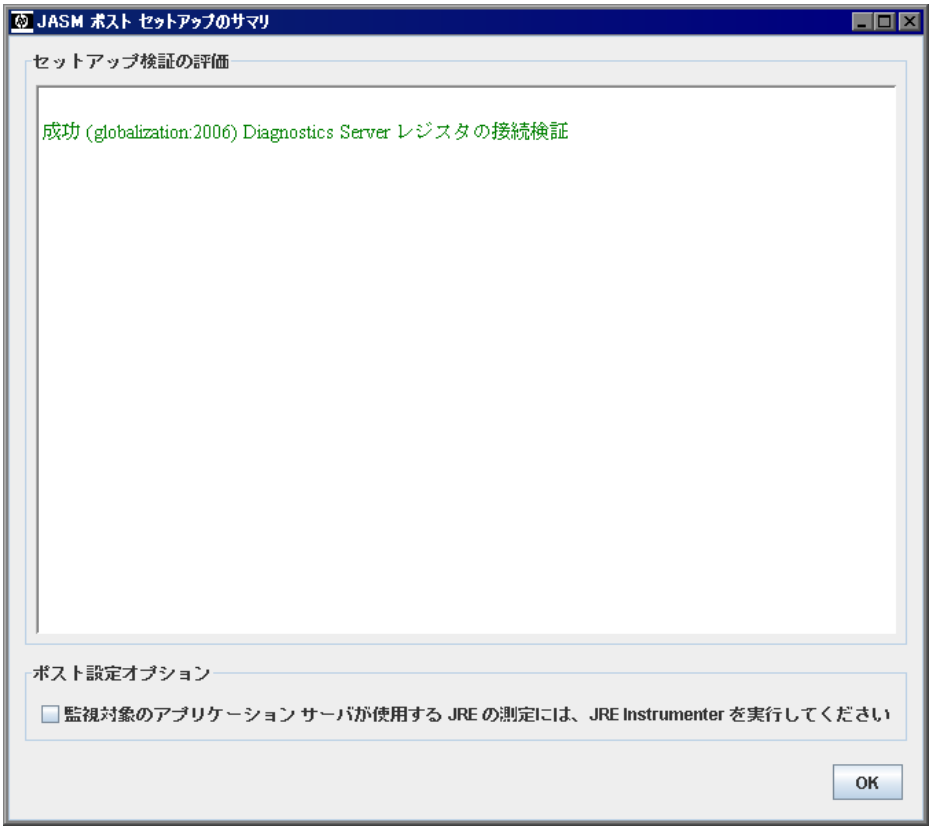
1［Diagnostics Profiler のみとして動作するように Java エージェントを設定します］オプションを選択します。

それ以外の設定オプションはありません。

設定を終了するには，［**完了**］をクリックしてください。

［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。

2 ［ポスト セットアップのサマリ］を確認し，**JRE Instrumenter** をここで実行するかどうかを指定します。



JRE Instrumenter をすぐに実行するには，チェック・ボックスを設定して［**OK**］をクリックします。

注：統合開発環境（IDE）で WebSphere と連携して動作するように Agent をインストールしている場合は，ここで述べる手順と若干異なる手順で JRE Instrumenter を実行する必要があります。詳細については，160 ページ「WebSphere IDE での JRE Instrumenter の実行」を参照してください。

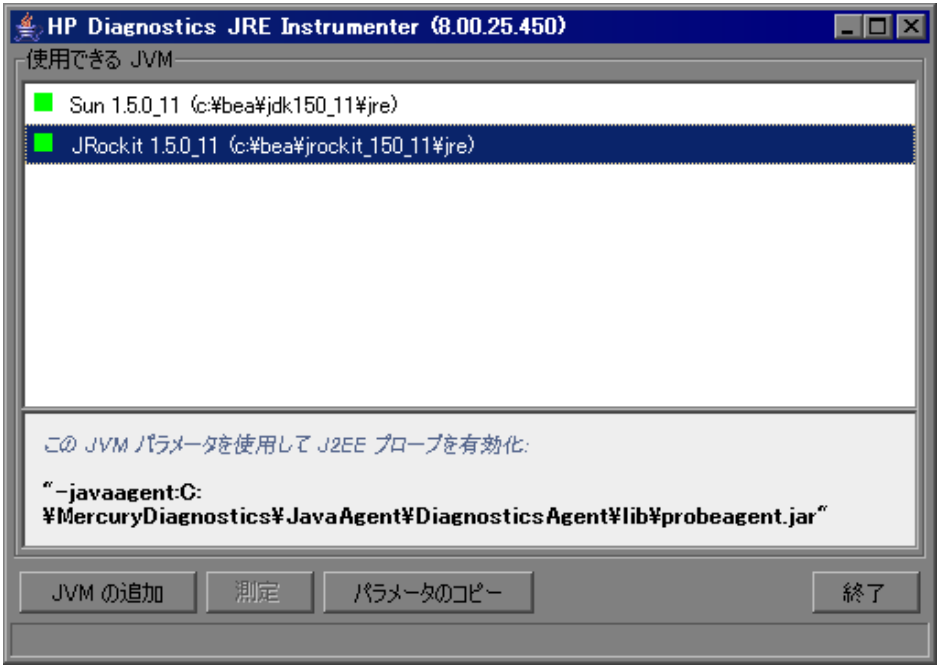
今すぐの JRE Instrumenter の実行をスキップして後で実行する場合は，チェック・ボックスをオフにして［**OK**］をクリックします。［JRE Instrumenter］ウィンドウが閉じ，Java Agent の設定が終了します。

JRE Instrumenter を Java Agent Setup プログラムとは別に実行する方法については，第 6 章「JRE Instrumenter の実行」を参照してください。

3 インストゥルメントする JVM を選択し，起動パラメータをコピーします。

［Instrumenter］ウィンドウが表示されます。

a インストゥルメントする JRE を選択して［**測定**］をクリックします。



ダイアログに［**使用できる JVM**］がリストされていない場合は，［**JVM の追加**］ボタンをクリックして JRE を参照します。たとえば，**D:¥bea** ディレクトリに WebLogic 8.1（Sun JDK を使用）をインストールした場合，JRE ディレクトリ **D:¥bea¥jdk142_05¥JRE** の bin フォルダに **java.exe** ファイルがあります。

b［**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。

重要：後の手順でクリップボードの内容を貼り付ける必要があるため，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。

---

c［**終了**］をクリックして［JRE Instrumenter］ウィンドウを閉じます。

4 インストール後の作業に進みます。

Agent のインストール，Java Agent としての設定，および JRE のインストゥルメントが終わったら次の作業を行う必要があります。

➤ Probe が監視対象アプリケーションと連携して起動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する。

詳細については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」を参照してください。

➤ 追加の Probe 設定を実行します。

環境に適用する Probe の詳細設定を確認します。詳細については，第 15 章「Java Probe およびアプリケーション・サーバの詳細設定」を参照してください。

Diagnostics Server と連携して動作する Java Agent の設定

Java Agent を Diagnostics Profiler と連携して動作するスタンドアロンの Java Agent として，あるいは Diagnostics Server と連携して動作する Java Agent として設定できます。

Java Agent を Diagnostics Server にレポートする Java Agent として設定するには，Java Agent Setup Module を使用します。

Java Agent のインストールが終わると，Java Agent Setup Module は自動的に起動します。［**スタート**］＞［**プログラム**］＞［**HP Java Agent**］＞［**Setup Module**］を選択することにより，いつでも Java Agent Setup Module を起動できます。

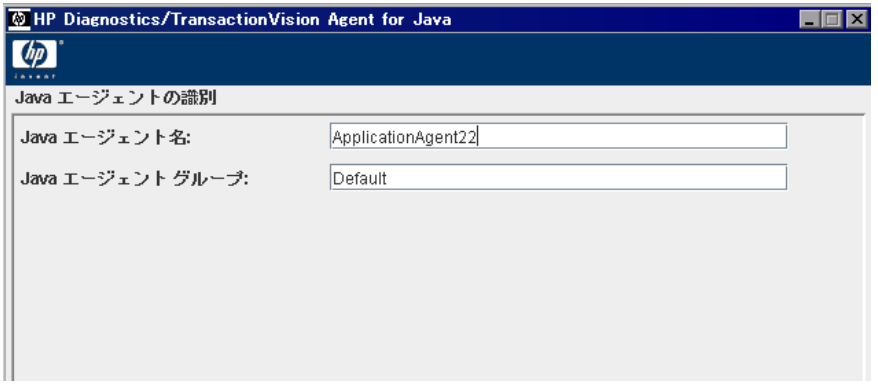
Java Agent を Diagnostics Server にレポートする Java Agent として設定するには

1 **［HP Diagnostics Server と動作する Diagnostics Java エージェント］オプションを選択します。**

Java Agent は Diagnostics Java Agent と同様に TransactionVision Sensor としても設定できます。Java Agent を TransactionVision Sensor として設定する場合は，本書では説明されていないダイアログが Java Agent Setup Module から表示されます（『**HP TransactionVision デプロイメント・ガイド**』を参照）。

続行するには［**次へ**］をクリックします。

2 **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。名前に使用できる文字は，**-**，**_** およびすべての英数字です。Agent 名は Probe 名になるように割り当てられます。

Agent に名前を割り当てる際，Agent が監視しているアプリケーションと Probe のタイプ（この場合は PetWorld アプリケーション・エージェント）を識別しやすい名前を選択します。

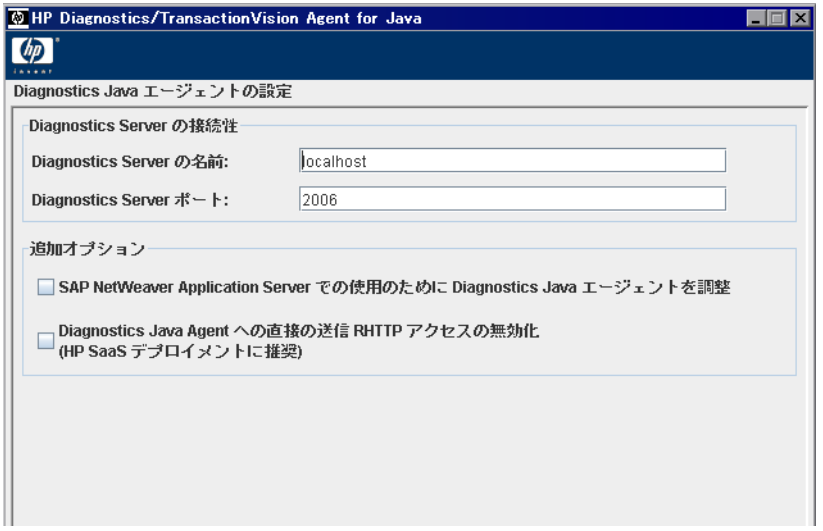
- Java Agent グループ名に対しては，既存のグループの名前か新規作成するグループの名前を入力します。Agent グループ名は大文字と小文字を区別します。

  Probe グループは，同じ Diagnostics Server に報告する Probe の論理グループです。Probe 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，場合によっては，Probe グループや個別の Probe のパフォーマンスを監視するために，1 つの Probe 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 Probe を割り当てる必要があります。

続行するには［**次へ**］をクリックします。

3 Diagnostics Server の設定情報と追加オプションを入力します。



a ［**Diagnostics Server の名前**］ボックスに，Diagnostics Server のホストのホスト名と IP アドレスを入力します。

Agent が実行する Diagnostics デプロイメントに Diagnostics Server が 1 台だけある場合は，Diagnostics Server 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 Server が複数ある場合は，Agent からイベントを受信するための Diagnostics Server の情報を入力します。

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。

b［**Diagnostics Server ポート**］ボックスに，Diagnostics Server のポート番号を入力します。

Diagnostics Server のデフォルトのポートは **2006** です。Diagnostics Server をインストールした後にポートを変更した場合は，デフォルト・ポートではなく，変更後のポート番号を指定する必要があります。

c この Agent を使って SAP NetWeaver アプリケーション・サーバをサポートする必要がある場合は，［**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。

d この Agent を HP SaaS 環境で動作させる必要がある場合は，［**Diagnostics Java Agent への直接の送信 RHTTP アクセスの無効化**］チェック・ボックスを設定します。

Diagnostics Server では，RHTTP を使って Java Agent から測定値をクエリします。HP Software-as-a-Service (SaaS) が Probe からデータ・ディレクトリにアクセスできないように，Probe の発信 RHTTP を無効にする必要があります。

適切なオプションを選択して，［**完了**］をクリックします。

設定プロセスが開始し，進行状況バーに設定の進捗状況が表示されます。

Diagnostics Server への接続性がテストされ，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。

Agent のインストール，Java Agent としての設定，JRE のインストゥルメントが終わったら，以下のインストール後の作業を実行する必要があります。

4 Probe が監視対象アプリケーションと連携して起動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。

詳細については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」を参照してください。

5 Probe のインストールを確認します。

Probe のインストールは，133 ページ「Java Agent のインストールの確認」に説明されているように，システムの状況モニタを使用して確認されます。

6 追加の Probe 設定を実行します。

環境に適用する Probe の詳細設定を確認します。詳細については，第 15 章「Java Probe およびアプリケーション・サーバの詳細設定」を参照してください。

UNIX での Java Agent のインストールと設定

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。ここでは，コンソール・モード・インストールを使って，ほとんどの UNIX 環境に Java Agent をインストールするのに必要な手順を紹介します。

グラフィック・インストーラを使って Java Agent を UNIX マシンにインストールする場合は，103 ページ「Windows での Java Agent のインストールと設定」を参照してください。この手順は，UNIX のグラフィック・インストーラを使用する場合にも適用されます。

重要：デフォルトでは，**＜ Probe 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 はユーザによって実行されている監視対象アプリケーションから測定値を収集できます。

組織のセキュリティ要件に応じては，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。

例：

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

場合によっては，標準の UNIX インストーラを使用できることもあります。そのような場合，128 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する必要があります。

以下の手順およびスクリーン・ショットは，Solaris マシンに Agent をインストールするためのものです。ほかの認定 UNIX プラットフォームでも同じ手順でインストールできます。

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は，付録 G「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」を参照してください。

本項の内容

Probe ファイルのファイル許可の設定（UNIX 専用）

Java Agent が正しく動作するためには，監視対象の JVM を起動するユーザには次のアクセス許可が必要です。

- **＜Probe 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。
- **＜Probe 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。
- **＜Probe 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。

  Diagnostics Profiler の［Configuration］タブが正しく動作するためには，監視対象の JVM を起動するユーザには次のアクセス許可が必要です。

- **＜Probe のインストール・ディレクトリ＞/etc** ディレクトリへの読み取り / 書き込みアクセス。

  デフォルトでは，**＜Probe 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 はユーザによって実行されている監視対象アプリケーションから測定値を収集できます。

  組織のセキュリティ要件に応じては，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。

例：

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

UNIX での Java Agent インストーラの起動

インストーラは，Diagnostics インストール・ディスクまたは Business Availability Center の［Diagnostics Downloads］ページから起動できます。

製品インストール・ディスクからインストーラを起動するには

1 **＜HP Diagnostics インストール・ディスク＞ /Diagnostics_Installers** ディレクトリから，Java Agent がインストールされるマシンに適切なインストーラ **JavaAgentSetup ＜プラットフォーム＞ _ ＜バージョン＞ .bin** をコピーします。

2 116 ページ「インストールの実行」に進みます。

Business Availability Center の［Diagnostics downloads］ページからインストーラを起動するには

1 Business Availability Center 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，環境に適したインストーラのリンクをクリックし，Java Agent をインストールするマシンに保存します。

注：Java Agent インストーラは，有効になっている場合にかぎり Diagnostics Server のインストール時に Business Availability Center で利用可能です。第 2 章「Diagnostics Server のインストール」の手順 3（48 ページ）を参照してください。

---

116 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

Probe をインストールするマシンにインストーラをコピーしたら，インストールを実行できます。

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は，付録 G「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」を参照してください。

注：以下の手順は，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，付録 I「UNIX コマンドの使用」を参照してください。

UNIX マシンに Java Agent をインストールするには

1 インストーラを実行します。

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。

➤ コンソール・モードでインストーラを実行するには，UNIX 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，次の手順で示されるようにコンソール・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，UNIX 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>
```

インストーラには，103 ページ「Windows での Java Agent のインストールと設定」のように，Windows インストーラのものと同じ画面が表示されます。

2 エンド・ユーザ使用許諾契約書に同意します。

エンド・ユーザのソフトウェア使用許諾契約書が表示されます。

契約書を読みます。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。

契約書の規約に同意し，［**次へ**］を選択してインストールを続行します。

3 Agent のインストール先を指定します。

［**ディレクトリ名**］プロンプトで，括弧内に表示されるデフォルトのインストール先を受け入れます。

［**次へ**］を選択してインストールを続行します。

4 インストール先を確認します。

インストール先と推定サイズがリストされます。

これらが問題なければ，［**次へ**］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ります。

Java Agent Setup Module が起動します。

Java Agent の Profiler 専用の設定

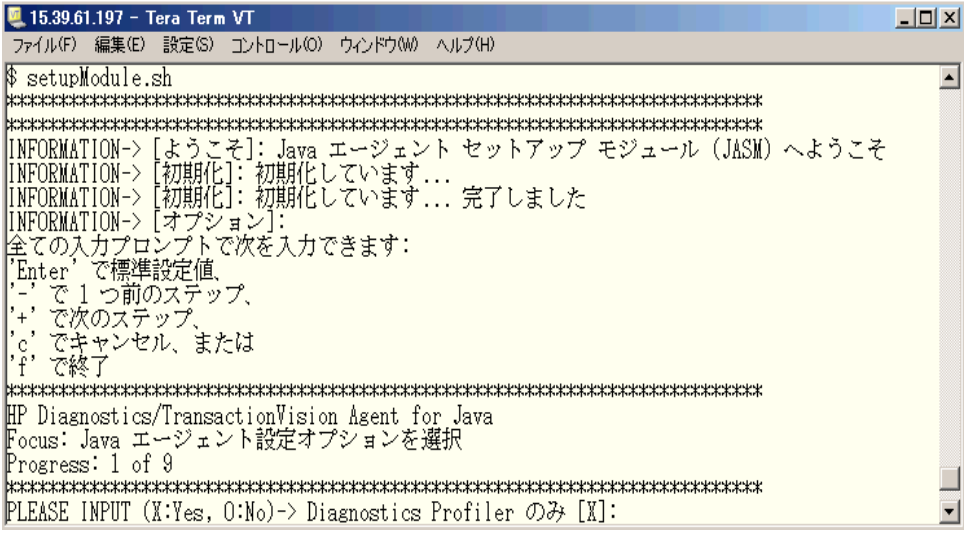
Java Agent を Diagnostics Profiler と連携して動作するスタンドアロンの Java Agent として，あるいは Diagnostics Server と連携して動作する Java Agent として設定できます。Probe を Profiler 専用で設定すると，今後 Diagnostics Server と連携して動作するように Probe を再設定できます。

Java Agent を Diagnostics Profiler として設定するには，Java Agent Setup Module を使用します。インストール・プログラムが終わると，Java Agent Setup Module は自動的に起動します。

＜ Probe のインストール・ディレクトリ＞ /bin/setupModule.sh を実行することにより，いつでも Java Agent Setup Module を起動できます。

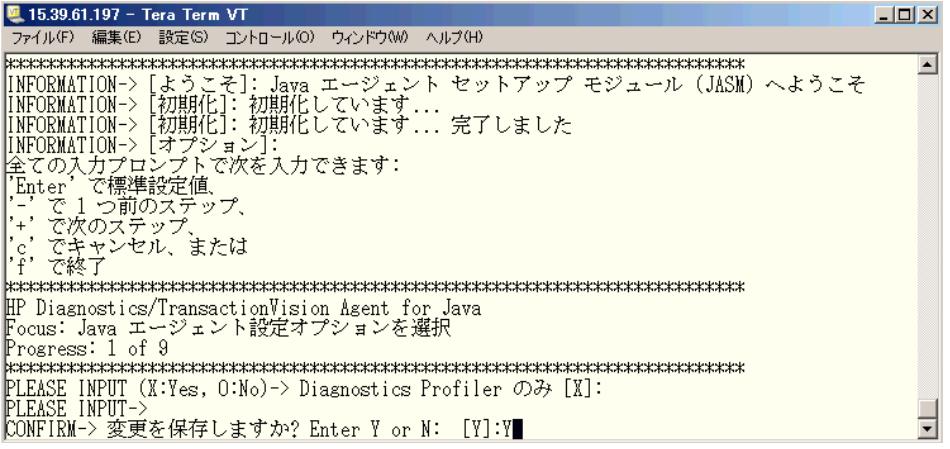
Java Agent を Diagnostics Profiler として設定するには

1 O を入力して［Diagnostics Profiler のみ］オプションを入力します。

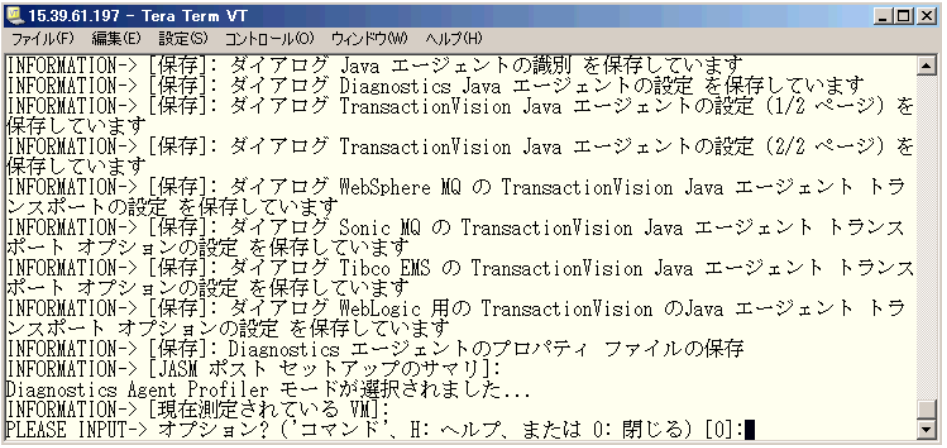


それ以外の設定オプションはありません。

2 Y を入力して，Java Agent Setup Module への変更を保存します。



3 JRE Instrumenter を実行します。



測定コマンドを次のように入力します。

- 特定のディレクトリの JVM をインストゥルメントし，classloader 情報を /classes/boot にコピーするには，jreinstrumenter.sh -b ＜ JVM のディレクトリ＞ を入力します。

  ＜ JVM のディレクトリ＞ は監視対象アプリケーション・サーバで使用する JRE がある JRE bin フォルダのパスです。たとえば，WebLogic 8.1（Sun JDK 使用）が **/opt/bea** ディレクトリにインストールされている場合は，次のように入力します。

  ```
  jreinstrumenter.sh -b  /opt/bea/jdk142_05/JRE
  ```

- 特定のディレクトリの JVM をインストゥルメントするには，jreinstrumenter.sh -i ＜ JVM のディレクトリ＞ を入力します。
- 特定のディレクトリの JVM をリストするには，jreinstrumenter.sh -a ＜ JVM のディレクトリ＞ を入力します。
- 既知のすべての JVM をリストするには，jreinstrumenter.sh -l を入力します。

注：統合開発環境（IDE）で WebSphere と連携して動作するように Agent をインストールしている場合は，ここで述べる手順と若干異なる手順で JRE Instrumenter を実行する必要があります。詳細については，160 ページ「WebSphere IDE での JRE Instrumenter の実行」を参照してください。

---

4 O（ゼロ）を入力して設定を終了します。

Agent のインストール，Java Agent としての設定，JRE のインストゥルメントが終わったら，以下のインストール後の作業を実行する必要があります。

- Probe が監視対象アプリケーションと連携して起動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する。

  詳細については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」を参照してください。

- 追加の Probe 設定を実行します。

  環境に適用する Probe の詳細設定を確認します。詳細については，第 15 章「Java Probe およびアプリケーション・サーバの詳細設定」を参照してください。

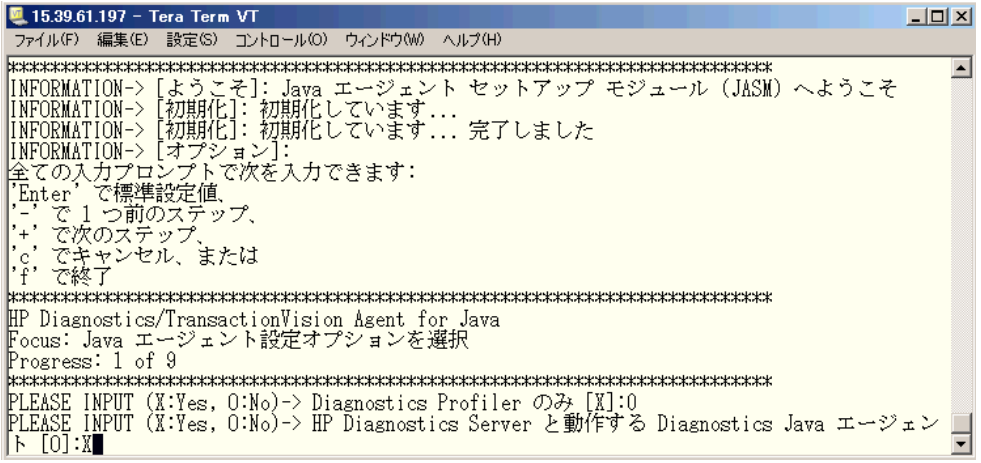
Diagnostics Server と連携して動作する Java Agent の設定

Java Agent を Diagnostics Profiler と連携して動作するスタンドアロンの Java Agent として，あるいは Diagnostics Server と連携して動作する Java Agent として設定できます。

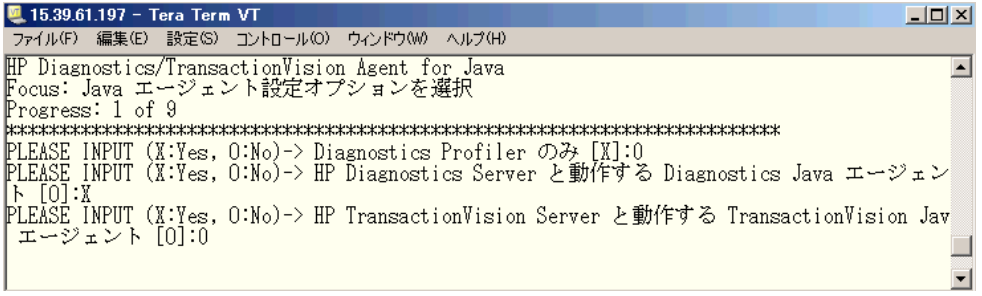
Java Agent を Diagnostics Server にレポートする Probe として設定するには，Java Agent Setup Module を使用します。
＜ Probe のインストール・ディレクトリ＞ /bin/setupModule.sh -console を入力して，Java Agent Setup Module をコンソール・モードで起動します。

Java Agent を Diagnostics Server にレポートする Java Agent として設定するには

1 **O を入力して［Diagnostics Profiler のみ］オプションをスキップし，X を入力して［HP Diagnostics Server と動作する Diagnostics Java エージェント］オプションを選択します。**



2 **Agent を TransactionVision sensor として実行するように設定するかどうかも指定します。**

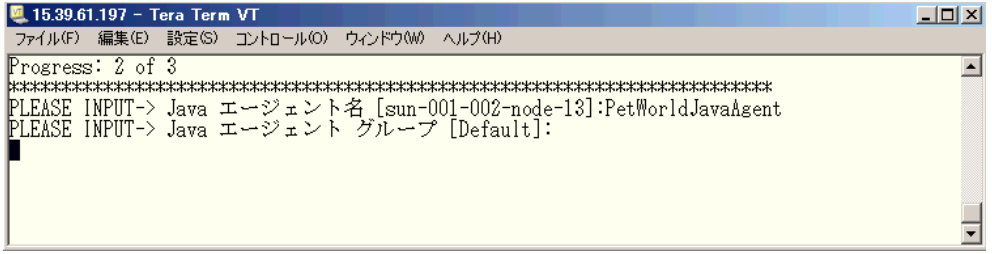


- Agent を TransactionVision sensor として設定しない場合は O を選択します。
- Agent を TransactionVision sensor として設定する場合は X を選択します。

  Java Agent は Diagnostics Java Agent と同様に TransactionVision Sensor としても設定できます。Java Agent を TransactionVision Sensor として設定する場合は，本書では説明されていないプロンプトが Java Agent Setup Module から表示されます。

TransactionVision Sensor として動作するように Java Agent を設定する場合は，『**HP TransactionVision デプロイメント・ガイト**』を参照してください。

3 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。名前に使用できる文字は，**-**，**_** およびすべての英数字です。Agent 名は Probe 名になるように割り当てられます。

  Agent に名前を割り当てる際，Agent が監視しているアプリケーションと Probe のタイプ（この場合は PetWorld アプリケーション・エージェント）を識別しやすい名前を選択します。

- 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。

Probe グループは，同じ Diagnostics Server に報告する Probe の論理グループです。Probe 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，場合によっては，Probe グループや個別の Probe のパフォーマンスを監視するために，1 つの Probe 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 Probe を割り当てる必要があります。

4 Diagnostics Server の設定情報を入力します。

a Diagnostics Server のホストのホスト名または IP アドレスを入力します。

Agent が実行する Diagnostics デプロイメントに Diagnostics Server が 1 台だけある場合は，Diagnostics Server のホスト名と通信ポート情報を入力します。

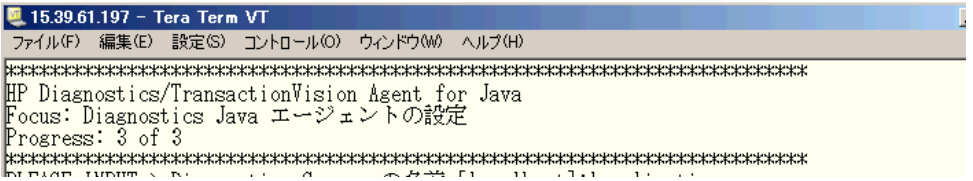
デプロイメントに Diagnostics Server が複数ある場合は，Agent からイベントを受信するための Diagnostics Server の情報を入力します。

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。

b Diagnostics Server のポート番号を入力します。

Diagnostics Server のデフォルトのポートは **2006** です。Diagnostics Server をインストールした後にポートを変更した場合は，デフォルト・ポートではなく，変更後のポート番号を指定する必要があります。

5 Agent を SAP NetWeaver アプリケーション・サーバで実行するためのチューニングを行うかどうかを指定します。

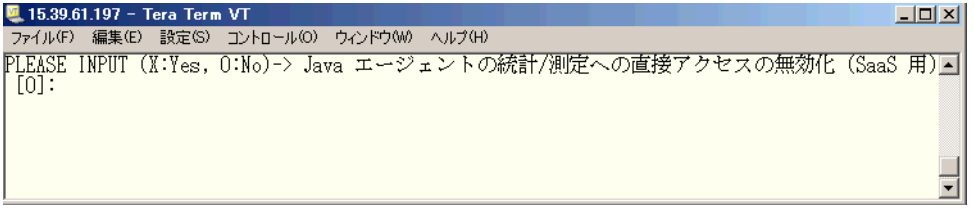


➤ SAP NetWeaver アプリケーション・サーバをサポートするには **X** を指定します。

➤ SAP NetWeaver アプリケーション・サーバをサポートするには **O** を指定します。

6 Agent が HP SaaS 環境で実行するかどうかを指定します。

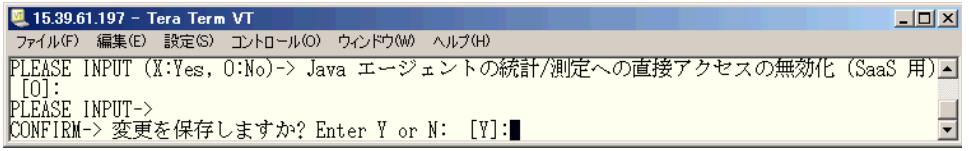
Diagnostics Server では，RHTTP を使って Java Agent から測定値をクエリします。HP Software-as-a-Service (SaaS) が Probe のデータに直接アクセスできないように，Probe の発信 RHTTP を無効にする必要があります。



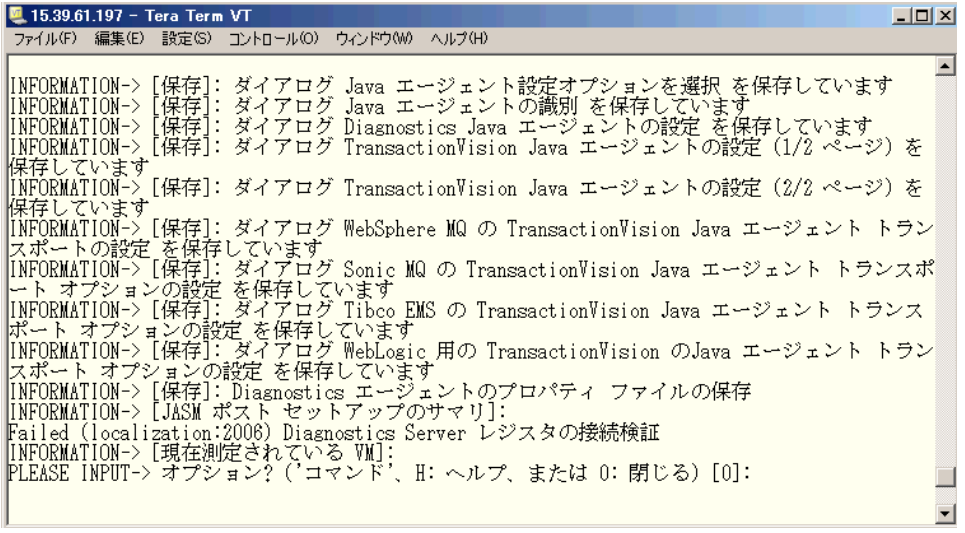
➤ 発信 RHTTP を無効にするには **X** を指定します。

➤ 発信 RHTTP を有効にするには **O** を指定します。

7 Java Agent Setup Module への変更を保存するようにプロンプトが表示されたら，Y を入力します。



8 JRE をインストゥルメントします。



インストゥルメンテーション・コマンドを次のように入力します。

➤ 特定のディレクトリの JVM をインストゥルメントし，classloader 情報を /classes/boot にコピーするには，jreinstrumenter.sh -b ＜ JVM のディレクトリ＞ を入力します。

＜ JVM のディレクトリ＞ は監視対象アプリケーション・サーバで使用する JRE がある JRE bin フォルダのパスです。たとえば，WebLogic 8.1（Sun JDK 使用）が **/opt/bea** ディレクトリにインストールされている場合は，次のように入力します。

```
jreinstrumenter.sh -b  /opt/bea/jdk142_05/JRE
```

- 特定のディレクトリの JVM をインストゥルメントするには，jreinstrumenter.sh -i ＜ JVM のディレクトリ＞ を入力します。
- 特定のディレクトリの JVM をリストするには，jreinstrumenter.sh -a ＜ JVM のディレクトリ＞ を入力します。
- 既知のすべての JVM をリストするには，jreinstrumenter.sh -l を入力します。

注：統合開発環境（IDE）で WebSphere と連携して動作するように Agent をインストールしている場合は，ここで述べる手順と若干異なる手順で JRE Instrumenter を実行する必要があります。詳細については，160 ページ「WebSphere IDE での JRE Instrumenter の実行」を参照してください。

---

9 0（ゼロ）を入力してセットアップを終了します。

Agent のインストール，Java Agent としての設定，JRE のインストゥルメントが終わったら，以下のインストール後の作業を実行する必要があります。

10 Probe が監視対象アプリケーションと連携して起動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。

詳細については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」を参照してください。

11 Probe のインストールを確認します。

Probe のインストールは，133 ページ「Java Agent のインストールの確認」に説明されているように，システムの状況モニタを使用して確認されます。

12 追加の Probe 設定を実行します。

環境に適用する Probe の詳細設定を確認します。詳細については，第 15 章「Java Probe およびアプリケーション・サーバの詳細設定」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

このセクションでは，Diagnostics インストール・ディスクに含まれる .tar 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 に Java Agent をインストールする際の重要な検討事項

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。

z/OS メインフレームでのプロパティ・ファイルの編集

z/OS 環境をインストールすると，Probe は，Diagnostics プロパティ・ファイルが 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。

z/OS へのシステム・ログの表示

SDSF でプライマリ・オペレータのコンソールにアクセスして，システム・ログを表示できます。

z/OS メインフレームでの測定値のキャプチャ

負荷テスト測定値は，z/OS に対してキャプチャされません。Probe は，システム・レベル測定値の制限数をキャプチャするように設定できます。

z/OS におけるシステム測定値のキャプチャについては，532 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 tar ファイルは，Diagnostics インストール・ディスクにあります。

z/OS メインフレームに Java Agent をインストールするには

1 HP Diagnostics インストール・ディスクの **Diagnostics_Installers** フォルダから，インストーラを解凍する z/OS マシンのディレクトリに **JavaAgentInstall_zOS.tar.gz** をアップロードします。

2 次の例のように gzip を使って **JavaAgentInstall_zOS.tar.gz** を解凍します。

```
gzip -d JavaAgentInstall_zOS.tar.gz
```

このコマンドでは，解凍ファイル **JavaAgentInstall_zOS.tar** が作成されます。

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

```
tar -xvf JavaAgentInstall_zOS.tar
```

このコマンドでは，解凍ディレクトリ **JavaAgentInstall** が作成されます。

4 スクリプトが実行できるようにスクリプトに権限を設定して，**jreinstrumenter.sh** の実行準備をします（o+x）。

5 JRE Instrumenter を実行します。z/OS では，次のコマンドを使用します。

```
＜ Java へのパス＞ -jar ＜ Java Instrumenter へのパス＞ -b ＜ JVM のディレクトリ＞
```

説明：

- ＜ Java へのパス＞ は，Java 実行ファイルへのパスです。
- ＜ Java Instrumenter へのパス＞ は，**＜ Probe のインストール・ディレクトリ＞ /lib/jreinstrumenter.jar** です。
- ＜ JVM のディレクトリ＞ は，アプリケーションで使用している Java インストールの JRE へのパスです。

6 135 ページ「Java Agent の詳細設定について」の説明に従って，Probe がほかの Diagnostics コンポーネントと連携して動作するように Probe プロパティを設定します。

7 Probe をインストールして JRE をインストゥルメントしたら，Probe と監視対象アプリケーションが連携して起動するように，アプリケーションの起動スクリプトを手動で修正する必要があります。詳細については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」を参照してください。

8 133 ページ「Java Agent のインストールの確認」の説明に従って，Probe のインストールを確認します。

9 z/OS のシステム測定値を収集する場合は，532 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。

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 インプリメンテーションの 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必要があることがあります。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

重要：このセクションは，Java Agent を Diagnostics Server にレポートする Java Probe として設定する場合にのみ適用されます。

Java Agent のインストーラは，コンポーネントが認定されている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ています。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て，ほかの Diagnostics コンポーネントと通信して，アプルケーションの処理を監視できるように Agent を Java Probe として手動で設定する必要があります。

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

1 HP Diagnostics インストール・ディスクの **Diagnostics_Installers** フォルダから **JavaAgentInstall.tar.gz** を見つけます。

2 次の例のように gzip を使って **JavaAgentInstall.tar.gz** を解凍します。

```
gzip -d JavaAgentInstall.tar.gz
```

このコマンドが完了すると，**JavaAgentInstall.tar** という解凍ファイルが作成されます。

3 tar ファイルを解凍するには，次の例のように tar コマンドを実行します。

```
tar -xvf JavaAgentInstall.tar
```

このコマンドでは，解凍ディレクトリ **JavaAgentInstall** が作成されます。

4 135 ページ「Java Agent の詳細設定について」の説明に従って，Probe がほかの Diagnostics コンポーネントと連携して動作するように Probe を設定します。

5 Probe が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・サーバを設定します。

a 137 ページ「JRE Instrumenter の実行」の説明に従って，JRE Instrumenter を実行します。

b Probe をインストールして JRE をインストゥルメントしたら，Probe と監視対象アプリケーションが連携して起動するように，アプリケーションの起動スクリプトを手動で修正する必要があります。詳細については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」を参照してください。

6 133 ページ「Java Agent のインストールの確認」の説明に従って，Probe のインストールを確認します。

Java Agent のサイレント・インストール

サイレント・インストールとは，ユーザが操作することなく自動的に実行されるインストールです。ユーザの入力の代わりに，サイレント・インストールでは，各インストール手順の応答ファイルからの入力を受け入れます。

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。

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

サイレント・インストールでは 2 つの応答ファイルを使用します。1 つは Java Agent のインストール用で，1 つは Java Agent Setup Module 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 ＜ installResponseFileName ＞
```

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 Setup Module 用に応答ファイルを作成するには

➤ 次のコマンド・ライン・オプションを使って Java Agent Setup Module を実行します。

Windows の場合：

```
＜ Probe のインストール・ディレクトリ＞ ¥bin¥setupModule.cmd  -
createBackups  -console  -recordFile ＜ JASMResponseFileName ＞
```

UNIX の場合：

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

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

➤ 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。

まず環境変数を設定し，次のようにコマンド・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
＜インストーラ＞ -options ＜ installResponseFileName ＞ -silent
```

UNIX システムでは，環境変数を指定する場合に引用符で囲む必要があります。

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

Java Agent Setup Module を使ってサイレント・インストールを実行するには

➤ 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。

まず環境変数を設定し，次のようにコマンド・ライン・オプション **-silent** を使って Java Agent Setup Module を実行します。

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

UNIX システムでは，環境変数を指定する場合に引用符で囲む必要があります。

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

アプリケーション・サーバの設定について

Probe を使ってアプリケーションを監視するには，いくつかの JRE バージョン用にインストゥルメントする JRE を設定して，アプリケーション・サーバの起動スクリプトを手動で修正し，いくつかのアプリケーション・サーバ用に SOAP メッセージ・ハンドラを設定する必要があります。

JRE が 1.5 以降の場合，JRE のインストゥルメントをスキップできます。1.5 よりも前のバージョンの JRE では，JRE を手動で設定する必要があります。

JRE のインストゥルメント

インストゥルメンテーション・プロセスは，アプリケーションで監視する Probe，クラスおよびメソッドに定義されます。また，これは生成されるパフォーマンス測定値の報告に使われるレイヤも制御します。

JRE Instrumeter を使ってインストゥルメンテーションを実行します。JRE Instrumenter は，ユーザが指示しない限り，Probe のインストール終了時に自動的に実行してアプリケーションのインストゥルメンテーションの準備を行います。

Probe のインストール中に JRE Instrumenter の実行を省略した場合，Probe を使ってアプリケーションを監視する前に JRE Instrumenter を手動で実行する必要があります。詳細については，第 6 章「JRE Instrumenter の実行」を参照してください。

アプリケーションの起動スクリプトの修正

Probe をインストールして JRE をインストゥルメントしたら，Probe と監視対象アプリケーションが一緒に起動するように，アプリケーションの起動スクリプトを手動で修正する必要があります。

アプリケーション・サーバの起動スクリプトを修正する方法については，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」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，Probe 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，インストゥルメンテーション・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。

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，第 7 章「Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定」を参照してください。

1 台のマシンに複数の JVM がある場合の Java Agent とアプリケーション・サーバの設定プロセスについては，383 ページ「複数のアプリケーション・サーバの JVM インスタンスで使用するための Probe の設定」で説明します。

Java Agent のインストールの確認

システムの状況モニタを使って，Probe のインストールを確認できます。このセクションは，Diagnostics Server と連携して動作するように Probe をインストールした場合だけ必要になります。

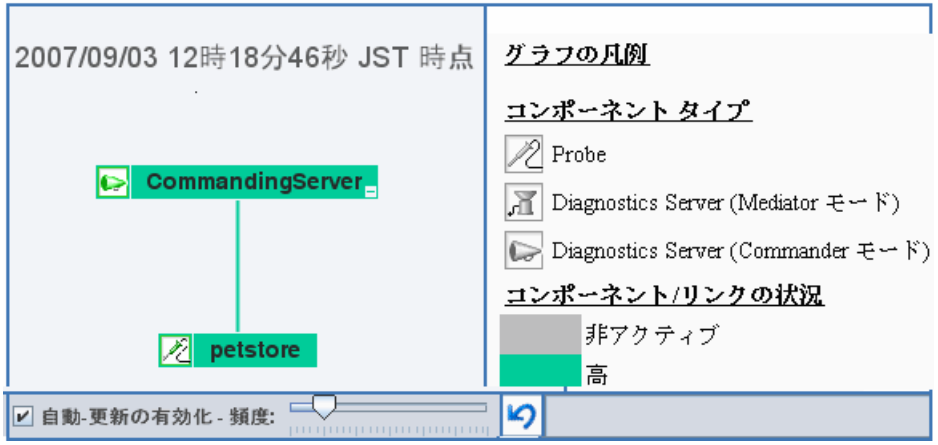
Probe は，起動するまで Diagnostics Server に登録されません。Probe は，インストゥルメント対象のアプリケーション・サーバが起動したときに立ち上がります。したがって，Probe とアプリケーション・サーバを設定しないと，システムの状況モニタを使って Probe のインストールを確認できません。

推奨インストール手順に従った場合，Java Agent のインストール後，以下を確認できます。

- Diagnostics Server（Commander モード）が正常にインストールされている。
- 追加の Diagnostics Server が正常にインストールされ，Diagnostics Server（CommandingServer）と通信している。
- Java Agent が正常にインストールされていて，Diagnostics Server との接続を確立している。

  Java Agent は，起動するまで Probe が Diagnostics Server に登録されないため，初めてインストールしたときにシステムの状況に表示されません。測定対象のアプリケーション・サーバが起動すると，Probe が立ち上がって Diagnostics Server に登録されます。

デプロイメントに応じて，新しい Java Agent は，システムの状況モニタに Diagnostics Server（Commander モード）または Diagnostics Server（Mediator モード）の子として表示されます。



Java Agent は，監視対象のアプリケーションが終了したときにシステムの状況モニタに灰色で表示されます。アプリケーションが起動すると，Probe は緑色になります。

システムの状況モニタの詳しい使い方については，付録 D「System Health Monitor の使用」を参照してください。

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。

Probe 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 **＜ Probe 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。
- Probe ログ・ファイル（**＜ Probe 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。
- バージョン番号は，システムの状況モニタの Probe の詳細の中にあります。

Java Agent のアンインストール

Java Probe をアンインストールするには

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

  または，**＜ Probe 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，**＜ Probe 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 ＜ Probe のインストール・ディレクトリ＞ /_uninst/uninstall.jar** を実行して Java Probe をアンインストールします。

Java Agent の詳細設定について

第 15 章「Java Probe およびアプリケーション・サーバの詳細設定」の内容を見直し，お使いの環境で Probe の動作の効率を高める Probe の構成設定があるかどうかを確認してください。

Java アプリケーションのカスタム・インストゥルメンテーションについて

さらに，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・インストゥルメンテーションポイントを作成できます。カスタム・インストゥルメンテーションの一般的な情報については，第 9 章「Java アプリーションのカスタム・インストゥルメンテーション」の各トピックを確認してください。

第 6 章

JRE Instrumenter の実行

JRE Instrumenter を実行して，Java Probe でアプリケーションを監視できるようにする方法について説明します。

本章の内容

JRE Instrumenter について

Java Probe をインストールする際，JRE Instrumenter を実行して，Probe で監視するアプリケーションのパフォーマンス・インストゥルメント値を収集できるようにする必要があります。JRE Instrumenter は，アプリケーションで使用している JVM の ClassLoader クラスをインストゥルメントし，＜ Probe 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ゥルメントされた ClassLoader を置きます。また，アプリケーション・サーバでインストゥルメントされたクラス・ローダ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，ユーザが実行しないように指示しない限り，Probe のインストール時に自動的に実行します。Probe のインストール時に JRE Instrumenter を省略した場合は，手動で JRE Instrumenter を実行する必要があります。

また，Probe で処理の監視を継続できるように，アプリケーション・サーバで使用される JDK（java.exe 実行ファイル）が変更された場合，JRE Instrumenter を再度実行する必要があります。

注：

- IBM の 1.4.2 J9 JRE をインストゥルメントする場合，適切な ClassLoader をインストゥルメントして，アプリケーションのコマンド・ラインで -Xj9 オプションを追加する必要があります。適切な ClassLoader は，**＜ Java ディレクトリ＞ ¥jre¥lib¥jclSC14** ディレクトリ（たとえば jreinstrumenter.sh -i ¥usr¥java14_64¥jre¥lib¥jclSC14）にあります。
- Probe が複数の JVM の監視に使われている場合，各 JVM で実行中のアプリケーションを Probe がインストゥルメントできるようにするために，各 JVM に対して JRE Instrumenter を一度実行する必要があります。詳細については，383 ページ「複数のアプリケーション・サーバの JVM インスタンスで使用するための Probe の設定」を参照してください。

JRE Instrumenter の機能

JRE Instrumenter は，次の機能を実行します。

- インストゥルメント可能な JVM を特定する。
- 指定したディレクトリで，追加 JVM を検索する。
- 指定した JVM をインストゥルメントし，インストゥルメントされた ClassLoader クラスの場所を示すために JVM 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 は，実行方法に応じて，インストゥルメントされた ClassLoader を別の場所に置きます。

  - Instrumenter を Probe インストーラから実行した場合，Instrumenter は，インストゥルメントした ClassLoader を **＜ Probe のインストール・ディレクトリ＞ /classes/boot** ディレクトリの配下のフォルダに置きます。
  - Instrumenter が Windows または UNIX 環境のグラフィカル・インタフェースを使って実行した場合，Instrumenter は，インストゥルメントした ClassLoader を **＜ Probe のインストール・ディレクトリ＞ /classes/ ＜ JVM_vendor ＞ / ＜ JVM_version ＞** ディレクトリの配下のフォルダに置きます。

- Instrumenter をコンソール・モードの UNIX 環境で実行した場合，Instrumenter は，指定した処理オプションに応じて，**＜ Probe のインストール・ディレクトリ＞ /classes/boot** ディレクトリまたは **＜ Probe のインストール・ディレクトリ＞ /classes/ ＜ JVM_vendor ＞ / ＜ JVM_version ＞** ディレクトリの配下のいずれかのフォルダにインストゥルメントした ClassLoader を置きます。UNIX の処理オプションについては，146 ページ「リストに表示された JVM のインストゥルメント」を参照してください。

JRE Instrumenter の実行

以下は，Windows 環境およびコンソール・モードの UNIX 環境で，JRE Instrumenter を実行するための手順です。

注： Probe のインストール中に JRE Instrumenter を実行しないように選択した場合，手動で実行し，Probe でアプリケーションのパフォーマンス測定値を収集できるようにする必要があります。

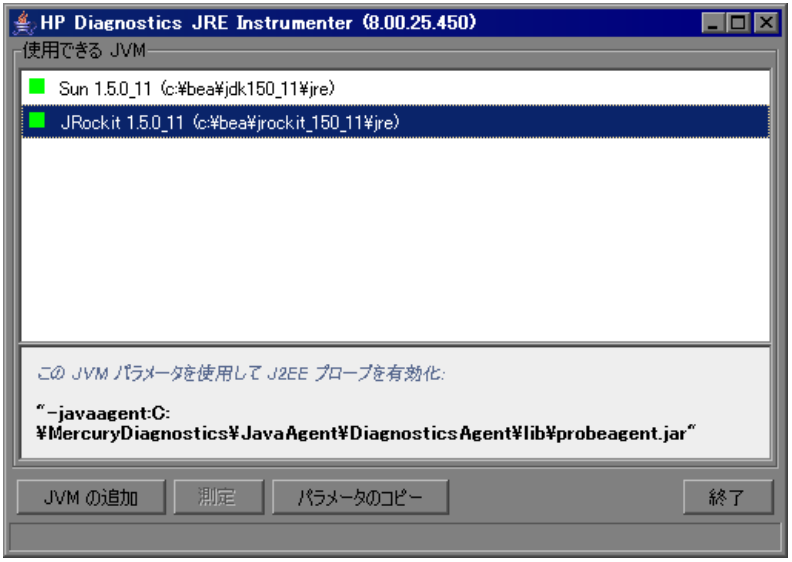
本項の内容

Windows マシンでの JRE Instrumenter の実行

JRE Instrumenter を Windows 環境で実行すると，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。Instrumenter をコンソール・モードで実行していない場合に，インストーラを UNIX マシンで実行したときも同じダイアログが表示されます。

Windows マシンでの JRE Instrumenter の起動

＜Probe 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。コマンド **jreinstrumenter.cmd** を実行します。Instrumenter が起動すると，［HP Diagnostics JRE Instrumenter］ダイアログが表示されます。



Instrumenter には，Instrumenter が検出した JVM が一覧表示され，インストゥルメンテーションに使用できます。インストゥルメントされた JVM には，JVM の名前の前に緑色の四角形が付きます。

［HP Diagnostics JRE Instrumenter］ダイアログから，以下のタスクを実行できます。

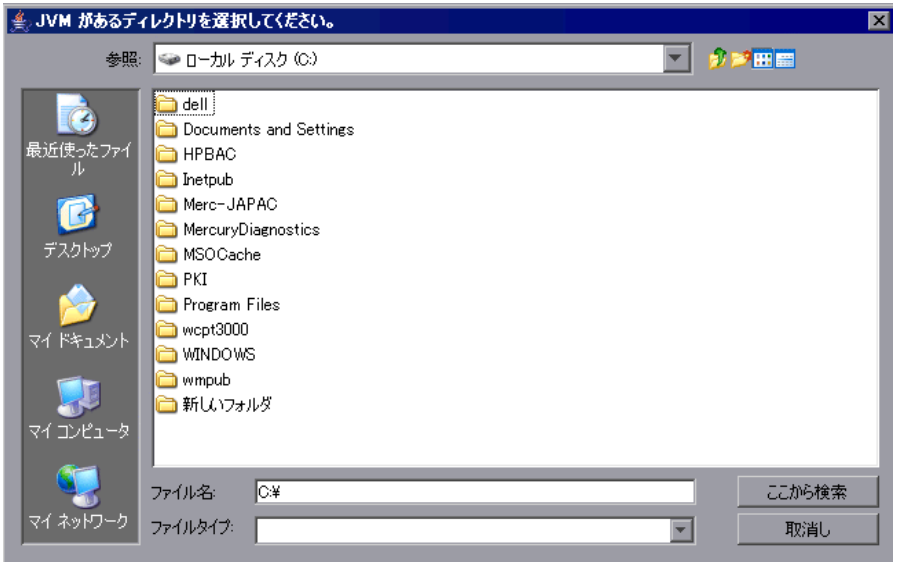
- ダイアログの［使用できる JVM］リストにインストゥルメントする JVM がない場合，141 ページ「使用可能な JVM リストへの JVM の追加」の説明に従ってリストに JVM を追加できます。
- リストにインストゥルメントする JVM があるものの，インストゥルメントされていない場合は，144 ページ「選択した JVM のインストゥルメント」の説明に従って JVM をインストゥルメントできます。

- リストにインストゥルメントする JVM があり，すでにインストゥルメントされている場合，144 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」の説明に従って，JVM の起動スクリプトに挿入する必要のある JVM パラメータをコピーして，Probe の監視をアクティブにできます。
- JRE Instrumenter の使用が完了したら，［**終了**］をクリックして，JRE Instrumenter を閉じます。

使用可能な JVM リストへの JVM の追加

1 ［JRE Instrumenter］ダイアログで，［**JVM の追加**］をクリックしてほかの JVM を検索し，［使用できる JVM］リストに追加します。

Instrumenter に［JVM があるディレクトリを選択してください］ダイアログ・ボックスが表示されます。



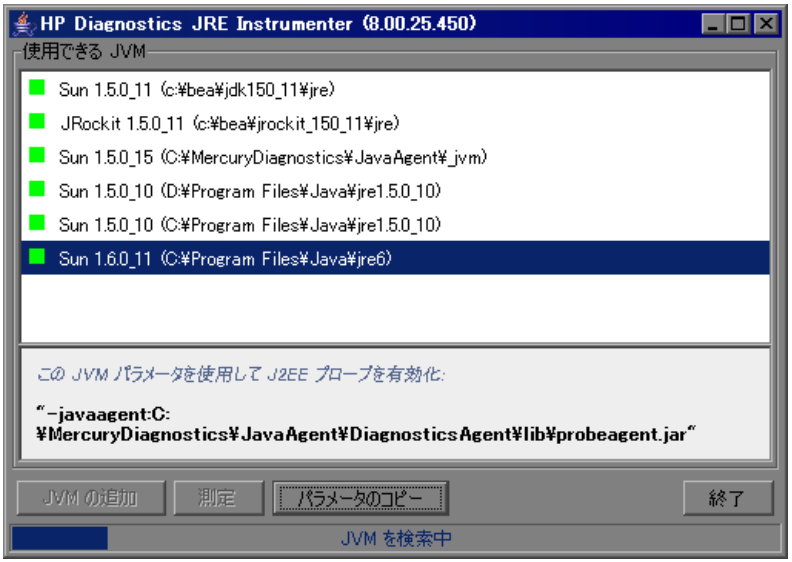
2 標準の Windows エクスプローラ型のナビゲーション・コンソールを使って，Instrumenter で JVM の検索を開始するディレクトリの場所に移動します。

3 ［**ファイル名**］ボックスに名前が表示されるように，検索を開始するファイルを選択します。

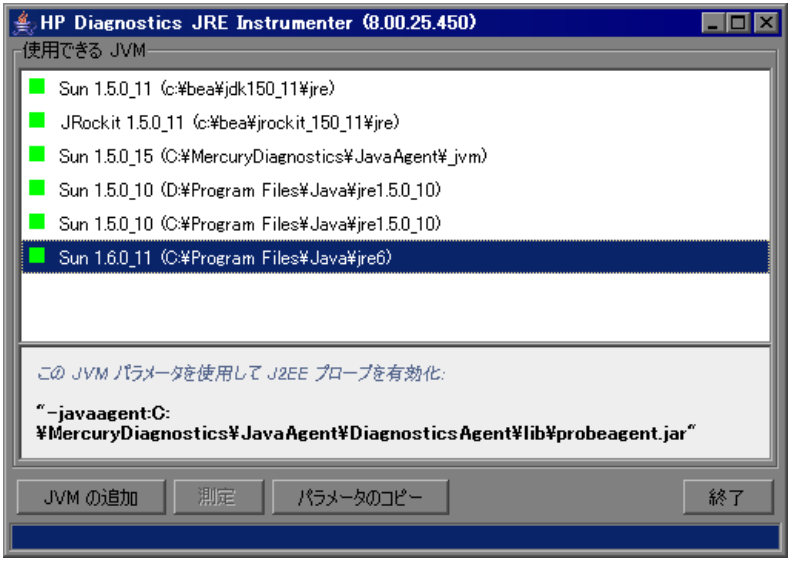
4［**ここから検索**］をクリックして，JVM の検索を開始します。

Instrumenter はダイアログ・ボックスを閉じ，もう一度［JRE Instrumenter］ダイアログ・ボックスを表示します。ダイアログのコマンド・ボタンは無効になり，Instrumenter は JVM を検索します。ダイアログ下部の進行状況バーに，Instrumenter が検索中であることと，検索プロセスに要する時間が表示されます。

ツールで JVM を見つかると，［**使用できる JVM**］リストに表示されます。

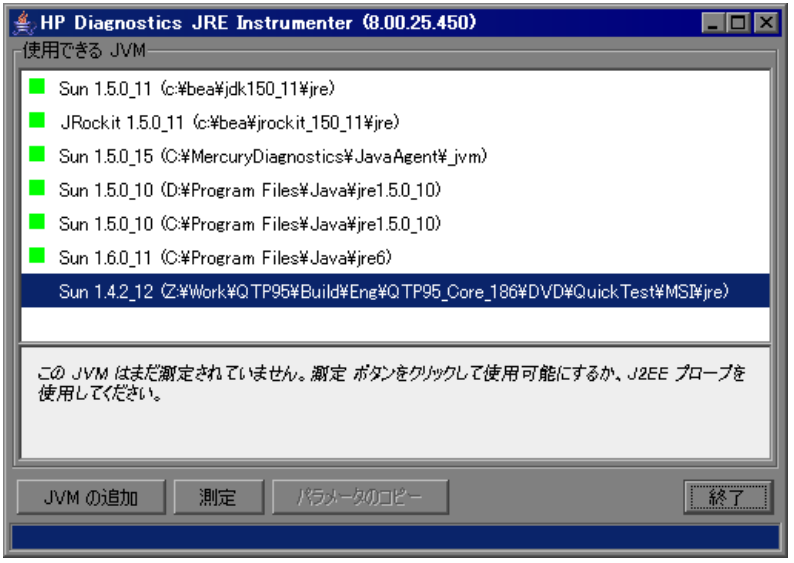


Instrumenter が検索を完了すると，ダイアログのコマンド・ボタンが有効になり，次のように表示されます。選択した行がインストゥルメントされた JVM の場合，［測定］ボタンが無効になります。



選択した JVM のインストゥルメント

［**使用できる JVM**］リストからインストゥルメントされていない JVM を選択し，［**測定**］をクリックします。



JRE Instrumenter は，選択した JVM の ClassLoader クラスをインストゥルメントし，＜ **Probe 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ゥルメントされた ClassLoader を置きます。また，［使用できる JVM］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。

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 をインストゥルメントする際，アプリケーションでインストゥルメントしたクラス・ローダを使うために，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。［使用できる JVM］リストからインストゥルメントされた JVM 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。

このボックスに表示された JVM パラメータをクリップボードにコピーするには，［**パラメータのコピー**］をクリックします。JVM パラメータがクリップボードにコピーされ，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えきます。

UNIX マシンでの JRE Instrumenter の実行

ここでは，グラフィック・モード・インストールまたはコンソール・モード・インストールを使って，JRE Instrumenter を実行するのに必要な手順を紹介します。

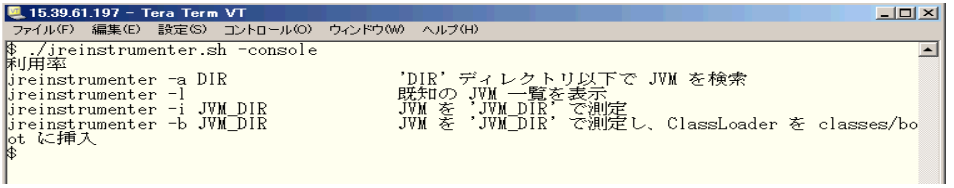
グラフィック・モードに表示される JRE Instrumenter 画面は，139 ページ「Windows マシンでの JRE Instrumenter の実行」で Windows インストーラに使用されているものと同じです。

UNIX マシンでの JRE Instrumenter の起動

＜ Probe 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 が起動すると，使用可能な処理オプションのリストが表示されます。



jreinstrumenter.sh コマンドを実行する際，-x オプションを指定して，オプションのリストを再表示できます。

```
./jreinstrumenter.sh -x
```

次の表は，各処理オプションの参照先を示します。

Instrumenter 機能	参照先
jreinstrumenter -l	146 ページ「インストゥルメント対象の JVM リストの表示」
jreinstrumenter -a DIR	146 ページ「使用可能な JVM リストへの JVM の追加」
jreinstrumenter -i JVM_DIR jreinstrumenter -b JVM_DIR	146 ページ「リストに表示された JVM のインストゥルメント」

インストゥルメント対象の JVM リストの表示

JRE Instrumenter の既知の JVM のリストを表示するには，次のコマンドを入力します。

```
./jreinstrumenter.sh -l
```

Instrumenter は，JVM ベンダ，JVM バージョンおよび JVM が置かれている場所を含め，行に JVM を一覧表示します。

使用可能な JVM リストへの JVM の追加

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 の既知の JVM のリストに追加するには，次のコマンドを入力します。

```
./jreinstrumenter.sh -a DIR
```

DIR を，Instrumenter で検索を開始する場所へのパスに置き換えます。

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。

リストに表示された JVM のインストゥルメント

［使用可能な JVM］リストに表示された JVM をインストゥルメントするには，次のいずれかのコマンドを使用します。

➤ ClassLoader への特定的なパス

```
./jreinstrumenter.sh -i JVM_DIR
```

JVM_DIR を，［使用可能な JVM］リストで指定された JVM の場所へのパスと置き換えます。

このコマンドは，JRE Instrumenter が，選択した JVM の ClassLoader クラスをインストゥルメントするように指示し，**＜ Probe のインストール・ディレクトリ＞ /classes/ ＜ JVM_vendor ＞ / ＜ JVM_version ＞** ディレクトリの下のフォルダにインストゥルメントされた ClassLoader を置きます。

このコマンドは，特に 1 台の Probe で監視する複数の JVM をインストゥルメントする場合に必ず使用します。

➤ ClassLoader への一般的なパス

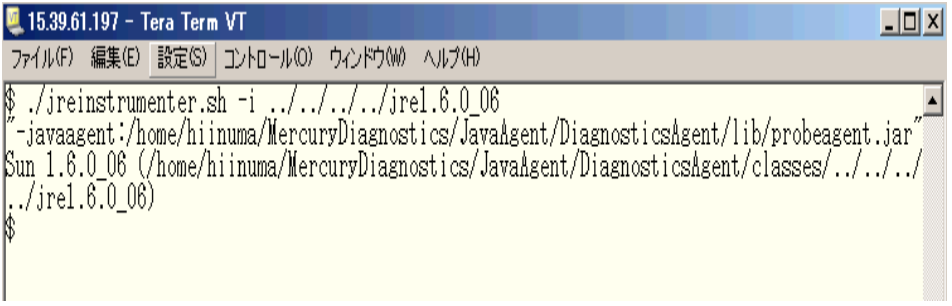
```
./jreinstrumenter.sh -b JVM_DIR
```

JVM_DIR を，［使用可能な JVM］リストで指定された JVM の場所へのパスと置き換えます。

このコマンドは，JRE Instrumenter が，選択した JVM の ClassLoader クラスをインストゥルメントするように指示し，**＜Probe のインストール・ディレクトリ＞ /classes/boot** ディレクトリの下のフォルダにインストゥルメントされた ClassLoader を置きます。

このコマンドは，Probe を使って 1 台の JVM を監視しており，-i コマンド・オプションを使用するときに作成される特定的なパスをそれ以上使いたくない何らかの理由があるときに使用します。

Instrumenter で JVM のインストゥルメントを終了すると，インストルメンテーションのアクティブ化に使う必要のある JVM パラメータが表示され，アプリケーションのパフォーマンスの監視を Probe で有効にします。JVM パラメータに続いて，Instrumenter は，次の例のように［使用可能な JVM］リストを再表示します。



アプリケーション・サーバの起動スクリプトへの JVM パラメータの追加

JRE Instrumenter が JVM をインストゥルメントする際，アプリケーションでインストゥルメントしたクラス・ローダを使うために，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。Instrumenter が JVM のインストゥ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。

第 7 章

Java Agent と連携して動作する アプリケーション・サーバ起動スクリプトの設定

アプリケーション・サーバを設定して，プローブでアプリケーションを監視できるようにする方法について説明します。

本章の内容

アプリケーション・サーバの起動スクリプトの設定について

Java Probe で JRE Instrumenter を実行したら，アプリケーションを監視する Probe がアプリケーションの起動時に立ち上がるように，アプリケーションの起動スクリプトを修正する必要があります。

アプリケーション・サーバの起動スクリプトを手動で更新して，アプリケーション・サーバを設定できます。次のセクションでは，アプリケーション・サーバを手動で更新する方法を説明します。

重要：特定のバージョンのアプリケーション・サーバに対する手順の例を示します。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，Diagnostics の製品可用性マトリックス（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。

サイト管理者は，設定を変更するためのサイト独自の手法を持つことができます。その場合，182 ページ「一般的なアプリケーション・サーバの設定」の一般的な手順を使用してください。サイト管理者が必要な設定の変更を行うための情報が記載されています。

注：次のセクションに，お使いのアプリケーション・サーバに適した手順がない場合は，182 ページ「一般的なアプリケーション・サーバの設定」の手順に従ってください。

---

1 台のマシンに複数の JVM がある場合の Java Probe とアプリケーション・サーバの設定プロセスについては，383 ページ「複数のアプリケーション・サーバの JVM インスタンスで使用するための Probe の設定」で説明します。

注：

- **＜プローブのインストール・ディレクトリ＞** を使って，Java Probe がインストールされているディレクトリを指定します。
- **-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必ず使ってください。

WebSphere アプリケーション・サーバの設定

WebSphere サーバは，WebSphere アプリケーション・サーバ管理コンソールを使って制御します。このコンソールでは，JVM コマンド・ラインを制御し，ユーザがクラスパス要素を追加したり，ランタイム変数（**-D** 変数）を定義したり，WebSphere の bootclasspath を設定できます。管理コンソールを使って，Xbootclasspath プロパティと，JVM コマンド・ラインで必要な追加引数を追加します。

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。WebSphere の各バージョンに，若干の変更が加えられることがあります。したがって，このセクションの例は，お使いの WebSphere のバージョンと完全に一致しない場合があります。これらの例では，コンソールの適切な場所で必要なパラメータを入力する必要のある情報を示します。

注：WebSphere は，タブの［**適用**］をクリックした場合のみ，各タブで加えた変更を適用します。［**適用**］をクリックするまで，変更は適用されません。

このセクションには，以下の例があります。

- WebSphere 5.x および 6.0
- WebSphere 6.1
- WebSphere IDE での JRE Instrumenter の実行

WebSphere 5.x および 6.0

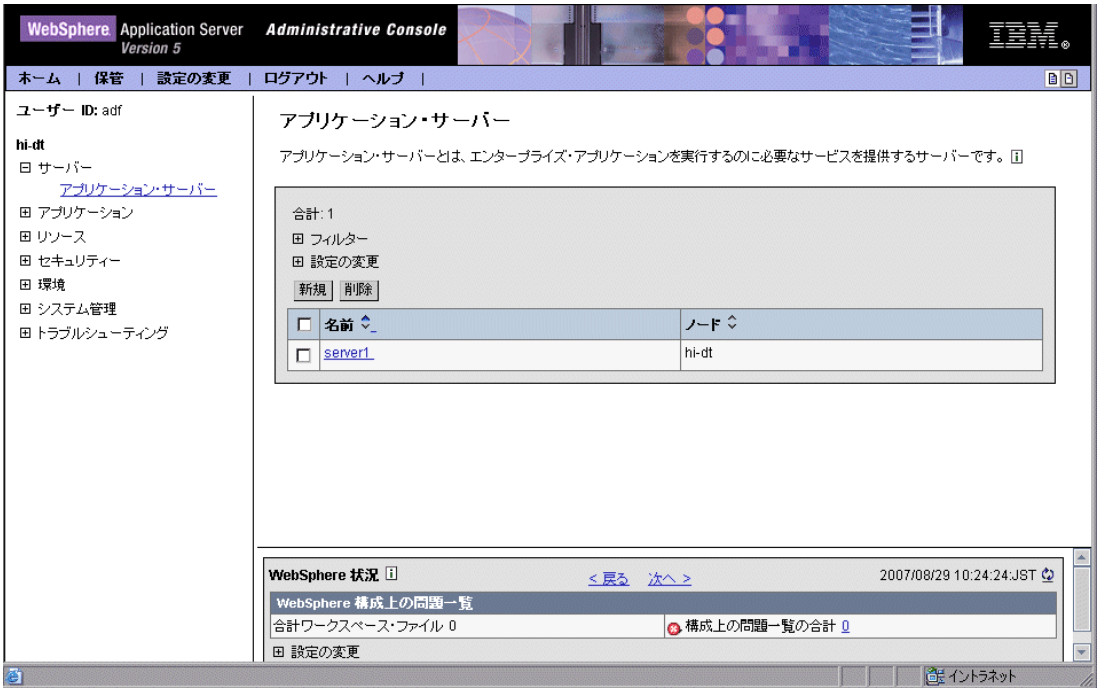
WebSphere 5.x および 6.0 アプリケーション・サーバを設定するには

1 Web ブラウザを使って，Probe がインストール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。

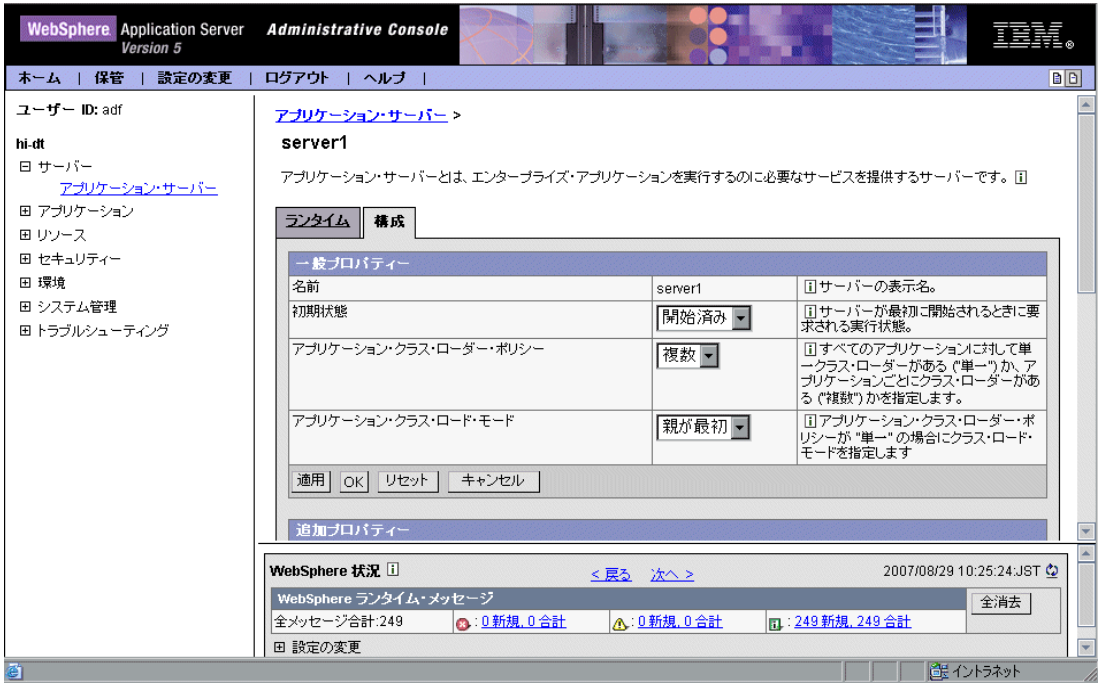
WebSphere アプリケーション・サーバ管理コンソールが開きます。



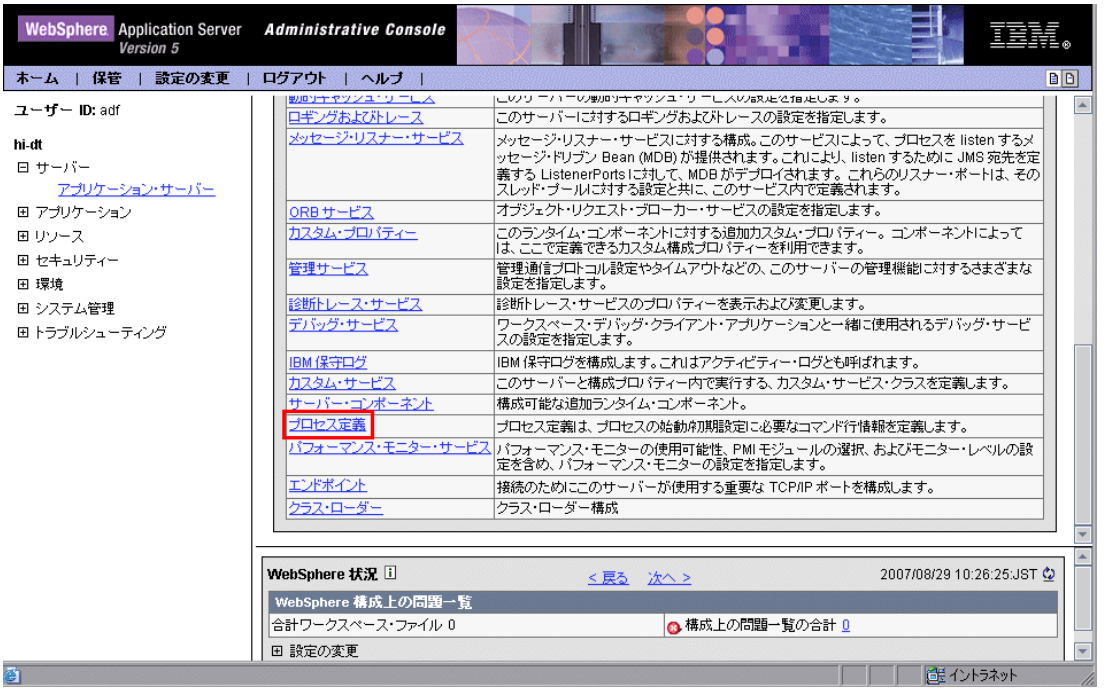
2 左のパネルで，［**サーバー**］＞［**アプリケーション・サーバー**］を選択します。

3 右のパネルのアプリケーション・サーバのリストから，Probe 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［構成］タブが表示されます。

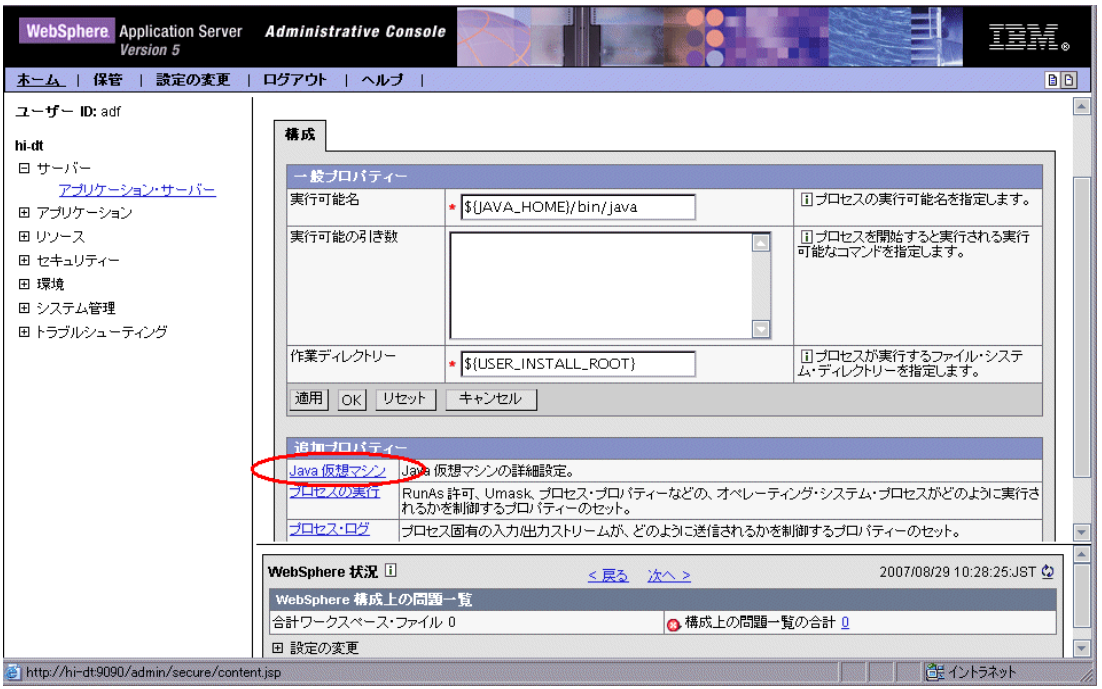


4［構成］タブを下にスクロールし，［**一般プロパティー**］列で，［**プロセス定義**］プロパティを探します。



5［**プロセス定義**］をクリックします。

6 右のパネルを下にスクロールして，**Java 仮想マシン**を探します。



7［**Java 仮想マシン**］をクリックします。

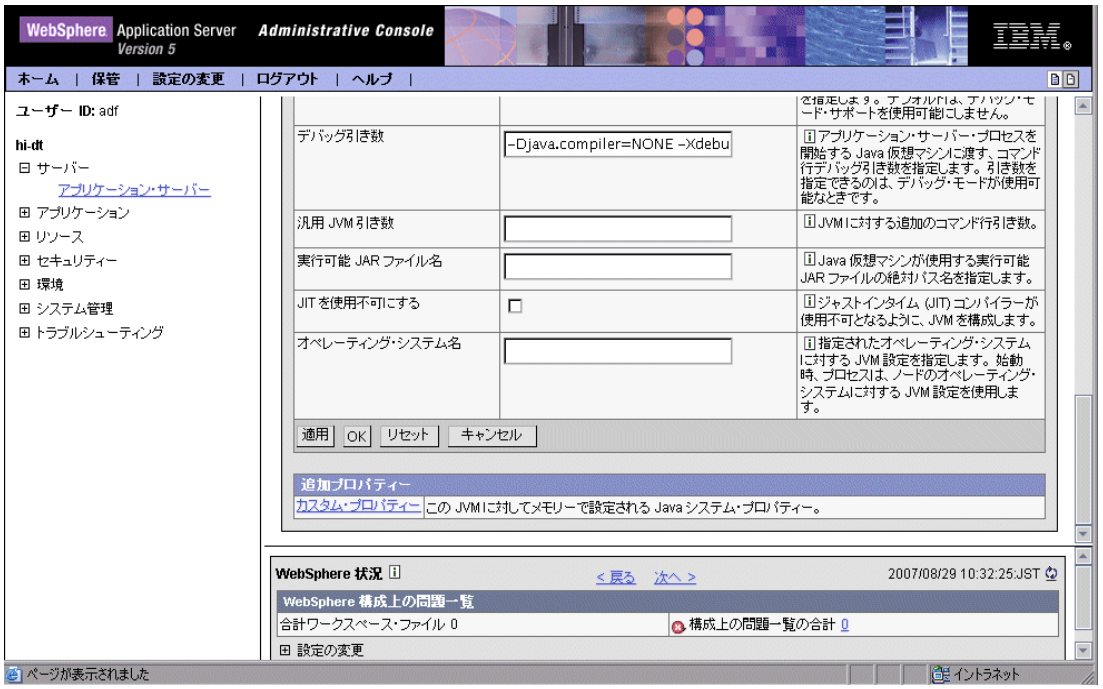
8 Java 仮想マシンの［構成］タブが表示されます。

9［**ブート・クラスパス**］ボックスで，Probe のブート・ディレクトリへのパスを次のように入力します。

```
＜Probe のインストール・ディレクトリ＞¥classes¥IBM¥1.4.2_06;
＜Probe のインストール・ディレクトリ＞¥classes¥boot
```

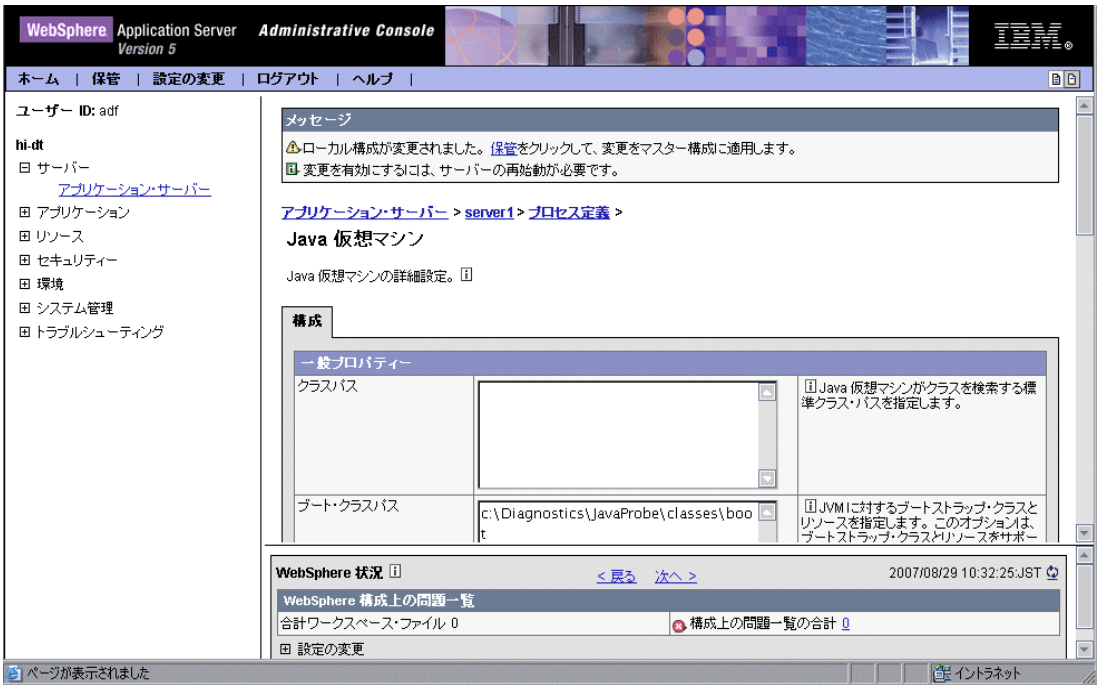
＜**Probe のインストール・ディレクトリ**＞は，Probe がインストールされている場所へのパスです。

10 コマンド・ボタンが表示されるまで［構成］タブを一番下にスクロールします。



［**適用**］をクリックします。

11 変更が適用された旨を伝えるメッセージが表示されます。



［**保管**］をクリックして，マスター設定に変更を適用します。

12 ［**マスター構成に保管**］エリアで，［**保管**］をクリックします。

13 WebSphere アプリケーション・サーバを再起動します。アプリケーション・サーバのホストは再起動する必要がありません。

14 Probe が正しく設定されていることを確認するには，**＜ Probe のインストール・ディレクトリ＞ ¥log¥ ＜ Probe ID ＞ .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** を正しく入力しなかったかのいずれかです。JRE Instrumenter の実行については，137 ページ「JRE Instrumenter の実行」を参照してください。

WebSphere 6.1

WebSphere 6.1 アプリケーション・サーバを設定するには

1 WebSphere アプリケーション・サーバ管理コンソールを開きます。

2 次のように［Java 仮想マシン］ページを開きます。

［アプリケーション・サーバ］＞［アプリケーション・サーバ・インスタンス名（server1 など）］＞［プロセス定義］＞［Java 仮想マシン］ページ

3 ［Java 仮想マシン］ページの［**汎用 JVM 引数**］ボックスに，次の JVM パラメータを入力します。

```
-javaagent: ＜ Probe のインストール・ディレクトリ＞
¥lib¥probeagent.jar
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされている場所へのパスです。

4 変更を適用して保存します。

5 WebSphere アプリケーション・サーバを再起動します。アプリケーション・サーバのホストは再起動する必要がありません。

6 Probe が正しく設定されていることを確認するには，**＜ Probe 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，Java Probe が正しく起動していないことを示します。

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ゥ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，10 の異なる java.exe 実行可能ファイルがあり選択できます。IDE の実行に使用する実行可能ファイルをインストゥルメントする必要があります。

正しい java.exe をインストゥルメントするには

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。下の表を参照してください。

3 137 ページ「JRE Instrumenter の実行」の説明に従って，JRE Instrumenter を実行します。

バージョン	実行可能ファイル
WAS 5.0	IDE INSTALL¥runtimes¥base_v5¥java¥bin¥java.exe
WAS 5.1	IDE INSTALL¥runtimes¥base_v51¥java¥bin¥java.exe

注：

- データをキャプチャする前に，お使いの IDE の命令ごとに，アプリケーションの JVM 起動パラメータに対して **Xbootclasspath** を修正する必要があります。
- IDE 内でアプリケーションを実行していない場合，次の手順に従って，WebSphere 管理コンソールを使って JVM パラメータを設定します。

WebLogic アプリケーション・サーバの設定

WegLogic アプリケーション・サーバは，アプリケーション・サーバの起動に使われるスクリプトに Xbootclasspath プロパティまたは JVM パラメータを追加して設定します。WebLogic は，UNIX 環境でシェル・スクリプトを実行するか，または Windows 環境でコマンド・スクリプトを実行して起動します。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，次のセクションでは，一般的なインプリメンテーションのための WebLogic アプリケーション・サーバの各認定バージョンに対する手順を紹介します。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。

注：Probe 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，および環境変数の使い方を理解しておいてください。変更を行う間に，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。

このセクションには，以下の例があります。

- WebLogic 8.1
- WebLogic 9.x および 10.0

WebLogic 8.1

Sun JVM に WebLogic 8.1 アプリケーション・サーバを設定するには

1 JRE Instrumenter を実行して，WebLogic が実行中の Sun JVM を追加します。

2 JVM を追加したら，［**パラメータのコピー**］ボタンをクリックします。クリップボードに **Xbootclasspath** パラメータがコピーされます。次に例を示します。

```
JAVA_OPTIONS="-Xbootclasspath/p: ＜ Probe のインストール・ディレクトリ＞¥classes¥Sun¥1.4.2_04; ＜ Probe のインストール・ディレクトリ＞¥classes¥boot"
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされているディレクトリへのパスです。

3 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。通常，このファイルは，次の例のようなパスにあります。

```
D:¥bea¥weblogic81¥config¥ ＜ドメイン名＞ ¥start ＜ドメイン名＞ .cmd
```

＜ドメイン名＞ をアプリケーションを起動するスクリプトの名前に置き換えます。

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。

```
D:¥bea¥weblogic81¥samples¥domains¥medrec¥startMedRecServer.cmd
```

4 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。

5 エディタを使って起動スクリプトを開きます。

6 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，クリップボードに保存されている **Xbootclasspath** パラメータを追加します。パラメータは，**-hotspot** または **-classic** などの JIT オプションに続いて，Java パラメータの先頭で置き換える必要があります。

以下は，**Xbootclasspath** パラメータを追加する前の WebLogic 起動スクリプトの例です。

```
"%JAVA_HOME%¥bin¥java" -hotspot -ms64m -mx64m"-Xbootclasspath/p:
＜ Probe のインストール・ディレクトリ＞¥
classes¥Sun¥1.3.1_04; ＜ Probe のインストール・ディレクトリ＞¥
classes¥boot" -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¥weblogic81/lib/weblogic.policy" weblogic.Server
```

注：起動スクリプトの例には，改行が入っています。しかし，実際のスクリプトに改行はなく，必要に応じてコマンドのテキストが画面上でワードラップします。

以下は，**Xbootclasspath** パラメータを追加した後の WebLogic 起動スクリプトの例です。

```
"%JAVA_HOME%¥bin¥java" -hotspot -ms64m -mx64m -
Xbootclasspath/p:"C:¥Program Files¥HP¥common¥JavaProbe¥classes¥boot"-
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¥weblogic81/lib/weblogic.policy"
weblogic.Server
```

7 起動スクリプトに変更を保存します。

8 WebLogic アプリケーション・サーバを再起動します。アプリケーション・サーバのホスト・マシンは再起動する必要がありません。

9 Probe が正しく設定されていることを確認するには，**＜ Probe 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 Instrumenter を実行していないか，**Xbootclasspath** を正しく入力しなかったかのいずれかです。

注：JRE Instrumenter の実行については，137 ページ「JRE Instrumenter の実行」を参照してください。

JRockit JVM に WebLogic 8.1 アプリケーション・サーバを設定するには

1 JRE Instrumenter を実行して，WebLogic が実行中の JRockit JVM を追加します。

2 JVM を追加したら，［**パラメータのコピー**］ボタンをクリックします。クリップボードに **Xbootclasspath** パラメータがコピーされます。

以下は，**Xbootclasspath** パラメータの例です。

```
-Xbootclasspath/p: ＜ Probe のインストール・ディレクトリ＞ ¥classes¥boot
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされているディレクトリへのパスです。

3 WebLogic アプリケーション・サーバを呼び出すコマンド・ファイルを特定します（たとえば，startWLS.cmd）。通常，このファイルは，次の例のようなパスにあります。

```
C:¥bea¥weblogic81¥server¥bin¥ startWLS.cmd
```

4 スクリプトに変更を加える前に，コマンド・ファイルのバックアップ・コピーを作成します。新しいコピーに startWLSWithJRockit.cmd のような名前を指定し，これを次の手順で操作するコマンド・ファイルの新しいバージョンとして使用します。

5 エディタを使って起動スクリプトを開きます。

6 JRockit に，WebLogic によって呼び出される JAVA 実行可能ファイルを設定します。

a **JAVA_VENDOR** パラメータの値が設定されるコマンド・ファイルでこの行を見つけます。

b 次のように，**JAVA_VENDOR** 変数の値を JRockit フォルダへのポイントに変更します。

```
set JAVA_VENDOR=<BEA_HOME_DIR>¥jrockit
```

次に例を示します。

```
set JAVA_VENDOR=BEA
```

7 アプリケーション・サーバを起動する Java コマンド・ラインを変更します。

a 次のように始まるコマンド・ファイルで，この行を見つけます。

```
%JAVA_HOME%¥bin¥java %JAVA_VM% %JAVA_OPTIONS% .....
```

b %JAVA_OPTIONS% 変数の直後に **Xmanagement:class** パラメータを指定して，JRockit 管理 URL を指定します。

以下は，**Xmanagement:class** パラメータの例です。

```
-Xmanagement:class=com.mercury.opal.capture.proxy.JRockitManagement
```

c %JAVA_OPTIONS% 変数の直後に，クリップボードに保存した **Xbootclasspath** パラメータを追加して，アプリケーション・サーバ・プロセスに Probe が接続できるようにします。

以下は，**Xmanagement:class** パラメータと **Xbootclasspath** パラメータを追加する前の WebLogic 起動スクリプトの例です。

```
"%JAVA_HOME%¥bin¥java" %JAVA_VM% %MEM_ARGS%
%JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%-
Dweblogic.management.username=%WLS_USER%-
Dweblogic.management.password=%WLS_PW% -
Dweblogic.management.server=%ADMIN_URL% -
Dweblogic.ProductionModeEnabled=%PRODUCTION_MODE% -
Djava.security.policy="%WL_HOME%¥server¥lib¥weblogic.policy"
weblogic.Server
```

注：起動スクリプトの例には，改行が入っています。しかし，実際のスクリプトに改行はなく，必要に応じてコマンドのテキストが画面上でワードラップします。

以下は，**Xbootclasspath** パラメータを追加した後の WebLogic 起動スクリプトの例です。

```
"%JAVA_HOME%¥bin¥java" %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Xmanagement:class=com.mercury.opal.capture.proxy.JRockitManagement -
Xbootclasspath/p:"C:¥Program Files¥HP¥common¥JavaProbe¥classes¥boot" -
Dweblogic.Name=%SERVER_NAME% -
Dweblogic.management.username=%WLS_USER% -
Dweblogic.management.password=%WLS_PW% -
Dweblogic.management.server=%ADMIN_URL% -
Dweblogic.ProductionModeEnabled=%PRODUCTION_MODE% -
Djava.security.policy="%WL_HOME%¥server¥lib¥weblogic.policy"
weblogic.Server
```

8 コマンド・ファイルに変更を保存します。

9 WebLogic アプリケーション・サーバを再起動します（コンピュータではなく，アプリケーション・サーバだけです)。

10 Probe が正しく設定されていることを確認するには，**＜ Probe 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 Instrumenter を実行していないか，**Xbootclasspath** パラメータを正しく入力しなかったかのいずれかです。

注：JRE Instrumenter の実行については，137 ページ「JRE Instrumenter の実行」を参照してください。

WebLogic 9.x および 10.0

WebLogic 9.0 および 10.0 アプリケーション・サーバを設定するには

1 JRE Instrumenter を実行して，WebLogic 9.x または 10.0 が実行中の JVM を追加します。

2 JVM を追加したら，［**パラメータのコピー**］ボタンをクリックします。クリップボードに JVM パラメータがコピーされます。次に例を示します。

```
JAVA_OPTIONS="-javaagent: ＜ Probe のインストール・ディレクトリ＞
¥lib¥probeagent.jar"
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされているディレクトリへのパスです。

3 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。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。

```
D:¥bea¥wlserver_10.0¥samples¥domains¥medrec¥bin¥startWebLogic.cmd
```

4 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。

5 エディタを使って起動スクリプトを開きます。

6 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，クリップボードに保存されている **JVM** パラメータを追加します。パラメータは，**-hotspot** または **-classic** などの JIT オプションに続いて，Java パラメータの先頭で置き換える必要があります。

注：起動スクリプトの例には，改行が入っています。しかし，実際のスクリプトに改行はなく，必要に応じてコマンドのテキストが画面上でワードラップします。

以下は，**JVM** パラメータを追加した後の WebLogic 起動スクリプトの例です。

```
set JAVA_OPTIONS= "-
javaagent:C:¥MercuryDiagnostics¥JAVAProbe¥lib¥probeagent.jar"
%SAVE_JAVA_OPTIONS%
```

7 起動スクリプトに変更を保存します。

8 WebLogic アプリケーション・サーバを再起動します。アプリケーション・サーバのホスト・マシンは再起動する必要がありません。

9 Probe が正しく設定されていることを確認するには，**＜ Probe 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 Instrumenter を実行していないか，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。

注：JRE Instrumenter の実行については，137 ページ「JRE Instrumenter の実行」を参照してください。

Oracle アプリケーション・サーバの設定

このセクションでは，以下の Oracle 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。

- 169 ページ「Oracle 9i の設定」
- 172 ページ「Oracle 10.1.3 の設定」
- 173 ページ「Oracle 10.1.2 の設定」

Oracle 9i の設定

Oracle9i アプリケーション・サーバは，アプリケーション・サーバの起動に使われる XML ファイルに Xbootclasspath プロパティを追加して設定します。Oracle9i が提供するファイルは，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。代わりに，次のセクションでは，一般的なインプリメンテーションに Oracle9i アプリケーション・サーバを設定するための手順を紹介します。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。

注：Probe 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，および環境変数の使い方を理解しておいてください。変更を行う間に，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。

Oracle9i アプリケーション・サーバを設定するには

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 変更を加える前に **opmn.xml** ファイルのバックアップ・コピーを作成してください。

3 エディタを使って編集する **opmn.xml** ファイルを開きます。

4 **Xbootclasspath** プロパティを追加します。このプロパティは，java-option value に追加する必要があります。

以下は，**Xbootclasspath** パラメータの例です。

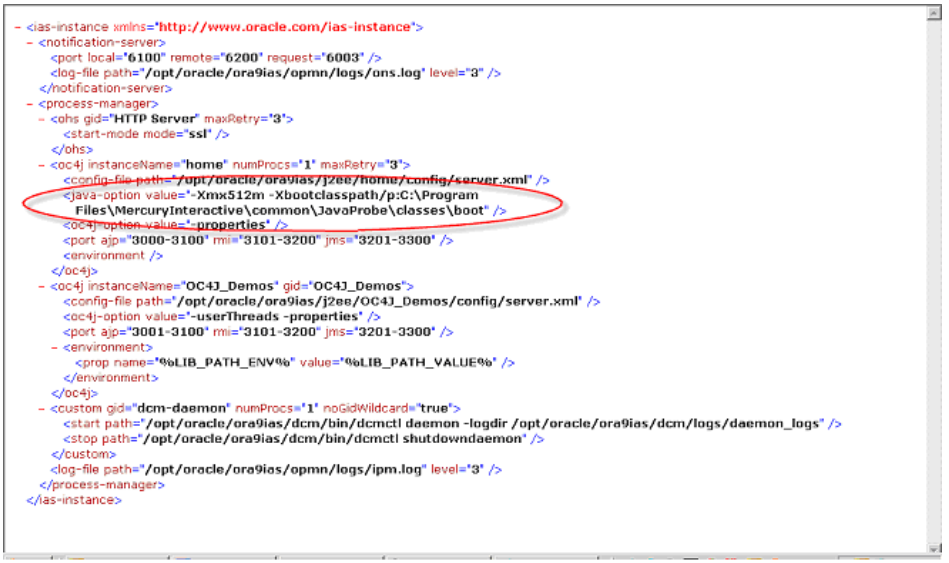
```
-Xbootclasspath/p: ＜ Probe のインストール・ディレクトリ＞¥
classes¥Sun¥1.4.2_04;＜ Probe のインストール・ディレクトリ＞¥
classes¥boot
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされているディレクトリへのパスです。

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，必ず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 XML ファイルに変更を保存します。

6 Oracle アプリケーション・サーバを再起動します。アプリケーション・サーバのホストは再起動する必要がありません。

7 Probe が正しく設定されていることを確認するには，**＜Probe 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 Instrumenter を実行していないか，**Xbootclasspath** パラメータを正しく入力しなかったかのいずれかです。JRE Instrumenter の実行については，137 ページ「JRE Instrumenter の実行」を参照してください。

Oracle 10.1.3 の設定

このセクションでは，Oracle 10.1.3 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。

Oracle 10.1.3 アプリケーション・サーバを設定するには

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 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 変更を加える前に **opmn.xml** ファイルのバックアップ・コピーを作成してください。

3 エディタを使って編集する **opmn.xml** ファイルを開きます。

4 java-option value に次のパラメータを追加します。

```
-javaagent: ＜ Probe のインストール・ディレクトリ＞ ¥lib¥probeagent.jar
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされているディレクトリへのパスです。

5 XML ファイルに変更を保存します。

6 Oracle アプリケーション・サーバを再起動します。アプリケーション・サーバのホストは再起動する必要がありません。

7 Probe が正しく設定されていることを確認するには，**＜ Probe 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，Java Probe が正しく起動していないことを示します。

Oracle 10.1.2 の設定

このセクションでは，Oracle 10.1.2 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。

Oracle 10.1.2 アプリケーション・サーバを設定するには

1 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。

2 **home** システム・コンポーネントをダブルクリックします。

3 ［**OC4J: home**］ページで，［**管理**］を選択します。

4［管理］ページで，［**サーバー・プロパティ**］を選択します。

5 ［サーバー・プロパティ］ウィンドウの［**コマンドライン・オプション**］で，［**Java オプション**］ボックスに Xbootclasspath プロパティを追加します。

注：Oracle 10.1.2 では，スイッチの前に（**^**）を追加する必要があります。そうしないと，Oracle は（**/**）スイッチ・オプションを（**¥**）に変更します。

以下は，（**^**）を挿入した Xbootclasspath パラメータの例です。

```
-Xbootclasspath^/p: ＜ Probe のインストール・ディレクトリ＞ /classes/boot
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされているディレクトリへのパスです。

6 変更を適用して Oracle サーバを再起動します。

JBoss アプリケーション・サーバの設定

このセクションでは，JBoss アプリケーション・サーバの設定方法について説明します。

重要：JBoss 4.0.5 の設定方法は，ほかのバージョンと異なります。詳細については，179 ページ「JBoss 4.0.5 以降」を参照してください。

JBoss アプリケーション・サーバは，アプリケーション・サーバの起動に使われるスクリプトに Xbootclasspath プロパティまたは JVM パラメータを追加して設定します。JBoss は，UNIX 環境でシェル・スクリプトを実行するか，または Windows 環境でコマンド・スクリプトを実行して起動します。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。代わりに，次のセクションでは，一般的なインプリメンテーションのための JBoss アプリケーション・サーバの各認定バージョンに対する手順を紹介します。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。

注：Probe 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，および環境変数の使い方を理解しておいてください。変更を行う間に，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。

このセクションでは，以下の JBoss 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。

- 177 ページ「4.0.5 よりも前のバージョンの JBoss」
- 179 ページ「JBoss 4.0.5 以降」

4.0.5 よりも前のバージョンの JBoss

4.0.5 よりも前のバージョンの JBoss アプリケーション・サーバを設定するには

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。通常，このファイルは，次の例のようなパスにあります。

```
D:¥JBoss¥bin¥run.bat
```

注： UNIX の場合，ファイル拡張子は .sh です。

---

2 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 エディタを使って編集する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，**Xbootclasspath** パラメータを追加します。パラメータは，**-hotspot** または **-classic** などの JIT オプションに続いて，Java パラメータの先頭で置き換える必要があります。

以下は，**Xbootclasspath** パラメータの例です。

```
-Xbootclasspath/p: ＜ Probe のインストール・ディレクトリ＞ ¥classes¥
Sun¥1.4.2_04;＜プローブのインストール・ディレクトリ＞ ¥classes¥boot
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされているディレクトリへのパスです。

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，必ず引用符を使用してください。

以下は，**Xbootclasspath** パラメータを追加する前の JBoss 起動スクリプトの例です。

```
"%JAVA%" %JAVA_OPTS% -classpath "%CLASSPATH%" org.jboss.Main
%ARGS%
```

以下は，**Xbootclasspath** パラメータを追加した後の JBoss 起動スクリプトの例です。

```
"%JAVA%" "-Xbootclasspath/p: ＜ Probe のインストール・ディレクトリ＞ ¥
classes¥Sun¥1.4.2_04;＜ Probe のインストール・ディレクトリ＞ ¥
classes¥boot" %JAVA_OPTS% -classpath "%CLASSPATH%" org.jboss.Main
%ARGS
```

注：起動スクリプトの例には，改行が入っています。しかし，実際のスクリプトに改行はなく，必要に応じてコマンドのテキストが画面上でワードラップします。

---

5 起動スクリプトに変更を保存します。

6 変更したスクリプトを実行して，Probe と一緒に JBoss アプリケーション・サーバを再起動します。アプリケーション・サーバのホストは再起動する必要がありません。

7 Probe が正しく設定されていることを確認するには，**＜ Probe 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 Instrumenter を実行していないか，**Xbootclasspath** パラメータを正しく入力しなかったかのいずれかです。JRE Instrumenter の実行については，137 ページ「JRE Instrumenter の実行」を参照してください。

JBoss 4.0.5 以降

4.0.5 以降の JBoss アプリケーション・サーバを設定するには

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。通常，このファイルは，次の例のようなパスにあります。

```
D:¥JBoss¥bin¥run.bat
```

注： UNIX の場合，ファイル拡張子は .sh です。

---

2 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 エディタを使って編集する起動スクリプトを開きます。

4 以下の JVM パラメータを追加します。

```
-javaagent: ＜ Probe のインストール・ディレクトリ＞
¥lib¥probeagent.jar
```

＜ Probe のインストール・ディレクトリ＞ は，Probe がインストールされているディレクトリへのパスです。

5 起動スクリプトに変更を保存します。

6 変更したスクリプトを実行して，Probe と一緒に JBoss アプリケーション・サーバを再起動します。アプリケーション・サーバのホストは再起動する必要がありません。

7 Probe が正しく設定されていることを確認するには，**＜ Probe 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，Java Probe が正しく起動していないことを示します。

注：ヒープ・ダンプに **-agentpath: ＜ Probe のインストール・ディレクトリ＞ ¥lib¥x86-windows¥jvmti.dll** を追加する際，次のオプションを使って JBoss を起動する必要があります。**run.bat -Djboss.platform.mbeanserver**

そうしないと，「Failed to locate MBeanServer via ManagementFactory." for jBoss 4（InvocationTargetException）」というエラーが発生します。

SAP NetWeaver アプリケーション・サーバの設定

ここでは，Probe でアプリケーションを監視できるように，SAP NetWeaver アプリケーション・サーバを設定する方法について説明します。

NetWeaver アプリケーション・サーバを設定すると，JVM がインストゥルメントされ，アプリケーション・サーバの起動に使われるスクリプトに Xbootclasspath プロパティが追加されます。次のセクションでは，一般的な NetWeaver のインプリメンテーション手順について説明します。サイト管理者は，これらの手順を使って，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。

注：Probe のアプリケーション・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，および環境変数の使い方を理解しておいてください。また，変更を行う前に必ずファイルのバックアップを作成してください。

SAP NetWeaver アプリケーション・サーバを設定するには

1 NetWeaver アプリケーション・サーバを実行する JVM を追加します。

2 JVM をインストゥルメントします。JRE Instrumenter では，**Xbootclasspath** パラメータを提供します。このパラメータは，次の手順に従って，NetWeaver Configtool の JVM パラメータ・ウィンドウに追加します。

3 NetWeaver アプリケーション・サーバ設定ツールを実行します。設定ツールは **configtool.bat** と呼ばれ，**usr¥sap¥j2e¥jc00¥j2ee¥configtool** ディレクトリに置かれます。

4 **-Xbootclass** パラメータを Java パラメータ・テキスト・ウィンドウに追加します。このウィンドウは，サーバ・インスタンスを選択したときに［General］タブにあります。たとえば，cluster-data | instance_ID70323 | server_ID7032350 となります。

5 変更を保存して，設定ツールを終了します。

6 **etc/capture.properties** ファイルのこれらのプロパティに，次の値を割り当てます。

minimum.buffer.size = 250000

initial.private.buffer.count = 50

maximum.private.buffer.count = 200

gentle.reserve.buffer.count = 50

hard.reserve.buffer.count = 50

7 NetWeaver アプリケーション・サーバを再起動します。

8 Probe が正しく設定されていることを確認するには，**＜ Probe 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 Instrumenter を実行していないか，**Xbootclasspath** パラメータを正しく入力しなかったかのいずれかです。

一般的なアプリケーション・サーバの設定

注：本書で特定のアプリケーション・サーバに対する設定方法が見つからない場合のみ，一般的なアプリケーション・サーバの設定方法を使用します。

サイト管理者は，代わりにサイト特有の方法を使ってアプリケーション・サーバを設定できます。その際，必要な変更を把握するのに一般的な手順が十分に役立つことがあります。

重要：変更を行う前に，すべての起動スクリプトのバックアップを作成してください。

アプリケーション・サーバの設定を更新するには

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って起動スクリプトを開きます。

4 以下の構文を使って，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 **Xbootclasspath** パラメータを追加します。

```
-Xbootclasspath/p: ＜ Probe のインストール・ディレクトリ＞
¥classes¥Sun¥1.4.2_04;＜ Probe のインストール・ディレクトリ＞ ¥classes¥boot
```

＜ Probe のインストール・ディレクトリ＞ は，Probe 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 Probe がアプリケーションに接続します。パラメータは，**-hotspot** または **-classic** などの JIT オプションに続いて，Java パラメータの先頭で置き換える必要があります。

以下は，**Xbootclasspath** 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
```

注：起動スクリプトの例には，改行が入っています。しかし，実際のスクリプトに改行はなく，必要に応じてコマンドのテキストが画面上でワードラップします。

以下は，**Xbootclasspath** 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: ＜
Probe のインストール・ディレクトリ＞ ¥classes¥Sun¥1.4.2_04; ＜ Probe のイン
ストール・ディレクトリ＞ ¥classes¥boot" -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。アプリケーション・サーバのホスト・マシンは再起動する必要がありません。

7 Probe が正しく設定されていることを確認するには，**＜ Probe 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 Instrumenter を実行していないか，**Xbootclasspath** パラメータを正しく入力しなかったかのいずれかです。

注：また，管理コンソールに入って，アプリケーション・サーバの JVM プロセス定義を設定できます。ただし，これは WebLogic サーバには適用されません。

JRE Instrumenter の実行については，137 ページ「JRE Instrumenter の実行」を参照してください。

アプリケーションの起動スクリプトを使用した Java Probe のヒープ・サイズの調整

ヒープのサイズは Java Probe のパフォーマンスに影響することがあります。

ヒープ・サイズのデフォルト値は 64 MB です。ヒープ・サイズは，次の VM 引数を使ってアプリケーションの起動スクリプトで設定します。

```
-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詳細なヘルプについては，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，Probe 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，インストゥルメンテーション・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ゥルメンテーションが Diagnostics に用意されています。

しかし，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，184 ページ「SOAP メッセージ・ハンドラの設定」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・インストゥルメンテーションを使って 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 の製品可用性マトリックス（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

SOAP メッセージ・ハンドラの無効化

1 デフォルトでは，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 のインストール・ディレクトリ＞ ¥etc¥inst.properties...¥
mercury.enable.autoLoadSOAPHandler = false
```

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。

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，以下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 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。

<install_root>¥config¥cell¥<Server>¥applications¥
<WebServiceEAR>¥deployments¥<WebServiceName>¥
<WebServiceJAR|WARName>¥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 のインストール・ディレクトリ > ¥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 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 install root>¥j2ee¥home¥applications¥<app name>¥<deployment name>¥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（<probe install dir>¥lib¥probeSOAPHandler.jar）を <OC4J install root>¥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。

第 8 章

.NET Agent（Probe）のインストール

.NET Agent は，Diagnostics .NET Probe と TransactionVision .NET Sensor の機能を 1 つのコンポーネントに組み合わせたものです。.NET Agent は，.NET ホスト上で TransactionVision Sensor と Diagnostics Probe の両方として同時に機能できます。

.NET Agent は，HP ソフトウェアの Business Availability Center アプリケーションと連携して機能する低オーバーヘッドなキャプチャ・ソリューションを提供します。.NET Agent は .NET アプリケーションからイベントをキャプチャし，そのイベント測定値を TransactionVision Analyzer，Diagnostics Server，あるいはその両方に送信します。

本章の内容

.NET Agent インストーラについて

.NET Agent のインストール時には，以下のセットアップと設定が行われます。

- ASP.NET アプリケーションの検出。このインストーラは，Agent がインストールされるシステム上の ASP.NET アプリケーションを自動的に検出しようとします。詳細については，192 ページ「ASP.NET アプリケーションの検出」を参照してください。
- デフォルトのエージェント設定。
  - このインストーラは，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。このエージェント設定は，**probe_config.xml** ファイルを使用して制御します。詳細については，192 ページ「検出した ASP.NET アプリケーション用の .NET Agent の自動設定」を参照してください。
  - デフォルトの **Asp.Net.points**，**Ado.points**，および **WCF.points** ファイルがインストールおよび有効化され，ASP.NET アプリケーションの監視を開始できるように標準の測定が提供されます。.NET および .NET WCF の要件と制限事項については，194 ページ「インストールの準備」を参照してください。ポイント・ファイルは，Agent がアプリケーションに対してキャプチャする負荷を制御します（.NET Remoting イベントを生成するには，Remoting.points も必要であり，インストゥルメンテーション定用のアプリケーションをセットアップする必要があります）。
  - 検出された各アプリケーション用の個別のインストゥルメンテーション・ポイント・ファイル。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インストゥルメンテーション・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。probe_config.xml ファイル内の各 appdomain 参照には，インストゥルメンテーション・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・インストゥ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。

- 追加のインストゥルメンテーション・ポイント・ファイルが作成されますが，無効になっています（第 10 章「.NET アプリーションのカスタム・インストゥルメンテーション」を参照）。インストーラは，以下のデフォルト・ファイルを作成します。

  Asp.Net.IExecutionStep.points，IIS.points，lwmd.points，msmq.points，webservices.points（WCF Web サービスでは使用されず，代わりに WCF.points ファイルが使用されます）。上記のポイント・ファイルを有効にするには，**probe_config.xml** ファイルに各ファイルへの参照を追加します（このインストゥルメンテーションを有効にする方法の詳細については，212 ページ「アプリケーションの標準インストゥ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）。

- デフォルトの .NET アプリケーション・レイヤのセットが設定されます。詳細については，第 12 章「インストゥルメンテーション・レイヤ」を参照してください。

- オプションのカスタム・インストゥルメンテーション。アプリケーション固有のメソッドをキャプチャするには，インストゥルメンテーション・ポイント・ファイルをカスタマイズします（第 10 章「.NET アプリーションのカスタム・インストゥルメンテーション」を参照）。

- オプション設定。probe_config.xml ファイルのエージェント設定を変更します（第 16 章「.NET Probe 設定ファイルについて」および第 17 章「.NET Probe の詳細設定」を参照）。

- .NET Agent をインストールして，設定を変更するか，（必要に応じて）カスタム・インストゥルメンテーションを作成した後は，ASP.NET アプリケーションとともに .NET Agent を使用する前に，IIS を再起動します。詳細については，214 ページ「IIS の再起動」を参照してください。

- IIS の再起動後，アプリケーション内の URL が少なくとも 1 回アクセスされると，.NET Agent はデフォルトのインストゥルメンテーション・ポイントと設定を使ってパフォーマンス・データの収集を開始できます。

ASP.NET アプリケーションの検出

.NET Agent インストーラでは，インストゥルメント可能な 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。インストーラが ASP.NET アプリケーションを検出できない例として，ASP.NET アプリケーションが仮想ディレクトリではなく Web ディレクトリとしてインストールされている場合が挙げられます。

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 Probe**］＞［**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 .NET Agent の自動設定

.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ゥ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ゥ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。

➤ **＜ Probe 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 16 章「.NET Probe 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オーバーライドすることにより，検出されたすべてのアプリケーションのインストゥルメンテーション効にします。アプリケーションのインストゥルメンテーションの有効化および無効化については，218 ページ「記録の無効化」を参照してください。

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。

SimpleConsoleHost.exe という名前のアプリケーションに対する probe_config.xml の設定例を次に示します。

```
<process name=="SimpleConsoleHost.points">
      <points file="SimpleConsoleHost.points"/>
      <logging level=" "/>
</process>
```

SimpleConsoleHost.exe という名前のアプリケーションに対するポイント・ファイルの設定例を次に示します。

```
[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・インストゥルメンテーション」を参照してください。

.NET Agent のインストール

次のセクションでは，.NET Agent のインストール方法について詳しく説明します。

本項の内容

インストールの準備

.NET 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。.NET Agent は，デフォルトではアプリケーションのパフォーマンスにほとんど影響を与えずにアプリケーションの監視を行うように設定されています。

.NET Agent（Probe）の詳細設定については，第 17 章「.NET Probe の詳細設定」を参照してください。

重要：.NET Agent のインストールを実行する前に，.NET Framework 1.1 以降をマシンにインストールする必要があります。

.NET Agent には SOAP Extension Handler も含まれています。.NET Agent をインストールすると，SOAP を使用する既存の Web アプリケーションを再起動する必要が生じることがあります。

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 以降が必要であり，以下のバインドのみがサポートされます。

- 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，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。

.NET Agent をホストするための推奨システム構成を次に示します。

.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。以下は，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	1.1 以降

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

.NET Diagnostics Profiler のユーザ・インタフェースは，IE6 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://<probehost>:<probeport>/profiler）にアクセス可能である必要があります。Probe は，Probe 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。デフォルトのポート範囲は 35000 ～ 35100 です。

.NET Agent インストーラの起動

.NET Agent インストーラは，Diagnostics インストール・ディスク，別の場所，または Business Availability Center の［Diagnostics Downloads］ページから起動できます。

製品インストール・ディスクからインストーラを起動するには

1 インストール・ディスクのルート・ディレクトリにある **setup.exe** ファイルを実行します。Diagnostics セットアップ・プログラムが起動し，インストール・メニュー・ページが表示されます。

2 メニューから［**Diagnostics Agent for .NET**］を選択します。これによって，32 ビット Windows バージョンの .NET Agent がインストールされます。64 ビット・バージョンの .NET Agent をインストールする場合は，**DVD を参照して Diagnostics_Installers** ディレクトリを見つけ，DotNetAgentSetup_x64_ ＜バージョン＞ .msi ファイルをダブルクリックする必要があります。

別の場所からインストーラを起動するには

1 **＜ HP Diagnostics インストール・ディスク＞ ¥Diagnostics _Installers** ディレクトリから，**DotNetAgentSetup_x86_ ＜バージョン＞ .msi**（32 ビット Windows 用）または **DotNetAgentSetup_x64_ ＜バージョン＞ .msi**（64 ビット Windows 用）ファイルを新しい場所にコピーし，そのファイルをクリックします。

2 197 ページ「インストールの実行」に進みます。

Business Availability Center の［Diagnostics downloads］ページからインストーラを起動するには

1 **Business Availability Center** 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 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 インストーラは Business Availability Center からアクセスするために必要なディレクトリに置かれていれば，Business Availability Center で利用できます。Diagnostic Server 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。

---

197 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

インストーラを起動したら，メイン・インストール手順を開始できます。

Windows マシンに .NET Agent をインストールするには

1 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，［**I accept the terms of the license agreement**］を選択します。

続行するには［**Next**］をクリックします。

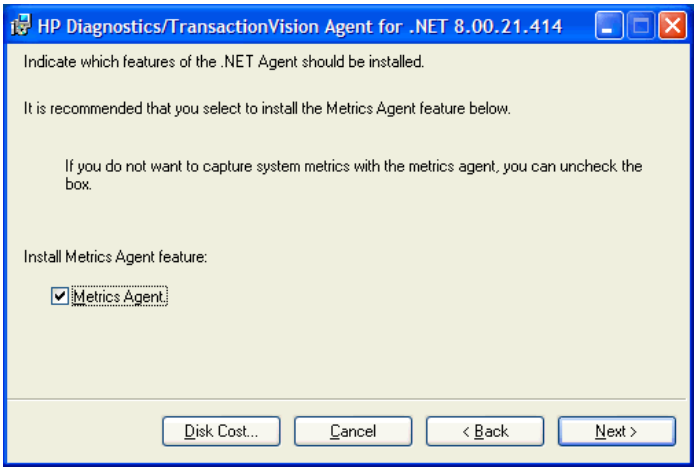
2 Agent のインストール先を指定します。

デフォルトでは，Agent は **C:¥MercuryDiagnostics¥.NET Probe** にインストールされます。

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。

続行するには［**Next**］をクリックします。

3 インストールする .NET Agent の機能を選択します。



デフォルト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。

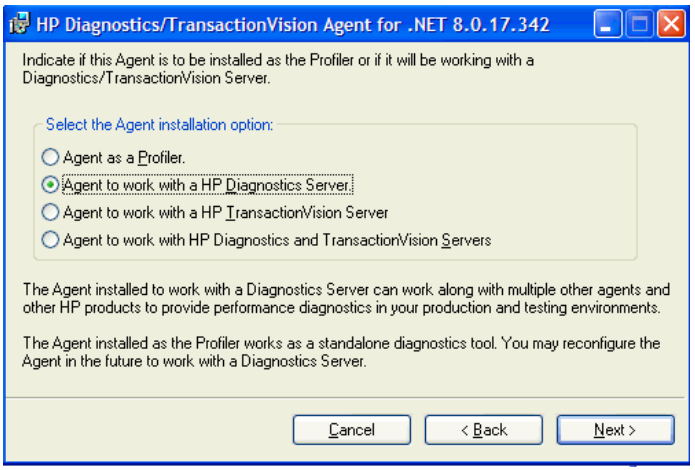
ただし，ホスト・マシン上のシステム測定値をキャプチャしない場合は，［**Metrics Agent**］ボックスをオフにできます。

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。

続行するには［**Next**］をクリックします。

4 Agent を，Diagnostics Profiler 専用，Diagnostics Server に報告するエージェント，TransactionVision Server に報告するエージェント，Diagnostics Server と

TransactionVision Server の両方に報告するエージェントのうち，どれでインストールするかを選択します。



Agent を使用する環境に適した項目を選択します。

- Agent を Diagnostics 環境で Profiler として使用するには，［**Agent as a Profiler**］を選択します。
- Agent を Diagnostics 環境でプローブとして使用するには，［**Agent to work with an HP Diagnostics Server**］を選択します。
- Agent を TransactionVision 環境で Agent として使用するには，［**Agent to work with an HP TransactionVision Server**］を選択します。
- Agent を TransactionVision 環境と HP Diagnostics 環境の両方で Agent として使用するには，［**Agent to work with HP Diagnostics and TransactionVision Servers**］を選択します。

続行するには［**Next**］をクリックします。

次の手順：ここでは，Agent をインストールしている環境によってインストール手順が異なります。

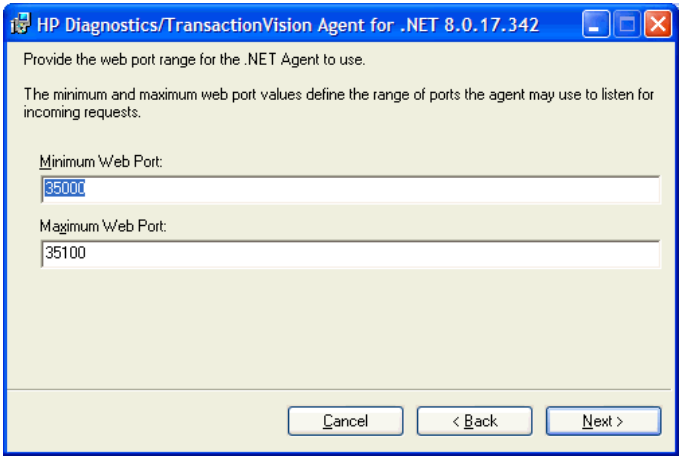
- Agent を Diagnostics Profiler 専用でインストールしている場合は，201 ページ「Agent を Profiler 専用でインストールする」に進んでください。
- Agent を HP Diagnostics Server と連携して動作するようにインストールしている場合は，203 ページ「Agent を Diagnostics Server と連携して動作させるためのインストール」に進んでください。
- Agent を TransactionVision 環境で動作するようにインストールしている場合は，209 ページ「Agent を TransactionVision 環境で動作させるためのインストール」に進んでください。
- Agent を TransactionVision 環境と HP Diagnostics 環境の両方で動作するようにインストールしている場合は，203 ページ「Agent を Diagnostics Server と連携して動作させるためのインストール」に進み，次に 209 ページ「Agent を TransactionVision 環境で動作させるためのインストール」に進んでください。

続行するには［**Next**］をクリックします。

Agent を Profiler 専用でインストールする

Agent を Diagnostics Profiler 専用で動作するようにインストールしている場合は，次の手順に進みます。

1 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。

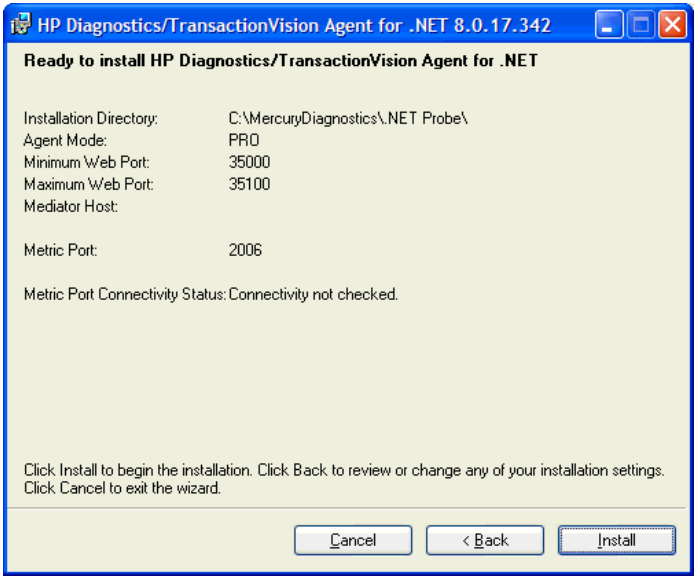
ポート範囲の最小サイズは，Agent のホストで同時に動作する Agent の最大数と等しくなります。

Web ポート範囲を設定する際の注意事項

- 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。

続行するには［**Next**］をクリックします。

2 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，［**Instal**］をクリックします。



注：Profiler だけ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。

---

3 .NET Agent のインストールが完了すると，インストーラのウィンドウにインストール後の作業に関する指示が表示されます。詳細については，212 ページ「アプリケーションの標準インストゥ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。

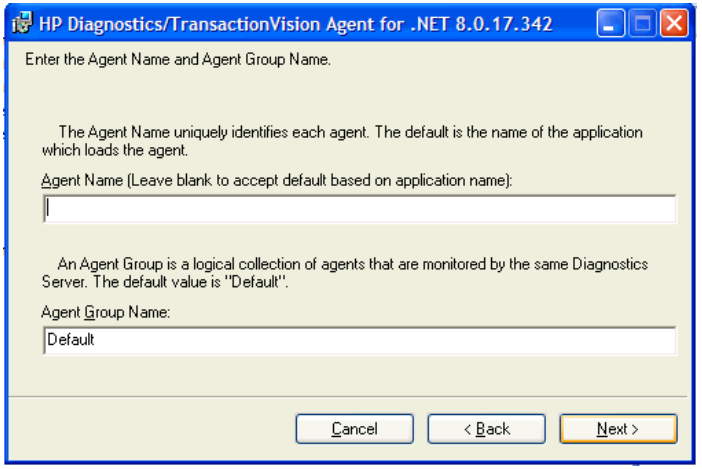
インストーラを終了するには，［**Finish**］をクリックしてください。

4 ASP.NET アプリケーションで .NET Agent を使用するには，インストーラの終了後，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。詳細については，214 ページ「IIS の再起動」を参照してください。

Agent を Diagnostics Server と連携して動作させるためのインストール

Agent を Diagnostics Server と連携して動作するようにインストールしている場合は，次の手順に進みます。

1 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，アプリケーションのドメイン名に基づいて Agent 名を自動生成します。

重要：［**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，以下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

- Agent 名には，文字，数字，ダッシュ，下線およびピリオドを使用できます。
- Agent が監視しているアプリケーションと Agent 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 名を次のように付けます。

  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。デフォルト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

  ［エージェント名］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デフォルトの Agent 名は，$(APPDOMAIN).NET の指定に相当します。

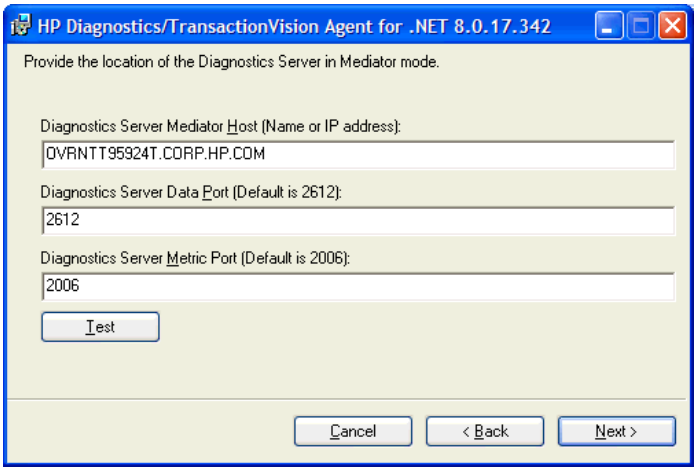
- ［**Agent Group Name**］: 既存のグループの名前か新規作成するグループの名前を入力します。Agent グループ名のデフォルト値は，Default です。Agent グループ名は大文字と小文字を区別します。Diagnostics では，この名前がプローブ・グループ名として使用されます。

  Probe グループは，同じ Diagnostics Server に報告する Probe の論理グループです。Probe 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，場合によっては，Agent グループや個別の Agent のパフォーマンスを監視するために，1 つの Agent 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 Agent を割り当てる必要があります。

続行するには［**Next**］をクリックします。

2 .NET Agent が Diagnostics Server（Mediator モード）と通信するために必要な情報を入力します。



a ［**Diagnostics Server Mediator Host or IP Address**］ボックスに，Diagnostics Server（Mediator モード）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 Server Data Port**］ボックスに，Diagnostics Server 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。デフォルトのポート番号は 2612 です。Diagnostics Server をインストールした後にポートを変更した場合は，デフォルト・ポートではなく，変更後のポート番号を指定する必要があります。

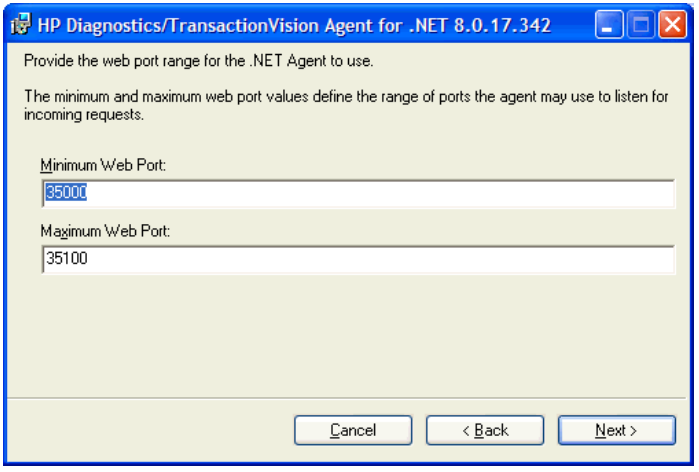
c ［**Diagnostics Server Metric Port**］ボックスに，Diagnostics Server が System Metrics Agent の通信のためにリスンしているポート番号を入力します。デフォルトのポート番号は **2006** です。Diagnostics Server をインストールした後にポートを変更した場合は，デフォルト・ポートではなく，変更後のポート番号を指定する必要があります。

d 接続チェックを実行して，Diagnostics Server 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 Diagnostics Server（Mediator モード）の情報に間違いがないかどうか，Diagnostics Server 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。Diagnostics Server（Mediator モード）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。

e 続行するには［**Next**］をクリックします。

3 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。

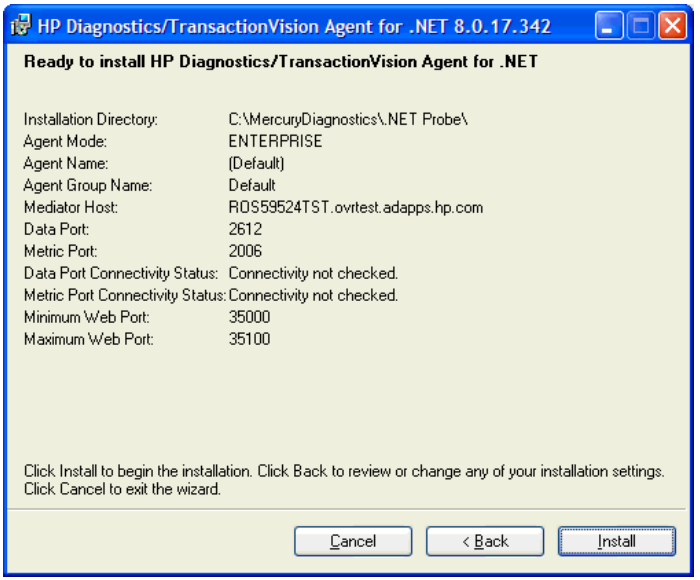
ポート範囲の最小サイズは，Agent のホストで同時に動作する Agent の最大数と等しくなります。

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。

続行するには［**Next**］をクリックします。

4 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。



5 .NET Agent のインストールが完了すると，インストーラのウィンドウにインストール後の作業に関する指示が表示されます。

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。

詳細については，以下のトピックを参照してください。

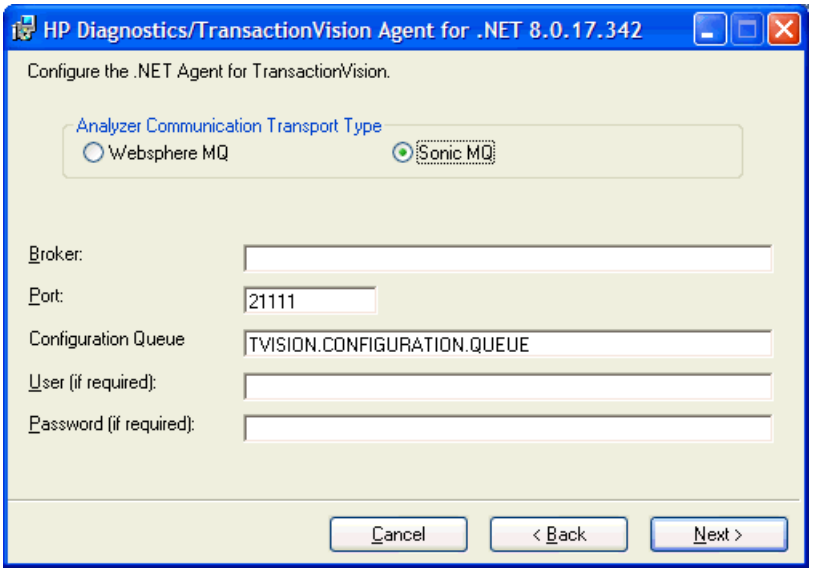
- ➤ 212 ページ「.NET Agent のカスタム設定およびインストゥルメンテーションについて」
- ➤ 212 ページ「アプリケーションの標準インストゥルメンテーションの有効化と無効化」
- ➤ 214 ページ「IIS の再起動」
- ➤ 215 ページ「.NET Agent のインストールの確認」

6 ASP.NET アプリケーションで .NET Agent を使用するには，インストーラを終了し，.NET Agent の設定と .NET アプリケーションのインストゥルメンテーションに対する変更やカスタマイズを完了した後で，IIS サービスを再起動する必要があります。詳細については，214 ページ「IIS の再起動」を参照してください。

Agent を TransactionVision 環境で動作させるためのインストール

Agent を TransactionVision 環境で動作するようにインストールしている場合は，次の手順に進みます。

1 ［Configure the .NET Agent for TransactionVision］ダイアログが表示されます。



2 メッセージング・ミドルウェア・プロバイダ（［Analyzer Communication Transport Type］）を選択します。［WebSphere MQ］と［SonicMQ］のいずれかを選択できます。

SonicMQ は .NET Agent に含まれています。このオプションを指定すると，Agent のインストールの一部として Sonic MQ .NET クライアント（Sonic.Client.dll - Progress SonicMQ .NET クライアント，バージョン 7.6.0.112）がインストールされます。

サードパーティーの WebSphere MQ インストールを代わりに使用できます。この場合は，監視対象のホストに MQ シリーズの .NET クライアント（amqmdnet.dll - .NET 用 WebSphere MQ クラス，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。

デフォルトでは，［SonicMQ］が選択されています。

3 SonicMQ の場合は，以下の項目を入力します。

［**Broker**］：TransactionVision 構成キューがホストされるブローカ。通常はホスト名。

［**Port**：ブローカが通信するポート。

［**Configuration Queue**］：設定キューの名前。

［**User**］：ユーザ ID（Sonic MQ のインストールで接続のために必要な場合）。

［**Password**］：パスワード（Sonic MQ のインストールで接続のために必要な場合）。これは，PassGen ユーティリティを使って作成した難読化形式で指定します。PassGen の詳細については，第 16 章「.NET Probe 設定ファイルについて」を参照してください。

4 WebSphere MQ の場合は，以下の項目を入力します。

［**Host**］：WebSphere MQ キュー・マネージャがあるホスト。

［**Port**］：WebSphere MQ キュー・マネージャのポート番号。

［**ConfigurationQueue**］：設定キューの名前。

［**User**］：ユーザ ID（WebSphere のインストールで接続のために必要な場合）。

［**Password**］：パスワード（WebSphere MQ のインストールで接続のために必要な場合）。これは，PassGen ユーティリティを使って作成した難読化形式で指定します。PassGen の詳細については，第 16 章「.NET Probe 設定ファイルについて」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**］：キュー・マネージャの名前。

続行するには［**Next**］をクリックします。

5 プレインストール・サマリ・ダイアログが表示されます。

6［**Install**］をクリックしてインストールを進めます。

7 ASP.NET アプリケーションで .NET Agent を使用するには，インストーラを終了し，.NET Agent の設定と .NET アプリケーションのインストゥルメンテーションに対する変更やカスタマイズを完了した後で，IIS サービスを再起動する必要があります。詳細については，214 ページ「IIS の再起動」を参照してください。

.NET Agent の接続状況の確認

実稼動環境（Business Availability Center または Diagnostics Standalone を使用）またはテスト環境（LoadRunner または Performance Center を使用）で .NET Agent を実行している場合，次の手順を使って Agent が接続していることを確認できます。

Agent が接続していることを確認するには

1 Web ブラウザを開き，Diagnostics Server（Commander モード）に対して http:// ＜ Diagnostics Server のホスト＞ : ＜ Diagnostics Server のポート＞ /registrar にアクセスします。サーバにログオンするように指示されることがあります。

2 ［**表示**］で，［**登録されたコンポーネント**］をクリックします。

3 ［**名前**］列に一覧されているアプリケーションを探します。一致するコンポーネントのリストを絞り込むには，［**Probe**］コンポーネント・タイプを選択して，Agent（プローブ）のリストのみ表示するように，［**送信**］ボタンをクリックします。

4 ［**Active**］列で Agent のステータスを探します。［Active］のステータスが［True］であれば，その Agent は接続されていることを示しています。

.NET Agent のカスタム設定およびインストゥルメンテーションについて

Diagnostics と連携して動作するように .NET Agent をインストールした場合，.NET Agent はデフォルトで Diagnostics Probe として動作するように設定され，アプリケーションのパフォーマンス測定値をキャプチャします。この設定は，環境や診断するパフォーマンス問題に合わせてカスタマイズできます。

インストーラは，ASP.NET アプリケーションと .NET Agent を，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設定します。インストーラが検出しない方法で 1 つ以上の ASP.NET アプリケーションをデプロイしたり，標準の測定を拡張して，アプリケーションのカスタム・クラスのパフォーマンス測定値をキャプチャできます。

HP 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 16 章「.NET Probe 設定ファイルについて」を参照してください。.NET Agent の高度な設定については，第 17 章「.NET Probe の詳細設定」を参照してください。

HP 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・インストゥルメンテーション・ポイントも作成できます。カスタム・インストゥ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・インストゥルメンテーション」を参照してください。

アプリケーションの標準インストゥ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 インストゥルメンテーションが有効になりますが，アプリケーション特有のインストゥルメンテーションは有効になりません。.NET Agent の **probe_config.xml** ファイルの属性を使って，インストゥルメンテーションを有効または無効にするアプリケーションを制御します。

アプリケーションのインストゥ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ゥ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。

アプリケーションのインストゥルメンテーションを有効または無効にするには

1 **process** タグの **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ゥ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 インストゥ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** タグの **enabled** 属性を **true** に設定します。

3 インストゥ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** タグの **enabled** 属性を **true** に設定します。

以下は，あるアプリケーションに対してインストゥ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です。

```
<process name="ASP.NET", <enablealldomains="false">
    <points file="ASP.NET.points" />
    <points file="ADO.points" />
    <appdomain name="myApplication", enabled="true">
        <points file="myApplication.points" />
    </appdomain>
    <appdomain name="myApplicationTwo", enabled="false">
            <points file="myApplicationTwo.points"/>
    </appdomain>
</process>
```

すべてのアプリケーションのインストゥルメンテーションを有効にするには

➤ **process** タグの **enablealldomains** 属性を **true に設定します。**これにより，各 appdomain タグの属性の設定が変更されるため，さまざまな属性の設定に入ることなく，インストゥルメンテーションを有効にできます。

以下は，すべてのアプリケーションに対してインストゥルメンテーションを有効にした場合の例です。

```
<process name="ASP.NET", <enablealldomains="true">
    <logging level=""/>
    <points file="ASP.NET.points" />
    <appdomain name="myApplication", enabled="false">
            <points file="myApplication.points" />
    </appdomain>
    <appdomain name="myApplicationTwo", enabled="false">
            <points file="myApplicationTwo.points"/>
    </appdomain>
</process>
```

IIS の再起動

.NET Agent をインストールし，Agent のインストゥルメンテーションまたは設定を変更したら，IIS を再起動して変更を反映させる必要があります。

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。

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ゥルメントされ，Agent が Diagnostics Server（Commander モード）に登録を行います。

注：ASP.NET は，アプリケーションが再デプロイされたことを検出した場合や，アプリケーションが設定されているリソースしきい値を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 Server（Commander モード）に登録し，Diagnostics Server（Mediator モード）に接続しているという一定期間の重複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 Availability Center 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。

.NET Agent のインストールの確認

このセクションは，Agent を Diagnostics Server と連携して機能するようにインストールした場合のみ関係があります。

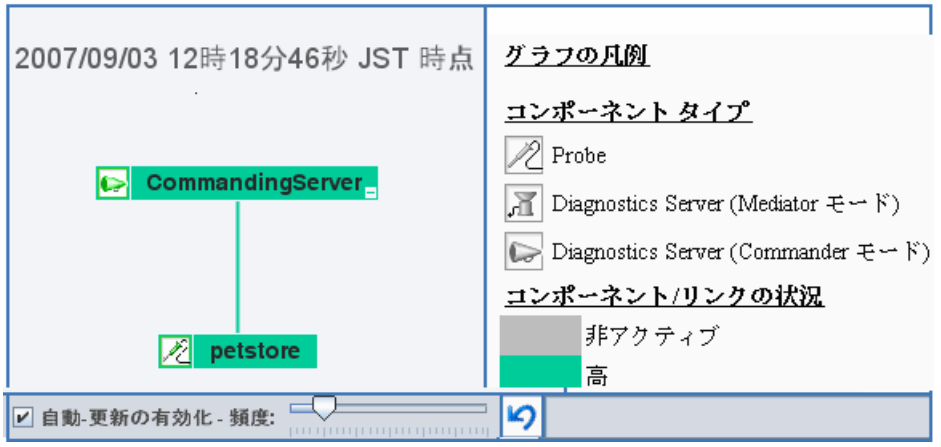
システムの状況モニタを使って，.NET Agent のインストールを確認します。システムの状況モニタの使い方に関する詳細は，付録 D「System Health Monitor の使用」を参照してください。

推奨インストール手順に従った場合は，.NET Agent をインストールしてアプリケーションをインストゥルメントした後に，以下を確認できます。

- Diagnostics Server（Commander モード）が正常にインストールされている。
- 追加の Diagnostics Server（Mediator モード）が正常にインストールされ，Diagnostics Server（Commander モード）と通信している。
- .NET Agent が正常にインストールされ，Diagnostics Server と通信している。

注：.NET Agent は，起動するまで Agent が Diagnostics Server に登録されないため，初めてインストールしたときに System Health に表示されません。測定されるアプリケーションが実行すると，Agent が起動して Diagnostics Server に登録されます。ASP.NET アプリケーションでは，インストゥルメントされるアプリケーションにページが要求されたときに初めて，これが発生します。Agent がシステムの状況に表示されない場合に，ログ・ファイルをチェックして問題解決に役立たせるために，.NET Agent のデフォルトの記録レベルは，**info** に設定されています。ログ・ファイルが表示されない場合は，Windows イベント・ビューアで確認できます。

デプロイメントに応じて，Diagnostics Server（Commander モード）か Diagnostics Server（Mediator モード）かを知らせるために，システム状況モニタに新しい .NET Agent が表示されます。



.NET Agent のバージョン確認

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

➤ **＜ 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。

または，Diagnostics のシステムの状況モニタを使って .NET Agent のバージョンを確認することもできます。

.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。

Diagnostics .NET Agent 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。

注：.NET Agent を有効にすると，SOAP Extension Handler が再び有効になり，それにより SOAP を使用しているいくつかの Web アプリケーションを再起動する必要が生じることがあります。また，.NET Agent を無効にする場合は，その前にホスト上で SOAP を使用しているすべての Web アプリケーションを停止する必要があります。

起動してパフォーマンス測定値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

➤ ［**スタート**］＞［**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

➤ ［**スタート**］＞［**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。システム測定値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。Probe の有効化または無効化は，次回 Probe アプリケーションを起動したときだけ有効になります。つまり，現在実行中のアプリケーションでは有効になりません。

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，Probe 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，インストゥルメンテーション・セクションの **logging level** タグを変更して，Probe のインストゥ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>
```

第 IV 部

Java および .NET アプリケーションの監視のカスタム・インストゥルメンテーション

第 9 章

Java アプリーションのカスタム・インストゥルメンテーション

HP Diagnostics がパフォーマンス測定値の収集を可能にするためにアプリケーションのクラスとメソッドに適用する測定の制御方法について説明します。

本章の内容

インストゥ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

インストゥ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 Probe 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。インストゥルメンテーションインストゥルメンテーションによって，Probe は実行時間の測定，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 Probe インスタンスのインストゥルメンテーション・ポイントは，キャプチャ・ポイント・ファイルに指定されます。

キャプチャ・ポイント・ファイルは，インストゥルメンテーションの範囲を制御できるようにするためのインストゥルメンテーション・メカニズムです。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測定定義は，どのメソッドをインストゥルメントするか，どのようにインストゥルメントするか，そしてどのインストゥルメンテーションをインストールするかを Probe 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，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。正規表現の使い方については，付録 J「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，デフォルト・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，Java EE 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，重要な情報の層に体系づけたり，インストゥルメンテーションの収集動作を制御するのに使われます。

Probe 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，デフォルト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。レイヤの詳細については，第 12 章「インストゥルメンテーション・レイヤ」を参照してください。

キャプチャ・ポイント・ファイルの検索

Java Agent（プローブ）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。

Java では，定義済みの Java EE ポイントが含まれるデフォルトのキャプチャ・ポイント・ファイルは，**＜ Probe のインストール・ディレクトリ＞ ¥etc¥ auto_detect.points** にあります。

注：

- デフォルトの **auto_detect.points** ファイルに別の名前を付けてコピーを作成し，そのコピーを使ってすべてのインストゥルメンテーションをカスタマイズします。これは，Probe 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 をオーバーレイするときにカスタム・インストゥルメンテーションを失わないための予防策です。
- デフォルトのファイル名は，**＜ Probe のインストール・ディレクトリ＞ ¥ etc¥probe.properties** で指定します。
- 代わりにコピーが使用されるように，デフォルト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

以下は，キャプチャ・ポイント・ファイルのポイントの定義に使用できる引数の一覧です。

```
[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= < 指定子のリスト >
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。

- 必須ポイント引数
- オプションのポイント・エントリ

必須ポイント引数

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なければなりません。

引数	説明
Point-Name	ポイントの一意の名前。
class	class 引数では，インストゥ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します。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ます。
method	method 引数は，インストゥルメントするメソッドの名前を指定します。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ます。
signature	signature 引数では，メソッド・シグネチャ（<jdk_install>/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）を指定します。
layer	layer 引数では，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します。レイヤは，インストゥ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ます。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。サブレイヤは，サブレイヤに続くほかのスラッシュとサブレイヤ名をエンコーディングして，独自のサブレイヤを持つように定義できます。 UI で，レイヤのサブレイヤは，親レイヤの配下に表示されます。たとえば，Web レイヤのサブレイヤが JSP および Struts の場合，それらのサブレイヤは Web レイヤの配下に表示され，Web から JSP および Struts にドリルダウンが存在します。

以下は，必須引数を含むカスタム・ポイントの例です。

```
[MyCustomEntry_1]
; コメントをここに入力…
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。正規表現の使い方については，付録 J「正規表現の使用」を参照してください。

オプションのポイント・エントリ

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** の存在は，特別な測定クラスによって処理されるインストゥルメンテーション・ポイントを示します。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ゥルメンテーション・クラス名になります。特別なインストゥルメンテーション・ポイントでは，ここで説明していない測定固有の引数を使用できます。
ignore_cl	**ignore_cl** 引数は，無視するクラス名または正規表現のカンマ区切りのリストを指定するのに使用します。**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ゥルメントされません。
ignore_method	**ignore_method** 引数は，無視するメソッドのカンマ区切りのリストを指定するのに使用します。**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ゥルメントされません。
ignore_tree	**ignore_tree** 引数は，クラスまたは正規表現のリストです。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ゥルメンテーションから除外されます。

引数	説明
method_access_filter	**method_access_filter** は，メソッド指定子のカンマ区切りのリストです。使用できる指定子は，**static**，**private**，**protected**，**package**，および **public** です。このポイントと一致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ゥルメントされません。
deep_mode	**deep_mode** 引数は，サブクラスの処理方法を指定します。この引数では，次の 3 つの値を受け入れます。 ➤ **none** - 値 none は，**deep_mode** 引数を指定しないものに類似します。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で，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ゥルメントするように要求します。 ➤ **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で，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ゥルメントするように要求します。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 モードは，広範なインストゥルメンテーションになり，Probe のオーバーヘッドが非常に高くなる恐れがあります。
scope	**scope** 引数は，インストゥルメンテーションが実行されるコンテキストを制約します。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ります。この引数の値には，任意の有効な正規表現を使用できます。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	**ignoreScope** 引数は，メソッド名または正規表現のリスト形式を取り，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ために使用されます。

引数	説明
detail	**detail** 引数は，より限定的なキャプチャ命令を指定します。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ゥルメンテーションを実行します。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ゥルメンテーションである呼び出される側のインストゥルメンテーションが実行されます。 ➤ **args:n** - **n-th** 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ます。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()** を呼び出します。スタティック・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します。 ➤ **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します。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ます。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・キーを指定します。コード・スニペットついては 234 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・キーついては 248 ページ「コード・スニペットの安全の確保」をそれぞれ参照してください。 ➤ **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します。「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ります。

引数	説明
detail（続き）	➤ **disabled** - バイトコードに挿入されたインストゥルメンテーションがデータを報告するのを禁止します。インストゥ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させることができます。この Web ページには，Profiler URL http:// ＜ Probe のインストール・ディレクトリ＞：＜ Probe のポート＞ /inst/layer を使ってアクセスできます。 ➤ **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します。また，このインストゥルメンテーション・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します。 ➤ **no-correlation** － 相関サポート・テクノロジを使用しない「outbound」ポイントで使用します。 ➤ **method-no-trim** － このポイントによってインストゥ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・ベースの除外を行わないことを示します。 ➤ **method-trim** － このポイントによってインストゥルメントされるメソッドのすべての呼び出しが「除外」される（つまり，報告されない）ことを示します。ただし，対応するコード・スニペットの副作用（がある場合）は，通常どおり発生します。 ➤ **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ゥルメンテーション・ポイントを識別します。 ➤ **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ゥ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します。 ➤ **is-statement** － java.sql.Statement クラスの呼び出しをマークします。 ➤ **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します。 ➤ **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 包含時間を収集します。

引数	説明
detail（続き）	➤ **condition** ー 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ゥルメンテーションを禁止します。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）で定義されます。 ➤ **when-root-rename** - このポイントによってインストゥ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します。 ➤ **diag** - HP Diagnostics に関係するものとしてポイントをマークします（デフォルト）。 ➤ **tv: ＜キー＞** - HP Transaction Vision に関係するものとしてポイントをマークします。 ➤ **no-tv** - HP Transaction Vision と競合するものとしてポイントをマークします。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 コードインストゥルメントが完全に禁止されます。 ➤ **add-field: ＜アクセス＞: ＜タイプ＞: ＜名前＞** - 指定したフィールドがインストゥルメント対象のクラスに追加されます。 ➤ **gen-instrument-trace** - このポイントがインストゥ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** - このポイントによってインストゥ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 に出力されます。 ➤ **trace** - このポイントによってインストゥルメントされる各メソッドの各入口または出口で，インストゥ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。 ➤ **sub-point: ＜キー＞** - インストゥルメンテーション時の追加処理を指定します。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ります。 ➤ **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ます。 ➤ **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ます。

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ゥ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ます。 ➤ **ws-operation** - インストゥルメンテーション・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します。また，このインストゥルメンテーション・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します。
rootRenameTo	**rootRenameTo** 属性は，**when-root-rename** detail が有効であるときに常にサーバ要求を識別するために使用されます。
layerType	**layerType** 引数は，インストゥルメントしたいくつかのメソッドに対して特殊な処理を指定します。この引数は，次の値を受け入れます。 ➤ **method** - 特殊な処理を行いません（デフォルト）。 ➤ **trended_method** - Trended Methods ビューに表示するメソッドを特定します。 ➤ **Portlet** - Portal Components ビューで使用するポートレット・ライフサイクル・メソッドを特定します。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します。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ゥルメンテーション・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ます。priority の高い方が優先され，デフォルトは 0 です。
active	active 引数は，ポイントをアクティブまたは非アクティブにします。true に設定すると，ポイントはアクティブになります。false に設定した場合，ポイントは非アクティブになり，Probe に無視されます。

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** 引数で指定したときに，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，またはインストゥルメントしたメソッドの 1 つ以上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，インストゥ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 Probe インスタンスの 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 Probe インストールで有効になります。code-key の詳細については，248 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，**＜ Probe 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。このファイルのコード・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・スニペットを使って，インストゥルメントしたメソッドでアクセス可能なメソッドに，インストゥ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・スニペットは，**toString()** をバイトコードに解析しているステートメントのスタックに残せる文字列またはオブジェクトで終了します。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，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，Probe 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，インストゥルメンテーション時に Probe 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 で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には，次の例のように detail を使う必要があります。

```
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ていることを示します。**code-key** 値は，コード・スニペットのコードの安全を保ち，ポイントと実際のコード・スニペットを関連付けるために使われます。

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，＜ **Probe 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()); ¥
"DIAG_ARG:type=jms&name=topic:"+ #topic + "&target=topic://" + #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でき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
配列 *	[A][B][C] [D][E][F]
null 定数	null

* すべての種類の一次元および多次元配列

➤ 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，インストゥ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するのに使われます。

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。この変数は，スタティック・メソッドを参照するときに使用できます。
#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：一部のインストゥルメンテーション・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・データをインストゥ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。

```
"DIAG_ARG:type=ws&ws_name="+ #servicename +"&ws_op="+ #operation
+¥ "&ws_ns="+ #ns +"&ws_port="+ #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 サービスの受信データを収集するには，どちらか一方のみを使用してください。

➤ 特殊フィールド（store-fragment）

デフォルト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 を格納するために使用します。
##WS_SOAP_fault_code	SOAP エラーのコードを格納するために使用します。
##WS_SOAP_fault_reason	SOAP エラーの理由を格納するために使用します。
##WS_SOAP_fault_detail	SOAP エラーの詳細情報を格納するために使用します。
##WS_inbound_service_name	受信 Web サービスの名前を格納するために使用します。
##WS_inbound_operation_name	受信 Web サービスの操作名を格納するために使用します。
##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ために使用します。
##WS_inbound_port_name	受信 Web サービスのポート名を格納するために使用します。

➤ 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，store-thread detail の設定とともに使用する必要があります。

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 getAndRemoveThreadContextValue(" 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 getAndRemoveThreadContextValue(" 文字列値 ") を呼び出してください。

変数	用途
##SOAPHandler_wsname	後でプローブの SOAP ハンドラが使用する Web サービスの名前を格納するために使用します。
## ＜任意の文字列＞	後続するコード・スニペットで取得する値を格納するために使用します。

➤ 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 @ 記号を付けて「スタティック」と特定し，@ 記号を使ってメソッドがアクセスされていることをマークすることによって，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 ("Hello World");
```

```
@com.mercury.diagnostics.capture.metrics.countingCollector@.incrementCou
nter();
```

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 ProbeCodeSnippetV5 という 2 つのヘルパー・クラスが用意されています。後者は，Java 5 以降でのみ使用できます。ProbeCodeSnippetHelper には，以下に示す機能があります。

```
/*
 * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
 */

package com.mercury.opal.capture.proxy;

/**
 * コード・スニペットを支援するために使用する
 */
public class ProbeCodeSnippetHelper {

  /**
   * 以下の文字列への参照を保持している特殊フィールドは
   * フラグメントのローカル・ストレージや
   * 呼び出しのローカル・ストレージには格納されない
   */
  public static final String DO_NOT_STORE = ...

  /**
   * int を Integer に変換するヘルパー
   * @param i
   * @return i の値を持つ新しい Integer オブジェクト
   */
  public static Object intToInteger(int i) {
    ...
  }

  /*
   * 現在のスレッドをマークする（まだマークされていない場合）
   * @return スレッドがすでにマークされていた場合のみ true
   */
  public static boolean testAndSetRecursiveFlag() {
    ...
  }

  /*
   * 現在のスレッドのマークを削除する
   */
  public static void clearRecursiveFlag() {
    ...
  }

  /**
   * ResourceBundle.getString() を呼び出して，スローされる可能性がある例外をキャッチする
   * ヘルパー・メソッド
   * @param theBundle getString が呼び出される ResourceBundle
   * @param key getString に渡すキー
   * @return getString から返された値，例外発生時は null
   */
  public static String getStringFromResourceBundle(ResourceBundle theBundle, String key) {
    ...
  }

}
```

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。このクラスによって提供されるメソッドを次に示します。

```
/*
 * (c) Copyright 2008 Hewlett-Packard Development Company, L.P.
 */

package com.mercury.opal.capture.jdk15.agent;

/**
 * Java 5 以降を使用するコード・スニペットを支援するために使用する
 */
public class ProbeCodeSnippetHelperV5 {

  /**
   * クラス，そのスーパークラス，またはその実装済みインタフェースから，
   * 指定されたタイプの注釈を取得する
   * @param theClass 注釈を取得するクラス
   * @param annClass 検索する注釈クラス
   * @return
   */
  public static Object getEndpointClassAnnotation(Class theClass, Class annClass) {
    ...
  }

  /**
   * クラス，そのスーパークラス，またはその実装済みインタフェースから，
   * 指定されたタイプのメソッド注釈を取得する
   * @param theClass クラス
   * @param methodName メソッド名
   * @param argCount 引数カウント
   * @param annClass クラス注釈のタイプ
   * @param methodAnnClass メソッド注釈のタイプ
   * @return
   */
  public static Object getEndpointMethodAnnotation(Class theClass, String methodName,
              String argCount, Class annClass, Class methodAnnClass) {
    ...
  }

  /**
   * 注釈要素の値を取得するヘルパー・メソッド。
   * 注釈に要素がない場合は null を返す。
   * @param annClass 注釈のクラス
   * @param instance 注釈インスタンス・オブジェクト
   * @param elementName 要素名
   * @return 注釈インスタンスの要素の値，または null
   */
  public static String getAnnotationElementValue(Class annClass, Object instance, String elementName) {
    ...
  }

  /**
   * このヘルパー・メソッドは，DOM ドキュメントをシリアル化するために使用する。
   * このメソッドでは，DOM Level 3 以降で使用できる API が使用されている。
   * これらの API は，1.5 以降の JVM で使用できる。
   * @param document
   * @return 入力 DOM ドキュメントのシリアル化された形式（XML）
   */
  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ことが予想されるときに，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 ("Hello World");¥
"Callee Name="+#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();

これは，次の Java ステートメントに相当します。

((com.myCompany.myFoo)arg1).myMethod();
```

```
@some.class.Foo@foo<com.myCompany.myFoo>.myMethod();

次の Java ステートメントに相当します。

((com.MyCompany.myFoo)some.class.Foo.foo).doSomething();
```

```
#foo = #arg1<bar>.b(); #foo.toString();

次の Java ステートメントを作成します。

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。以下は，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です。

```
#arg1.toString()
```

```
#arg2.getSomething().getSomethingElse()
```

```
#callee.getSomething("foo", #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("Here I am!")
```

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り，ほかの状況で便利なことが予想される **CLApplicationDiscoveryPoint** などのインストゥルメンテーションで必要です。

```
@java.lang.System@out.println("Look out!");
#arg2.getSomething();
```

➤ ローカル・メンバの割り当て

サポートされているデフォルトの「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・メンバを作成する場合，ローカル・メンバの前に「#」記号を入力すると，パーサによってローカル・メンバが作成されます。

```
#myBar = #arg2.getName();¥
#myUpperBar = #myBar.toUpper();¥
"Target Name=http://"+myUpperBar+"/services";
```

➤ 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。

特殊フィールドを使用するには，インストゥルメンテーション・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。

特殊フィールドを使用するには，インストゥルメンテーション・ポイントで，**store-thread** detail キーワードとともに，「##」記号の後ろに特殊フィールドの名前を入力します。

```
# Used by [SOA_Broker_Payload_Handler]
##SOA_Manager_Inbound_Payload=#callee.getRequestDocument();"";
```

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("SOA_Manager_Inbound_Payload");
```

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( ("SOA_Manager_Inbound_Payload");
```

➤ 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許可します。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を比較できます。この構文を使って，リテラル値とプリミティブを比較することはできません。

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();
```

コード・スニペットの安全の確保

デフォルトで，インストゥルメンテーション中に Probe 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，他者が誤ってインストゥルメンテーションを実行し，Probe のオーバーヘッドが著しく高くなるのを防ぎます。

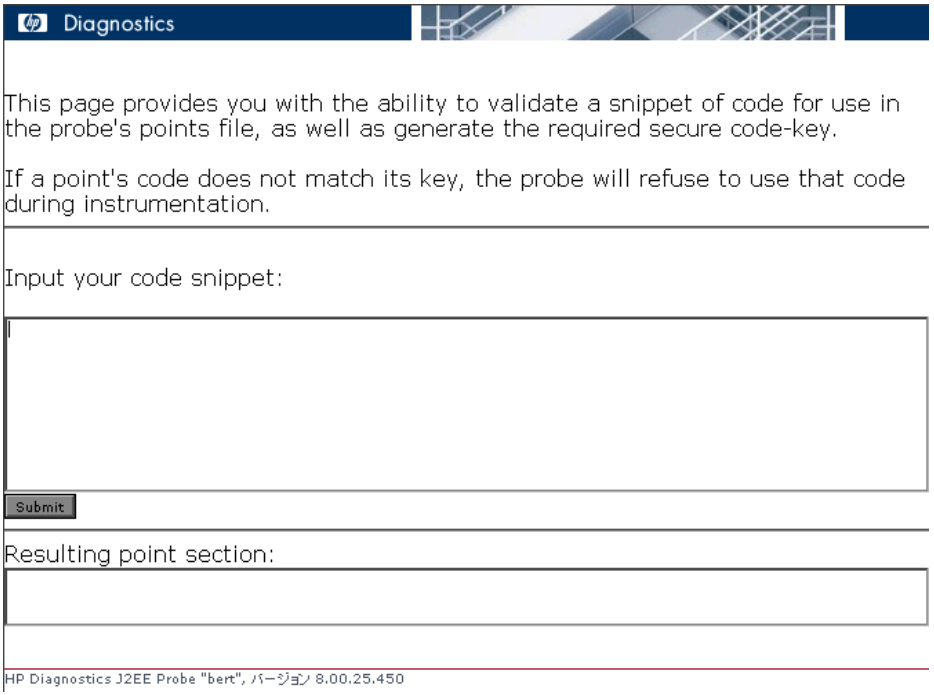
コードキーを生成する際，Diagnostics では，キーを生成する前にコード・スニペットの構文が有効なことを確認します。Diagnostics がアプリケーションをインストゥルメントする際，code-key 引数に入力された値が，ポイントのコード・スニペットに対して計算されたコードキーと一致することを確認します。コードキーが一致しない場合，Diagnostics は，コード・スニペットを無視してインストゥルメンテーション・ポイントを作成しません。

コード・スニペットのコードキーの生成

Java Probe 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

1 ブラウザで, http:// ＜プローブのホスト＞ : ＜プローブのポート＞ /inst/code-key を開きます。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

デフォルトで，Diagnostics は，code-key 引数の値が，アプリケーションインストゥ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。値 false を使って，＜ **Probe 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て，このセキュリティ・チェックを無効にすることができます。

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。

インストゥルメンテーションの例

このセクションの例は，キャプチャ・ポイント・ファイルを作成および変更して，アプリケーションのインストゥ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
- Trended Methods 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除外
- 直接再帰の記録禁止
- 呼び出し側のインストゥルメンテーションの実行
- アロケーション分析の設定
- Light-Weight Memory Diagnostics（LWMD）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
- メソッドが除外されないように指定する
- メソッドが常に除外されるように指定する
- メソッドの CPU 時間の収集を有効にする
- ポイントのインストゥ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）
- 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
- 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

カスタム・レイヤおよびサブレイヤ

➤ 次の例のポイントは，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

➤ 次の例のポイントは，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

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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
```

Trended Methods ビューのキャプチャ・メソッド

➤ 次の例のポイントは，必要なデータをキャプチャして myMethod メソッドの Trended Methods 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

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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

➤ 次の例のポイントは，MyCompany.MyFoo クラスの java.lang.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 = ()Ljava¥lang¥String
layer = FOO/BAR
```

制御された範囲でのキャプチャ

➤ 次の例のポイントは，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，268 ページ「呼び出し側のインストゥ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。次の例のポイントは，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，268 ページ「呼び出し側のインストゥ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");
   }
}
```

➤ 次の例のポイントは，Class1 クラスの「callerMethod」をキャプチャします。

```
[Training-1]
class      = Interface1
method = !.*
signature = !.*
deep_mode  = soft
layer      = Training
```

➤ 次の例のポイントは，Class 1 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 = !.*
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。

➤ メソッド myCompany.myFoo.myMethod() をインストゥルメントしようとしており，「myFoo」にはこれとは別に文字列を返す getComponentName() という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()
```

➤ 次の例のポイントは，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 を使用した場合は，2 番目の引数が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

➤ 以下のポイントを検討します。

```
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。

➤ 次に，以下のポイントを検討します。

```
class      = Statement
method   = execute
layer      = Database/JDBC/Execute
detail     = when-root-rename
```

rootRenameTo プロパティが省略されています。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この場合も「RootRename」になります。

クラスへのフィールドの追加

➤ 以下のポイントを検討します。

```
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，上記の名前（に「display_」というプレフィックスが付いたもの）とそれに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除外

➤ 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ゥルメントします。

```
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あることに注意してください。com.corp.Foo.bar が別のインストゥルメント対象のメソッドを呼び出し，そのメソッド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ゥルメンテーションの実行

➤ 次のポイントでは，（デフォルトの呼び出される側のインストゥルメンテーションではなく）呼び出し側のインストゥ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。

呼び出し側のインストゥルメンテーションを行うには，前述のように，「scope」プロパティを使用する方法もあります。詳細については，268 ページ「呼び出し側のインストゥルメンテーションの使用」を参照してください。

```
class     = com.corp.Foo
method   = bar
detail    = caller
```

アロケーション分析の設定

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，次の 2 つの相違点があります。

➤ 1 つ目の例（**detail = leak**）では，追跡が管理されています。つまり，ユーザがプロファイラで「start」をクリックしたときに追跡を開始し，「stop」をクリックしたときに停止します。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，通常のインストゥ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。ユーザがプロファイラで「start」をクリックしたときに追跡を開始し，「stop」をクリックしたときに停止します。

```
[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）。

Light-Weight Memory Diagnostics（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。

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 User's Guide**』（英語版）の「Collections and Resources View」を参照してください。

```
[Light-Weight Memory Diagnostics]
scope    = !com¥.mycompany¥.mycomponent¥..*
class    = java.lang.Integer
keyword = lwmd
active   = true
```

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

オブジェクト・ライフサイクルの監視を可能にする事前定義のインストゥルメンテーション・ポイントがいくつかありますが，これらはデフォルトでは無効になっています。便宜上，そのうちの 2 つを以下に示します。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 User's Guide**』（英語版）の「Analyzing Memory and Object Lifecycle」の［アロケーション / ライフサイクル分析］タブに関するセクション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 を見つけます。「mercury.enable.resourcemonitor.jdbcResultSet=true」を設定します。

2 （以下に示す）対応する「open」インストゥルメンテーション・ポイントの範囲を指定します。

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・ライフサイクルの監視を実行します。

```
[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¥.rsadapt
er¥.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，インストゥ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。インストゥルメンテーションは行われるが，データは収集されないということです。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，http:// ＜プローブ・ホスト＞ : ＜プローブ・ポート＞ /inst/layer ページにアクセスしてレイヤ **myLayer** を探し，［**Enable**］をクリックします。

```
[My Example]
class      = Example
method = !.*
layer   = myLayer
detail    = disabled
```

インストゥルメンテーションをまったく行わない場合は，**active=false** を使用する必要があります。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。

メソッドが除外されないように指定する

➤ 次の例では，レイテンシの除外が Example.myMethod() に適用されません。

```
[My Example]
class      = Example
method = myMethod
detail    = method-no-trim
```

メソッドが常に除外されるように指定する

➤ 次の例では，メソッド Example.myMethod() が報告されません。

```
[My Example]
class      = Example
method = myMethod
detail    = method-trim
```

メソッドの CPU 時間の収集を有効にする

➤ 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
```

ポイントのインストゥ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。詳細は，次のとおりです。

➤ **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ゥルメント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，インストゥ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・インストゥルメンテーションのオーバーヘッドについて

カスタム・インストゥ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ゥルメントに注意する必要があります。クラスでインストゥルメントを実行するほど，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，インストゥ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・インストゥ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，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，インストゥ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 10ms かかり，インストゥルメンテーションの応答が 11ms 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，インストゥルメント対象のメソッドが頻繁に呼び出されると，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・インストゥルメンテーションを採用しています。これにより，インストゥ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ゥルメンテーションを選択できます。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，およびインストゥ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（JNDI ルックアップ）

のみ，インストゥ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インストゥルメントされます。

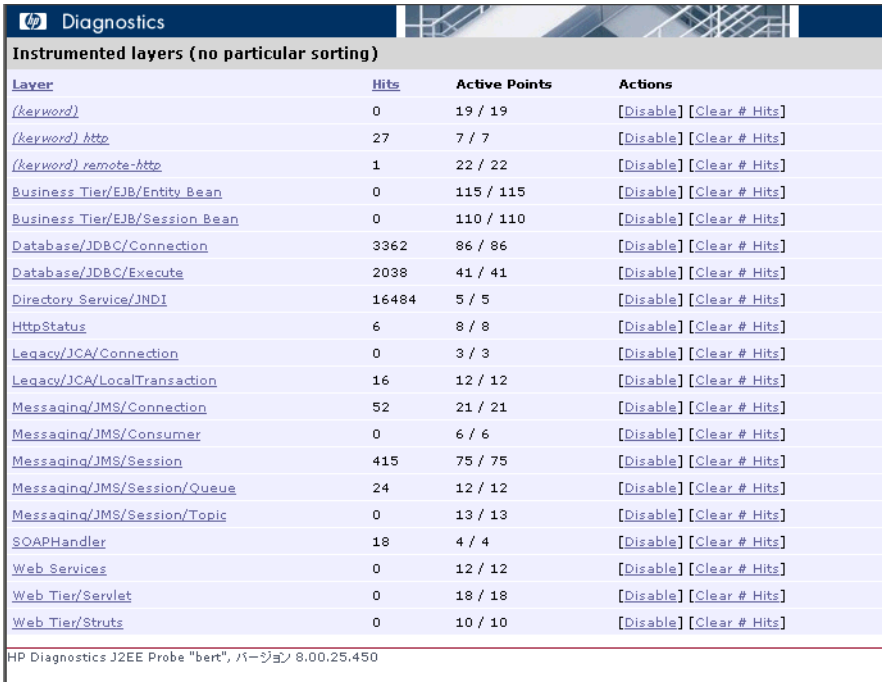
アプリケーションのインストゥルメンテーションをカスタマイズする際は，Diagnostics データのオーバーヘッドも考慮する必要があります。インストゥルメントするメソッドが多いほど，Probe がシリアル化し，ネットワークを通じて Diagnostics Server に渡すデータの量が増えます。監視中のシステムのパフォーマンスに不要な影響を与えないように，Probe のデフォルト設定を変更して Probe に Diagnostics のデータ量を調整させることができます。Probe の設定を不適切に変更すると，Probe がある物理マシンで CPU，メモリおよびネットワークのオーバーヘッドが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 15 章「Java Probe およびアプリケーション・サーバの詳細設定」を参照してください。

インストゥルメンテーションの制御

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，Probe 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ゥルメントできます。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度を要求します。インストゥルメンテーションをメソッド内に置くと，Probe の［Instrumented Layers］ページを使って，データ収集と実行中の CPU およびメソッドのレイテンシ・オーバーヘッドがレイヤごとに制御されます。

［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


Diagnostics

Instrumented layers (no particular sorting)

Layer	Hits	Active Points	Actions
(keyword)	0	19 / 19	[Disable] [Clear # Hits]
(keyword) http	27	7 / 7	[Disable] [Clear # Hits]
(keyword) remote-http	1	22 / 22	[Disable] [Clear # Hits]
Business Tier/EJB/Entity Bean	0	115 / 115	[Disable] [Clear # Hits]
Business Tier/EJB/Session Bean	0	110 / 110	[Disable] [Clear # Hits]
Database/JDBC/Connection	3362	86 / 86	[Disable] [Clear # Hits]
Database/JDBC/Execute	2038	41 / 41	[Disable] [Clear # Hits]
Directory Service/JNDI	16484	5 / 5	[Disable] [Clear # Hits]
HttpStatus	6	8 / 8	[Disable] [Clear # Hits]
Legacy/JCA/Connection	0	3 / 3	[Disable] [Clear # Hits]
Legacy/JCA/LocalTransaction	16	12 / 12	[Disable] [Clear # Hits]
Messaging/JMS/Connection	52	21 / 21	[Disable] [Clear # Hits]
Messaging/JMS/Consumer	0	6 / 6	[Disable] [Clear # Hits]
Messaging/JMS/Session	415	75 / 75	[Disable] [Clear # Hits]
Messaging/JMS/Session/Queue	24	12 / 12	[Disable] [Clear # Hits]
Messaging/JMS/Session/Topic	0	13 / 13	[Disable] [Clear # Hits]
SOAPHandler	18	4 / 4	[Disable] [Clear # Hits]
Web Services	0	12 / 12	[Disable] [Clear # Hits]
Web Tier/Servlet	0	18 / 18	[Disable] [Clear # Hits]
Web Tier/Struts	0	10 / 10	[Disable] [Clear # Hits]

HP Diagnostics J2EE Probe "bert", バージョン 8.00.25.450

［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。

高度な測定

本項の内容

呼び出し側のインストゥルメンテーションの使用

デフォルトで，Diagnostics のすべてのインストゥルメンテーションは呼び出される側のインストゥルメンテーションで，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ゥルメンテーションは，中ではなくインストゥルメントするメソッドへの呼び出しの近くにインストゥ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ゥルメンテーションでは，インストゥルメンテーションの配置を微調整できますが，スコープに指定された各クラスを，ポイントに指定されているクラス / メソッドの参照で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ゥ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ゥルメントします。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ゥ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ゥルメントするには，呼び出し側のインストゥルメンテーションを使用する必要があります。

次の例のポイントで，**javax.xml.parsers.SAXParser** および **javax.xml.parsers.DocumentBuilder** の「parse」メソッドは，任意のクラスから任意（!.*）のメソッドの「parse」の呼び出しの近くにバイトコードを置いてインストゥルメントされます。**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ゥ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
```

➤ 次の例のポイントは，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ゥルメントし，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;
detail = args:1
scope = !com¥.myCompany¥.myFoo¥..*
deep_mode  = soft
layer   = FOO/JNDI
```

注：

➤ このインストゥルメンテーションは，競合する可能性のあるほかのポイントよりも上に置く必要があります。

➤ 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，**＜ Probe のインストール・ディレクトリ＞ /log/ ＜ probeName ＞ / 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。

➤ アプリケーションのパフォーマンス問題に対する最後の優先順位付け手順で，アプリケーションの疑いのあるエリアに呼び出されるすべてのメソッドで，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ゥ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 の呼び出しプロファイルの次のレベルに入力する際に便利な方法です。

次の例のポイントは，「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ゥルメントします。

```
[MethodsCalledByFoo.myMethod]
class = !.*
method = !.*
scope = !com¥.myCompany¥.myFoo¥.myMethod.*
layer = FOO/other
```

以下は，**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別のポイント例です。

```
[SetMethodsCalledByFoo.myMethod]
class = !.*
method = !set.*
scope = !com¥.myCompany¥.myFoo¥.myMethod.*
detail = args:1
layer = FOO/other
```

URI 集計のインストゥルメンテーション

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://server/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。

次のポイント例では，Diagnostics コンソールに後続の ServerRequest を一意に表示できます。

```
http://server/myApplication?page=home
http://server/myApplication?page=openReport
```

```
[HttpCorrelation]
args_by_class=!.*&page
```

次の例のように，URI 解析に複数の URI パラメータを使用できます。

```
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 値の使用を避ける必要があります。

WebLogic 環境で，URI 集計を使うときに **＜ Probe のインストール・ディレクトリ＞ /etc/inst.properties** で **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する必要があります。

RMI インストゥ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（クロス VM）ポイントは，デフォルト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している Probe 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 インストゥルメンテーション

＜ Probe 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 インストゥルメンテーションでクロス VM RMI パフォーマンス・値をキャプチャする方法を制御します。デフォルトで，**weblogic.t3.rmi** は false に設定されているため，一般的な RMI インストゥ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 インストゥ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 インストゥ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 クロス VM をキャプチャします。これにより，Probe は，RMI インストゥルメンテーションが有効な調査対象の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

この例は，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。デフォルトでは，特定のアプリケーション・サーバでしか SOAP ペイロードがキャプチャされないことに注意してください。サポート・マトリクスの詳細については，419 ページ「SOAP の失敗ペイロード・データの設定」を参照してください。

こ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べき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。この場合，次のポイントを用意します。

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プローブに指示します。これは，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（「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」に進んでください。）

```
[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;"";
```

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

この例は，ポイントとコード・スニペットの次の機能を示します。

- 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

Diagnostics では，Web サービスは特別な方法で処理されるため，いくつか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，236 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ゥルメンテーション・ポイントを見つけます。インストゥルメント対象のアプリケーションでは，1 つのクラスからこれらすべてのフィールドにアクセスできるとします。このクラスの名前は com.myCompany.MyWSContext であるとします。上記のすべてのフィールドが設定されたときに，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。これがインストゥルメントすべきメソッドになります。

ここでのインストゥルメンテーション・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
```

このポイントの最初の 3 行は，com.myCompany.MyWSFactory.create(MyWSContext, *) に一致するすべてをインストゥルメントするようにプローブに指示します。

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以下の追加情報（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 つ目の 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@.getThreadContextVal
ue("My_Inbound_Payload");¥
#consumer_id = (#soap_payload == null ? null
:@com.mercury.opal.capture.proxy.ProbeCodeSnippetHelper@.getConsumerId
FromDocument(##WS_inbound_service_name<java.lang.String>,#soap_payloa
d<org.w3c.dom.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，メソッドの第 1 引数（ここでは MyWSContext オブジェクト）をローカル変数（wsContext）に渡します。

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，236 ページ「コード・スニペットの構文」の「Special Fields」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，236 ページ「コード・スニペットの構文」の「Special Fields」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，236 ページ「コード・スニペットの構文」の「Special Fields」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，236 ページ「コード・スニペットの構文」の「Special Fields」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，インストゥルメント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。この詳細については，前の例を参照してください。この例では，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 だった場合は，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，410 ページ「コンシューマ ID の設定」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，236 ページ「コード・スニペットの構文」の「Special Fields」に記載されています。

カスタム・インストゥ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ゥ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（デフォルト）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()
```

ARGS には以下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

以下に InstrumentationPoint 注釈を使ったコードの例を示します。

```
/*
 * (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 がインストゥルメントされ，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）より短くても除外されません。さらに，このメソッドによる包括的な（子を含む）CPU 消費量が測定され，報告されます。

第 10 章

.NET アプリーションのカスタム・インストゥルメンテーション

HP Diagnostics がパフォーマンス測定値の収集を可能にするために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ゥ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

インストゥ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

インストゥ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。インストゥルメンテーションによって，プローブは実行時間の測定，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・インスタンスのインストゥルメンテーション・ポイントは，キャプチャ・ポイント・ファイルに指定されます。

キャプチャ・ポイント・ファイルは，インストゥルメンテーションの範囲を制御できるようにするためのインストゥルメンテーション・メカニズムです。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ゥルメンテーション定義は，どのメソッドをインストゥルメントするか，どのようにインストゥルメントするか，そしてどのインストゥ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。正規表現の使い方については，付録 J「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，デフォルト・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ゥルメンテーションする場合を除き，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 Probe ではほとんどの場合，キャプチャ・ポイント・ファイルで .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測定値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，重要な情報の層に体系づけたり，インストゥルメンテーションの収集動作を制御するのに使われます。

Probe 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，デフォルト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（第 12 章「インストゥルメンテーション・レイヤ」を参照してください）。

.NET Probe 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デフォルトのキャプチャ・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¥** にあります。これらのファイルには **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 デプロイ済み ASP.NET アプリケーション・ドメインを検出すると，検出した各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・インストゥルメンテーション・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，**＜ Probe のインストール・ディレクトリ ＞ ¥etc¥ ＜ ApplicationDomain ＞ .points** ファイルにあります。これらのポイント・ファイルは，.NET Agent によってデフォルトのポイント・ファイルと一緒に読み取られます。

インストール時には，デフォルトのポイント・ファイルである **Asp.Net.points**，**Ado.points**，および **WCF.points** だけが有効になります。以

下のデフォルトの .NET ポイント・ファイルは追加されますが，有効にはなりません。

デフォルトのポイント・ファイル（最初は無効）	インストゥルメンテーション・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
Remoting.points	.NET Remoting
WebServices.points	ASP.NET Web サービス

上記のポイント・ファイルは，**probe_config.xml** ファイルで，それらのファイルに参照を追加して有効にすることができます。probe_config.xml ファイルの各要素の詳細については，第 16 章「.NET Probe 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 Probe インストゥルメンテーションについては，『**HP TransactionVision デプロイメント・ガイド**』を参照してください。

重要：

➤ インストールのアップグレード時に変更が失われるため，デフォルトのポイント・ファイルを変更することはお勧めしません。アプリケーション特有のインストゥルメンテーション・ポイントは，自動または手動で作成されたカスタム・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。

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

以下は，キャプチャ・ポイント・ファイルのポイントの定義に使用できる引数の一覧です。

```
[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 >
keyword               = < キーワード >
layer                 = < レイヤ名 >
layerType             = < レイヤ・タイプ >
```

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，次の引数が必ず含まれなければなりません。

引数	説明
Point-Name	ポイントの一意の名前。
class	class 引数では，インストゥ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します。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ます。
method	method 引数は，インストゥルメントするメソッドの名前を指定します。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ます。

引数	説明
layer	layer 引数では，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します。レイヤは，インストゥ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ます。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。サブレイヤは，サブレイヤに続くほかのスラッシュとサブレイヤ名をエンコーディングして，独自のサブレイヤを持つように定義できます。

以下は，必須引数を含むカスタム・ポイントの例です。

```
[MyCustomEntry_1]
; コメントをここに入力…
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。正規表現の使い方については，付録 J「正規表現の使用」を参照してください。

オプションのポイント・エントリ

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 の存在は，特別なインストゥルメンテーションを示します。特定の機能を有効にするために使用できます（たとえば，WCF キーワードを指定すると WCF 機能が有効になります）。また，ポイントの定義を特別な機能に関連付けるためにも使用できます（たとえば，297 ページ「Remoting」で説明する RemotingServer キーワードなど）。 ➤ **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します。 ➤ **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ます。 ➤ **WCF**：WCF 機能を有効にします。 ➤ **lwmd**：lwmd のインストゥルメンテーションを有効にします。 ➤ **Remoting**：.NET Remoting フレームワークのインストゥルメンテーションを有効にします。 ➤ **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ために使用します。
ignoreClass	**ignoreClass** 引数は，無視するクラスのカンマ区切りのリストを指定するのに使います。**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ゥルメントされません。
ignoreMethod	**ignoreMethod** 引数は，無視するメソッドのカンマ区切りのリストを指定するのに使用します。**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ゥルメントされません。
ignoreTree	**ignoreTree** 引数は，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ゥルメントを無視します。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ます。

引数	説明
deep_mode	**deep_mode** 引数は，サブクラスの処理方法を指定します。この引数では，次の 3 つの値を受け入れます。 ➤ **none** - 値 none は，**deep_mode** 引数を指定しないものに類似します。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で，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ゥルメントするように要求します。 ➤ **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で，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ゥルメントするように要求します。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 モードは，広範なインストゥルメンテーションになり，Probe のオーバーヘッドが非常に高くなる恐れがあります。
scope	**scope** 引数は，インストゥルメンテーションが実行されるコンテキストを制約します。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ります。この引数の値には，任意の有効な正規表現を使用できます。scope 値は，クラス名およびメソッド名のカンマ区切りのリストです。
ignoreScope	**ignoreScope** 引数は，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するために使用します。

引数	説明
layerType	**layerType** 引数は，インストゥルメントしたいくつかのメソッドに対して特殊な処理を指定します。この引数では，次の 3 つの値を受け入れます。 ➤ **trended_method** - Trended Methods ビューに表示するメソッドを特定します。 ➤ **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します。これらは HP Diagnostics によって設定され，変更できません。
signature	**signature** 引数は，シグネチャ（戻り値とパラメータの型）を指定します。たとえば，System.String(System.int32, System.String) などです。任意の有効な正規表現を使用できます。

インストゥルメンテーションの例

このセクションの例は，キャプチャ・ポイント・ファイルを作成および変更して，アプリケーションのインストゥ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

一般的な例

カスタム・レイヤおよびサブレイヤ

➤ 次の例のポイントは，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

➤ 次の例のポイントは，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

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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
```

Trended Methods ビューのキャプチャ・メソッド

➤ 次の例のポイントは，必要なデータをキャプチャして myMethod メソッドの Trended Methods 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

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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
```

➤ 次の例のポイントは，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

➤ 次の例のポイントは，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ゥルメンテーション

デフォルトでは，Diagnostics のすべてのインストゥルメンテーションは呼び出される側のインストゥ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ゥルメンテーションは，中ではなくインストゥルメントするメソッドへの呼び出しの近くに測定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ゥルメンテーションでは，インストゥルメンテーションの配置を微調整できますが，スコープに指定された各クラスを，ポイントに指定されているクラス / メソッドの参照で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。

インストゥ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ゥルメンテーションを示しています。

- 次の例のポイントは，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。次の例のポイントは，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
```

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");
   }
}
```

➤ 次の例のポイントは，Class1 クラスの「callerMethod」をキャプチャします。

```
[Training-1]
class      = Interface1
method = !.*
deep_mode  = soft
layer      = Training
```

➤ 次の例のポイントは，Class 1 のすべてのメソッドをキャプチャします（つまり，「callerMethod」，「calleeMethod1」および「calleeMethod2」）。

```
[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

このセクションでは，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ゥ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。このセクションの文脈では，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ゥルメンテーションは Windows アプリケーションのインストゥルメンテーションとみなされます。わかりやすくするため，これらのアプリケーションを Windows アプリケーションと呼びます。

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ときに知っておくべき重要なポイントは，.NET Probe 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。その基本的な原因は，Windows アプリケーションがすぐに終了すると，プローブが Diagnostics Server または .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にあります。

次の例は，非常に簡単な Windows アプリケーションです。この例を使って，Windows アプリケーションのインストゥルメンテーションを設定するときに検討すべき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)
            {
// 何らかの処理を行う

                  // コマンド・ラインを読み込んで終了する
                  clReader myClReader = new clReader();
                  String cl;
                  cl = myClReader.readCl();
            }
      }
      // コマンド・ラインの読み込み処理
      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，Main() で 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，終了します。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，476 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と仮定して，次のコマンドを実行します。

```
Reflector.exe Hello_dotNet.exe
```

出力が stdout に送られ，ほかの情報とともに，次のような推奨される 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
```

推奨の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() のようなメソッドがある場合は別です。つまり，インストゥルメントしても，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，インストゥルメンテーション・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ゥ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 のほかのメソッドは（あれば）インストゥルメントします。

インストゥ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 アプリケーションをインストゥルメントするように .NET Probe を設定するには，次の XML を **¥etc¥probe_config.xml** ファイルに追加する必要があります。これは，probe_config.xml 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 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。インストゥルメンテーション，記録，およびモードは省略可能です。インストゥ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，次のインストゥルメントの設定を使用できます。

```
<instrumentation>
   <logging level="points ilasm" />
</instrumentation>
```

Remoting

.NET Probe 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ゥルメンテーションをサポートするカスタム・インストゥ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
- .NET 1.1 および 2.0 Remoting アプリケーション

設定

デフォルトでは，.NET Probe は Remoting アプリケーションをインストゥ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・インストゥルメンテーション・ポイントを追加する必要があります。

Remoting に関しては，次の 2 つのキーワードがあります。

Remoting：Remoting キーワードは，Remoting フレームワークのさまざまなポイントをインストゥルメンテーションできるようにするために使用します。

RemotingServer：RemotingServer キーワードは，Remoting メソッドをインストゥルメンテーションし，そのクラスのメソッドのインストゥ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ゥルメンテーションとを区別するクラスを特定するために使用します。

クライアントの例

次の例は，非常に簡単な Windows アプリケーションの例です。この例を使って，Remoting クライアント・アプリケーションのインストゥルメンテーションを設定するときに検討すべき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));
      }
}
```

293 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 Remoting 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ゥルメントするように Probe を設定するには，次の XML を **probe_config.xml** ファイルに追加する必要があります。

```
<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 の測定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
```

上記の設定は推奨設定としては適切ですが，これではデータを生成するインストゥルメンテーションは作成されません。その理由は，293 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
```

これで有用な設定になり，データも生成されますが，もう少し設定を調整して，設定の精度を高めることをお勧めします。その理由は，主にプローブのパフォーマンスにあります。インストゥ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ゥルメントされるアプリケーションに与える影響は大きくなります。したがって，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，測定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ゥ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，インストゥルメントする必要性はほとんどありません。

しかし，doit メソッドは Remoting クライアントの呼び出しをラップするため，インストゥルメント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

次の Windows アプリケーションの例を使って，Remoting サーバ・アプリケーションのインストゥルメンテーションを設定するときに検討すべき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。

リモート処理可能オブジェクトの C# コード・スニペット：

```
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;
            }
      }
}
```

SimpleConsoleServer.exe の C# コード・スニペット：

```
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ゥ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** ファイルに追加する必要があります。

```
<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。

Remoting 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，Remoting サーバのインストゥルメンテーションに関するもので，必要に応じて追加できます。その他のセクションは，Remoting メソッドまたは Remoting 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ゥ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。

リモート処理可能オブジェクトは，いずれかのアセンブリにあります。ここでは，RemotableObjects.dll にあるとします。この場合も，RemotableObjects.dll に対して Reflector を実行し，以下を含む出力を確認します。

```
---------------------------------------------------------------------
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。まず，リモート処理可能オブジェクトをインストゥ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る必要があることを観察します。上記の Reflector の出力からは，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ゥ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，Probe 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ゥルメントされます。したがって，インストゥルメントする Remoting メソッドを指定する必要はありません。すべての Remoting メソッドがインストゥルメンテーションされます。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ゥルメントされるメソッドのインストゥルメントが，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ゥ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 Remoting サーバのセクションを作成します。このセクションを作成する必要があるのは，Remoting 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。この場合も，SimpleConsoleServer.exe に対して Reflector を実行し，以下を含む出力を確認します。

```
---------------------------------------------------------------------
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()
```

この場合も，293 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，Remoting メソッドまたは Remoting 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ゥ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・インストゥルメンテーションのオーバーヘッドについて

カスタムインストゥ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測定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，インストゥ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・インストゥ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，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，インストゥ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 10ms かかり，インストゥルメンテーションの応答が 11ms 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，インストゥルメント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ゥルメンテーションを採用しています。これにより，インストゥ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ゥルメンテーションを選択できます。インストゥ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，インストゥ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，インストゥルメントされます。

アプリケーションのインストゥ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。インストゥルメントするメソッドが多いほど，Probe がシリアル化し，ネットワークを通じて Diagnostics Server 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，Probe のデフォルト設定を変更して Probe に Diagnostics のデータ量を調整させることができます。Probe の設

定を不適切に変更すると，Probe 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 17 章「.NET Probe の詳細設定」を参照してください。

第 11 章

Profiler からの Java アプリケーションのインストゥルメントと Probe の設定

Java Profiler の［設定］タブを使って，インストゥルメンテーション・ポイントを管理したり，Java Probe のいくつかのプロパティを設定したりできます。

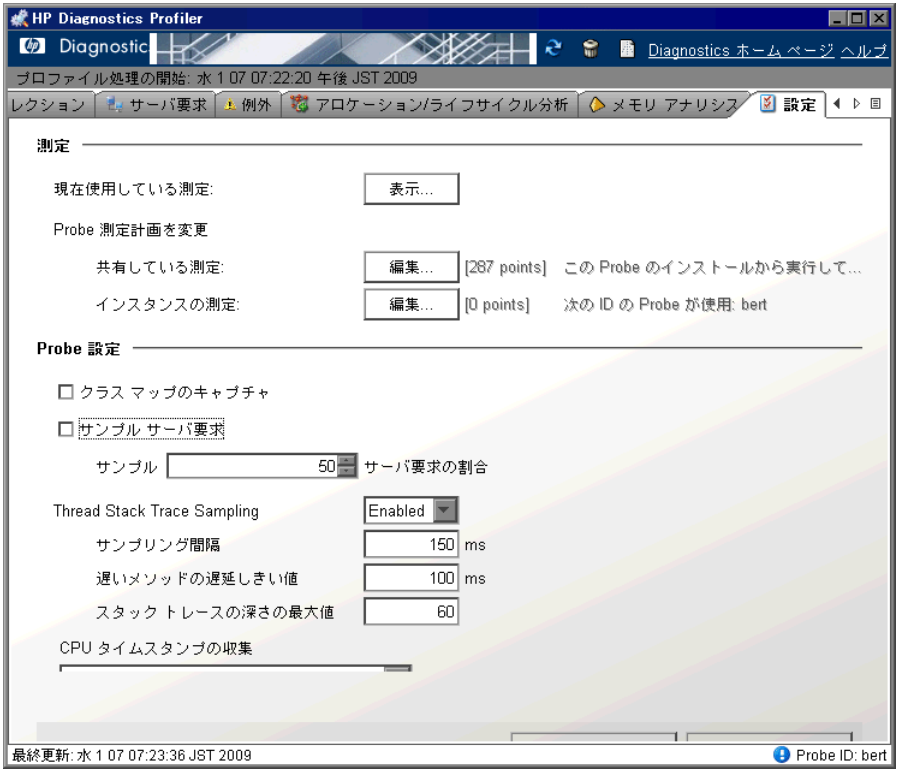
本章の内容

Java Profiler の［設定］タブについて

Java Diagnostics Profiler の［設定］タブは，キャプチャ・ポイント・ファイルやプロパティ・ファイルを手動で編集することなくインストゥルメンテーション・ポイントと Probe 設定の管理する手段となります。

プロファイル処理を Probe で開始しているかどうかに応じて，Java Diagnostics Profiler から［設定］タブにアクセスできます。

Java Diagnostics Profiler の［設定］タブは，次の図のように［測定］セクションと［Probe 設定］セクションに分かれています。

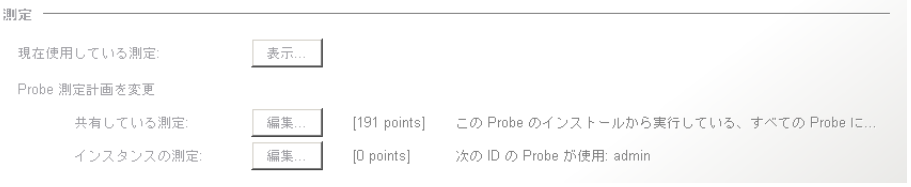


［測定］セクションでは，Probe が監視しているアプリケーションのインストゥルメンテーションを表示および更新できます。［測定］セクションからアクセスする［編集］ダイアログでは，Diagnostics でアプリケーションのインストゥルメンテーションに使うキャプチャ・ポイント・ファイルに定義したように，インストゥルメンテーション・ポイントを表示および編集することができます。

［設定］タブの［Probe 設定］セクションでは，アプリケーションのパフォーマンス測定値をキャプチャするときに Probe が使用するクラス・マップのキャプチャ，サーバ要求のサンプリング，スレッドのスタック・トレースのサンプリング，および CPU タイムスタンプを設定できます。［Probe 設定］セクションの詳細については，323 ページ「［設定］タブからの Probe 設定の保守」を参照してください。

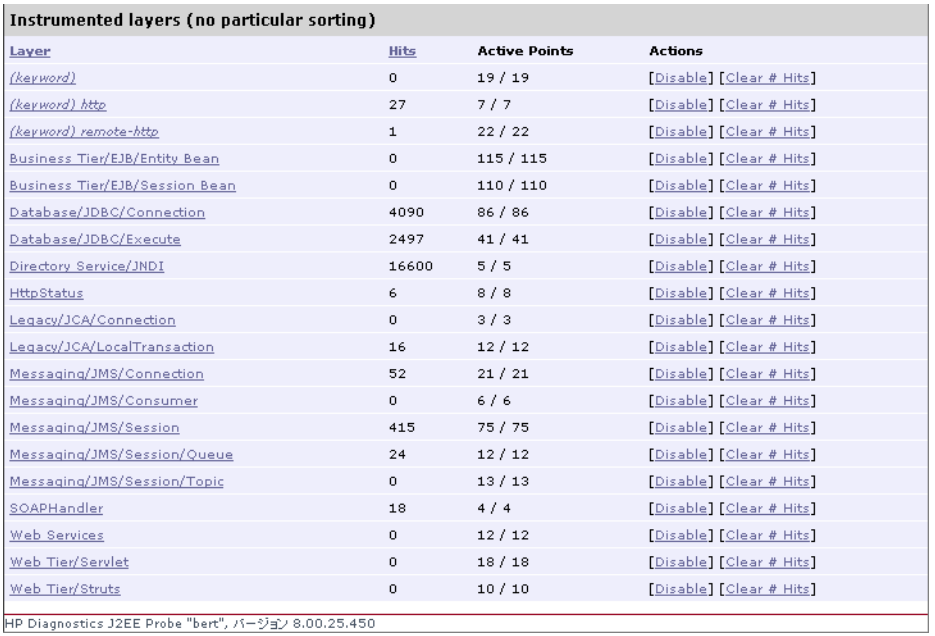
［設定］タブからのインストゥルメンテーションの保守

［設定］タブの［測定］セクションから，Diagnostics がアプリケーションに適用するインストゥルメンテーションを制御できます。



現在のインストゥ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ゥ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，［設定］タブの［測定］セクションにある［**表示 ...**］をクリックします。次の図のように，Profiler に［Instrumented layers］ページが表示されます。



Instrumented layers (no particular sorting)

Layer	Hits	Active Points	Actions
(keyword)	0	19 / 19	[Disable] [Clear # Hits]
(keyword) http	27	7 / 7	[Disable] [Clear # Hits]
(keyword) remote-http	1	22 / 22	[Disable] [Clear # Hits]
Business Tier/EJB/Entity Bean	0	115 / 115	[Disable] [Clear # Hits]
Business Tier/EJB/Session Bean	0	110 / 110	[Disable] [Clear # Hits]
Database/JDBC/Connection	4090	86 / 86	[Disable] [Clear # Hits]
Database/JDBC/Execute	2497	41 / 41	[Disable] [Clear # Hits]
Directory Service/JNDI	16600	5 / 5	[Disable] [Clear # Hits]
HttpStatus	6	8 / 8	[Disable] [Clear # Hits]
Legacy/JCA/Connection	0	3 / 3	[Disable] [Clear # Hits]
Legacy/JCA/LocalTransaction	16	12 / 12	[Disable] [Clear # Hits]
Messaging/JMS/Connection	52	21 / 21	[Disable] [Clear # Hits]
Messaging/JMS/Consumer	0	6 / 6	[Disable] [Clear # Hits]
Messaging/JMS/Session	415	75 / 75	[Disable] [Clear # Hits]
Messaging/JMS/Session/Queue	24	12 / 12	[Disable] [Clear # Hits]
Messaging/JMS/Session/Topic	0	13 / 13	[Disable] [Clear # Hits]
SOAPHandler	18	4 / 4	[Disable] [Clear # Hits]
Web Services	0	12 / 12	[Disable] [Clear # Hits]
Web Tier/Servlet	0	18 / 18	[Disable] [Clear # Hits]
Web Tier/Struts	0	10 / 10	[Disable] [Clear # Hits]

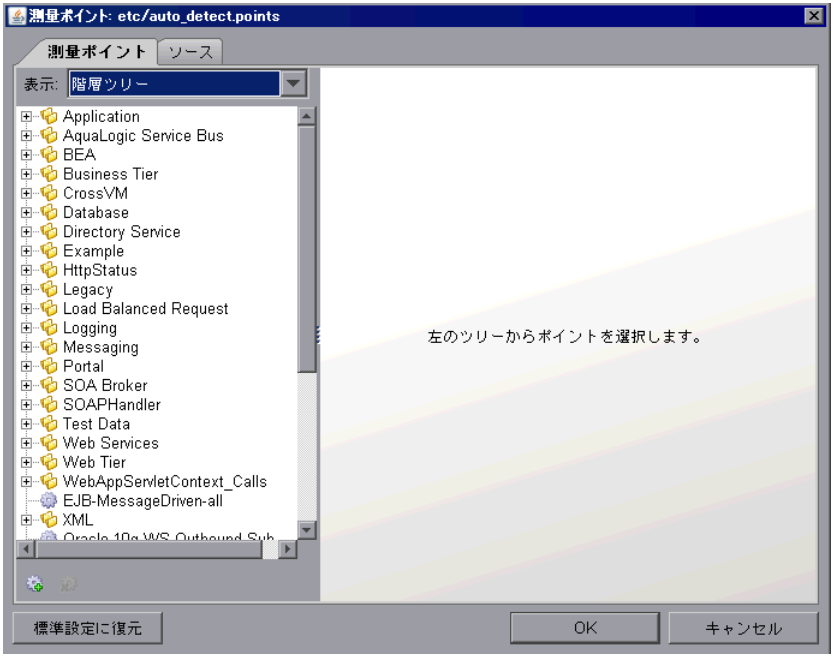
HP Diagnostics J2EE Probe "bert", バージョン 8.00.25.450

［Instrumented layers］ページには，レイヤのインストゥルメンテーション・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と一緒に，インストゥルメントされたレイヤが一覧表示されます。次の表では，このページの列について説明します。

列	説明
Layer	インストゥルメントされたレイヤの一覧が表示されます。この列のレイヤ名はリンクになっていて，Probe 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移動することができます。**注**：実際にインストゥ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ます。
# 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ます。［**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数を表示します。
Actions	一覧のレイヤの情報を操作するためのリンクが表示されます。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。インストゥ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。ポイントのデフォルトの有効ステータスを変更する場合は，226 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。 ➤ **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。

インストゥルメンテーション・ポイントの保守

アプリケーションで監視する Probe を伝えるインストゥ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［設定］タブに移動して，［共有している測定］または［インスタンスの測定］で［**編集 ...**］をクリックします。次のような［測量ポイント］ダイアログが表示されます。

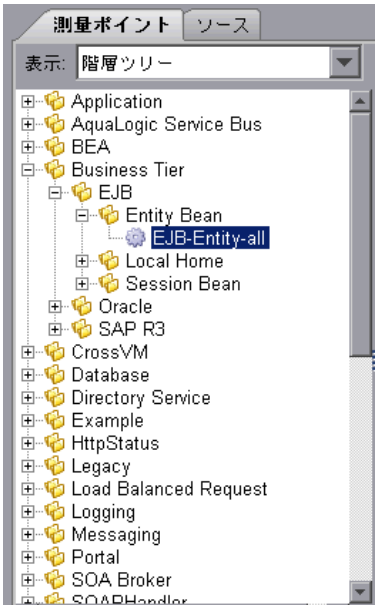


インストゥルメンテーションを編集する方法は 2 つあります。［測量ポイント］タブでポイントのリストまたはツリーを視覚的に使うか，または［ソース］タブのキャプチャ・ポイント・ファイルのソースを通じて編集します。

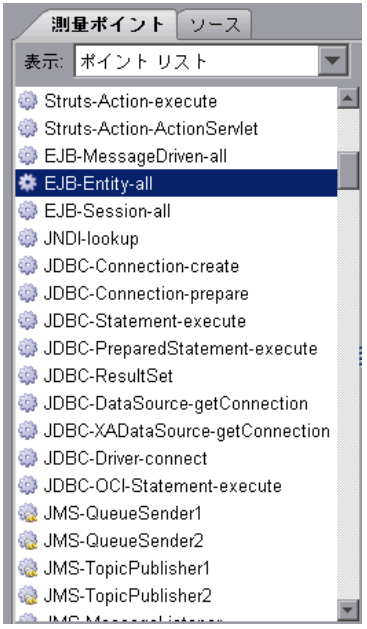
既存のポイントの選択および表示

［測量ポイント］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。

［**表示**］ドロップダウンから［**階層ツリー**］を選択すると，Probe は，次の図のように，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します。



［**表示**］ドロップダウンから［**ポイント リスト**］を選択すると，Probe は，次の図のように，キャプチャ・ポイント・ファイルを昇順のアルファベット順に表示します。

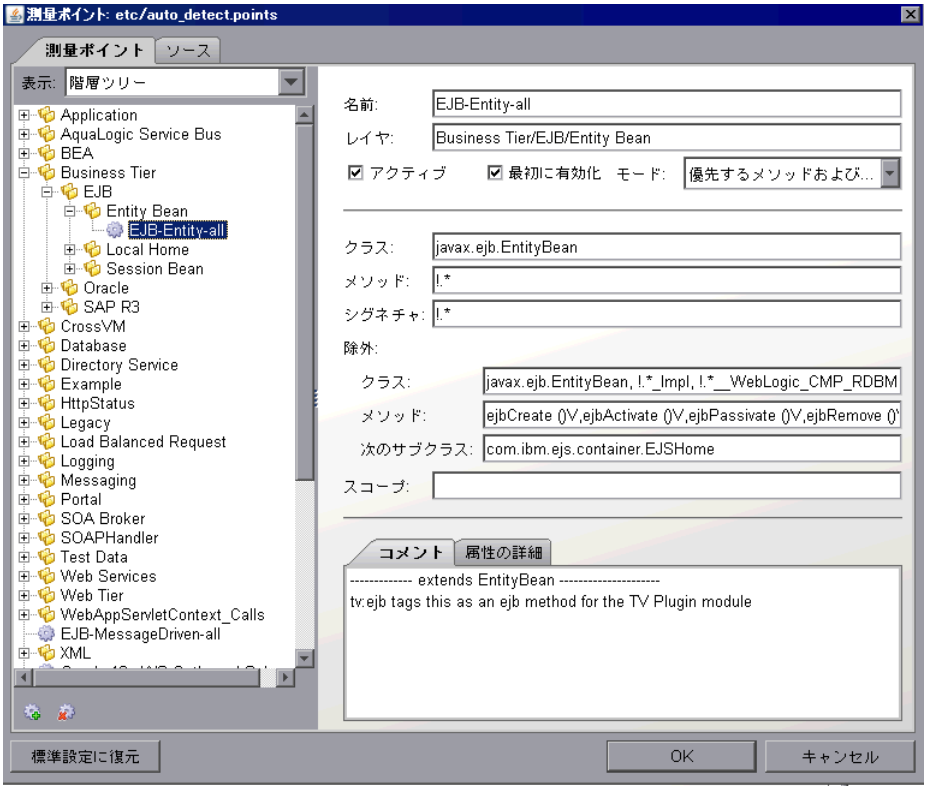


表示または保守するポイントが見つかったら，ナビゲーション・バーでポイントを選択します。Probe には，［表示 / 編集］パネルに選択したポイントの詳細が表示されます。このパネルでポイントを操作できます。

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。次の図のように，Profiler には，［表示 / 編集］パネルの［測量ポイント］タブにポイントの詳細情報が表示されます。



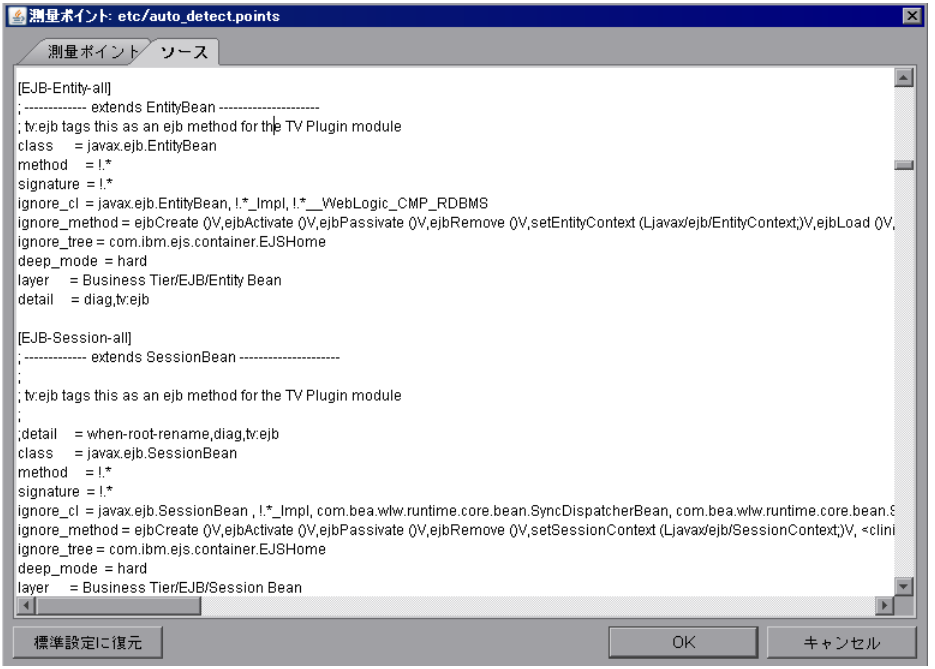
キャプチャ・ポイント・ファイルのポイントを定義するときに一般的に使われる引数は，別のデータ・フィールドとして表示され，簡単に必要な変更を加えることができます。また，［測量ポイント］タブの下部にある［属性の詳細］タブに，詳細な引数が表示されます。ポイントのコメントは，［コメント］タブに表示されます。

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，226 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。

次の表は，［測量ポイント］に表示される引数と，このセクションに記載されているキャプチャ・ポイント・ファイルの相互参照表として役立ちます。また，この相互参照表は，［ソース］タブに表示されるポイントを確認する際も便利です。

［測量ポイント］タブ	**キャプチャ・ポイント・ファイルの引数**
名前	Point-Name
Layer	layer
アクティブ	keyword = disabled
モード	deep_mode
クラス	class
メソッド	method
シグネチャ	signature
クラスの除外	ignore_cl
メソッドの除外	ignore_method
次のサブクラスの除外	ignore_tree
スコープ	scope
属性の詳細	detail，code-key，tierDefinition，layerType，merge_classes，merge_url_to，ignore_tree，ignoreScope

次の図は［ソース］タブの例です。



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

1 ［測量ポイント］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［設定］タブからすべてのインストゥルメンテーション・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，［測量ポイント］ダイアログを閉じます。

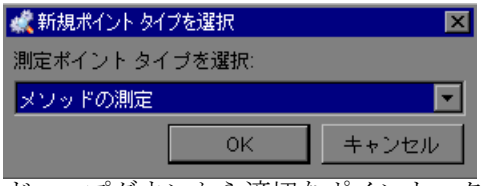
4 ［**変更を適用**］をクリックして，［設定］タブを使って加えた変更をすべて適用します。

新しいポイントの追加

インストゥ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

1 ［新規ポイント］をクリックします。次の図のように，Profiler に［新規ポイント タイプを選択］ダイアログ・ボックスを表示します。



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。

［測量ポイント］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［設定］タブからすべてのインストゥルメンテーション・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，［測量ポイント］ダイアログを閉じます。

5 ［**変更を適用**］をクリックして，［設定］タブを使って加えた変更をすべて適用します。

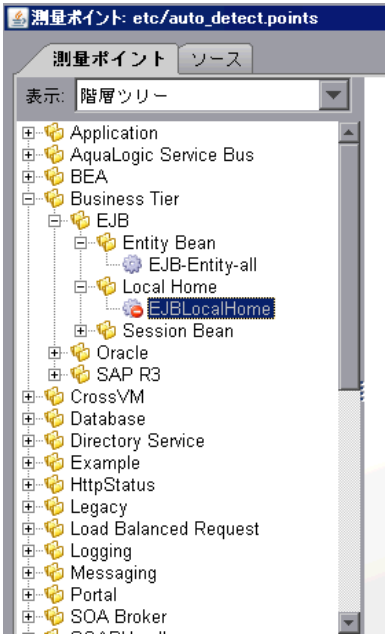
OVTA のようなポイントのアクティブ化

ServletFilter および EJB ローカル・ホーム・メソッドに対する Java Probe のインストゥルメンテーションにポイントが含まれます。これらのインストゥルメンテーション・ポイントは，OVTA（OpenView Transaction Analyzer）Java モニター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。

デフォルトで，EJBLocalHome，ServletFilter および関連する HttpCorrelation2 インストゥルメンテーション・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ゥルメンテーション・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ために，**auto_detect.points** ファイルで active=true に設定する必要があります。これは，UI から操作するか，またはファイルを直接編集して行います。

315 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，インストゥルメンテーション・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。

［**OK**］をクリックして，［測量ポイント］ダイアログを閉じます。

［**変更を適用**］をクリックして，［設定］タブを使って加えた変更をすべて適用します。ポイントをアクティブにするために，Probe（アプリケーション・サーバ）を再起動するように指示されます。

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，デフォルトのインストゥルメンテーションを復元して，より堅牢なインストゥ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。

測定にデフォルトの設定を復元するには

1 ［**標準設定に復元**］をクリックします。

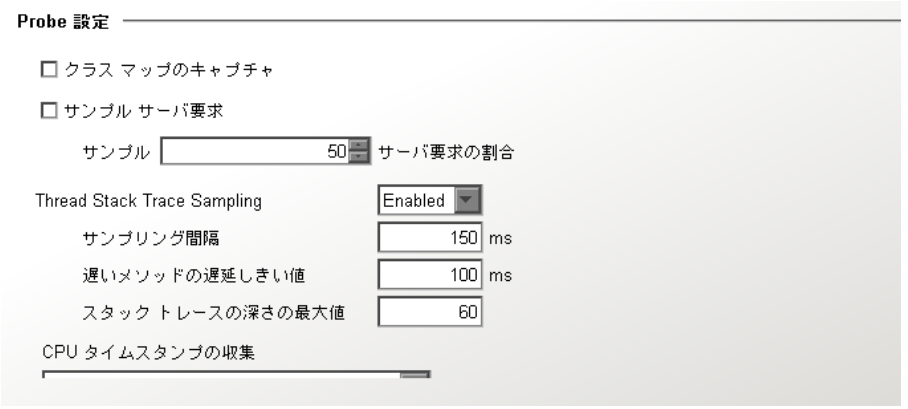
Profiler の［設定］タブからすべてのインストゥルメンテーション・ポイントの更新を適用するまで，インストゥルメンテーション・ポイントは，実際にキャプチャ・ポイント・ファイルに追加されません。

2 ［**OK**］をクリックして，［測量ポイント］ダイアログを閉じます。

3 ［**変更を適用**］をクリックして，［設定］タブを使って加えた変更をすべて適用します。

［設定］タブからの Probe 設定の保守

［設定］タブの［Probe 設定］セクションでは，アプリケーションのパフォーマンス測定値をキャプチャするときに Probe が使用するクラス・マップのキャプチャ，サーバ要求のサンプリング，スレッドのスタック・トレースのサンプリング，および CPU タイムスタンプを設定できます。



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る上で役立ちます。クラス・マップ情報は，312 ページ「現在のインストゥルメンテーションの確認」の説明に従って，［Instrumented layers］ページに記載されているレイヤでドリルダウンすることで表示できます。クラス・マップを使用すると，マップが Probe のホスト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。

クラス・マップ・キャプチャを有効にするには

1 ［**クラス マップのキャプチャ**］チェック・ボックスをオンにします。

2 ［設定］タブに変更を加えたら，［**変更を適用**］をクリックします。

3 Probe を復元して，Probe でクラス・マップのキャプチャを開始できるようにします。

クラス・マップ・キャプチャを無効にするには

1 ［**クラス マップのキャプチャ**］チェック・ボックスをオフにします。

2 ［設定］タブに変更を加えたら，［**変更を適用**］をクリックします。

3 Probe を復元して，Probe でクラス・マップのキャプチャを停止できるようにします。

サーバ要求サンプリングの制御

サーバ要求サンプリングによって，Diagnostics は，アプリケーションで実行されたすべてのサーバ要求のサブセットに関する詳細情報を収集することができます。これにより，サンプリングされていないサーバ要求にキャプチャされる情報がないため，Probe のオーバーヘッドが低減します。サンプリング・レートが低すぎない限り，サンプリングされたすべてのサーバ要求の平均結果は，アプリケーションのパフォーマンスを理解するために，妥当に正確で重要な結果を提供する必要があります。

いつサンプリングを有効にするか，何パーセントのサーバ要求をサンプリングするかを制御します。サーバ要求のサンプリングを制御するプロパティは，**dynamic.properties** にあります。

- enable.server.request.sampling
- server.request.sampling.rate
- server.request.sampling.method
- sample.colored.requests

Profiler からサーバ要求のサンプリングを有効にするには

1 ［**サンプル サーバ要求**］チェック・ボックスをオンにします。

2 スピナを使って，サンプリングの割合を設定します。

3 ［設定］タブに変更を加えたら，［**変更を適用**］をクリックします。

変更はただちに有効になります。Probe を再起動する必要はありません。

Profiler からサーバ要求のサンプリングを無効にするには

1 ［**サンプル サーバ要求**］チェック・ボックスをオフにします。

2 ［設定］タブに変更を加えたら，［**変更を適用**］をクリックします。

変更はただちに有効になります。Probe を再起動する必要はありません。

スレッドのスタック・トレースのサンプリング設定

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どのメソッドが実行されたかが Call Profile ビューに表示されます。これは，この期間中にインストゥ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 User's Guide**』（英語版）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。

スレッドのスタック・トレースのサンプリングを有効にして設定するには，複数のプロパティを使用します。

dynamic.properties には，次のプロパティがあります。

- **enable.stack.trace.sampling** - スレッドのスタック・トレースのサンプリングを有効にするためのプロパティ。取り得る値は false，auto，true で，auto がデフォルトです。

  動的なプロパティである enable.stack.trace.sampling を auto に設定すると，選択された（認定された）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，このプロパティを true に設定する必要があり，JVM で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 のリリース・ノート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までに，インストゥルメンテーション・ポイントにヒットせずにインストゥルメント対象のメソッドを実行しなければならない最短時間。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** - これは，連続する次のサンプリング試行を行うまでに経過しなければならない時間です。

  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）。

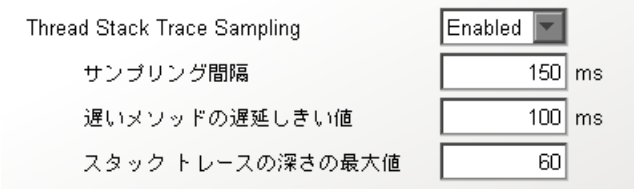
- **stack.trace.depth.max** - これは，JVM によって取得されるスタック・トレースの深さに対する制限です。ほとんどの場合，この値を調整する必要はありません。

dispatcher.properties には，次のプロパティがあります。

- **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を除きます。true に設定すると，追加作成される呼び出しツリー・ノードの数が減りますが，追加ノードの呼び出し数が既知で小さ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は呼び出し数をまったく表示できません。
- **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。

上記のプロパティは，すべて動的に変更できます。

最初の（dynamic.properties の）4 つのプロパティは，Diagnostics Java Profiler の［設定］タブを使ってリモートで変更できます。



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，時々このメソッドの完了までに 1.4 秒かかります。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，約 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っているため，それらをインストゥ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。これにより，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。しかし，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ゥ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。

この場合は，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，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を確認します。

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，多くの呼び出しプロファイル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 Probe で提供されるデフォルトインストゥ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ゥルメンテーションを追加すること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 ms 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。しかし，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 300 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ゥルメンテーション・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

Question 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ません。なぜですか。

Answer : 以下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 以降を使用しているかどうかを確認します。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が発信呼び出しかどうかを確認します。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ゥルメンテーション・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が no-layer-recurse としてマークされているかどうかを確認します。こ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，minimum.method.latency，あるいはその両方の値を小さくしてみます。呼び出しプロファイルに表示される最後のメソッドが除外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，あるいはその両方を行います。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 の値を小さくしてみます。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す。Java コードは，ガベージ・コレクション中には実行されません。これには，スタック・トレースのサンプリング・コードも含まれます。

Question 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。

Answer：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，タイマーの精度，および 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の，3 つの要因が影響します。

Question 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。

Answer：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。何らかの結果を得るためには，注目するサーバ要求ごとに少なくとも数個のサンプルが得られるように計画し，場合によってはさらにほかのサンプリング・パラメータも調整する必要があります。

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 時間を計算するのに使います。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。詳細については，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 399 ページ「VMware 上で実行中の Probe の時刻の同期」を参照してください。

デフォルトでは，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 時間測定値の収集は，プロパティ・ファイルで（407 ページ「CPU 時間測定値の収集の設定」参照）または下記の Java Diagnostics Profiler UI を使って設定できます。

1 **Profiler** UI で，［**設定**］タブを選択します。この Probe 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 ［設定］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。

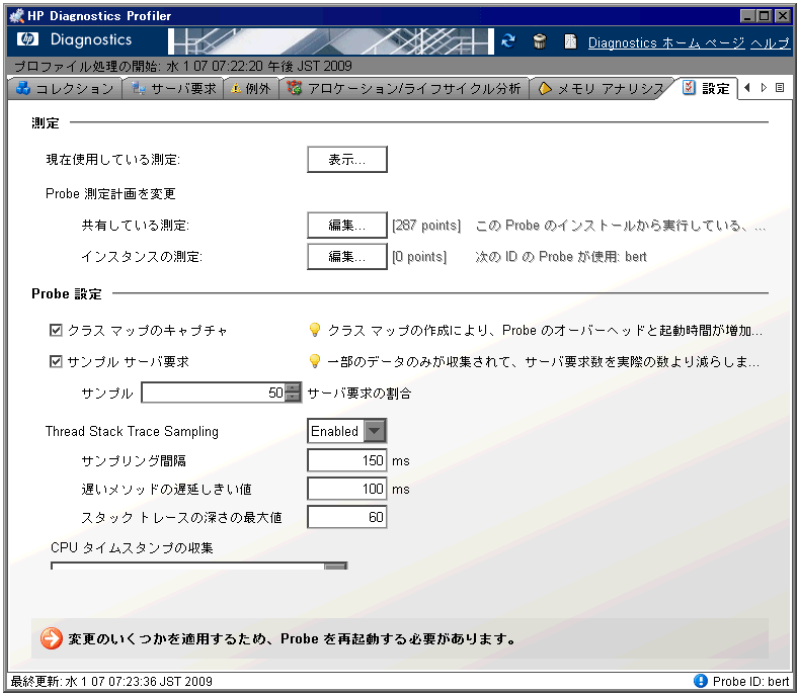
CPU タイムスタンプ収集メソッド	**説明**
なし	CPU タイムスタンプはありません。
サーバ要求に対してのみ	CPU タイムスタンプはサーバ要求に対してのみ収集されます。
サーバ要求とポートレット・メソッドに対して	CPU タイムスタンプはすべてのサーバ要求とポータル・コンポーネントでインストゥルメントされたライフサイクル・メソッドに対して収集されます（layertype=portlet）。
サーバ要求とすべてのメソッドに対して	CPU タイムスタンプはすべてのサーバ要求とすべてのメソッドに対して収集されます。

3 ［設定］タブに変更を加えたら，［**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。Probe を再起動する必要はありません。

Probe の再起動

［設定］タブで［**変更を適用**］をクリックすると，［設定］タブの［測定］セクションと［Probe 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。行った変更を有効にするために Probe を再起動する必要がある場合は，［設定］タブの左下角にリマインダとしてメッセージが表示されます。Probe の再起動が必要な唯一のフィールドは，［クラス マップのキャプチャ］チェック・ボックスです。



第 12 章

インストゥルメンテーション・レイヤ

デフォルトのインストゥルメンテーション・レイヤは，プローブを最初にインストールしたときにインストゥルメンテーション・ポイントに定義されます。

本章の内容

インストゥルメンテーション・レイヤについて

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。

ポイントおよびレイヤの保守については，Probe のカスタムインストゥルメンテーションに関する項を参照してください。

Diagnostics レイヤについて

次の表は，一般的な Java EE と .NET のクラスおよびメソッドに定義されているデフォルトのレイヤとサブレイヤの一覧で，クラスとメソッドがレイヤに割り当てられているキャプチャ・ポイント・ファイルのポイントを示します。この情報は，キャプチャ・ポイント・ファイルのポイントを解釈およびカスタマイズする上で役立ちます。

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	ポイントの定義
Web 階層	JSP Servlets Struts		JSP-_jspService Servlet-all Struts-Action
JSP		Web 階層	JSP-_jspService
Servlets		Web 階層	Servlet-all
Struts		Web 階層	Struts-Action
ビジネス階層	EJB		EJB-Entity-all EJB-Session-all
EJB	Entity Bean Session Bean	ビジネス階層	EJB-Entity-all EJB-Session-all
Entity Bean		EJB	EJB-Entity-all
Session Bean		EJB	EJB-Session-all
Directory Service	JNDI		JNDI-lookup
JNDI		Directory Service	JNDI-lookup

レイヤ	サブレイヤ	親レイヤ	ポイントの定義
Database	JDBC		JDBC-Connection-create JDBC-Connection-prepare JDBC-Statement-execute JDBC-PreparedStatement-execute JDBC-ResultSet JDBC-DataSource-getConnection JDBC-XADataSource-getConnection JDBC-Driver-connect JDBC-OCI-Statement-execute
JDBC	Execute Connections	Database	JDBC-Connection-create JDBC-Connection-prepare JDBC-Statement-execute JDBC-PreparedStatement-execute JDBC-ResultSet JDBC-DataSource-getConnection JDBC-XADataSource-getConnection JDBC-Driver-connect JDBC-OCI-Statement-execute
Execute		JDBC	JDBC-Connection-create JDBC-Connection-prepare JDBC-Statement-execute JDBC-PreparedStatement-execute JDBC-ResultSet JDBC-OCI-Statement-execute
Connections		JDBC	JDBC-DataSource-getConnection JDBC-XADataSource-getConnection JDBC-Driver-connect

レイヤ	サブレイヤ	親レイヤ	ポイントの定義
Messaging	JMS		JMS-MessageProducer JMS-MessageListener JMS-MessageConsumer
JMS	Producer Listener Consumer	Messaging	JMS-MessageProducer JMS-MessageListener JMS-MessageConsumer
Producer		JMS	JMS-MessageProducer
Listener		JMS	JMS-MessageListener
Consumer		JMS	JMS-MessageConsumer

.NET レイヤ

レイヤ	サブレイヤ	親レイヤ	ポイントの定義
Web 階層	ASP.NET		
Database	ADO		ADO 1 ADO 2 ADO 3 ADO 4
ADO	Execute Connection	Database	ADO 1 ADO 2 ADO 3 ADO 4
Execute		ADO	ADO 1 ADO 2 ADO 3
Connection		ADO	ADO 4

レイヤ	サブレイヤ	親レイヤ	ポイントの定義
Messaging	MSMQ		MSMQ Synchronous MSMQ Asynchronous MSMQ ReceiveCompleted EventHandler MSMQ PeekCompleted EventHandler
MSMQ		Messaging	MSMQ Synchronous MSMQ Asynchronous MSMQ ReceiveCompleted EventHandler MSMQ PeekCompleted EventHandler

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・レイヤについては，『**HP Diagnostics User's Guide**』（英語版）を参照してください。

第 13 章

ソリューション・テンプレートの使用

ソリューション・テンプレートは，複数の主要アプリケーション・ミドルウェア・サーバおよびフレームワークのために作成されたキャプチャ・ポイントに含まれる，定義済みのインストゥルメンテーション・ポイントです。ソリューション・テンプレートを使うと，Diagnostics は，プラットフォーム特有のパフォーマンス測定値をキャプチャし，Diagnostics ビューにそれらの測定値を表示することができます。

本章の内容

ソリューション・テンプレートについて

HP Diagnostics では，次のサーバおよびフレームワークのためのソリューション・テンプレートを提供します。

- **WebLogic**：WebLogic Portal Server ソリューション・テンプレートでは，BEA WebLogic Portal Server WL 8.1 SP3，SP4，SP5，9.2 の測定値を収集します。
- **WebSphere**：WebSphere Portal Server ソリューション・テンプレートでは，WebSphere 5.1，6.1 の測定値を収集します。
- **Oracle 11i**：Oracle 11i ソリューション・テンプレートでは，Web ベースのアプリケーション・フレームワークの測定値を収集します。
- **SAP**：SAP ソリューション・テンプレートでは，SAP Web Application Server 6.0，7.0 の iViews の測定値を収集します。

ソリューション・テンプレート・データについて

HP Diagnostics では，アプリケーションが呼び出して処理を実行するリソースに基づいて，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**と**サブレイヤ**にグループ化します。これらのレイヤは，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。レイヤの詳細は，第 12 章「インストゥルメンテーション・レイヤ」を参照してください。

ソリューション・テンプレートを使って，Diagnostics アプリケーションの新しいレイヤの，さまざまなミドルウェア・サーバおよびフレームワークに関連付けられているビジネス・サービスのパフォーマンス測定値を確認できます。

ソリューション・テンプレート・パフォーマンス測定値の表示

HP Diagnostics では，各ソリューション・テンプレートの一意のパフォーマンス測定値を表示します。以下のセクションでは，各ソリューション・テンプレートのインストゥルメンテーション・ポイントによって収集されるパフォーマンス測定値について説明します。

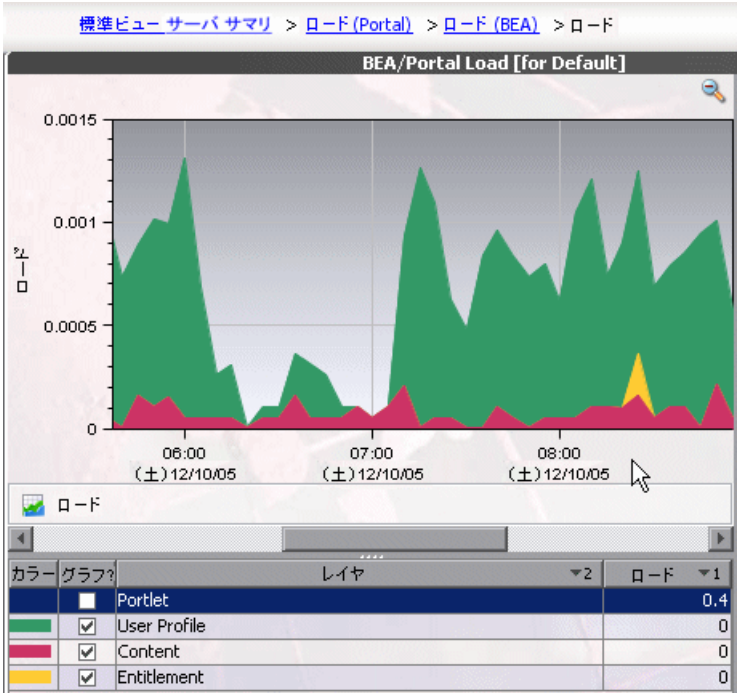
WebLogic

WebLogic Portal Server ソリューション・テンプレートを使うと，Diagnostics アプリケーションのレイヤの次のビジネス・サービスに対するパフォーマンス測定値を表示できます。

サービス	レイヤ	説明
Entitlements	Portal/BEA/Entitlement	エンタイトルメント API
Content management	Portal/BEA/Content	コンテンツ・マネージメント API
User profiles	Portal/BEA/UserProfile	ユーザ・プロファイル API
Personalization	Portal/BEA/Personalization	パーソナリゼーション API
Portal components	Portal/BEA/Portlet	ポータル・コンポーネント・レンダリング API

注：デフォルトで，BEA Web Service を表す Porta/BEA/Web Services レイヤは，ポイント・ファイルでコメントアウトされます。

以下は，ソリューション・テンプレートを使って収集された測定値で，WebLogic ビジネス・サービス・レイヤを表示する Diagnostics ビューの例です。



注：HP Diagnostics の Portal ビューで WebLogic パフォーマンス測定値を表示することもできます。Portal ビューの使用については，『**HP Diagnostics User's Guide**』（英語版）を参照してください。

WebSphere

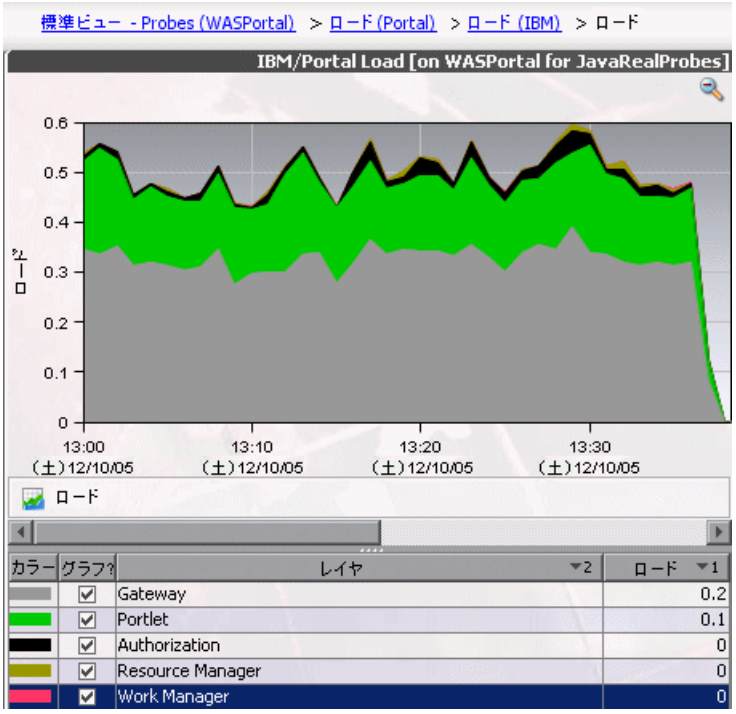
WebSphere Portal Server ソリューション・テンプレートを使うと，Diagnostics アプリケーションのレイヤの次のビジネス・サービスに対するパフォーマンス測定値を表示できます。

サービス	レイヤ
WAS Portal Gateway Servlet	Portal/IBM/Gateway
WAS Portal Authentication	Portal/IBM/Authentication
WAS Portal PortalAuthorization	Portal/IBM/Authorization
WAS Portal Resource Manager	Portal/IBM/ResourceManager
WAS Portal Portlet Adapter Interface Actions	Portal/Jetspeed/Portlet/Action
WAS Portlet Lifecycle	Portal/Jetspeed/Portlet/Lifecycle
WAS Portal Rendering	Portal/IBM/Portlet/Rendering
WAS Portal Work Manager	Portal/IBM/WorkManager
WAS Portal Portlet Services	Portal/IBM/Portlet/Services
WAS Portal Phases	Portal/IBM/Portlet/Phases
WAS Portal Services	Portal/IBM/Portlet/Services
WAS Portal Portlet Filter	Portal/IBM/Portlet/Filter

注 :

- デフォルトで，WAS Portal Phases を表す Porta/IBM/Portlet/Phase レイヤは，ポイント・ファイルでコメントアウトされます。Portal Phases は，コンテナが，ポートレットの初期化 - アクション - レンダ・サイクルの各フェーズをどのように処理するのかを理解したい場合を除いて，キャプチャする必要はありません。
- また，デフォルトで WAS Portal Services を表す Porta/IBM/Portlet/Services レイヤは，ポイント・ファイルでコメントアウトされます。AccessControlService サービスや PortletFilterService などの数多くのポータル・サービスがほかのレイヤにさらされますが，ポータル・コンテナによって提供されるほかのサービスに関心がある場合，このセクションをコメントアウトしないことがあります。

以下は，ソリューション・テンプレートを使って収集された測定値で，WebSphere ビジネス・サービス・レイヤを表示する Diagnostics ビューの例です。



Oracle

Oracle 11i ソリューション・テンプレートを使って，ルート **Business Tier** レイヤの下にある次の 2 つのレイヤに関するパフォーマンス測定値を表示できます。

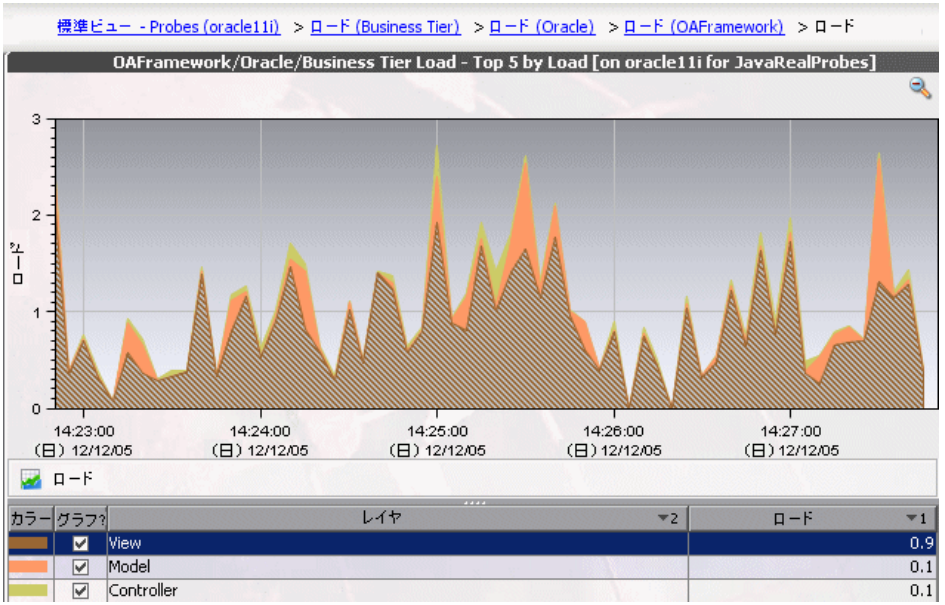
- Caching
- OAFramework（Oracle アプリケーション・フレームワーク）

OAFramework レイヤをドリルダウンして，次の 3 つのレイヤを表示できます。

- Model
- View
- Controller

MVC レイヤ（Model，View および Controller）は，Oracle 11i アプリケーション・フレームワークのさまざまな部分に関連します。

以下は，ソリューション・テンプレートを使って収集された測定値で，MVC レイヤを表示する Diagnostics ビューの例です。



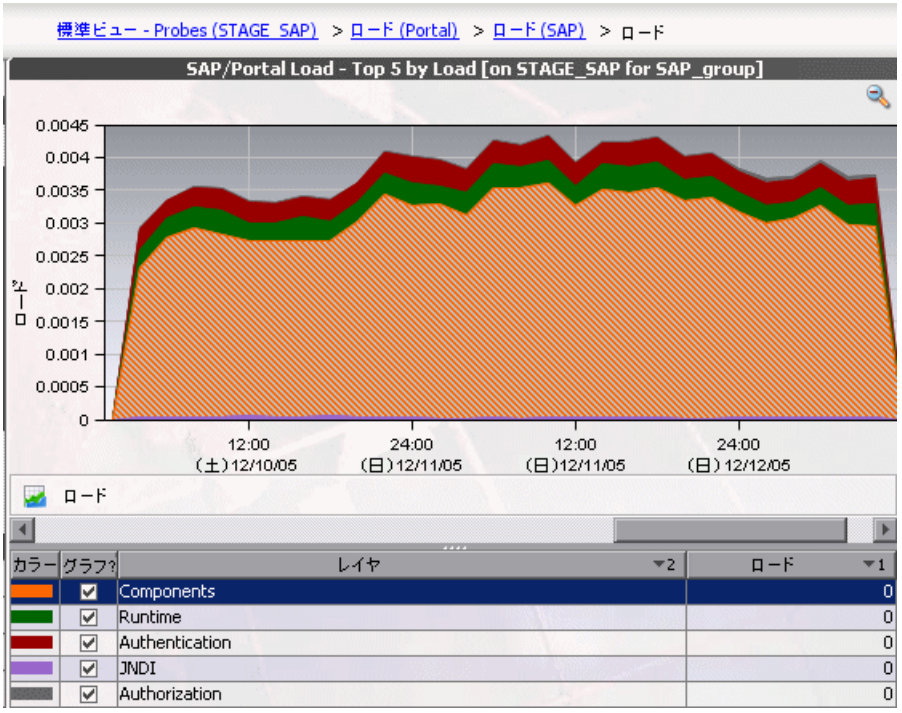
SAP

SAP Portal Server ソリューション・テンプレートを使うと，Diagnostics アプリケーションのレイヤの次のビジネス・サービスに対するパフォーマンス測定値を表示できます。

サービス	レイヤ	コメント
Runtime	Portal/SAP/Runtime	インフラストラクチャの一般ランタイム部分
Authorization	Portal/SAP/Authorization	承認サービス専用の中枢アプリケーション・コンポーネント
認証	Portal/SAP/Authentication	認証サービス専用の中枢アプリケーション・コンポーネント

サービス	レイヤ	コメント
JNDI Services:	Portal/SAP/JNDI	JNDI サービス専用の中枢アプリケーション・コンポーネント
Components Runtime	Portal/SAP/Components/Runtime	Portal コンポーネントの一般ランタイム部分
Content Generation（Portal コンポーネント）	Portal/SAP/Components/Content Generation	Portal コンポーネントのコンテンツ生成部分
Content Generation（Service コンポーネント）	Portal/SAP/Components/Service	コンテンツ生成サービス・コンポーネント（注：コメントアウトされ，使用前に最適化）
Content Generation（要求オブジェクト専用）	Portal/SAP/Components/Request	コンテンツ生成要求オブジェクト特有（注：コメントアウトされ，使用前に最適化）
Content Generation（プロファイル・コンポーネント）	Portal/SAP/Components/Profile	コンテンツ生成のプロファイル・コンポーネント
Content Generation（Config コンポーネント）	Portal/SAP/Components/Config	コンテンツ生成の設定コンポーネント
Content Generation（要求オブジェクト専用）	Portal/SAP/Components/Response	Content generation（要求オブジェクト専用）
R3 Content Generation	Business Tier/SAP R3	R3 バックエンド特有のコンテンツ生成
Dynpage Content Generation	Web Tier/SAP/DynPage	DynPage 特有のコンテンツ生成
JSP Dynpage Content Generation	Web Tier/SAP/JSPDynPage	JSP DynPage 特有のコンテンツ生成
Page Processor Component	Web Tier/SAP/Page Processor	Page Processor 特有のコンテンツ生成

以下は，ソリューション・テンプレートを使って収集された測定値で，SAP ビジネス・サービス・レイヤを表示する Diagnostics ビューの例です。レイヤの名前は，リモート関数呼び出し（RFC）の名前に相当します。

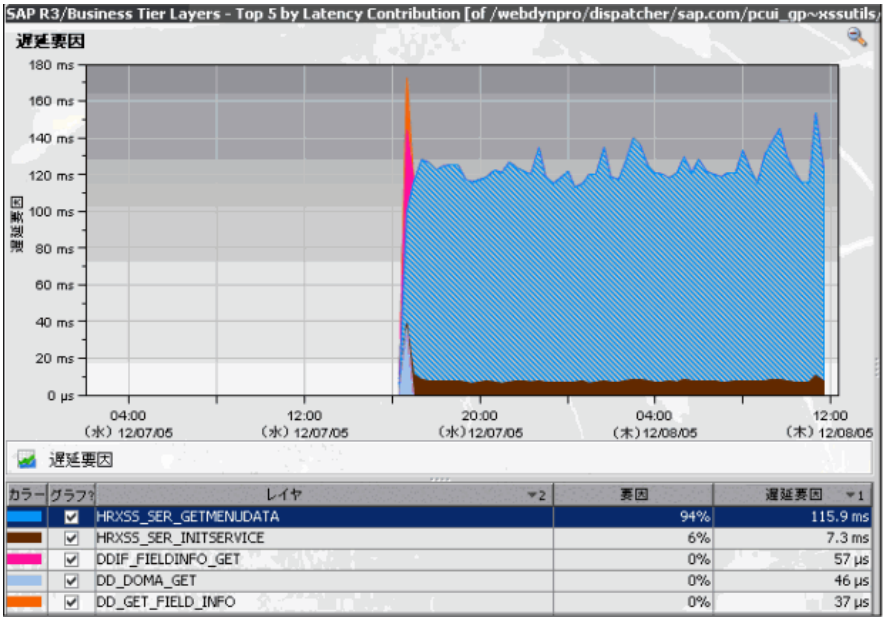


SAP R3 Layer

一般的な SAP Web Application Server デプロイメント（WAS）では，SAP WAS はバックエンド SAP R3 Server に接続します。SAP R3 Server との通信は，SAP JCO（Java Connector）を使用し，RFC を通じて行われます。HP Diagnostics では，各呼び出しをキャプチャし，サブレイヤに表します。

これらのサブレイヤは，Business Tier レイヤからドリルダウンすることでアクセスします。たとえば，HP Diagnostics の Server Summary ビューから起動すると，次のように SAP R3 サブレイヤにドリルダウンできます。［**標準ビュー**］＞［**サーバ サマリ**］＞［**サーバ要求**］＞［**レイヤ（Business Tier）**］＞［**レイヤ（SAP R3）**］＞［**レイヤ**］

以下は，ソリューション・テンプレートを使って収集された測定値で，SAP R3 サブレイヤを表示する Diagnostics ビューの例です。



注：HP Diagnostics の Portal ビューで SAP Portal パフォーマンス測定値を表示することもできます。Portal ビューの使用については，『**HP Diagnostics User's Guide**』（英語版）を参照してください。

第 V 部

Diagnostics Server と Java および .NET Agent の詳細設定

第 14 章

Diagnostics Server の詳細設定

Diagnostics Server 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

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 Server（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 が特定されるローカル時刻にデータを正しく表示できるようになります。

以下のセクションでは，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 Probe 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，399 ページ「VMware 上で実行中の Probe 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 Server（Commander モード）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 Server 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 Server 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 Server からコンタクトできることを確認してください。コンタクトできない場合は，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 Availability Center コア・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 Availability Center 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 Server（Commander モード）を設定する必要もあります。

➤ **SERVER**：Diagnostics Server（Commander モード）を**時刻ソース**に使用することを示します。

これは，Diagnostics Server 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 Server は，Diagnostics Server（Mediator モード）と Diagnostics Server（Commander モード）の間の時間差を確認して時刻を同期します。

Diagnostics Server（Commander モード）が**時刻ソース**と同期していない場合，Diagnostics Server（Mediator モード）は「非同期」とみなされます。非同期の Diagnostics Server（Mediator モード）は，成功するまで 15 秒ごとに時刻の同期を試みます。

Diagnostics Probe が Diagnostics Server（Mediator モード）または Diagnostics Server（Commander モード）に接続するときに，Diagnostics Server と Probe の時間差が特定されます。

Probe が非同期の Diagnostics Server に接続しようとすると，Probe 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 Server（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 マシンが同期された GMT 時刻から離れている分，変更されて表示されます。

Diagnostics Server での時刻の同期の設定

次の手順を実行して，Diagnostics Server（Commander モード）を同期します。

注：Diagnostics Server（Mediator モード）の時刻は，Diagnostics Server（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 Server（Commander モード）の時刻を同期できるようにするには

1 Diagnostics Server のデフォルト設定は，**＜Diagnostics Server 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 Server にインターネット接続があり，**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，デフォルト設定が機能し，変更は不要です。

また，Business Availability Center は，デフォルトで時刻の同期に NTP を使用するため，これは推奨される設定です。

2 Diagnostics Server にインターネット接続がなく，**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，次のいずれかを実行する必要があります。

➤ Diagnostics Server（Commander モード）がコンタクト可能なローカル NTP サーバをセットアップします。**＜Diagnostics Server 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することをお勧めします。

➤ Business Availability Center または HP Software as a Service（SaaS）環境で Diagnostics を使用している場合，**＜ Diagnostics Server 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 Availability Center を示すことができます。これにより，Diagnostics Server が登録されている Business Availability Center 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：Business Availability Center をセットアップして Diagnostics を使用する方法については，第 23 章「Diagnostics を使用するための Business Availability Center のセットアップ」を参照してください。

➤ Business Availability Center で使わずに Diagnostics Server（Commander モード）をスタンドアロン・モードで使用し，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，**＜ Diagnostics Server 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 Server 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 Server 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，Diagnostics Server が GMT と同期していないときに，保持されたデータがキャプチャされたことが原因です。Diagnostics Server が GMT と同期されているときに，後でキャプチャされたデータがキャプチャされた場合，データを何度も再集計したり，属さない時刻バケットに記録することができます。

大規模インストールの Diagnostics Server の設定

20 以上の Probe で Diagnostics Server（Mediator モード）を使用している場合，Diagnostics Server 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 Server（Commander モード）では，Diagnostics Server（Mediator モード）として機能するように Probe 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 Server（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 Server（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 Server（Mediator モード）でガベージ・コレクションが長時間遅れます。

ヒープ・サイズのデフォルト値は 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 Server のインストール・ディレクトリ＞ ¥nanny¥windows¥dat¥nanny¥**，Solaris では **＜ Diagnostics Server 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 Server（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 Server（Mediator モード）のヒープ・サイズを調整するには

1 次の表に従って，Diagnostics Server（Mediator モード）で必要なヒープの量を決めます。

Probe の数	推奨ヒープ・サイズ
Java Probe 50 台以内	512 MB
Java Probe 100 台以内	750 MB
Java Probe 200 台以内	1280 MB
.NET Probe 10 台以内	350 MB
.NET Probe 20 台以内	700 MB
.NET Probe 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 G ある場合，ヒープ・サイズは 768 MB 以内にします。

---

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 Server のインストール・ディレクトリ＞ ¥ nanny¥windows¥dat¥nanny¥**，Solaris では **＜ Diagnostics Server 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¥loa
ding.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¥loa
ding.jar;C:¥MercuryDiagnostics¥Server¥lib¥common.jar;C:¥MercuryDiagnostics¥Server
¥lib¥mercury_picocontainer-1.1.jar" com.mercury.opal.mediator.util.DiagnosticsServer
```

Probe 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メディエータ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，除外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，サーバがプローブから取得するデータの頻度を減らしたりすることをお勧めします。

プローブの主な除外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，101ms まで増やすことができます）

除外の詳細に関する参照先を示します。サーバについては 367 ページ「フラグメント名に基づく除外の設定」，.NET Probe については 478 ページ「レイテンシの除外およびスロットリングの設定」および 483 ページ「深さの除外の設定」，Java Probe については 391 ページ「Probe におけるメソッドの自動除外の制御」および 393 ページ「Probe のスロットリングの制御」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 の rhttp.gzip.replies を true** に設定します。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。デフォルトでは，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メディエータの **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下げることをお勧めします。それでも十分でない場合は，**probe.trends.pull.interval** を（たとえば 10 秒に）変更してトレンドの取得頻度を下げることもできます。

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すことをお勧めします。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メディエータに移動することをお勧めします。そうしないと，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，デフォルトのパージ設定（5 GB）では十分でない可能性があります。詳細については，656 ページ「データの保存サイズとデータ保存」を参照してください。

デフォルトの Diagnostics Server 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 Server（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，Diagnostics Server（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 Server（Commander モード）が Diagnostics Server（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 Server（Mediator モード）が Diagnostics Server（Commander モード）への登録に使うデフォルトのホスト名を変更できます。

Diagnostics Server（Mediator モード）のデフォルトのホスト名を変更するには，**＜ Diagnostics Server 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 Server（Commander モード）が Diagnostics Server（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 Server ポートの変更

Diagnostics Server のホスト設定で，デフォルトの Diagnostics ポートを使用できない場合，Diagnostics Server とプローブおよびほかの Diagnostics Server との通信に別のポートを指定することができます。

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 をデプロイした後で，ほかのポート番号の使用を決めた場合，デプロイメントの必要な各コンポーネントについて，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 Server（Commander モード）	**＜ Diagnostics Server のインストール・ディレクトリ＞ /etc** ➤ webserver.properties - jetty.port ➤ mediator.properties - commander.url ➤ probe/etc/dispatcher.properties - registrar.url
Diagnostics Server（Mediator モード）	**＜ Diagnostics Server のインストール・ディレクトリ＞ /etc** ➤ mediator.properties - commander.url **＜ Diagnostics Server のインストール・ディレクトリ＞ /probe/etc** ➤ dispatcher.properties - registrar.url
Probe	**＜ Probe のインストール・ディレクトリ＞ /etc** ➤ dispatcher.properties - registrar.url

マルチホーム環境向けの Diagnostics Server の設定

Diagnostics Server をホストするマシンは，1 枚以上のネットワーク・インタフェース・カード（NIC）で設定でき，Diagnostics Server プロセスは，そのホストのすべてのインタフェースをリスンします。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 Server 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 Server ホストに複数のネットワーク・インタフェースがあり，Diagnostics Server でリ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 Server 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 Server 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 Server のリ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。デフォルトで，Diagnostics Server に含まれる **jetty.xml** ファイルには，定義済みのリスナがないため，Diagnostics Server はすべてのインタフェースをリスンします。

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 Server を設定するには

1 **＜ Diagnostics Server 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 この行の後に以下のコード・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 Server 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 Server にアクセスするコンポーネントが，Diagnostics Server 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 Server ホストのほかの IP アドレスに解決することができます。詳細については，362 ページ「デフォルトの Diagnostics Server 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 Server がループバックでリスンする Diagnostics Server 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 Server 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 Server 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 Server は，Diagnostics Server 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 Server 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 Server（Commander モード）が実行し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 Availability Center と Diagnostics Server（Commander モード）の間，または Business Process Monitor と Business Availability Center の間にネットワーク接続の問題がある場合，Diagnostics Server でトランザクションが完了したときに通知する ELT を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 Server は，ELT が送信されておらず，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容量を空けるものと判断します。

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 Server でメモリ不足が生じると，場合によっては，トランザクションの完了までに Diagnostics Server 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必要があります。このプロパティの値を調整する際は，十分に注意してください。複数の Probe が Diagnostics Server にデータを送信したり，Diagnostics Server でアクティブなトランザクションがアイドル状態になったりするため，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間違って報告されることがあります。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 秒長くすることをお勧めします。

フラグメント名に基づく除外の設定

フラグメント名に基づく除外を使うと，Probe の設定やインストゥルメンテーションを変更せずに，Diagnostics Server にパフォーマンスの問題を生じさせる

ように見えるサーバ要求をフィルタリングするように Diagnostics Server を設定できます。

注：フラグメント名に基づく除外は，Probe で設定するレイテンシや深さの除外の代わりに使用することはできません。

＜ Diagnostics Server のインストール・ディレクトリ＞ ¥etc¥ trimming.properties の **trim.fragment** プロパティを使って，Diagnostics で除外する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除外します。

デフォルトで，**trim.fragment.1** および **trim.fragment.2** プロパティは，**trimming.properites** でコメントアウトされます。除外するフラグメントを指定するには，プロパティの 1 つをコメントアウトせず，Diagnostics ビューに一覧表示されたときに除外するフラグメントの名前を入力します。除外するフラグメントが 3 つ以上ある場合，追加の **trim.fragment** プロパティを作成して，末尾の数を増やして，各プロパティ名が一意になるようにすることができます。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除外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。

可用性の高い Diagnostics Server の準備

Diagnostics デプロイメントで，Diagnostics Server に高い可用性が必要な場合，以下の手順に従って，各 Diagnostics Server にスタンバイ Diagnostics Server を作成することができます。スタンバイは，ハードウェア障害や，Diagnostics Server のホストにほか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 Server の作成

Diagnostics Server をスタンバイ・マシンにインストールし，プライマリ Diagnostics Server のデータをスタンバイ Diagnostics Server に定期的に複製することで，各 Diagnostics Server のスタンバイを作成できます。

スタンバイ Diagnostics Server を設定するには

1 スタンバイ・マシンに Diagnostics Server 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 Server は，プライマリ・サーバの Diagnostics Server と同じバージョンを使用してください。

2 スタンバイ Diagnostics Server 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 Server 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 Diagnostics Server 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 ユーティリティを使用し，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 Server 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 Server のフェールオーバー

プライマリ Diagnostics Server のホストで障害が発生した場合，プライマリ Diagnostics Server として機能するようにスタンバイ Diagnostics Server を設定する必要があります。

スタンバイ Diagnostics Server をプライマリ Diagnostics Server にするには

1 障害が発生したプライマリ Diagnostics Server のホストのホスト名に一致するように，スタンバイ Diagnostics Server のホスト名を変更します。これにより，Probe が起動したときに Diagnostics Server に再接続できるようになります。

2 スタンバイ Diagnostics Server 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** スクリプトを使って起動します。

3 Probe は Diagnostics Server に再接続します。Probe は，Diagnostics Server 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。

4 以上で，スタンバイ Diagnostics Server がプライマリ Diagnostics Server になりました。368 ページ「スタンバイ Diagnostics Server の作成」の説明に従って，新しいスタンバイ Diagnostics Server を設定します。

注：障害が発生した Diagnostics Server が復旧したら，新しいプライマリ Diagnostics Server の使用中に Probe から収集したデータが失われるため，復旧した Diagnostics Server をプライマリにする必要があります。

LoadRunner / Performance Center Diagnostics Server の割り当て

デフォルトの Diagnostics Server の割り当て

デフォルトで，LoadRunner または Performance Center に指定した Probe は，**＜ Probe のインストール・ディレクトリ＞ /etc/dynamic.properties** に指定されている Diagnostics Server を使用します。

Diagnostics Server の割り当ての変更

実行のために Probe を起動したときに，Probe の設定を変更することができます。この場合，Diagnostics Server（Commander モード）のマッピング・ファイルを変更して，Probe に対する Diagnostics Server の割り当てを変えることができます。

これは，LoadRunner / Performance Center と Business Availability Center の複合環境で Diagnostics を実行している場合に便利です。このような場合，Probe が LoadRunner / Performance Center で実行していて，Business Availability Center の監視を実行しているときに，Probe でさまざまな Diagnostics Server を使いたいことがあります。そのようなときは，次の Diagnostics Server の変更手順に従います。

Probe 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：Probe が実行中でない場合，**＜ Probe 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 Server が必ず使用されます。

Probe に対して Diagnostics Server の割り当てを変更する場合，Diagnostics Server（Commander モード）ホスト・マシンで **＜ Diagnostics Server 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 Probe の ID に置き換えます。
- **<Server.id>** を Diagnostics Server 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。したがって，このファイルに加えた変更は，Diagnostics Server（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のサイズに合わせた Diagnostics Server 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 Server（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 Server（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する必要があります。

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

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 Server（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 Server のインストール・ディレクトリ＞ /data/ ＜ newest ディレクトリ＞ で，Diagnostics Server（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 Server（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 Server（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・データの細分性レベルが下がるため，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デフォルト設定は **5s**（5 秒）で，Diagnostics Server（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・サイズは約 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 Server のインストール・ディレクトリ＞/etc/mediator.properties** で，Diagnostics Server（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 Server（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

1 Diagnostics Server（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 Server（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，ブラウザで［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 の倍数でなければなりません。たとえば，45s に設定します。

4 **LoadRunner / Performance Ceter 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 の倍数でなければなりません。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 Server（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 Server（Mediator モード）の再起動を求めるリマインダが表示されます。その他のリマインダとして，［**Diagnostics Server の再起動**］ボタンが表示されます。

［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，567 ページ「Diagnostics Server 設定ページを使用した Diagnostics の設定」を参照してください。

設定の変更を有効にするには，［**Diagnostics Server の再起動**］をクリックして Diagnostics Server（Mediator モード）を再起動します。

Business Availability Center のサンプル・キュー・サイズと Web サービス CI の頻度の設定

以下の設定は Business Availability Center の統合に適用できます。

Business Availability Center のサンプル・キュー・サイズの設定

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 BAC のデータが消失します。ログには次のメッセージが記録されます。BAC samples being dropped since too many are waiting for delivery

サーバ・プロパティ **bac.webservice.delivery.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 によってデフォルトの頻度で自動的に作成され，uCMDB に追加されます。

場合によっては，Web サービス CI を uCMDB に追加するプロセスのタイミングを変更する必要が生じることがあります。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（デフォルトは 360，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（デフォルトは 1d）

Diagnostics Server 設定ページを使用した Diagnostics の設定

Diagnostics Server の設定ページでは，Diagnostics Server とほかの Diagnostics コンポーネントの通信方法，および Probe から受信したデータを Diagnostics Server 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するための方法を提供します。

注：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，これらのページを使って Diagnostics Server プロパティを更新する必要があります。

Diagnostics Server 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 Server の管理ページ」を参照してください。

実稼動環境の Diagnostics Server の処理 Probe 数増加の最適化

1 つの Diagnostics Server が処理できる Probe 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。以下は，1 つのサーバ・プロセスあたりに処理できる Probe の数を増やすのに役立つ最適化を示します。

- デフォルトで，Diagnostics Server は，各 Probe から 5 秒間隔でトレンドを取得し，45 秒間隔で各 Probe のツリーを取得するように設定されています。1 つの Diagnostics Server プロセスで 25 以上の Probe 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。実稼動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 Server のインストール・ディレクトリ＞¥Server¥etc¥server.properties** で設定できます。

```
# Probe からトレンドを取得する間隔 probe.trends.pull.interval = 30s

# Probe からツリーを取得する間隔 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** で決められます。最大ヒープ・サイズのデフォルト設定は 512MB です。Probe からの負荷に基づいて，適宜，最大ヒープ・サイズを増やします。処理される

Probe 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 Probe を処理している場合，Diagnostics Server の実稼動環境では 1Gbps 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 Probe を処理している場合は，Jetty スレッドの数を増やします。スレッドの数を数を見極める一般的な目安は，数の 2 倍か，または Probe に 40 を足した数です。デフォルト値は 200 です。Jetty スレッドの数は，**＜Diagnostics Server 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やすことができます。例を次に示します。

```
jetty.threads.max=300
```

第 15 章

Java Probe およびアプリケーション・サーバの詳細設定

Java Probe とアプリケーション・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。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

➤ スタンドアロン・モードの Java Diagnostics Profiler での認証と承認（404 ページ）

➤ CPU 時間測定値の収集の設定（407 ページ）

➤ コンシューマ ID の設定（410 ページ）

➤ SOAP の失敗ペイロード・データの設定（419 ページ）

➤ REST サービスの設定（420 ページ）

➤ JMS 一時キュー / トピックのグループ化（420 ページ）

詳細設定の手引き

以下の情報は，お使いの環境特有の問題を解決するように Probe を設定する上で役立ちます。

➤ Probe を HP ソフトウェア製品と連動するようにインストールしていて，ほかの製品とも連動させる場合は，379 ページ「ほかの HP ソフトウェア製品との連携のための Probe の設定」を参照してください。

➤ お使いのアプリケーション・サーバで複数の JVM を使用している場合，または複数の JVM のデータをキャプチャする場合は，383 ページ「複数のアプリケーション・サーバの JVM インスタンスで使用するための Probe の設定」の手順に従って Probe の追加設定を行う必要があります。

➤ Profiler モードで Probe を他者に使わせないようにするには，387 ページ「Java Diagnostics Profiler の無効化」を参照してください。

➤ 1 つの Probe インストールを使用する JVM が複数ある場合，場合によっては，アプリケーションの起動コマンドで Java システム・プロパティの一部の Probe プロパティを設定する必要があります。詳細については，387 ページ「Java システム・プロパティとしての Probe プロパティの設定」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 Probe ログに送る場合，388 ページ「Probe の記録の制御」の手順に従ってログ・レベルを調整できます。

➤ 同じホストに複数の Probe をインストールしている場合，389 ページ「Probe のログ・ディレクトリの変更」の手順に従って，各 Probe のログ・メッセージを別のファイルに保存することができます。

- Diagnostics で報告された測定値から通常除外される可能性のある処理のパフォーマンスを調べる場合，391 ページ「Probe におけるメソッドの自動除外の制御」の手順に従って，除外レベルを下げたり，除外機能を完全にオフにすることができます。
- 場合によっては，Probe のスロットリングをオフにして，アプリケーション自体に与えるパフォーマンスの影響を気にせずに，得られるすべてのパフォーマンス測定値を取得できていることを確認する必要があります。スロットリングとスロットリングの制御方法については，393 ページ「Probe のスロットリングの制御」を参照してください。
- Probe と Diagnostics Server（Commander モード）の間にプロキシが存在する場合，Probe に Diagnostics Server（Commander モード）の URL を通知するように現在のプロパティを設定する必要があります。詳細については，396 ページ「プロキシ・サーバへの Probe の設定」を参照してください。
- HP Software as a Service（SaaS）環境で使用するために Java Probe をインストールしている場合，Probe と Diagnostics Server（Mediator モード）の間のリバース HTTP（RHTTP）通信を無効にすることをお勧めします。詳細については，396 ページ「SaaS の Probe へのリバース HTTP の設定」を参照してください。

ほかの HP ソフトウェア製品との連携のための Probe の設定

本項の内容

Probe の設定への HP ソフトウェア製品の追加

Java Probe は，開発から実装および実稼動までを通して，アプリケーションを監視できる設定が可能なライフサイクル Probe です。Java Probe は，複数の HP ソフトウェア製品と連動するように設定したり，ほかの Diagnostics コンポーネントや HP ソフトウェア製品を使用せずに Diagnostics Profiler として機能するように設定できます。

Java Probe が動作可能な製品は，**＜Probe 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** プロパティの値で決まります。

active.products プロパティの値は，Java Probe のインストール時に設定されます。Probe のインストールについては，第 5 章「Java Agent のインストール」を参照してください。Probe でほかの製品からデータをキャプチャできるようにするには，プロパティ・ファイルを編集し，アプリケーション・サーバを再起動して，**active.products** の値を設定する必要があります。

注：ほかの HP ソフトウェア製品と一緒に Diagnostics Profiler for Java にダウンロードした Java Probe を使用する場合は，HP ソフトウェアのソフトウェア・カスタマ・サポートに連絡してください。以下の手順は，HP ソフトウェア・カスタマ・サポートのサポートを受けずに Java Diagnostics Profiler にインストールした Probe では使用できません。

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，追加の設定手順を実行する必要があります。お使いの HP ソフトウェア製品に該当する章を参照して，設定を完了してください。

以下のセクションでは，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

Diagnostics Profiler for Java の場合，プロダクト・モードは PRO です。このモードで，Probe は追加の測定値を収集し，それらの測定値をユーザ・インタフェースに表示して，Probe ホストの URL で確認できるようにします。

Development モードで Java Diagnostics Profiler の一部として Java Probe を実行している場合，Probe で処理可能な負荷を制限するための制限が Probe に設定されています。

Diagnostics Standalone の一部として Java Probe 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っています。

Enterprise プロダクト・モード

Enterprise モードに設定されている場合，Probe は，Business Availability Center，LoadRunner，Performance Center および Diagnostics Standalone などの HP ソフトウェア製品と連携して機能します。

Probe を Enterprise モードに設定するには

1 **active.products** プロパティを **Enterprise** に設定して，アプリケーション・サーバを再起動します。

2 Probe を設定して，Diagnostics Server（Commander モード）に登録します。

Diagnostics Server（Commander モード）の機能の 1 つでは，Diagnostics コンポーネントを追跡し，それらの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常に知らせます。

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

以下は，**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/
```

3 Probe を設定して，Diagnostics Server（Mediator モード）に登録します。

Business Availability Center，LoadRunner および Performance Center でレポートを受信，処理および表示できるように，Probe は収集する処理測定値を Diagnostics Server（Mediator モード）に送信できる必要があります。LoadRunner および Performance Center で，Diagnostics は Probe に Diagnostics Server を割り当てます。Business Availability Center では，Diagnostics Server が連動するプローブを指定する必要があります。

Diagnostics Server（Mediator モード）と通信するように Probe を設定するには，**＜Probe のインストール・ディレクトリ＞ ¥etc¥dynamic.properties** にある **mediator.host.name** および **mediator.port.number** プロパティを使って，ホスト名およびポートを設定する必要があります。

Diagnostics Server（Mediator モード）のデフォルトのポートは 2006 です。Diagnostics Server（Mediator モード）のインストール中に Diagnostics Server（Mediator モード）のポートにデフォルト以外の値を設定する場合，Probe に **mediator.port.number** プロパティを設定するときに，同じポート番号を使用する必要があります。

以下は，これらのプロパティを示す **dynamic.properties** ファイルの抜粋です。

```
#the host the Topaz mediator is running on
mediator.host.name=host01.company.com

#the port the Topaz mediator is listening on (default is 2612)
mediator.port.number=2612
```

AM プロダクト・モード

実稼動環境の Business Availability Center の Probe が LoadRunner または Performance Center の実行に不用意に含まれないように保護するには，**active.products** を AM に設定します。AM モードにすると，Probe は，LoadRunner または Performance Center の使用可能な Probe に表示されません。

AD プロダクト・モード

QA 環境の Probe が追加リソースを使ったり，負荷テストが実行されていないときに Diagnostics コンソール・データセットへのデータの報告を続けないようにするには，**active.products** を AD に設定します。

Probe 設定からの製品の削除

Probe のプロダクト・モードは，**＜Probe のインストール・ディレクトリ＞¥etc¥probe.properties** ファイルにある **active.products** プロパティを使って設定されます。

製品をサポートしないように Probe を設定するには，プロパティ値に製品が含まれないように **active.products** プロパティを更新します。

複数のアプリケーション・サーバの JVM インスタンスで使用するための Probe の設定

お使いのアプリケーション・サーバで複数の JVM を使用している場合，または複数の JVM のデータをキャプチャする場合は，Probe の追加設定を行う必要があります。設定方法は 2 つあり，ホストの JVM ごとに Probe を 1 つインストールすることも，すべての JVM で共有する Probe を 1 つインストールすることもできます。

本項の内容

1 つの Probe インストールに対する複数 JVM の設定

複数の JVM で 1 つの Probe インストールを共有するには，各 JVM に Probe の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，以下が有効になります。

- JVM と Probe 間で通信を確立する。
- Probe を JVM ごとに特定する。
- 監視するパフォーマンス測定値について Probe に命令する JVM をインストゥルメンテーションする。

複数の JVM と連動するように Java Probe を設定するには

1 複数の JVM バージョンを監視するように Probe を設定している場合，JRE Instrumenter を JVM バージョンごとに実行して，JVM で実行中のアプリケーションのイベントを Probe で監視できるようにする必要があります。

Probe が連動する各 JVM バージョンに JRE Instrumenter を実行していない場合は，ただちに実行してください。詳細については，137 ページ「JRE Instrumenter の実行」を参照してください。

注：

- Probe をインストールしたときに，インストーラに JRE のインストゥルメントを許可した場合も，JRE Instrumenter を実行する必要があります。JRE Instrumenter は，複数の JVM をサポートするようにアプリケーションを準備します。
- 複数の JVM バージョンを使用しており，JRE Instrumenter を何度も実行した場合は，**-Xbootclasspath** パラメータに Instrumenter の出力どおりに 2 つのパスを含める必要があります。1 つ目のパスは特定の JVM のもので，2 つ目のパスはデフォルトの「ブート」ディレクトリのものです。
  例：-Xbootclasspath/p:/path/to/javaprobe/classes/IBM/1.3.1:/path/to/javaprobe/classes/boot

---

2 Probe が自動的に選択できるポートの範囲を指定します。Java Probe は，ミニ Web サーバを使って通信します。Probe が監視している各 JVM に対し，Probe との通信のために別々のポートが割り当てられます。デフォルトで，ポート番号の範囲は **35000** ～ **35100** です。Probe 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 Probe が隔離される場合，指定した範囲内のポートを使って通信できるようにファイアウォールを設定する必要があります。詳細については，第 19 章「ファイアウォール環境で動作する Diagnostics の設定」を参照してください。デフォルトとは異なる範囲で Probe 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，以下の説明に従って，ポート範囲の値を適宜更新してください。

---

a **＜ Probe のインストール・ディレクトリ＞ /javaprobe/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，Probe 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って設定します。

  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って設定します。

jetty.max.port=35100

3 Java コマンド・ラインを使って，各 JVM の Probe にカスタム Probe ID を割り当てます。

-Dprobe.id=<Unique_Probe_Name>

この Java コマンド・ラインで定義された Probe ID によって，Probe の **id** プロパティを使って **probe.properties** ファイルに定義された Probe 名が無効になります。

以下は，**probe.id** パラメータを追加する前の WebLogic 起動スクリプトの例です。

```
"%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
```

以下は，**probe.id** パラメータを追加した後の WebLogic 起動スクリプトの例です。

```
"%JAVA_HOME%¥bin¥java" -hotspot -ms64m -mx64m"-
Xbootclasspath/p:C:¥MercuryDiagnostics¥JAVAProbe¥classes¥Sun¥1.4.1_0
3;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
```

注：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。

---

4 Probe で使うポイント・ファイルを指定します。デフォルトで，LoadRunner / Performance Center は，points.file. 名を auto_detect.points に設定します。場合によっては，複数のカスタム・インストゥルメンテーション・プランを使う必要がある場合，または 1 つの Probe インスタンスを使って同じマシンで複数の JVM を持っているとき，および 1 つ以上の JVM でレイヤをサポートするカスタム・インストゥルメンテーションに特定のメソッドやクラスを必要なときに，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="<Custom_AutoDetect_Points_File>"
```

各 JVM への個別 Probe インストール設定

1 つのホストに複数の JVM がある場合，それぞれの JVM インスタンスに個別の Probe をインストールすることができます。各 JVM に個別の Probe を使用するには，Probe を複数回インストールし，各 Probe のインストール・ディレクトリにある **probe.properties** ファイルで Probe の **id** プロパティを設定して，各 Probe のインスタンスを定義します。

インストールされている各 Probe のインスタンスを定義するには

1 Probe が連動する各 JVM に JRE Instrumenter を手動で実行していない場合は，ただちに実行してください。JRE Instrumenter の実行については，137 ページ「JRE Instrumenter の実行」を参照してください。

2 **＜ Probe のインストール・ディレクトリ＞ /javaprobe/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 Server に一意の名前を割り当てます。

```
id=<uniqueProbeName>
```

Probe インスタンスが起動すると，Probe のログ・メッセージが保存される **＜ Probe のインストール・ディレクトリ＞ /javaprobe/log** ディレクトリにログ・ファイルが作成されます。

注：この設定により，診断を実行できるようになりますが，Probe の各インストールの設定に時間を取る必要があります。複数の JVM に 1 つの Probe を使用すると，同じ診断情報を提供できる一方で，コンポーネントの設定に要する時間が短縮されます。

Java Diagnostics Profiler の無効化

Java Probe で Diagnostics Profiler for Java を無効にして，不用意にアクセスされるのを防ぐことができます。Java Diagnostics Profiler が無効になると，Java Diagnostics Profiler URL：http:// ＜ Probe のホスト＞：＜ Probe 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，**＜ Probe 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 のデフォルト値は，**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。

Java システム・プロパティとしての Probe 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，すべての Java Probe プロパティは，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 Probe 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。

Probe 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 **D** およびプロパティ・ファイル名の最初の部分を付けます。以下，いくつかの例を挙げて詳しく説明します。

- 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 **D** とプロパティ・ファイル名の **probe** をつなげて，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 **D** とプロパティ・ファイル名の **probe** をつなげて，指定しているプロパティの名前，つまり **active.products** に付加します。

```
-Dprobe.active.products=AD,AM
```

- 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 **D** とプロパティ・ファイル名の **dispatcher** をつなげて，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
```

Probe の記録の制御

Probe プロパティを使って，Probe 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することができます。

ログ・メッセージ・レベルの制御

標準出力に記録される Probe のメッセージ・レベルは，**＜Probe 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。このプロパティのデフォルト設定は **OFF** になっており，ほとんどすべての Probe メッセージがコンソールに記録されません。

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することができます。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。

Probe のログ・ディレクトリの変更

Probe のログ・ディレクトリのデフォルトの場所は **＜ Probe 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 Probe がある場合，**log.dir** プロパティを使って，Probe ごとにログ・ディレクトリの場所を変更することができます。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，**＜ Probe 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，396 ページ「プロキシ・サーバへの Probe の設定」を参照してください。

Probe ホスト・マシン名の設定

Probe のホスト名を使って，Probe を Diagnostics Server（Commander モード）に登録します。Diagnostics Server（Commander モード）では，Probe のホスト名を使用して Probe と通信し，Probe が監視しているサーバのシステム測定値と一緒に Diagnostics ビューに表示します。

通常，Probe 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，Probe 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 Probe 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，Probe はホストを検出できないことがあります。

Probe 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 ルックアップを介してホスト名を特定するように Probe に命令することも，または Probe プロパティを使ってホスト名を指定することもできます。

Probe のリバース DNS ルックアップ使用の指定

Probe のホスト設定によって Probe 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 ルックアップを使ってホスト名を検出するように Probe に命令することができます。このプロパティは，
＜ Probe 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 プロパティのデフォルト値は 'false' です。これにより，リバース DNS を使わずにルックアップを実行するように Probe に指示します。このプロパティを 'true' に設定して，Probe にリバース DNS ルックアップを使うように命令します。

Probe のホスト名の手動指定

registered_hostname を使って，Probe のホスト・マシン名を手動で設定したり，Probe の自動ルックアップ機能を停止することができます。

Probe のデフォルトのホスト・マシン名を設定するには，Java Probe のプロパティ・ファイル **＜ Probe のインストール・ディレクトリ＞ /etc/dispatcher.properties** にある **registered_hostname** プロパティをマシン名または 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。ただし，Business Availability Center 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。

Probe におけるメソッドの自動除外の制御

Probe のデフォルト設定には，メソッドの除外を制御する設定が含まれます。除外機能は，**レイテンシ**と呼ばれるメソッドが実行に要する時間に基づき，メソッド呼び出しの**スタック**の**深さ**で制御できます。デフォルト設定では，レイテンシと深さの両方で除外を行うように Probe に命令します。

Diagnostics の特定の状況に対して，除外のレベルを下げたり，除外機能を完全にオフにすることができます。**＜Probe のインストール・ディレクトリ＞/etc/capture.properties** の **minimum.method.latency** および **maximum.stack.depth** プロパティを使って除外を制御できます。

レイテンシの除外の制御

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，関心の低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除外されます。

注：レイテンシが除外プロパティよりも小さいときに，メソッドのレイテンシが除外されない場合があります。

- 呼び出しツリーのルートであるメソッドは除外されません。
- Diagnostics Server（Mediator モード）にイベントを送信でいないことが原因で Probe が深刻なスロットリングの問題に直面しない限り，例外を送ったメソッドは除外されません。

スレッド処理およびバッファリングの動作が原因で，除外されたメソッドに関する一部の情報が Profiler のバッファ（Development モード）に保存されたり，Diagnostics Server（Production および Test モード）に送信されることがあります。Diagnostics Server 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。実行ですべてのメソッドの情報をキャプチャする必要がない限り，このメッセージは無視してください。

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，以下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。
- Business Availability Center，LoadRunner および Performance Center のキャプチャで，Diagnostics Server が送信されるイベントの数に対応できない場合，スロットリング機能は，メソッドの有効な最小レイテンシの値を自動的に増やします。

深さの除外の制御

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，関心の低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除外されます。

次に例を示します。

- **maximum.stack.depth** が 3 の場合
- /login.do calls a() calls b() calls c()
- /login.do，a および b だけがキャプチャされます。

maximum.stack.depth プロパティを設定する際は，以下の点に留意してください。

- **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。
- AM モードで，**maximum.stack.depth** を Diagnostics Server の深さの除外よりも高く設定しても意味がありません。Diagnostics Server の深さの除外は，**＜ Diagnostics Server のインストール・ディレクトリ＞ /etc/server.properties** で **trimming.type**=depth を使って設定します。このプロパティはデフォルトで無効になっています。

Probe のスロットリングの制御

Java Probe のデフォルト設定は，Diagnostics 情報の収集に対するパフォーマンスの影響を最小化するように設定されています。また，Probe は，処理によってアプリケーションのパフォーマンスに悪影響が出始めていることを検出したときに，キャプチャする情報の量を自動的に減らすことができます。Probe がキャプチャする情報の量を減らすプロセスを**スロットリング**と呼びます。

スロットリングについて

Probe は，比較的早く生じるメソッド呼び出しを省略して，収集する Diagnostics の情報の量をスロットリングします。また，Probe は，収集しているデータの量に基づいて，比較的早い方法を正確に特定します。省略されたメソッド呼び出しは，設定されたレイテンシの除外の最小値（デフォルトで 51 ms）で開始する期間を持つことができます。負荷が増えると，レイテンシの除外の最小値が大きくなります。負荷が減ると，除外の値が設定された最小値まで小さくなります。

スロットリングの制御に使われる Probe プロパティは，次のとおりです。これらのプロパティは，**＜ Probe のインストール・ディレクトリ＞ /etc/capture.properties** にあります。

- **gentle.reserve.buffer.count**

  gentle reserve buffer は，負荷スロットリング測定がデプロイしている負荷スパイクに一時的に割り当てられます。デフォルトで，**gentle.reserver.buffer.count** プロパティは，**maximum.private.buffer.count** プロパティと同じ値に設定されています。

- **hard.reserve.buffer.count**

  hard reserve buffer は，gentle reserve buffer では対応できないと判断されたときに負荷スパイクに割り当てられます。デフォルトで，**hard.reserve.buffer** プロパティは，**maximum.private.buffer.count** プロパティと同じ値に設定されています。

- **buffer.wait.time**

  buffer.wait.time プロパティを使って，イベントがバッファされるまでに Probe が待機する時間を制御します。このプロパティは，すべてのスロットリングの試行を使い果たし，イベントをバッファできないときの実行内容を Probe に伝えます。

  - -1 = 処理は可能な限り待機する必要があることを示します。
  - 0 = イベントをただちに中止する必要があることを示します。
  - # = 処理は，イベントを中止する前に # ミリ秒間待機する必要があることを示します。

スロットリング機能の停止

Probe を使って実稼動システムを監視している場合，Probe で自動的にスロットリングできるようにすることができます。ただし，何らかのパフォーマンス問題を診断しているときは，場合によって，すべての診断イベントがキャプチャできるようにするために，アプリケーションのパフォーマンスを犠牲にすることがあります。

スロットリングしないように Probe を設定するには，**＜ Probe のインストール・ディレクトリ＞ /etc/capture.properties** ファイルで次のプロパティを編集する必要があります。

- **gentle.reserve.buffer.count** を 0 に設定します。
- **hard.reserve.buffer.count** を 0 に設定します。
- **buffer.wait.time** を -1 に設定します。

注： スロットリングをオフにしている場合，これら 3 つすべてのプロパティを上記のように設定してください。

スロットリングの調整

Probe のログを見直して，スロットリングがいつ開始して，いつ停止するのかを確認することができます。予想よりもスロットリングの実行時間が長い場合，Probe のスレッド設定が，アプリケーション・サーバに設定されているスレッドの数と同期していることを確認する必要があります。アプリケーション・サーバに，Probe で処理できる数を上回るスレッドがある場合，スロットリングはただちに作動し，解除されないことがあります。

Probe が開始した後にログ・メッセージを確認して，Probe にイベントを作成できるスレッドの最大数を確認できます。次のようなメッセージが記録されているはずです。

```
2006-02-01 12:37:30,203 INFO class com.mercury.opal.capture.util.BufferPool
[main] BufferPool ready:buffer size=6600000 (max 66 threads), # events in
throttle overflow=4000000
```

この例で，スレッドの最大数は 66 になっています。

イベントを作成可能なスレッドの最大数は，**＜ Probe のインストール・ディレクトリ＞ /etc/capture.properties** プロパティ・ファイルで **maximum.private.buffer.count** プロパティを使って制御することができます。

ログ・メッセージが示すスレッドの最大数が，アプリケーション・サーバで処理可能なスレッドの数よりも少ない場合，このプロパティの値を大きくすることができます。これにより，Probe のメモリのオーバーヘッドが増えるため，この値を大きくするときは注意してください。メモリのオーバーヘッドに心配がある場合は，最大バッファ数の増分に比例して，**maximum.buffer.size** および **minimum.buffer.size** プロパティの値を小さくする必要があります。

プロキシ・サーバへの Probe の設定

重要： このセクションは，Diagnostics Server で Probe を使用している場合だけ必要になります。

Probe に Diagnostics Server（Commander モード）の URL を伝えるためのプロパティは 2 つあります。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**

  Probe をインストールするときに，**＜ Probe 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。Probe と Diagnostics Server（Commander モード）の URL の間に直接接続がある場合，このプロパティの値は Probe によって使用されます。

- **webserver.properties** の **registrar.url**

  プロキシがある場合，**＜ Probe 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 Server（Commander モード）の URL を指定する必要があります。

SaaS の Probe へのリバース HTTP の設定

HP Software as a Service（SaaS）環境で使用するために Java Probe をインストールしている場合，Probe と Diagnostics Server（Mediator モード）の間のリバース HTTP（RHTTP）通信を無効にすることをお勧めします。この設定によって，SaaS 環境のほかのカスタマがアプリケーションから不用意に測定値を取得しないように防ぎます。

次の手順に従って，Probe をインストールした後に RHTTP を無効または有効にすることができます。

リバース HTTP の無効化

Probe をインストールしたときに RHTTP を無効にし忘れた場合，次の手順に従って手動で無効にすることができます。

RHTTP を無効にするには，次の例のように **＜ Probe のインストール・ディレクトリ＞ /etc/modules.properities** の **dispatcher.objects** プロパティ値をコメントアウトします。

```
############################################################
## Dispatcher Module properties
############################################################
dispatcher.objects=¥
  com.mercury.diagnostics.capture.correlation.CorrelationSink, ¥
  com.mercury.opal.capture.dispatcher.ac.AppCriticTarget, ¥
  com.mercury.diagnostics.capture.correlation.CacheTarget, ¥
  com.mercury.diagnostics.capture.onlinecache.JavaprobeCacheStructure, ¥
  com.mercury.diagnostics.common.onlinecache.ProbeTreeSelector, ¥
  com.mercury.diagnostics.common.net.InternalCommSecurityManager, ¥
  com.mercury.diagnostics.common.modules.HostNameResolver, ¥
  com.mercury.diagnostics.probe.enterprise.RegistrarCommunication, ¥
  com.mercury.diagnostics.probe.enterprise.ServerCommunication, ¥
  com.mercury.diagnostics.probe.enterprise.ServerUpdate, ¥
  com.mercury.diagnostics.probe.DataPullingTokenProvider, ¥
  com.mercury.diagnostics.probe.enterprise.MediatorManager
#  , ¥
#  com.mercury.diagnostics.probe.enterprise.ReverseClientProbeImpl
```

リバース HTTP の有効化

Probe で誤って RHTTP を無効にした場合，次の手順を使って RHTTP を手動で有効にすることができます。

RHTTP を有効にするには，次の例のように **＜Probe のインストール・ディレクトリ＞ /etc/modules.properities** の **dispatcher.objects** プロパティ値をコメントアウトします。

```
#################################################################
## Dispatcher Module properties
#################################################################
dispatcher.objects=¥
  com.mercury.diagnostics.capture.correlation.CorrelationSink, ¥
  com.mercury.opal.capture.dispatcher.ac.AppCriticTarget, ¥
  com.mercury.diagnostics.capture.correlation.CacheTarget, ¥
  com.mercury.diagnostics.capture.onlinecache.JavaprobeCacheStructure, ¥
  com.mercury.diagnostics.common.onlinecache.ProbeTreeSelector, ¥
  com.mercury.diagnostics.common.net.InternalCommSecurityManager, ¥
  com.mercury.diagnostics.common.modules.HostNameResolver, ¥
  com.mercury.diagnostics.probe.enterprise.RegistrarCommunication, ¥
  com.mercury.diagnostics.probe.enterprise.ServerCommunication, ¥
  com.mercury.diagnostics.probe.enterprise.ServerUpdate, ¥
  com.mercury.diagnostics.probe.DataPullingTokenProvider, ¥
  com.mercury.diagnostics.probe.enterprise.MediatorManager , ¥
  com.mercury.diagnostics.probe.enterprise.ReverseClientProbeImpl
```

VMware 上で実行中の Probe の時刻の同期

VMware ゲストで実行する Probe 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 Probe 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考えられます。

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 Probe 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 Tools ツールのインストール方法については，VMware ESX Server の『Basic System Administration』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。

例外ツリー・データの制限

Probe は例外情報を収集し，それを使っ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。

デフォルトでは，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，注目していない例外は，表示，データ収集，転送処理の負荷がかかるため，すべての例外情報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，Probe によって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。

収集する例外ツリー・データは，次の 2 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と，例外タイプの数を制限する方法です。

特定の例外タイプの制限

＜ Probe 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。

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。デフォルトでは，このプロパティは 4 に設定されています。これは Probe のデータ収集サイクル中に発生した最初の 4 つの例外だけを使って例外ツリーを構築することを示します。必要に応じてこの設定を増減できます。

例

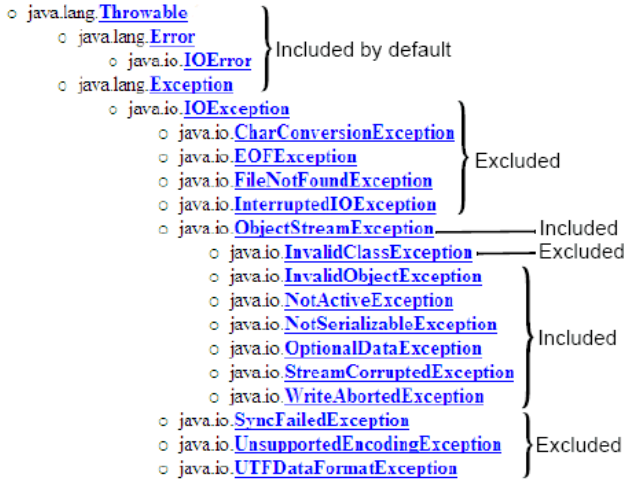
次の例では，タイプが ClassNotFoundException とそのすべてのサブタイプの例外が無視されます。

```
...
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.InvalidClassExceptionexception
.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 Probe の管理ページ

Diagnostics Probe の管理ページを使って，Java Probe および Profiler の設定を行うことができます。Diagnostics Probe の管理ページには，ブラウザから直接アクセスできます。

Diagnostics Probe の管理ページへのアクセス

ブラウザで Diagnostics Probe の管理ページを開くことができます。

Diagnostics Probe の管理ページにアクセスするには

1 ブラウザで, http:// ＜ Probe のホスト＞ : ＜ Probe のポート＞ にアクセスします。

Probe 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 を開く**］: 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，403 ページ「Diagnostics Probe の［コンポーネント］ページ」を参照してください。
- ［**権限と認証の管理**］: Probe 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。
  - Probe が Diagnostics Server と一緒に動作するように設定されている場合，Probe（Profiler）の認証と承認設定は，この Probe が接続されている Diagnostics Server（Commander モード）から管理されます。この場合，このオプションをクリックすると，その Diagnostics Server（Commander モード）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。

➤ Probe が Profiler 専用で動作するように設定されており，Diagnostics Server 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，編集および削除したり，ユーザの権限を変更することができます。詳細については，404 ページ「スタンドアロン・モードの Java Diagnostics Profiler での認証と承認」を参照してください。

Diagnostics Probe 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］ページにアクセスすることができます。

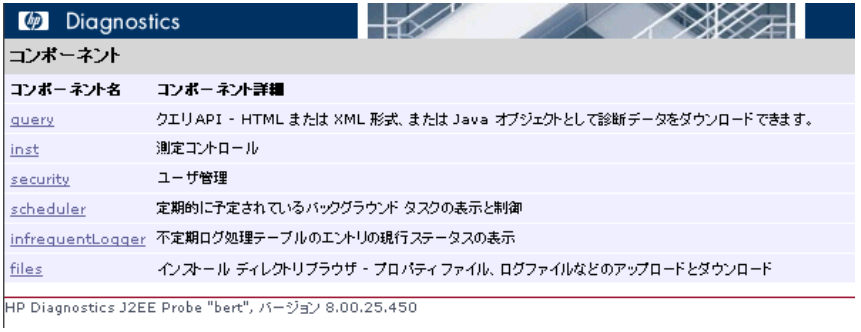
［コンポーネント］ページにアクセスするには

1 401 ページ「Diagnostics Probe の管理ページへのアクセス」の説明に従って，Diagnostics Probe 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 いずれかのオプションをクリックします。

➤ ［**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ゥルメンテーション・オプションが含まれます。Probe の測定の詳細については，223 ページ「Java アプリーションのカスタム・インストゥルメンテーション」を参照してください。

➤ ［**security**］：Probe 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。

- Probe が Diagnostics Server と一緒に動作するように設定されている場合，Probe（Profiler）の認証と承認設定は，この Probe が接続されている Diagnostics Server（Commander モード）から管理されます。この場合，このオプションをクリックすると，その Diagnostics Server（Commander モード）にリダイレクトされます。詳細については，573 ページ「ユーザの認証と承認」を参照してください。
- Probe が Profiler 専用で動作するように設定されており，Diagnostics Server 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，編集および削除したり，ユーザの権限を変更することができます。詳細については，404 ページ「スタンドアロン・モードの Java Diagnostics Profiler での認証と承認」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**］：不定期のロギング・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。

スタンドアロン・モードの Java Diagnostics Profiler での認証と承認

Java Probe を Profiler 専用でインストールしている（Diagnostics Server に接続していない）場合，Diagnostics Probe 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します。

注： Java Probe が Diagnostics Server と連携して動作するように設定されている場合，Probe（Profiler）の認証と承認設定は，この Probe が接続されている Diagnostics Server（Commander モード）から管理します。詳細については，573 ページ「ユーザの認証と承認」を参照してください。

Standalone Java Diagnostics Profiler のユーザの認証と承認を管理するには

1 **Diagnostics Probe の管理ページにアクセスします。**

ブラウザで，http:// ＜Probe のホスト＞：＜Probe のポート＞ にアクセスします。Probe は，35000 で始まる最初の空きポートに割り当てられます。

Diagnostics Probe 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，および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

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ゥ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** および **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することができます。

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

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

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 時間測定値の収集の設定

CPU 時間測定値は，Transaction ビュー，Probes ビュー，Call Profile ビュー，Portal Components ビューなど，各種のビューの［Details］ペインに表示されます。CPU 時間測定値の収集の有効化，無効化，設定を行うことができます。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 時間測定値は，次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。

- Windows
- Solaris 10+
- AIX 5.2+
- HP-UX 11i Itanium
- Linux 5.x および SUSE Linux 10.x

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・システムによってだけでなく，プラットフォーム・アーキテクチャ（たとえば SPARC と x86 など）によっても異なります。特定の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 の製品可用性マトリックス（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。詳細については，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 399 ページ「VMware 上で実行中の Probe の時刻の同期」を参照してください。

デフォルトでは，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 時間測定値の収集を無効にしたり，CPU 時間測定値の収集をプロパティ・ファイルまたは Java Diagnostics Profiler UI で設定したりできます。

Java Probe の場合は，CPU 時間測定値の収集は次の 2 つのプロパティによって制御されます。

- **＜Probe 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

  このプロパティはデフォルトで **true** に設定されており，CPU 時間測定値の収集が有効になっています。ただし，どの CPU スタンプが収集されるかは，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 時間測定値は，Probe 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。

- **＜Probe 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 ー** CPU タイムスタンプはありません。
- **1 ー** 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。

- デフォルト値は **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。しかし，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，デフォルト値は 0 になります。
- **2 ー** 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** ー 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ゥ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 つの方法は，次のように Java Diagnostics Profiler の［設定］タブを使用することです。

1 **Profiler** UI で，［**設定**］タブを選択します。この Probe 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 ［設定］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ゥ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 ［設定］タブに変更を加えたら，［**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。Probe を再起動する必要はありません。

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）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 ヘッダに表示される値
- SOAP エンベロープまたはボディに表示される値
- HTTP ヘッダに表示される値
- 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・プロパティまたはヘッダ
- 特定の IP アドレスまたは IP アドレスの範囲

重要：SOAP のヘッダ，エンベロープ，またはボディに基づいてコンシューマ ID を定義するには，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・サーバでは，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ゥルメンテーションが Diagnostics に提供されます。

しかし，WebSphere 5.1 JAX-RPC と Oracle 10g JAX-RPC では手動設定が必要です。詳細については，184 ページ「SOAP メッセージ・ハンドラの設定」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・インストゥルメンテーションを使って 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ことがあります。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 の製品可用性マトリックス（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 Availability Center に対しても有用です。Business Availability Center 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見て，パフォーマンスの特徴を比較できます。

注：Diagnostics が BAC と統合されている環境では，コンシューマ ID 集計機能は BAC バージョン 7.5 以上でのみサポートされます。

コンシューマ ID の設定は任意です。SOAP over HTTP/S Web サービスのデフォルトのコンシューマ ID としては IP アドレスが使用され，SOAP over JMS Web サービスのデフォルトのコンシューマ ID としては受信キュー名（またはトピック名）が使用されます。

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 コンシューマ別にグループ化する Java Probe ごとに，413 ページ「Java Probe のルール構文と例」で説明されているように consumer.properties ファイルを書き換えます。

2 5 つよりも多くのコンシューマ・タイプを追跡する予定である場合は，dispatcher.properties ファイルで max.tracked.ids.per.probe の設定を書き換えます。

3 probe/files/log ディレクトリにある ＜ Probe の名前＞ _id.properties ファイルを確認します。＜ Probe の名前＞ _id.properties ファイルを完全に削除するか，または直前の手順で行った consumer.properties の変更にあわせて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，Probe ごとの制限に達すると，ほかのすべてのコンシューマは「Other」として分類されます。

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。すなわち，SOAP ルール，HTTP ヘッダ・ルール，JMS Web サービス・ルール，および 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，次に JMS ルールが適用され，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。

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 ルールを使って，SOAP ヘッダ，エンベロープ，またはボディの 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。一致するものがあれば，Probe 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 ヘッダ内に要素が見つからない場合，このルールはスキップされ，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。

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）にすることができます。

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 Probe のルール構文と例

コンシューマ ID の割り当ては **consumer.properties** でルールを指定することによって制御されます。

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 という <pattern-name> を入力した場合，パターンが一致するには Web サービス名に大文字の T と大文字の S がなければなりません。

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なければなりません。

<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限り，測定値は CallerA に対してさまざまな値でグループ化されます。以下は，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>
     <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>
```

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。

<rule-name>**;HTTP_WS;**<pattern-name> **= soap-envelope;**<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なければなりません。

<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なければなりません。

<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なければなりません。

<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限り，測定値は User-Agent に対してさまざまな値でグループ化されます。以下は，ヘッダのコンシューマ ID にマップされる HTTP ヘッダからの抜粋です。

```
 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なければなりません。

<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「¥:」を使っ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なければなりません。

<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「¥:」を使っ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なければなりません。

<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 でなければなりません。

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「¥:」を使ってエスケープする必要があります。

特定の IP アドレス

ある 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，次の書式を使用します。

<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-wildcards> = <consumerID-string><rule-name>; **IP**; <IP-address-ranges> = <consumerID-string>

<IP オクテット > は整数，*，または で指定する範囲です。

次のルールは，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 から 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 の失敗ペイロード・データの設定

SOAP の失敗が検出されると，SOAP の失敗データに SOAP ペイロードが追加されます。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。

重要：SOAP ペイロードをキャプチャするには，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・サーバでは，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ゥルメンテーションが Diagnostics に提供されます。

しかし，WebSphere 5.1 JAX-RPC と Oracle 10g JAX-RPC では手動設定が必要です。詳細については，184 ページ「SOAP メッセージ・ハンドラの設定」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・インストゥルメンテーションを使って 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ことがあります。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 の製品可用性マトリックス（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

Java Probe では，**＜Probe 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デフォルトの 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 サービスの設定

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。**＜ Probe のインストール・ディレクトリ＞ ¥ etc¥rest.properties** を参照してください。

現在，HTTP のみ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

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，その種類に応じて 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，**＜ Probe 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。

第 16 章

.NET Probe 設定ファイルについて

.NET Probe 設定ファイル ＜**Probe 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 Probe の設定を制御します。

本章の内容

.NET Probe 設定ファイルについて

以下の項では，.NET Probe の設定ファイルである ＜**Probe 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を定義します。このファイルは，＜**Probe のインストール・ディレクトリ**＞**/etc/probe_config.xsd** ファイルで定義されます。各要素について，その目的，属性，親要素および子要素を説明して定義していきます。ここに記載されている事柄について疑問がある場合は，必ず ＜**Probe のインストール・ディレクトリ**＞**/etc/probe_config.xsd** ファイルを確認してください。TransactionVision に固有の .NET Probe の追加設定要素については，『**HP TransactionVision デプロイメント・ガイド**』を参照してください。

<appdomain> 要素

目的

複数の appdomain 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。

値がない場合は，すべてに適用されます。

属性

属性	有効な値	デフォルト値	説明
enabled	truefalse	true	appdomain のインストゥルメントを許可します。
name	文字列	なし	.NET AppDomain 設定が適用される名前。

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="MSPetShop"/>
```

<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	デフォルト値	説明
username		admin	
password		admin	**＜ Probe 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ります。

要素

発生数	0 ～多数
親要素	profiler
子要素	なし

例

```
<profiler autheticate="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	デフォルト値	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	デフォルト値	説明
enabled	truefalse	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	デフォルト値	説明
enabled	truefalse	false	コンシューマ ID ルールの評価を行います。

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	デフォルト値	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 Server 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	デフォルト値	説明
username		なし	Diagnostics Server での確認に使います。
password		なし	Diagnostics Server での確認に使います。
authenticate	true，false	true	認証を有効 / 無効にします。D4 以前のリリースとの下位互換性のために，これを無効にする必要があります。

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="" password="" authenticate="true"/>
```

<depth> 要素

目的

深さの除外を設定します。

属性

属性	有効な値	デフォルト値	説明
enabled	truefalse	true	深さの除外を有効にします。
depth	数値	25	深さの除外の深さを設定します。

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 Server 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	デフォルト値	説明
url	レジストラの URL。http://＜ホスト＞:＜ポート＞	なし	レジストラに接続するための 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	
proxypassword	プロキシのパスワード	なし	
registeredhostname	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

```
<diagnosticsserver url="http://localhost:2006/commander" delay="2"
keepalive="15" proxy="?" proxyuser="?" proxypassword="?"
registerhostname="?" register_byip="false"/>
```

<exceptiontype> 要素

目的

例外タイプを定義します。

属性

属性	有効な値	デフォルト値	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（詳細については，この機能に関するセクションを参照してください)。

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

除外するコレクション・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

```
<exclude>System.Collections.HashTable</exclude>
<exclude>System.Collections.ArrayList</exclude>
```

<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測定値をフィルタリングします。

属性

属性	有効な値	デフォルト値	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"/>
```

<httpheaderule> 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	デフォルト値	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> 要素のすべてが含まれます。

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

ID およびグループ ID を提供します。

属性

属性	有効な値	デフォルト値	説明
probeid	文字，数字，アンダースコア，ダッシュ，ピリオド	$(AppDomain).NET	LoadRunner / Performance Center およびシステムの状況で使用する Probe に名前を付けます。
probegroup		デフォルト値	レポート測定値で使う Probe のグループ化を定義します。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig
子要素	なし

例

```
<id probeid="$(AppDomain).NET" probegroup="Default"/>
```

<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

その他のすべての除外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 の記録設定が含まれます。

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	logging

例

```
<instrumentation>
    <logging level="off" threadids="true"/>
</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	デフォルト値	説明
id	文字列	なし	コンシューマ ID ルールの評価を行います。
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> 要素のすべてが含まれます。

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

レイテンシの除外を設定します。

属性

属性	有効な値	デフォルト値	説明
enabled	truefalse	true	レイテンシの除外を有効にします。
throttle	truefalse	true	レイテンシのスロットリングを有効にします。
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	スロットリングを増やす前のバッファの使用割合。
decrement threshold	数値	50	スロットリングを減らす前のバッファの使用割合。

例

```
<trim>
    <latency enabled="true" throttle="true" min="2" max="100" increment="2"
incrementthreshold="75" decrementthreshold="50"/>
</trim>
```

要素

発生数	1
親要素	trim
子要素	なし

<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。

属性

属性	有効な値	デフォルト値	説明
enabled	truefalse	true	
maxdirsize	数値	1,024 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

Probe のインストゥルメンテーション・プロセスの記録レベルを設定します。

属性

属性	有効な値	デフォルト値	説明
level	offassertbreaksev erewarninginfode bugpointsehsigch icilclassmapilas msymboldeepmo deloadallchecksu mproperty	"" これは「info」と同等です	
threadids	truefalse	true	

注：info よりも下にある有効な値は，通常は使用すべきではありません。Probe 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=""/ threadids="true">
</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 Probe 処理の記録レベルを設定します。

属性

属性	有効な値	デフォルト値	説明
level	offseverewarninginfodebugeventspropertywebserverhttpsymbolsprobemetricsregistrarthreadpoolauthentication|bufferpoolrumbacforsoavmwareexceptions|	"" これは「info」と同等です	

注： info よりも下にある有効な値は，通常は使用すべきではありません。Probe 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 level=""/>
```

<lwmd> 要素

目的

Light-Weight Memory Diagnostics 機能を設定します。

属性

属性	有効な値	デフォルト値	説明
enabled	truefalse	false	LWMD キャプチャを有効にします。
sample	文字列	1m	サンプル間隔（h- 時間 /m- 分 /s- 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 Server を指定します。

属性

属性	有効な値	デフォルト値	説明
host	ホスト名	なし	Mediator の名前
port	数値	2612	メディエータのポート
ssl	true/false	false	Diagnostics Server の URL が http で始まる場合，デフォルト値は false です。 Diagnostics Server の URL が https で始まる場合，デフォルト値は true です。
metricport	数値	2006	ヒープの使用量や可用性など，Probe が Probe 測定値を送信するポート。
block	true/false	false	Mediator 接続が確立するまでブロックします。
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	デフォルト値	説明
localportstart	数値	4000	Diagnostics Server（Mediator モード）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	Diagnostics Server（Mediator モード）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"/>
```

<modes> 要素

目的

Probe に実行中のモードを伝えます。

属性

属性	有効な値	デフォルト値	説明
enterprise	truefalse	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 モードの Probe（Probe は，Diagnostics Server と連動します）。
ent	truefalse	➤ false	enterprise 属性の省略形
ad	truefalse	➤ false	
am	truefalse	➤ false	
pro	truefalse	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 モードの Probe。
tv	truefalse	false	TransactionVision イベントのキャプチャを有効にします。必要なトランスポート設定の設定方法については，『**HP TransactionVision デプロイメント・ガイド**』を参照してください。
auto	auto	pro が true の場合，auto は auto に設定されます。	

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

```
<modes enterprise="false" ent="false" ad="false" am="false" pro="true"
auto="auto"/>
```

<points> 要素

目的

インストゥルメンテーションの包含のキャプチャ・ポイント・ファイルを一覧表示します。

属性

属性	有効な値	デフォルト値	説明
file	文字列	なし	インストゥルメンテーション・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

Probe 設定に，containing（包含）ルート要素を提供します。

属性

なし。

要素

発生数	1
親要素	なし
子要素	appdomain，Cbufferpool，Ccaptureexceptions，Cconsumeridrules，Ccredentials，Cdiagnosticsserver，Ceventserverhost，Cid，Cinstrumentation，Cipaddress，Clogging，Clwmd，Cmodes，Cpoints，Cprocess，Cprofiler，Csample，Csoappayload，Ctrim，Cwebserver，Ctopology，Cvmware

例

```
<probeconfig>
</probeconfig>
```

<process> 要素

目的

Probe に適用される包含フィルタ・リストを提供します。

ない場合もあります。

属性

属性	有効な値	デフォルト値	説明
enablealldomains	truefalse	false	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。
monitorheap	文字列	false	値を true に設定したときに，Heap タブとヒープ・ブレークダウン処理を有効にします。

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="false" name="ASP.NET" monitorheap="false">
```

<profiler> 要素

目的

Probe の Profiler 機能の設定を含みます。

属性

属性	有効な値	デフォルト値	説明
authenticate	true，false	なし	着信 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。
inactivitytime-out	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・アクセスを無効にします。このため，ユーザ / パスワードは公開されなくなりますが，Probe のマシンへの telnet/RemoteDestTop を実行して Profiler をローカルで実行することは可能です。

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 autheticate="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（有効 / ヘッダ）を制御します。

属性

属性	有効な値	デフォルト値	説明
enable	truefalse	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		＜ Probe 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ります。

要素

発生数	親 1 つあたり 1 回
親要素	probeconfig
子要素	なし

要素の例

```
<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 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	デフォルト値	説明
method	percent，count，period	percent	サンプリング方法を設定します。 ➤ 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・レートを設定します。

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

```
<sample method="percent" rate="0"/>
```

<soappayload> 要素

目的

SOAP 失敗時の SOA 用 BAC SOAP ペイロード・キャプチャ機能を設定します。

SOAP 失敗時の SOA 用 BAC SOAP ペイロード・キャプチャ機能により，SOAP の失敗に関連する SOAP ペイロードが提供されます。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	デフォルト値	説明
enabled	truefalse	true	SOAP ペイロード・キャプチャ機能を有効または無効にします。
size	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。指定されており，設定にエラーがある場合もデフォルト値が使われます。

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" />
```

<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	デフォルト値	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> 要素のすべてが含まれます。

属性

なし。

要素

発生数	1
親要素	consumeridrules
子要素	soaprules

例

```
<soaprules>
</soaprules>
```

<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	デフォルト値	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	なし

例

```
<symbols maxuri="1000" maxuriname="Maximum number of unique URIs exceeded" maxsql="1000" maxsqlname="Maximum number of unique SQLs exceeded"/>
```

<topology> 要素

目的

topology 設定プロパティを制御します。

属性

属性	有効な値	デフォルト値	説明
enable	truefalse	true	トポロジ情報の収集と，それらの情報を Diagnostics Server 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">
```

<trim> 要素

目的

除外機能を設定して Probe と Diagnostics Server 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 Server に送信する必要がないため，Profiler ユーザ・インタフェースは，深さの除外やレイテンシの除外といった，設定されているすべての除外設定を無視します。

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>
```

<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	デフォルト値	説明
attempttime-stampadjust-ments	truefalse	false	VMware 環境でのタイムスタンプの調節を可能にします。

要素

発生数	1
親要素	probeconfig
子要素	なし

例

```
<vmware attempttimestampadjustments="false"/>
```

<webserver> 要素

目的

Probe 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	デフォルト値	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			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>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>
```

第 17 章

.NET Probe の詳細設定

.NET Probe 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

VMware 上で実行中の Probe の時刻の同期

VMware ホストで実行する .NET Probe 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 Probe 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 Probe 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考えられます。

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 Probe 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 Tools ツールのインストール方法については，VMware ESX Server の『Basic System Administration』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ゥルメンテーションのカスタマイズ

.NET Probe をインストールすると，監視対象サーバにおいて Probe がすべての ASP.NET 処理に適用する標準のインストゥルメンテーションで，**ASP.NET.points** が作成されます。

アプリケーション特有のインストゥルメンテーション・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ゥルメンテーション・ポイントは，**＜ Probe 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，Probe 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することができます。Reflector の使用方法については，476 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ゥルメンテーション・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 Probe 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，インストゥ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，**＜ Probe 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 Probe 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，インストゥルメンテーション・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。

以下は，Probe 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表す例です。

```
[IBuySpy Callee]
class = !IBuySpy.*
method = !.*
signature =
scope =
ignoreScope =
layer = Custom.IBuySpy
```

インストゥ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・インストゥルメンテーション」を参照してください。

3 **probe_config.xml** の Probe 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。

```
<appdomain name="your app name">, enabled="true"
        <points file="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。

```
<?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="MSPetShop", enabled="true">
                  <points file="MSPetShop.points"/>
            </appdomain>
      </process>
</probeconfig>
```

4 214 ページ「IIS の再起動」の手順に従って IIS を再起動します。

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 Probe にインストールされている Reflector 自動検出ツールを使って，Probe のインストゥ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ることができます。Reflector の実行可能ファイルは，**＜ Probe 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

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。

```
＜ Probe 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。

```
＜ Probe 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。

```
＜ Probe 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あ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。

＜ Probe 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。

472 ページ「ASP.NET アプリケーションのインストゥ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ゥルメンテーションをカスタマイズします。

レイテンシの除外およびスロットリングの設定

Probe でキャプチャしているデータの量に Diagnostics Server 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 Probe で特定すると，**レイテンシの除外**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すことができます。デフォルトで，Probe が必要に応じて作業負荷を調整できるように，レイテンシの除外は有効になっています。

レイテンシの除外が有効になると，.NET Probe は，最小レイテンシしきい値を下回る合計レイテンシが生じているメソッドを無視して，キャプチャするメソッドの数を減らします。除外は，Probe がフリーズしたり実行を停止するよりも，関心が低いメソッドをキャプチャしないことでレイテンシを短縮する方がいいという考えに基づきます。除外によって，Probe は実行を継続し，レイテンシの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，除外されたメソッドに関する一部の情報が Diagnostics Server に送信されることがあります。Diagnostics Server 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：

- レイテンシの除外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。
- Diagnostics Server を設定して，Diagnostics ユーザ・インタフェースに表示される Probe のデータの細分性に影響する Probe データの追加除外を適用することができます。

レイテンシの除外の無効化

デフォルトで，除外は .NET Probe で有効になっています。除外を無効にするには，Probe の設定を変更する必要があります。

レイテンシの除外を無効にするには

次の例のように，
＜ Probe 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>
```

レイテンシの除外をオンにする latency 要素の属性は，**enabled** です。**enabled** を true に設定すると，レイテンシの除外が有効になります。**enabled** 属性を false に設定すると，レイテンシの除外が無効になります。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 16 章「.NET Probe 設定ファイルについて」を参照してください。

レイテンシの除外の有効化

デフォルトで，除外は .NET Probe で有効になっています。後で除外を無効にした場合，Probe の設定を再び有効にする必要があります。

レイテンシの除外を有効にするには

次の例のように，
＜ Probe 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>
```

レイテンシの除外をオンにする latency 要素の属性は，**enabled** です。**enabled** を **true** に設定すると，レイテンシの除外が有効になります。**enabled** 属性を **false** に設定すると，レイテンシの除外が無効になります。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 16 章「.NET Probe 設定ファイルについて」を参照してください。

レイテンシの除外のしきい値の設定

デフォルトで，レイテンシの除外のしきい値は，2 ms 未満のレイテンシを持つメソッドが除外され，100 ms より長いレイテンシを持つメソッドを除外しないように設定されています。

min 属性の値を調整して，除外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，除外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，
＜ Probe 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>
```

除外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。レイテンシの除外が有効になると，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除外されます。この属性に値を指定しない場合，デフォルト値 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，除外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除外を無効にします。

- **max**

  レイテンシの最大しきい値を設定します。レイテンシの除外が有効になると，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除外されます。値を指定しない場合，この属性のデフォルト値は 100 ms になります。

latency 要素の属性と要素については，第 16 章「.NET Probe 設定ファイルについて」を参照してください。

レイテンシの除外のスロットリングの設定

デフォルトで，レイテンシの除外はスロットリングされます。スロットリングを有効にすると，Probe が行う除外の量は，Diagnostics Server の処理中バックログによって使われている Probe 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除外されます。

Probe で使われるリソースの割合が，設定スロットリング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除外されるように，有効な除外しきい値が高くなります。使われる Probe 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除外されるように，有効な除外しきい値が再度高くなります。使われる Probe リソースの割合がスロットリングの増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，有効な除外しきい値が低くなります。

有効な除外しきい値は，最大のメソッド除外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。この属性のデフォルト値は true です。

- **increment**

  使われる Probe リソースの割合が **incrementthreshold** を超えたときに，有効な除外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，有効な除外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 Probe リソースの割合が，この属性の値以上になったとき，有効な除外しきい値が高くなるようにスロットリングがトリガされます。この属性のデフォルト値は **75**% です。

- **decrementthreshold**

  使われる Probe リソースの割合が，この属性の値以下になったとき，有効な除外しきい値が低くなるようにスロットリングがトリガされます。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 16 章「.NET Probe 設定ファイルについて」を参照してください。

深さの除外の設定

.NET Probe は，**深さの除外**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 Server が，Probe がキャプチャしているデータの量に対応できない場合，Probe は深さの除外を使って，リソース不足にならないように助けます。デフォルトで，深さの除外は有効になっています。

注：深さの除外は，Profiler ユーザ・インタフェースに無視されます。

深さの除外が有効になると，.NET Probe 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，Probe が実行を続け，呼び出しスタックの高いところ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，呼び出しスタックの関心の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。

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 は除外されます。

以下は，除外属性が含まれる **probe_config.xml** 設定ファイルの **depth** 要素の例です。

```
<trim>
        <depth enabled="true" depth="10" />
</trim>
```

除外を制御する **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，深さの除外が有効になります。この属性を false に設定すると，深さの除外が無効になります。この属性のデフォルト値は true です。

➤ **depth**

深さの除外に使われるしきい値を設定します。深さの除外が有効になると，この属性以下で呼び出されるメソッドは除外されます。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 16 章「.NET Probe 設定ファイルについて」を参照してください。

Light-Weight Memory Diagnostics 用の .NET Probe の設定

.NET Probe には，Light-Weight Memory Diagnostics（LWMD）の 3 つの異なるレベルがあります。

➤ LWMD オフ（ヒープまたはコレクション測定値をキャプチャしません）

➤ ヒープ測定値だけをキャプチャ

➤ ヒープ測定値とコレクション測定値をキャプチャ

.NET Probe がインストールされている場合，.NET Probe のデフォルト設定では Light-Weight Memory Diagnostics（LWMD）がオフになっているはずです。LWMD は probe_config.xml で有効にできます（447 ページ「<lwmd> 要素」を参照してください）。

コレクション測定値のために LWMD を有効にする場合，以下の手順に従って Probe の設定を変更する必要があります。

注： Probe を有効してコレクション測定値をキャプチャすると，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。

コレクション測定値のキャプチャを有効にするには

➤ **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 Probe 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 **＜ Probe 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，コレクション測定値のキャプチャを有効にするのに必要なインストゥルメンテーション命令が含まれます。

process タグに points タグを追加すると，**probe_config.xml** 設定ファイルは次の例のようになります。

```
<process name="ASP.NET", <enablealldomains="false">
    <points file="ASP.NET.points" />
    <points file="ADO.points" />
    <points file="Lwmd.points"/>
    <appdomain name="your app name", enabled="true">
        <points file="your app.capture points" />
    </appdomain>
</process>
```

appdomain タグに points タグを追加すると，**probe_config.xml** 設定ファイルは次の例のようになります。

```
    <process name="ASP.NET", <enablealldomains="false">
    <points file="ASP.NET.points" />
    <points file="ADO.points" />
    <appdomain name="your app name", enabled="true">
            <points file="your app.capture points" />
            <points file="Lwmd.points"/>
    </appdomain>
</process>
```

LWMD を無効にするには

➤ **Lwmd.points** ファイルの **points** タグを **process** タグ，または **probe_config.xml** 設定ファイルの適切な **appdomain** タグから削除します。

設定ファイルで **points** を使用しないと，Probe は **Lwmd.points** ファイルを見つけられないため，LWMD を有効にするための命令がわからなくなります。

例外スタック・トレース・データの制限

Probe 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，Probe によって収集するスタック・トレース・データの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。

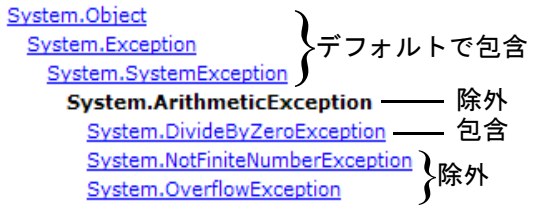
特定の例外タイプの制限

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

デフォルトでは，.NET Probe 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 Server 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］タブを読みやすくすることができます。**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 Server 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，Probe 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，インストゥルメンテーション・セクションの **logging level** タグを変更して，Probe のインストゥルメンテーションの記録機能を無効にすることができます。

```
<instrumentation>
        <logging level="off"/>
</instrumentation>
```

デフォルトの Probe ホスト・マシン名の変更

registered_hostname プロパティを使って，Probe が Diagnostics Server（Commander モード）への登録に使用するデフォルトのホスト・マシン名を変更することができます。ファイアウォールまたは NAT がある場合，または Probe 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，デフォルトのホスト・マシン名を変更しない限り，Diagnostics Server（Commander モード）は Probe と通信できません。

Probe のデフォルトのホスト・マシン名を変更するには，**probe_config.xml ファイル**の .NET Probe **<registrar>** タグにある **registered_hostname** 属性を，Diagnostics Server（Commander モード）が Probe 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。**<registrar>** タグは任意です。したがって，タグ全体を入力しなければならないことがあります。サーバがリスンするように設定されているポートとして，下の例にポート番号 2006 を入力します。以下は，変更の例です。

```
<registrar url="http://foo:2006/registrar" registered_hostname="bar"/>
```

注：

- 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 Availability Center 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。

ホストにインストールされている Probe の一覧表示

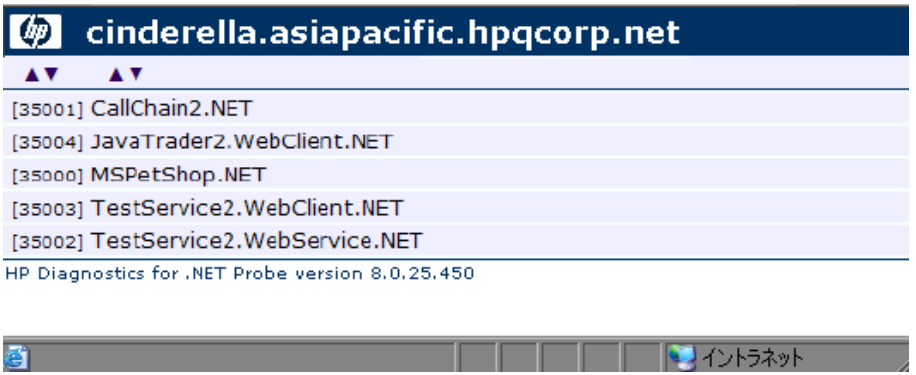
1 台のホストに複数の Probe がインストールされている場合，割り当てられているポートが，Probe の起動時に使用可能なものに基づくため，各 Probe で使用するポートを把握することができません。アプリケーションを起動および停止するときに，特定のアプリケーションの Probe に割り当てられているポートが変更されることがあります。

次の URL にアクセスして，ホストにインストールされている Probe と，それらが使用しているポートを特定することができます。

```
http:// ＜ Probe 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。

Probe とポートの一覧は，次の例のように表示されます。



スタンドアロン・モードの .NET Profiler での認証と承認

.NET Probe を Profiler 専用でインストールしている場合（Diagnostics Server に接続されていない），**＜ Probe のインストール・ディレクトリ＞ /etc/probe_config.xml** ファイルで，Profiler のユーザの認証と承認を管理します。

注：.NET Probe が Diagnostics Server と一緒に動作するように設定されている場合，Probe（Profiler）の認証と承認設定は，この Probe が接続されている Diagnostics Server（Commander モード）から管理されます。詳細については，573 ページ「ユーザの認証と承認」を参照してください。

Diagnostics Server から Probe にアクセスする場合，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 Probe が Profiler 専用でインストールされている場合，デフォルトで，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 へのアクセスに新しいユーザ名と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

➤ **＜ Probe 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。

スタンドアロン .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

1 **＜ Probe 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。

2 **Probe 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，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。

重要：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 BAC に対しても有用です。BAC 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見て，パフォーマンスの特徴を比較できます。

注： Diagnostics が BAC と統合されている環境では，コンシューマ ID 集計機能は BAC バージョン 7.5 以上でのみサポートされます。

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 ヘッダに表示される値
- 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. 494 ページ「.NET Probe のルール構文と例」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 Probe 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 ヘッダの XML 要素から取得できます。

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。

.NET Probe のルール構文と例

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 ヘッダ・ルール

SOAP ヘッダ・ルールにより，コンシューマ ID を SOAP ヘッダの XML 要素から取得できます。ルールと consumeridfield 属性はどちらも SOAP 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。

id は，ユーザがルールの識別に使用できるものであれば任意の名前が可能です。この属性は .NET Probe には使用されません。

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。一致するものがあれば，.NET Probe 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。コンシューマ ID 内の要素は完全された名前が可能です。つまり，名前空間名とローカル部分または修飾されていない名前で構成され，名前空間は関連付けられていません。

SOAP ヘッダ内に要素が見つからない場合，このルールはスキップされ，Probe は定義されている次のルールに進みます。

例：

```
<consumeridrules enabled="true">   <soaprules> <soaprule id="SOAP1"
rule="TestService.*" consumeridfield="Caller"/>
  </soaprules>
</consumeridrules>
```

consumeridfield で定義された要素を正しく見つけるために，SOAP ヘッダを XML で構造化する方法についていくつかの制限があります。consumeridfield で定義された要素は，<soap:header> の下の要素でなければなりません。また，1 レベルのネスト化が可能です。次に例を示します。

```
<soap:header>
        <namespace:Caller>consumer ID</Caller>
</soap:header>
```

SOAP ヘッダ・ルールの例（1 レベルのネスト）

```
<soap:header>
       <namespace:header1>
          <namespace:Caller>consumer ID</Caller>
       </namespace:header1>
</soap:header>
```

ホワイトスペースについて特殊な制限事項がいくつかあります。XML では，ホワイトスペースは空白，キャリッジ・リターン，ライン・フィード，タブとして定義されています。2 つ目の例にホワイトスペースが含まれていると，ユーザは修飾された名前を使用できなくなります。

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 Probe は consumeridfield で定義された要素の 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，Probe 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 Probe 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 の失敗データの設定

SOAP の失敗が検出されると，SOAP の失敗データに SOAP ペイロードが追加されます。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。

.NET Probe では，**＜ Probe のインストール・ディレクトリ＞ ¥etc¥ probe_config.xml** 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。たとえば，次のエントリは，SOAP のペイロードの長さをデフォルトの 5000 から 10000 に増やしています。

```
<soappayload enabled="true" maxsize="10000"/>
```

enabled を false に設定すると，この機能が無効になります。

第 VI 部

プロキシおよびファイアウォールの通信設定

第 18 章

Diagnostics Server，JavaAgent，および .NET Agent への HTTP プロキシの設定

HP 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，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。

本章の内容

Diagnostics Server での HTTP プロキシ通信の有効化

以下のセクションでは，Diagnostics Server（Commander モード）と Diagnostics Server（Mediator モード）を設定して，HTTP プロキシを通じて相互通信を行う手順を紹介します。

Diagnostics Server に HTTP プロキシ通信を設定するには

1 **＜Diagnostics Server 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 Server（Commander モード）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 Server を再起動します。詳細については，52 ページ「Diagnostics Server の起動および停止」を参照してください。

Java Probe での HTTP プロキシ通信の有効化

以下のセクションでは，HTTP プロキシを通じて Java Probe と通信するように Diagnostics Server（Commander モード）を設定する手順について説明します。

Java Probe に HTTP プロキシ通信を設定するには

1 **＜ Probe 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.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 インストゥルメント対象のアプリケーション VM を再起動します。

.NET Probe での HTTP プロキシ通信の有効化

以下のセクションでは，HTTP プロキシを通じて Diagnostics Server（Commander モード）と通信するように .NET Probe を設定する手順について説明します。

.NET Probe に HTTP プロキシ通信を設定するには

1 **＜ Probe 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 Server ホストを指定します。

- **uri** を Diagnostics Server 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 **＜ Probe 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 インストゥ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。

第 19 章

ファイアウォール環境で動作する 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 Probe と Diagnostics コンポーネント，あるいは LoadRunner，Performance Center または Business Availability Center 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 Availability Center のドキュメントを参照してください。

本章の内容

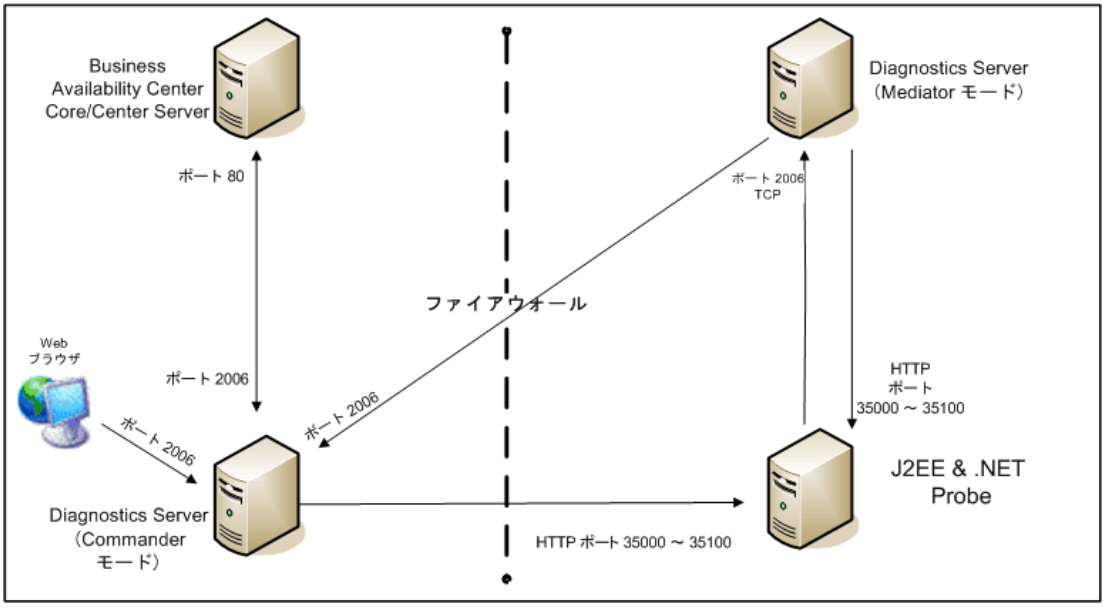
注：これらの設定手順は，HP Diagnostics に関する深い知識を有する熟練ユーザだけが使用してください。Diagnostics コンポーネントの構成設定を変更する際は十分に注意してください。

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 Availability 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 Business Availability Center 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。



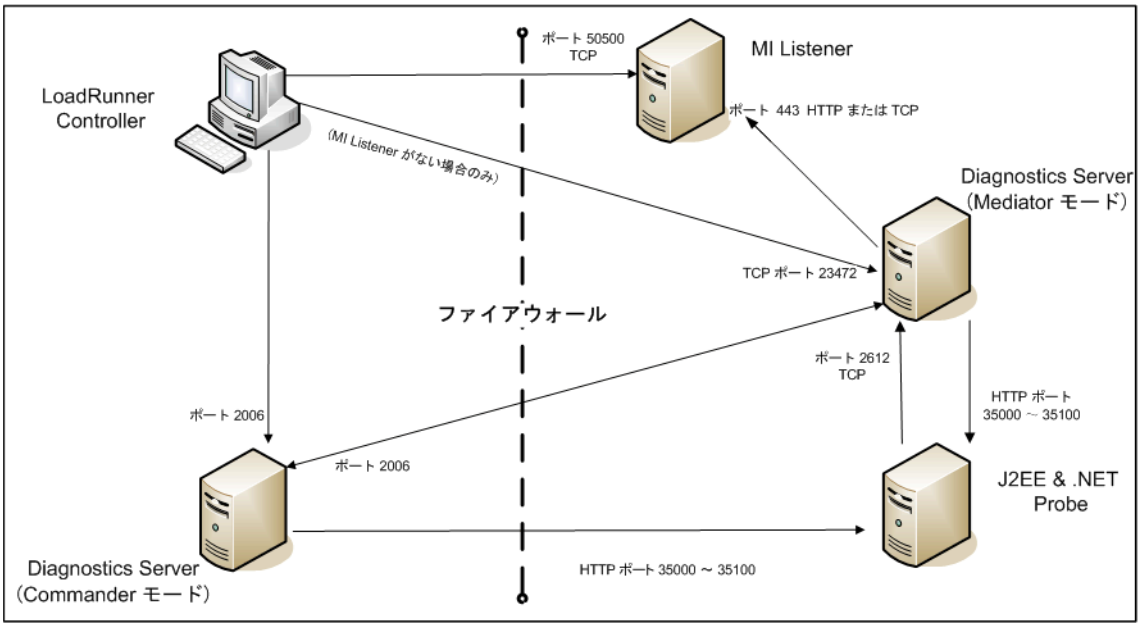
Diagnostics コンポーネント間で通信できるようにファイアウォールを設定するには，ポートを開いて以下を許可します。

- Business Availability Center コア / センター・サーバから，ポート 2006 の Diagnostics Server（Commander モード）への HTTP 要求。
- Diagnostics Server（Commander モード）から，ポート 80 の Business Availability Center コア・サーバへの HTTP 要求。
- Diagnostics Server（Mediator モード）から Probe のポート 35000 ～ 35100 への HTTP 要求。

- Web ブラウザ・クライアント・マシンから，ポート 2006 の Diagnostics Server（Commander モード）への HTTP 要求。
- Diagnostics Server（Mediator モード）から，ポート 2006 の Diagnostics Server（Commander モード）への HTTP 要求。
- Probe から，ポート 2612 の Diagnostics Server（Mediator モード）への TCP 要求。
- Probe から，ポート 2006 の Diagnostics Server（Mediator モード）への HTTP 要求。
- Diagnostics Server（Commander モード）から，ポート 35000 ～ 35100 の Probe への HTTP 要求。通信を許可する実際のポートは，Probe を設定したときに有効にしたポート番号とインストゥルメント対象の VM の数によって異なります。Probe のポート範囲の設定については，383 ページ「複数のアプリケーション・サーバの JVM インスタンスで使用するための Probe の設定」を参照してください。

LoadRunner と Performance Center

下の図は，ファイアウォールによって Probe 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，ポートを開いて以下を許可します。

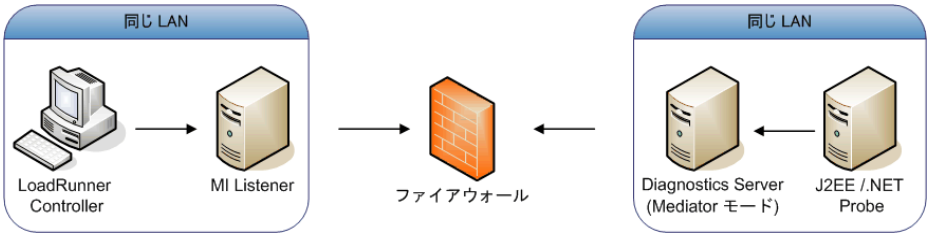
- Diagnostics Server（Mediator モード）から，ポート 2612 および 2006 の Diagnostics Server（Commander モード）への HTTP 要求。
- Probe から，ポート 2612 の Diagnostics Server（Mediator モード）への TCP 要求。
- Probe から，ポート 2006 の Diagnostics Server（Mediator モード）への HTTP 要求。
- Diagnostics Server（Commander モード）から，ポート 35000 ～ 35100 の Probe への HTTP 要求。通信を許可する実際のポートは，Probe を設定したときに有効にしたポート番号とインストゥルメント対象の VM の数によって異なります。Probe のポート範囲の設定については，383 ページ「複数のアプリケーション・サーバの JVM インスタンスで使用するための Probe の設定」を参照してください。

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 Server（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 Probe のある Diagnostics Server 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 Server の間にファイアウォールがある場合，Controller と Diagnostics Server 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 Server にアクセスするには，以下を参照してください。

- MI Listener のインストールと設定
- Firewall と連動するための Diagnostics Server（Mediator モード）の設定
- Diagnostics Firewall と連動するための 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 の設定方法については，『**HP LoadRunner Controller ユーザーズ・ガイド**』（英語版）を参照してください。MI Listener for Performance Center の設定方法については，『**HP Performance Center Administrator's Guide**』を参照してください。

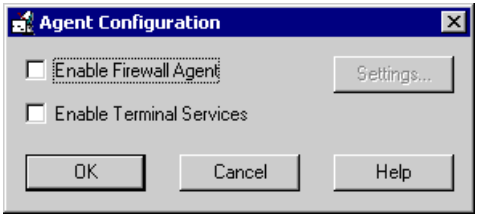
Firewall と連動するための Diagnostics Server（Mediator モード）の設定

Diagnostics Server（Mediator モード）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，以下の追加手順を行う必要があります。Diagnostics Server（Mediator モード）をインストールおよび設定していない場合，以下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。Diagnostics Server（Mediator モード）のインストールについては，第 2 章「Diagnostics Server 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 Diagnostics Server（Mediator モード）を設定するには

1 **＜ Diagnostics Server 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。

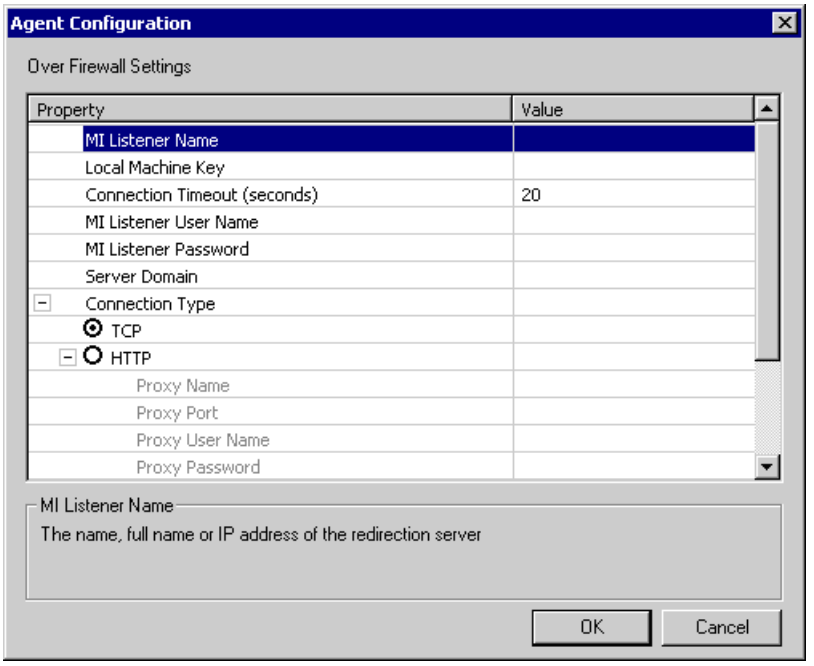
［Agent Configuration］ダイアログ・ボックスが開きます。



2 ［**Enable Firewall Agent**］を選択します。［**Settings**］ボタンが有効になります。

3 ［**設定**］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 の Name 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。



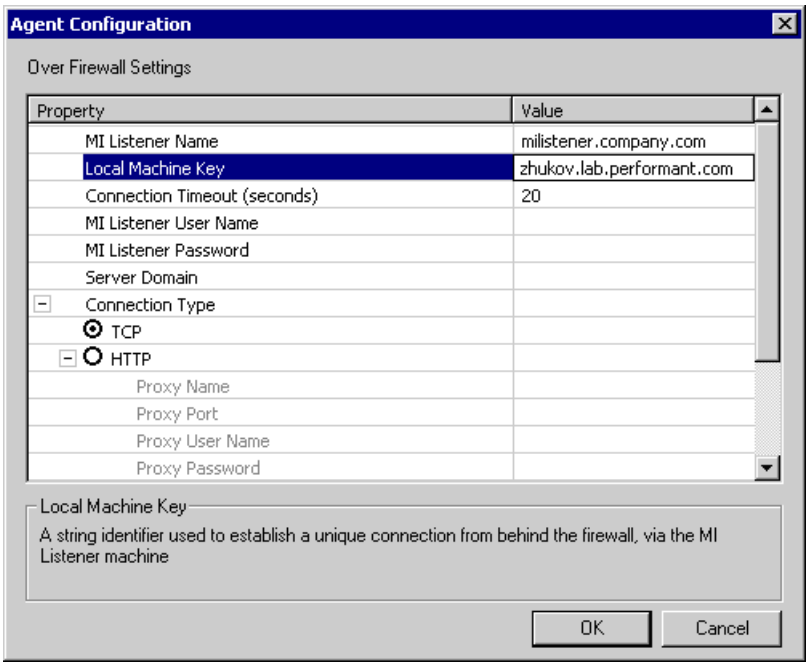
5［**Local Machine Key**］プロパティで，Diagnostics Server（Mediator モード）のホストのマシン名を入力します。

重要：

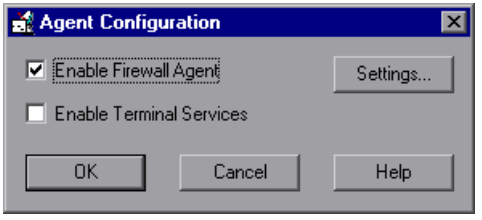
- Diagnostics コンポーネントのホスト名を入力する際，完全修飾ホスト名，つまりマシン名とドメイン名を必ず使用してください。
- システムの状況モニタを使って，Diagnostics Server（Mediator モード）のホストのマシン名を特定します。システムの状況モニタについては，付録 D「System Health Monitor の使用」を参照してください。

---

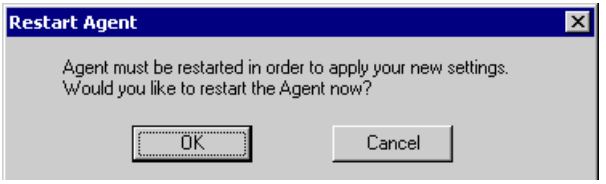
6 ［OK］をクリックして，ダイアログ・ボックスを閉じます。



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 Diagnostics Server（Mediator モード）を設定するには

1 **＜ Diagnostics Server 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。

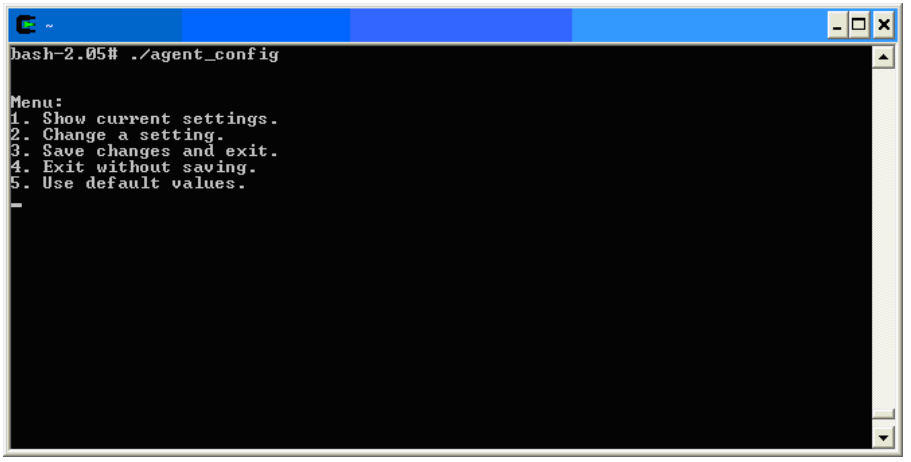
Solaris および Linux の場合（＜プラットフォーム＞ は solaris または linux になります）：

```
export LD_LIBRARY_PATH=.export M_LROOT= ＜ Diagnostics Server 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：

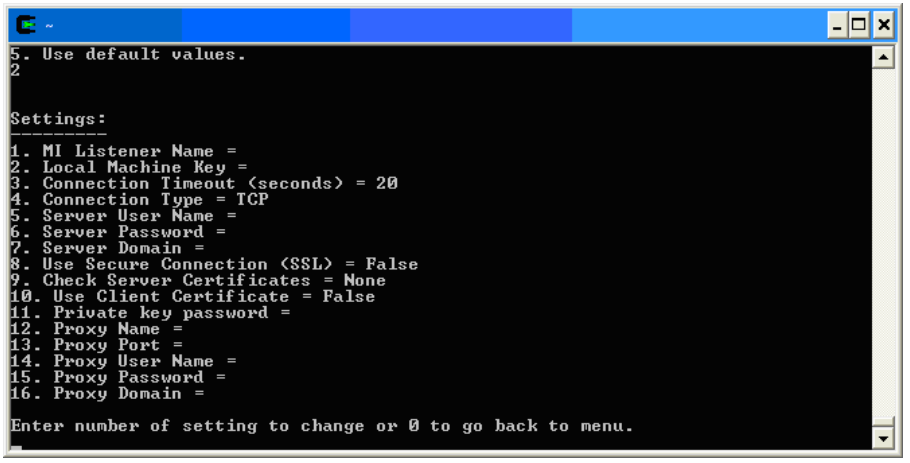
```
export SHLIB_PATH=.export M_LROOT= ＜ Diagnostics Server のインストール・ディレクトリ＞ /nanny/hpuxcd $M_LROOT/bin./agent_config
```

3 ［Agent Configuration Utility］ウィンドウで，**2** を押して［**Change a setting**］を選択します。



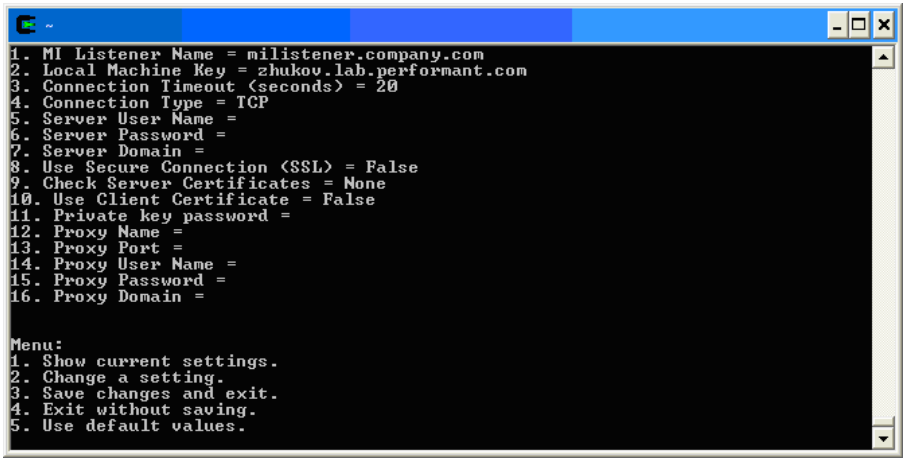
4 設定リストが表示されます。

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，システムの状況モニタの［ホスト:］フィールドに表示されるとおりに，Diagnostics Server（Mediator モード）のホストのマシン名を入力します。

システムの状況モニタを使って，Diagnostics Server（Mediator モード）のホストのマシン名を特定します。システムの状況モニタについては，付録 D「System Health Monitor 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 Diagnostics Server（Mediator モード）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である可能性があります。

Diagnostics Firewall と連動するための 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 Administrator's Guide**』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，『**HP 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 VII 部

Diagnostics 測定値コレクタの設定

第 20 章

システムおよび JMX 測定値コレクタの概要

システムおよび JMX 測定値のキャプチャと，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

システムおよび JMX 測定値コレクタは，プローブと一緒にインストールされます。コレクタでは，Probe ホストからシステム測定値を，アプリケーション・サーバから JMX 測定値を収集します。システムおよび JMX 測定値コレクタは設定可能なため，ユーザは収集する測定値を制御することができます。

測定値コレクタのエントリの設定

プローブ・インストールのシステムおよび JMX 測定値コレクタは，測定値設定ファイルで定義されます。測定値設定ファイル **＜ Probe 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：測定値設定ファイルを更新すると，測定値コレクタは自動的に再起動し，変更が有効になります。

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，測定値コレクタに特定の測定値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，Probe がホストまたは JVM から測定値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，収集された測定値を 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

```
＜コレクタ名＞ / ＜測定値の設定＞ =RATE ＜ rate_multiplier ＞ ( ＜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。

- **＜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，526 ページ「カスタム・システム測定値の収集」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，536 ページ「JMX 測定値のキャプチャ」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** ファイルで一意でなければなりません。metric_id の値がデフォルト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 の値がデフォルトの測定値と異なる場合，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
  - 負荷

- **＜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，一連の測定値をグループ化します。このパラメータは，Diagnostics ビューの［詳細テーブル］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ら，円マーク「¥」，スペース，またはコロン「:」の前に，拡張文字「¥」を追加する必要があります。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。

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。

測定値のキャプチャの開始

追加測定値の情報を収集するには，520 ページ「測定値コレクタのエントリについて」の説明に従って，テンプレートを使用し，**metrics.config** ファイルに適切な測定値コレクタに測定値のエントリを追加します。

キャプチャ済み測定値の変更

metrics.config ファイルで，測定値コレクタのデフォルト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することができます。

第 21 章

システム測定値のキャプチャの設定

システム測定値のキャプチャ・プロセスと，システム測定値コレクタを設定してシステム測定値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，Probe と一緒にインストールされます。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・レベルの測定値を Probe 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なため，ユーザは収集するシステム測定値を制御することができます。

ホストで起動している Probe 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。Probe 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 Probe 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 Probe 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 Probe を停止すると，Probe 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。

デフォルトでキャプチャされるシステム測定値

以下は，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，測定値コレクタがデフォルト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デフォルト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することができます。詳細については，525 ページ「システム測定値コレクタの設定」を参照してください。Windows，Solaris および 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・システム測定値を作成できます。詳細については，526 ページ「カスタム・システム測定値の収集」を参照してください。

z/OS システム測定値については，532 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。

システム測定値コレクタの設定

測定値設定ファイル ＜ **Probe のインストール・ディレクトリ**＞ **/etc/metrics.config** のプロパティを変更して，お使いの環境で実行し，関心のあるシステム測定値を収集および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。

注：測定値設定ファイルを更新すると，システム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。

デフォルトのポートの変更

測定値コレクタのデフォルトのポートは **35000** です。Probe 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，この値を変更することができます。

デフォルトのポートを変更するには

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。デフォルトのポートは **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，Probe の Web サーバのポートとは関係ありません。

測定値コレクタのエントリの設定については，519 ページ「測定値コレクタのエントリの設定」を参照してください。

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。

カスタム・システム測定値の収集

Windows，Solaris および Linux プラットフォームでは，カスタム・システム測定値を作成できます。カスタム測定値は，システム測定値コレクタで監視することができます。

以下のセクションでは，測定値を作成したり，システム測定値のエントリを更新して，カスタム測定値を監視するための手順について説明します。

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・モニタで追跡および報告され，Diagnostics 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

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。

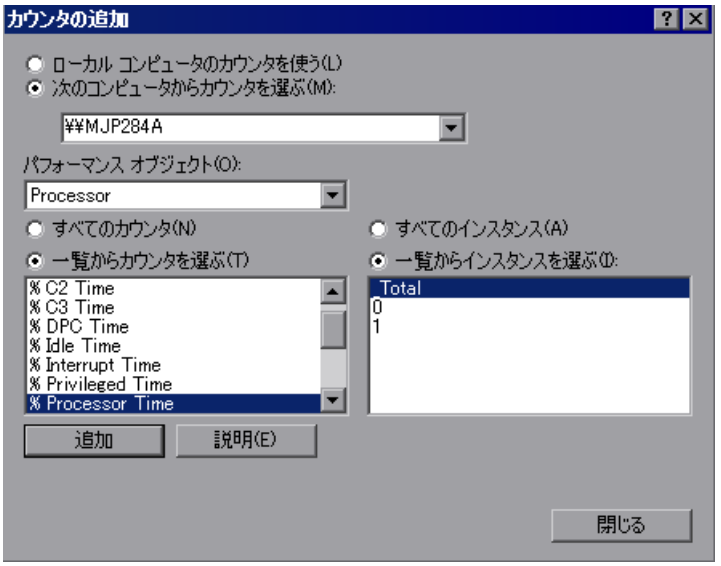
   b ［**ファイル名を指定して実行**］ダイアログ・ボックスの［**名前**］ボックスに，perfmon と入力します。

   ［**パフォーマンス**］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することができます。以下は，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，520 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

1 **＜ Probe 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 <metric_config> 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 最初のエントリで，円マーク「¥」，スペースまたはコロン「:」の前に円マーク「¥」を入れて，最初のエントリをフォーマットします。

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することができます。
system/¥¥¥¥RemoteMachine¥¥Processor(_TOTAL)¥¥%¥ Processor¥ Time=Processor¥ Time(Remote Machine)|percent

---

Solaris ホストでカスタム・システム測定値をキャプチャする

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

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，520 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 円マーク「¥」，スペースまたはコロン「:」の前に円マーク「¥」を入れて，最初のエントリをフォーマットします。

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，測定値情報がある場所に基づいて **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

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なければなりません。

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 kBMemFree:1552 kBBuffers:41616 kBCached:152084
kBSwapCached:46064 kBActive:402720 kBInactive:75328 kBHighTotal:0
kBHighFree:0 kBLowTotal:515548 kBLowFree:1552 kBSwapTotal:1048568
kBSwapFree:779192 kBDirty:4544 kBWriteback:0 kBMapped:300056 kBSlab:28764
kBCommitted_AS:801364 kBPageTables:3184 kBVmallocTotal:499704
kBVmallocUsed:2184 kBVmallocChunk:497324
kBHugePages_Total:0HugePages_Free:0Hugepagesize:4,096 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>= <metric_id>|<metric_units>
```

このテンプレートは，520 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。**<metric_config>** 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 円マーク「¥」，スペースまたはコロン「:」の前に円マーク「¥」を入れて，最初のエントリをフォーマットします。

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

デフォルトで，システム測定値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 システム測定値のキャプチャを有効にするには，次の設定手順を実行する必要があります。

z/OS システム測定値のキャプチャを有効にするには

1 **＜ Probe 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。これは，次の例のようなコマンドを使って行います。

```
chmod -R 770...
```

2 **＜ Probe 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。これは，次の例のようなコマンドを使って行います。

```
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。

第 22 章

JMX 測定値のキャプチャの設定

JMX 測定値のキャプチャ・プロセスと，測定値コレクタを設定して JMX 測定値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 Probe には，あらかじめ定義された JMX 測定値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

また，Java Probe 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 Probe 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。収集された測定値は，HP Diagnostics および Java Diagnostics Profiler 両方のユーザ・インタフェースに表示されます。

JMX 測定値の収集のための WebSphere の設定

JMX 測定値の受信を開始するために，場合によっては，WebSphere サーバにパフォーマンスの監視サービスを設定する必要があります。

このセクションでは，WebSphere 5.x および 6.x を JMX 測定値コレクションに設定する例を紹介します。

WebSphere 5.x サーバに JMX 測定値の収集を設定するには

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，［**サーバ**］＞［**アプリケーション サーバ**］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 ［アプリケーション サーバ リスト］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［**ランタイム**］タブと［**構成**］タブが表示されます。

4 ［**構成**］タブをクリックします。

5 ［**構成**］で次の操作を行います。

- ［**パフォーマンス・モニター・サービス**］をクリックします。
- ［**一般プロパティー**］の下の［**始動**］チェック・ボックスを選択します。
- ［**初期仕様レベル**］を［**標準**］に設定します。
- ［**保管**］をクリックします。

6 ［**適用**］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 5.x インストール・ディレクトリ＞ /properties/server.policy**）を開き，次のセキュリティ権限をファイルに追加して，JMX の収集を許可します。**grant codeBase "file:/ ＜ Probe のインストール・ディレクトリ＞ /lib/probe-jmx.jar" { permission java.security.AllPermission; }**

8 アプリケーション・サーバを再起動します。

ここでは，WebSpere 6.x に JMX 測定値の収集を設定する方法について説明します。

WebSphere 6.x サーバに JMX 測定値の収集を設定するには

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，［**サーバ**］＞［**アプリケーション サーバ**］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［アプリケーション サーバ リスト］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［**ランタイム**］タブと［**構成**］タブが表示されます。

4［**構成**］タブをクリックします。

5［**構成**］で次の操作を行います。

- ［**パフォーマンス**］ヘッダで，［**Performance Monitoring Infrastructure (PMI)**］をクリックします。
- ［**一般プロパティー**］ヘッダで，［**Performance Monitoring Infrastructure (PMI) を使用可能にする**］チェック・ボックスを選択します。
- ［**現在モニターされている統計セット**］ヘッダで，［**拡張**］を選択します。

6［**適用**］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 インストール・ディレクトリ＞ /work/tools/ibm-6.0/websphere/appserver/profiles/default/properties/server.policy**）を開き，次のセキュリティ権限をファイルに追加して，JMX の収集を許可します。
grant codeBase "file:/ ＜ Probe のインストール・ディレクトリ＞ /lib/probe-jmx.jar" { permission java.security.AllPermission; }

grant codeBase "file:/ ＜ Probe のインストール・ディレクトリ＞ /lib/probe-jmx-was6.jar" { permission java.security.AllPermission; }; アプリケーション・サーバを再起動します。

JMX 測定値コレクタの設定

JMX 測定値コレクタは設定可能なため，ユーザは収集する JMX 測定値を制御することができます。JMX 測定値コレクタは，**＜ Probe 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。

測定値コレクタの設定については，519 ページ「測定値コレクタのエントリの設定」を参照してください。

カスタム JMX 測定値のキャプチャ

Java Probe には，533 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 測定値コレクタが一緒にインストールされています。519 ページ「測定値コレクタのエントリの設定」の説明に従って，定義済みコレクタから JMX 測定値エントリを変更したり削除することができます。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 測定値がある場合に，新しいコレクタを作成することもできます。

以下のセクションでは，追加 JMX 測定値を監視できるように，JMX 測定値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。

JMX 測定値のキャプチャ

Java Probe 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 測定値がある場合や，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測定値が公開されている場合もあります。以下では，次のテンプレートに基づいて JMX 測定値エントリを作成する方法を紹介します。

```
＜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，520 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 測定値をキャプチャするには

1 **＜ Probe のインストール・ディレクトリ＞ /etc/metrics.config.** を開いて，Java Probe によって監視されている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 次のテンプレートを使って，**＜測定値の設定＞** パラメータを作成します。

```
＜ MBean オブジェクト名パターン＞ . ＜属性名＞
```

JMX 測定値の場合，**＜測定値の設定＞** 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。

- **＜ 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。
- **＜属性名＞** は，測定値を表す属性の名前です。

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 <metric_config> パラメータは次のように設定されます。

```
*:Type=ExecuteQueueRuntime,*.ServicedRequestTotalCount
```

測定値パターンの詳細については，538 ページ「測定値パターンについて」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 円マーク「¥」，スペース，等号「=」またはコロン「:」の前に円マーク「¥」を入れて，最初のエントリをフォーマットします。

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。

測定値パターンについて

JMX 測定値の場合，<metric_config> 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。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
```

第 VIII 部

他の HP ソフトウェア製品との統合のセットアップ

第 23 章

Diagnostics を使用するための Business Availability Center のセットアップ

HP Diagnostics と統合できるように HP Business Availability Center をセットアップする方法について説明します。

本章の内容

Diagnostics を使用するための Business Availability Center のセットアップ

HP Diagnostics と Business Availability Center 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 Availability Center に入力する必要があります。

Business Availability Center に Diagnostics をセットアップするには

1 **Diagnostics Server の情報を指定します。**

Business Availability Center に Diagnostics Server の情報を入力します。詳細については，543 ページ「Diagnostics Server の情報の指定」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，547 ページ「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 Availability Center の［Diagnostics configuration and application］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，548 ページ「Windows 2003 を使用した Diagnostics のページへのアクセス」を参照してください。

4 **Diagnostics Server ホストでクッキーを有効にします。**

クッキーは，Business Availability Center に Diagnostics データを表示するために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 Server 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。

Diagnostics Server の情報の指定

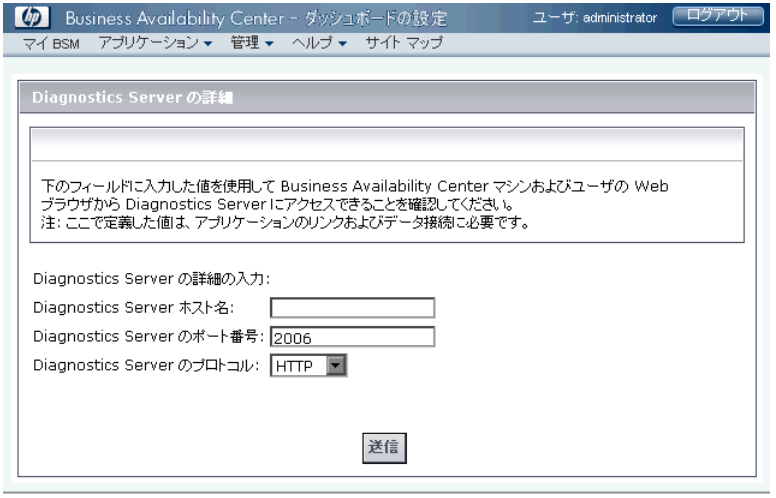
Business Availability Center から HP Diagnostics にアクセスできるようにするために，Diagnostics Server の情報を指定します。

注：Windows 2003 を使用している場合，［診断の設定］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，548 ページ「Windows 2003 を使用した Diagnostics のページへのアクセス」を参照してください。

Business Availability Center に Diagnostics Server の情報を指定するには

1 Business Availability Center にログオンします。

2 ［**管理**］＞［**Diagnostics**］を選択して，［診断の設定］ページを開きます。



注：Diagnostics Server 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 Server 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，［診断の設定］ページを開きます。

---

3 Diagnostics Server（Commander モード）の情報を入力します。

- **Diagnostics Server のホスト名**：Diagnostics Server（Commander モード）のホスト・マシンの名前を入力します。

  Diagnostics Server が Business Availability Center と同じシステムにインストールされている場合も，［**診断サーバ 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 Availability Center 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 Server ホストを登録する必要もあります。

- **Diagnostics サーバのポート番号**：Diagnostics Server（Commander モード）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 Availability Center 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，付録 C「Diagnostics コンポーネント間での HTTPS 有効化」を参照してください。

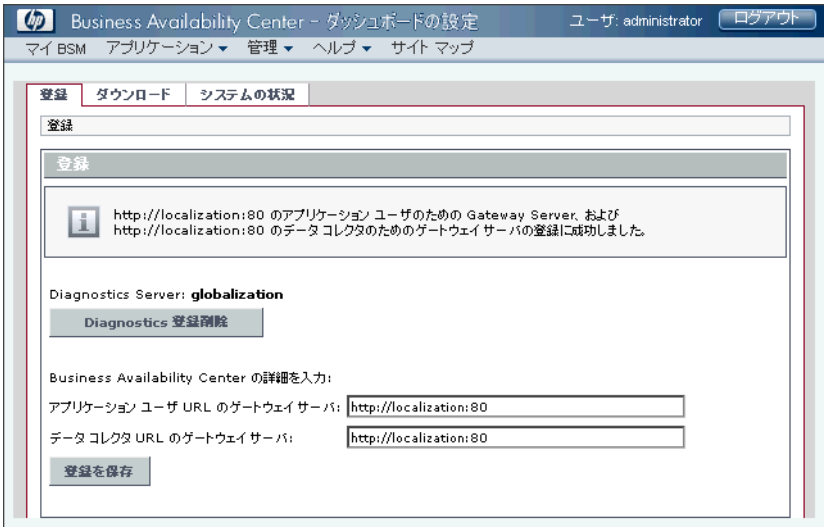
---

4 Diagnostics Server 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 Server 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 Server の情報が Business Availability Center に保存され，Business Availability Center のサーバ情報が Diagnostics Server マシンに自動的に登録されます。

［診断の設定］ページの［**登録**］タブが開き，入力した Diagnostics Server の情報と Business Availability Center のサーバ情報が表示されます。



必要に応じて，［登録］タブの［**Business Availability Center の詳細を入力**］セクションで Business Availability Center のサーバ情報を手動で変更できます。

注：［**センタ サーバ URL**］ボックスのルート URL が，Business Availability Center 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。

Diagnostics Server の情報の変更

場合によっては，Diagnostics Server の情報を変更したり，Diagnostics の登録をすべて削除する必要があります。

Diagnostics の登録を削除するには

1 ［**管理**］＞［**Diagnostics**］を選択します。

2 ［**登録**］タブで，［**Diagnostics 登録削除**］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 Server を登録するには，［**管理**］＞［**Diagnostics**］を選択して，543 ページ「Diagnostics Server の情報の指定」の手順に従います。

［診断の設定］ページについて

［**管理**］＞［**Diagnostics**］を選択して，［診断の設定］ページにアクセスします。［診断の設定］ページは，次の 3 つのタブで構成されます。各タブの詳細については，このセクションで説明します。

- ［登録］タブ
- ［ダウンロード］タブ
- ［システムの状況］タブ

［登録］タブ

［登録］タブには，次の情報が表示されます。

- Business Availability Center に登録した Diagnostics Server の情報。情報を変更する場合は，546 ページ「Diagnostics Server の情報の変更」を参照してください。
- Business Availability Center マシンに自動的に登録された Diagnostics Server のサーバ情報。［**BAC の詳細を入力**］セクションで，Business Availability Center のサーバ情報を手動で変更できます。

Business Availability Center に Diagnostics を初めて登録する際は，543 ページ「Diagnostics Server の情報の指定」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 Probe 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 Probe または Collector をダウンロードできます。

Diagnostics Server のインストール中に Probe 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 2 章「Diagnostics Server のインストール」を参照してください。

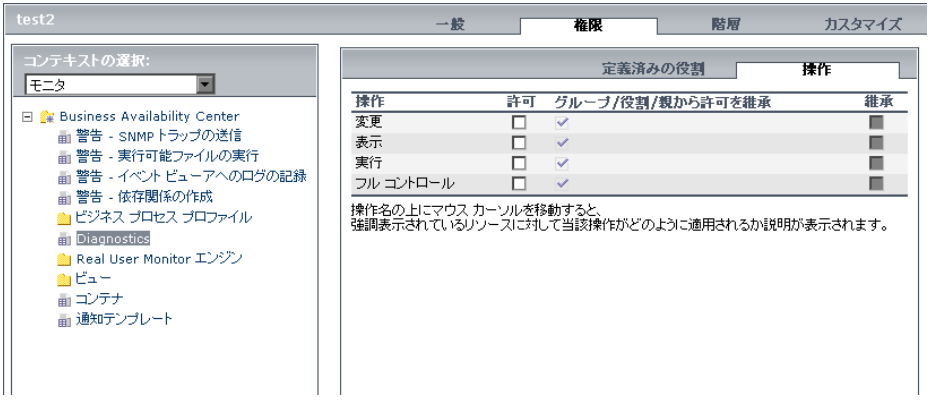
［システムの状況］タブ

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。システムの状況モニタの詳細については，付録 D「System Health Monitor の使用」を参照してください。

Diagnostics ユーザへの権限の割り当て

Business Availability Center 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することができます。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。

次の Business Availability Center の［プラットフォームの管理］ページの画面例には，Diagnostics の操作権限が表示されています。



Business Availability Center 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 Availability Center 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**：しきい値の設定など，HP Diagnostics UI での設定の変更を有効にします。
- **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することができます。

Business Availability Center でユーザ権限を割り当てる方法の詳細については，『**HP Business Availability Center Documentation Library**』の「**Platform Administration**」を参照してください。

Windows 2003 を使用した 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 Availability Center の［Diagnostics configuration and application］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

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 Server の名前を入力します。

- Business Availability Center で Diagnostics Server 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 Availability Center 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://，およびポート番号を含めます。

  http:// ＜ Diagnostics Server のホスト名＞ :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 Availability Center に送信されるデータ・サンプルと Web サービス CI

Diagnostics と Business Availability Center 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 Availability Center に送信します。Diagnostics は次のデータ・サンプルを Business Availability Center 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_ru_t（Probe サンプル）
- appmon_vu_t（Transaction（BPM）サンプル）

データ・サンプルの詳細については『**HP Business Availability Center Documentation Library**』を参照してください。

Diagnostics 7.50 以降では，Diagnostics は SOA サンプル（ws_perf_aggr_t）の CI を作成し，それを CMDB に直接追加します。Diagnostics からのそれ以外のタイプのデータ・サンプルの場合，Business Availability Center は対応する CI を作成します。

まれなケースですが，これらの Web サービス CI を uCMDB 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。

またこれもまれなケースですが，これらの Web サービス CI に対して Diagnostics と Business Availability Center を強制的に同期化させる場合は，**同期化**関数を Diagnostics Server で使用できます。565 ページ「Diagnostics Server 管理ページへのアクセス」を参照してください。

注：Business Availability Center 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 に現在表示されている Web サービスを Business Availability Center に転送する前に，厳密な同期を手動で行う（または 24 時間待機する）必要があります。厳密な同期を行うには，Diagnostics Server の管理ページに移動して［**synchronize**］を選択し，［**全カスタマ**］に対して［**ハード**］を選択します。

第 24 章

LoadRunner Diagnostics Add-in のインストール

LoadRunner Diagnostics Add-in を使うと，LoadRunner 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測定値を収集したり，Diagnostics UI に接続したり，システムの状況モニタを使用したりできるようになります。

本章の内容

LoadRunner Diagnostics Add-in をインストールする前に

LoadRunner Diagnostics Add-in をインストールする前に，Diagnostics Server（Commander モード）と LoadRunner をインストールする必要があります。LoadRunner のインストール方法については，『**HP LoadRunner インストール・ガイド**』を参照してください。

Diagnostics 8.00 に付属するアドインを使って，Diagnostics 8.00 を LoadRunner 9.10 以降と統合できます。

注：Diagnostics 8.00 サーバでは，以前のバージョンの LoadRunner アドインは機能しません。

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：Diagnostics 8.00 リリースでは，LoadRunner Diagnostics Add-in が変更され，起動時に小さいブートストラップ・プログラムを使って Diagnostics Server 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。

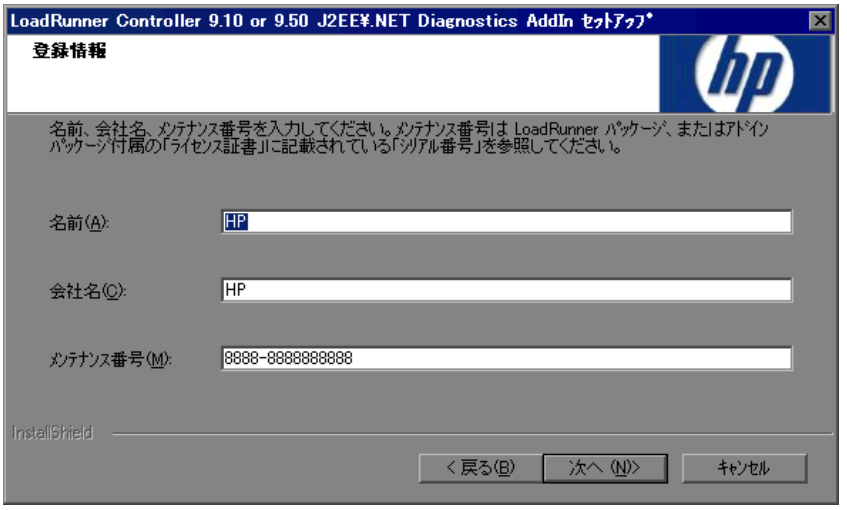
LoadRunner Diagnostics Add-in をインストールするには

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，［**はい**］をクリックして同意します。

4 ［登録情報］ダイアログ・ボックスが開きます。



［登録情報］ダイアログ・ボックスに，名前，会社名およびお手元の 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。

［**次へ**］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。

5 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，［**完了**］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 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 25 章「HP Diagnostics を使用するための LoadRunner のセットアップ」を参照してください。

第 25 章

HP Diagnostics を使用するための LoadRunner のセットアップ

LoadRunner を設定して，負荷テストで HP Diagnostics を使用できるようにする方法について，一般的な情報を提供します。

本章の内容

Diagnostics を使用するための LoadRunner のセットアップ

LoadRunner から Diagnostics UI にアクセスするには，あらかじめ LoadRunner が 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ために，Diagnostics Server の情報を指定します。これらの情報は，Diagnostics で LoadRunner を初めて使用するときだけ入力する必要があります。Diagnostics Server の情報を指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 Server の情報を指定する前に，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。

LoadRunner に Diagnostics Server の情報を指定するには

LoadRunner シナリオを設定した HP Diagnostics の使用

負荷テスト・シナリオで Diagnostics 測定値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 Probe 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，シナリオを実行するたびに Diagnostics パラメータを設定し直す必要はありません。

大きなオフライン分析ファイルの転送を改善する

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 Server 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 Server 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。

第 26 章

HP Diagnostics を使用するための Performance Center のセットアップ

Performance Center を設定して，負荷テストで HP Diagnostics を使用できるようにする方法について，一般的な情報を提供します。

本章の内容

HP Diagnostics を使用するための Performance Center のセットアップ

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を助ける情報を提供するように設計された統合製品です。

Diagnostics に Performance Center からアクセスできるようにするには，Performance Center で次の Diagnostics Server の詳細情報を指定します。

- **サーバ名**：Diagnostics Server（Commander モード）のホスト・マシンの名前。
- **ポート番号**：Diagnostics Server（Commander モード）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，576 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 Server 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「Diagnostics 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 Administration Site］の［Diagnostics］ページで入力します。

HP 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 測定値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 Probe 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 Server の間にファイアウォールがある場合，Controller と Diagnostics Server 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 Server（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，505 ページ「ファイアウォール環境で動作する 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。

- Probe 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，これらにファイルを削除できるようにするため，Diagnostics Servers を Mediator モードで設定する必要があります。

これを行うには，**＜ Diagnostics Server 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 property** の期間保存されます（デフォルトは 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

付録 A

Diagnostics Server の管理ページ

Diagnostics プロパティの設定を行うために，Diagnostics Server の管理ページにアクセスする方法について説明します。また，設定ページの表示方法およびページ内のプロパティを変更する方法についても説明します。

本章の内容

Diagnostics Server 管理ページへのアクセス

ユーザ権限を設定したり，Diagnostics の設定を行ったり，Diagnostics Server の管理ページから直接 Diagnostics を開くことができます。

Diagnostics Server の管理ページにアクセスするには

1 ブラウザで http://＜diagnostics_server_host＞:2006 にアクセスするか，または［**スタート**］＞［**プログラム**］＞［**HP Diagnostics Server**］＞［**Administration**］を選択して管理ページを開きます。URL のポート番号 **2006** は，Diagnostics Server のデフォルトのポートです。ほかのポートを使うように Diagnostics Server 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。

ブラウザに，Diagnostics Server 管理ページが開きます。

2 実行する活動項目に該当するオプションをクリックします。

- ［**Diagnostics を開く**］: Diagnostics Web ページを開いて，Diagnostics Server に報告を行っている Probe で収集されたパフォーマンス測定値を表示できます。パフォーマンス測定値は，標準の Diagnostics ビューに表示されます。

  Diagnostics Server（Commander モード）から Diagnostics を直接開く方法については，『**HP Diagnostics User's Guide**』（英語版）を参照してください。

- ［**Diagnostics の設定**］: Diagnostics Server 設定ページへのリンクがある［コンポーネント］ページを開きます。

  Diagnostics プロパティの設定に関する詳細は，567 ページ「Diagnostics Server 設定ページを使用した Diagnostics の設定」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，573 ページ「ユーザの認証と承認」を参照してください。

Diagnostics Server 設定ページを使用した Diagnostics の設定

Diagnostics Server の設定ページでは，Diagnostics Server とほかの Diagnostics コンポーネントの通信を制御したり，Probe から受信したデータを Diagnostics Server 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ます。

注：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 Server プロパティを変更することをお勧めします。

Diagnostics Server の設定ページへのアクセス

Diagnostics Server の設定ページには，［コンポーネント］ページからアクセスします。

Diagnostics Server の設定ページにアクセスするには

1 ブラウザで http:// ＜ Diagnostics Server のホスト＞ :2006 にアクセスするか，または［**スタート**］＞［**プログラム**］＞［**HP Diagnostics Server**］＞［**Administration**］を選択して管理ページを開きます。URL のポート番号 **2006** は，Diagnostics Server のデフォルトのポートです。ほかのポートを使うように Diagnostics Server 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。

ブラウザに，Diagnostics Server 管理ページが開きます。

2 ［**Diagnostics の設定**］をクリックします。

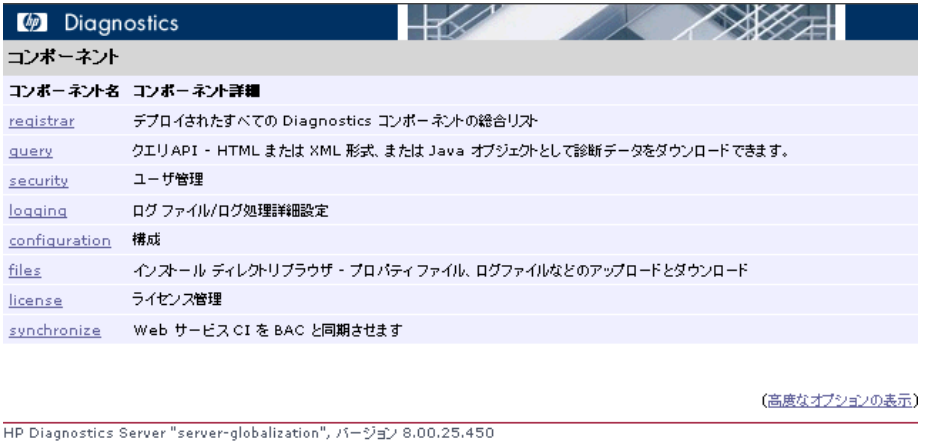
3 まだ Diagnostics Server にログオンしていない場合，ユーザ名とパスワードの入力を求められます。これは，有効なユーザ名でなければならず，「**表示**」権限と「**変更**」権限の両方が必要です。有効なユーザ名と権限については，付録 B「ユーザの認証と承認」を参照してください。

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。

ログオンすると，Diagnostics Server の［コンポーネント］ページが開きます。



リンクをクリックして［コンポーネント］ページを開きます。

- ［**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。

- ［**security**］：ビルトイン・ユーザ管理。
- ［**logging**］：ログ・ファイルとログ詳細の設定。
- ［**configuration**］：Diagnostics サーバの設定。
- ［**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。
- ［**license**］：ライセンス管理。
- ［**synchronize**］：Web サービス CI と BAC を同期化します。ハード同期化（Business Availability Center 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 Availability Center と同期化する）を強制的に実行できます。

［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

- 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。

追加の詳細設定オプションが表示されます。

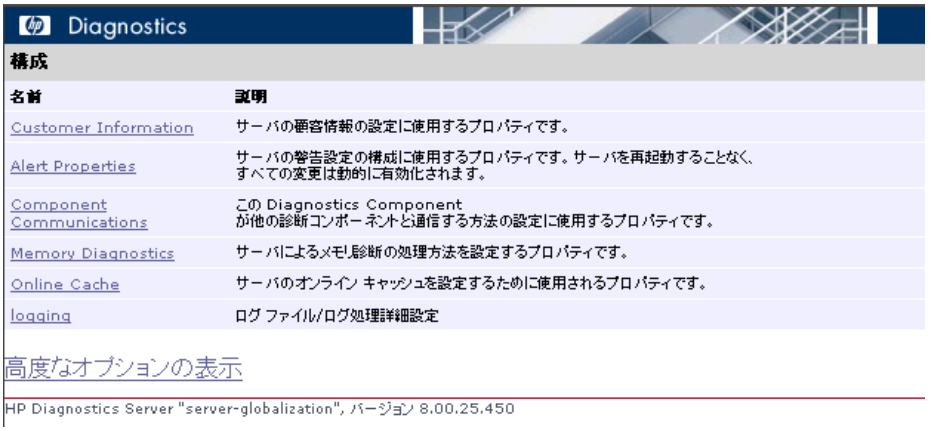
詳細設定オプションを隠すには

- 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。

設定コンポーネントへのアクセス

Diagnostics Server の管理ページで［Diagnostics の設定］を選択し，［コンポーネント］ページで［**構成**］を選択して，Diagnostics Server の［構成］ページにアクセスします。



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。以下を設定可能です。

- Customer Information
- Alert Properties
- Component Communications
- Memory Diagnostics
- Online Cache
- logging

［構成］ページに表示される設定オプションは，一般的に設定されるオプションです。デフォルトで，詳細設定オプションは隠れています。

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。

詳細設定オプションを表示するには

➤ 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。

追加の詳細設定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。

Diagnostics Server プロパティの変更

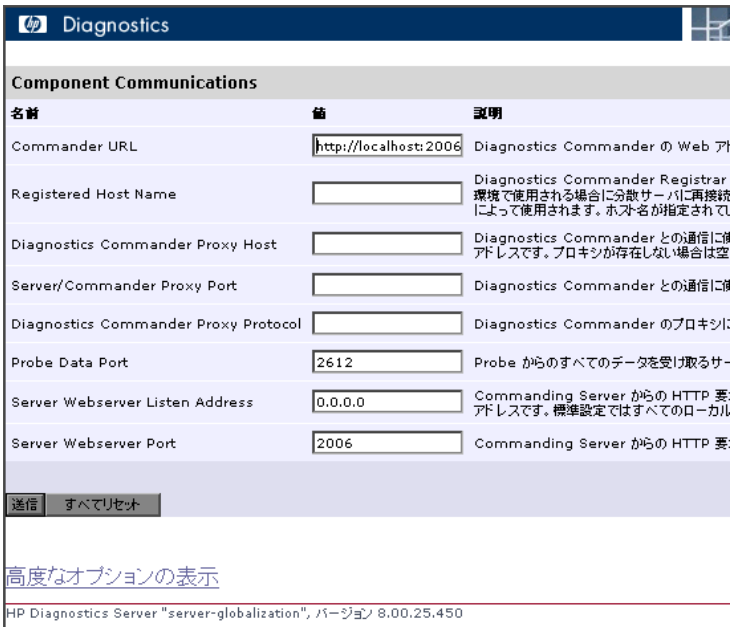
［構成］ページから Diagnostics Server のプロパティを変更します。

Diagnostics Server プロパティを変更するには

1 567 ページ「Diagnostics Server の設定ページへのアクセス」の説明に従って，Diagnostics Server の［構成］ページにアクセスします。

2 更新するプロパティのリンク（［**Component Communications**］など）をクリックします。

3 表示されたプロパティを見直し，更新します。



4 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 Server の再起動を求めるメッセージが表示されます。Diagnostics Server 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 Server 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 Server 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。
- 記録レベルの詳細の変更では，Diagnostics Server 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。

付録 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

すべての Diagnostics コンポーネントのユーザの認証および承認設定は，Diagnostics Server（Commander モード）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。

ユーザ名を作成および編集し，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 Server に接続している Probe の Profiler（.NET Diagnostics Profiler または Java Diagnostics Profiler）のユーザ権限および特権も，Diagnostics Server（Commander モード）で定義されます。ユーザに，特定の Probe グループの Profiler にアクセスするための権限を 1 セット，Diagnostics Server 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。

ユーザを管理し，Diagnostics Server のオープニング・ダイアログで［権限と認証の管理］を選択して［権限］ページにアクセスしてユーザ権限を割り当てます。

重要：

- Probe を Profiler 専用でインストールしている（Diagnostics Server に接続していない）場合，Probe 自体で Profiler のユーザの権限および承認を管理します。
- Profiler 専用でインストールしている Java Probe の認証および承認の管理については，404 ページ「スタンドアロン・モードの Java Diagnostics Profiler での認証と承認」を参照してください。
- Profiler 専用でインストールしている .NET Probe の認証および承認の管理については，491 ページ「スタンドアロン・モードの .NET Profiler での認証と承認」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 Server（Commander モード）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 Server 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，必ずブラウザを閉じて，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・データを消去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，ユーザ情報を保守することができます。Profiler では，これは，ヒープダンプの取得やインストゥ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** および **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 Server に登録するのに Probe が必要とする権限など）。また，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することができます。

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，587 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 Server および特定の Probe 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。

以下の例を検討します。

所有している Probe 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 Probe システムで表示権限を必要とする 2 つの開発チーム（Dev1 および 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 Probe グループ	[Dev1All] = 表示，実行，変更 [Dev2View] = 表示
Dev2 Probe 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。

デフォルトのユーザ名を使用した Diagnostics へのアクセス

Diagnostics には，次のデフォルトのユーザ名が定義されています。

デフォルトのユーザ名	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，および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，および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。

これらのデフォルトのユーザ名を使って，Diagnostics の機能にアクセスできます。

デフォルト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。

デフォルト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 つのデフォルトのユーザ **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 Server 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］ページで，Diagnostics ユーザを作成，編集および削除したり，特定のユーザ名の権限を管理することができます。

［権限］ページへは，Diagnostics Server の管理ページ，または［メンテナンス］ページからアクセスします。

Diagnostics Server の管理ページから［権限］ページにアクセスするには

1 ブラウザで, http://＜Commanding Diagnostics Server のホスト＞:2006/ にアクセスするか，または［**スタート**］＞［**プログラム**］＞［**HP Diagnostics Server**］＞［**Administration**］を選択して，Diagnostics Server の管理ページを開きます。

URL のポート番号 **2006** は，Diagnostics Server のデフォルトのポートです。ほかのポートを使うように Diagnostics Server 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。

2 Diagnostics Server の管理ページで，［**権限と認証の管理**］をクリックすると［権限］ページが開きます。このダイアログで，Diagnostics ユーザを作成，編集および削除したり，特定のユーザ名の権限を管理することができます。

Diagnostics UI から［権限］ページにアクセスするには

1 ［Diagnostics Views］ウィンドウの右上にある［**メンテナンス**］をクリックします。［メンテナンス］ページが開きます。

2 ［**security**］をクリックすると［権限］ページが開きます。

［権限］ページが開いたときに，Diagnostics Server 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，少なくても「**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 Server［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 Server および Probe 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。

デフォルトで，特定の Diagnostics Server 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 Probe へのアクセスも同様に認証（および権限）が与えられます。

➤ ［**次に接続されている Probe を制御 : ＜ Commanding Diagnostics Server ＞ :**］：このセクションでは，Probe 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 Probe グループの Profiler にアクセスするための権限を 1 セット，Diagnostics Server 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 Probe 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 User's Guide**』（英語版）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）。

➤ Probe 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ゥ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，修正，および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		
	ビュー	実行	変更	ビュー	編集	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			
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			

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		
	ビュー	実行	変更	ビュー	編集	画面の編集
［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					
カスタマ・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（メモリおよび割り当て分析）			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	
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	

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		
	ビュー	実行	変更	ビュー	編集	画面の編集
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，削除および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，編集および削除

「**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，ユーザを削除することができます。「**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

1 579 ページ「[権限］ページへのアクセス」の説明に従って，Diagnostics Server 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 任意で，このユーザに役割を割り当てます。役割は括弧で囲んでください（[aRole] など）。役割はカンマで区切ることができます（[Role1],[Role2] など）。

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（Diagnostics デプロイメントでの権限の割り当て，および Probe グループへの権限の割り当て）。

---

8 ［**変更を保存**］をクリックします。

デフォルトで，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，587 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。

ユーザを削除するには

1 579 ページ「［権限］ページへのアクセス」の説明に従って，Diagnostics Server 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」および「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

1 579 ページ「［権限］ページへのアクセス」の説明に従って，Diagnostics Server 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

1 579 ページ「[権限］ページへのアクセス」の説明に従って，Diagnostics Server 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，576 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

1 579 ページ「[権限］ページへのアクセス」の説明に従って，Diagnostics Server 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，ユーザ権限を変更することができます。

3 変更する権限を持つユーザの名前を見つけます。

重要：585 ページ「ユーザの作成，編集および削除」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」および「実行」権限を割り当てる場合，**newuser** を見つけて次のように行を編集します。

```
newuser = view,execute
```

［Editing Enterprise Permissions］ページには，一連のデフォルト・ユーザも含まれます。これらのユーザについては，578 ページ「デフォルト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。これらのデフォルト・ユーザの権限を変更することができます。

Probe 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 Probe グループに属す Probe 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。

デフォルトで，特定の Diagnostics Server 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 Probe 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 Server に対して持っているものとは異なる Probe 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，576 ページ「ユーザ権限について」を参照してください。

Probe 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 Probe 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することができます。

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 Probe グループで，特定の権限を持つユーザのデフォルトのユーザ・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。デフォルトで，次のユーザ・グループは，すべての Probe 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 Server の administration（変更）権限を持つユーザを参照します。デフォルトで，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 Probe の管理権限を持ちます。
any_diagnostics_superuser	このグループは，Diagnostics Server の superuser（実行）権限を持つユーザを参照します。デフォルトで，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 Probe の実行権限を持ちます。
any_diagnostics_user	このグループは，Diagnostics Server の user（表示）権限を持つユーザを参照します。デフォルトで，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 Probe の表示権限を持ちます。

特定の Probe 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

1 579 ページ「[権限］ページへのアクセス」の説明に従って，Diagnostics Server 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 Probe を制御 : <Commanding Diagnostics Server>:**］セクションで，［**< Probe グループの名前>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，ユーザ権限を変更することができます。

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 Probe 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

1 579 ページ「[権限］ページへのアクセス」の説明に従って，Diagnostics Server 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 Probe を制御 : <commanding_Diagnostics_server>:**］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，ユーザ権限を変更することができます。

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 Probe 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 Server に接続されている将来のすべての Probe グループは，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 Availability Center，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。

- 591 ページ「Business Availability Center ユーザの認証と承認」
- 592 ページ「Performance Center および LoadRunner ユーザの認証と承認」

Business Availability Center ユーザの認証と承認

Business Availability Center で，Diagnostics のユーザ権限を定義できます。詳細については，547 ページ「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 Availability Center ユーザが Business Availability Center から Diagnostics を開くと，Business Availability Center 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 を開くと，Business Availability Center ユーザ権限への更新だけが取得されます（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 Availability Center 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 Availability Center へのログインを信用します。

Business Availability Center 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 Availability Center 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，585 ページ「ユーザの作成，編集および削除」を参照してください。

注：Diagnostics Server は，ユーザに加えられた権限の変更を検出するのに最大で 5 分かかります。

Performance Center および LoadRunner ユーザの認証と承認

Diagnostics との統合時に LoadRunner または Performance Center をセットアップする場合，LoadRunner / Performance Center 内に Diagnostics Server の詳細を指定します。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 Server 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 Server 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

1 次のいずれかの方法で Diagnostics Server の管理ページを開きます。

- ➤［**スタート**］＞［**プログラム**］＞［**HP Diagnostics Server**］＞［**Administration**］を選択します。
- ➤ ブラウザで http:// ＜ Diagnostics Server のホスト＞ :2006 にアクセスします。URL のポート番号 **2006** は，Diagnostics Server のデフォルトのポートです。ほかのポートを使うように Diagnostics Server 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 Server の管理ページが開きます。

2［**Diagnostics の設定**］をクリックします。

3 まだ Diagnostics Server 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，有効なユーザ名でなければならず，「**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，576 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 Server の［コンポーネント］ページが開きます。

4 ［**ログ**］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 Server 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。

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 Server（Commander モード）のみで使用できます。

JAAS は，以下の 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，javax.security.auth.login.Configuration のドキュメント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，以下の 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，Commanding Server 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，管理者に問い合わせてください（LDAP など）。
- 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（594 ページ「JAAS を使用するための Diagnostics の設定」参照）を使うように Diagnostics を設定し，Commanding Server で以下を設定する必要があります。

1 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。以下は設定例です。

```
Diagnostics
{
      com.mercury.diagnostics.server.jaas.spi.LDAPLoginModule required
        providerURL="ldap://ldap.yourdomain.com:636"
        baseDN="ou=People,o=yourdomain.com"
        useSSL="true"
        serverCertificate="etc/ldap.keystore";
};
```

以下は，JAAS 設定ファイルで LDAP LoginModule に指定可能な全オプションの一覧です。

オプション名	説明	必須 / 任意	デフォルト値	例
providerURL	LDAP サーバへの URL。JNDI ルックアップの実行に使います。	必須		providerURL="ldap://ldap.yourdomain.com:636"
baseDN	認証するユーザに対応する DN（識別名）LDAP エントリの一部の文字列。uidAttribute オプションは，DN を完成させるために使います。したがって，［例］列が baseDN で，デフォルト値が uidAttribute の場合，LDAP に送信される結果の DN は「uid=testuser,ou=People,o=hp.com」になります。	必須		baseDN="ou=People,o=yourdomain.com"
uidAttribute	LDAP サーバのユーザ・エントリを一意に識別する DN を完了させるために，baseDN と一緒に使われる LDAP 属性名。ログイン時に送信されるユーザ名は，この属性の値になります。	任意	uid	uidAttribute="employeeNumber"

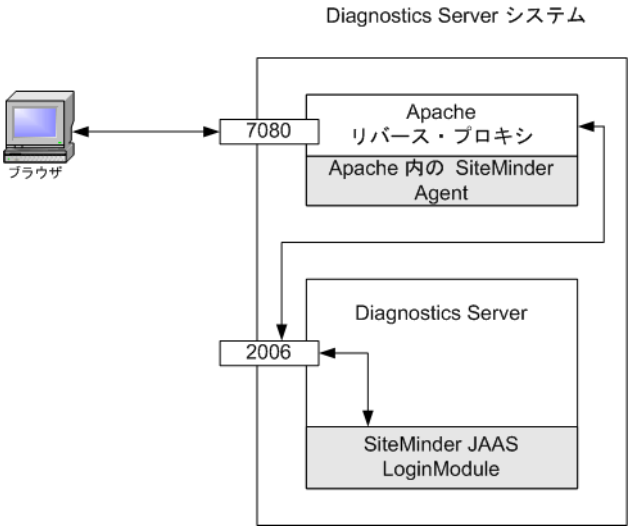
オプション名	説明	必須 / 任意	デフォルト値	例
useSSL	LDAP サーバへの接続に SSL が必要かどうか。	任意	false	useSSL="true"
serverCertificate	useSSL が true に設定されている場合に適用可能。LDAP サーバの証明書が含まれる truststore ファイルへのパス。サーバのインストール・ホームへの絶対パスまたは相対パス。プラットフォームとは無関係にフォワード・スラッシュを使います。	任意	<インストール・ディレクトリ> /etc/ ldap.keystore	serverCertificate="etc/ myldap.keystore"
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="Super Users"
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。	任意	roles	roleAttributes="employeeType,hpJobFunction"
allowAnonymous	LDAP サーバで匿名ログインを許可するかどうか。true に設定し，パスワードが入力されないと，ログイン・モジュールは，ログインを拒否するのではなく LDAP サーバへの接続を試みます。	任意	false	allowAnonymous="true"

注： server.properties や jaas.configuration ファイルに変更を加えた後，Diagnostics Commanding Server を再起動する必要があります。

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 Server 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 Server 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 Server 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，以下のプロパティを追加します。

➤ ProxyPass / http:// ＜ Diagnostics Server の IP アドレス＞ :2006/

（2006 は，Diagnostics Server のデフォルト・ポート。Diagnostics Server に設定したポートを使用します。）

➤ ProxyPassReverse / http:// ＜ Diagnostics Server の IP アドレス＞ :2006/

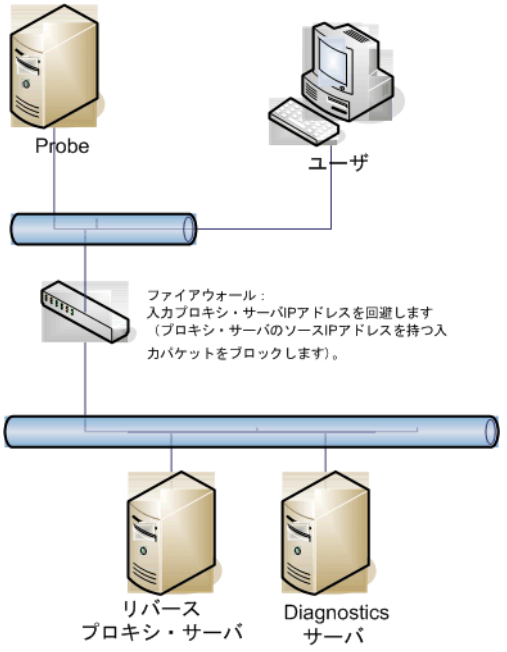
（2006 は，Diagnostics Server のデフォルト・ポート。Diagnostics Server に設定したポートを使用します。）

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，任意で以下の手順を実行してセキュリティを高めることができます。

➤ Diagnostics Server 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 Server 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することができます。

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，Commanding Server で以下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（594 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。デフォルトで，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** プロパティをコメントアウトします（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。

以下は例です。

```
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

以下は，JAAS 設定ファイルで JAAS LoginModule に指定可能な全オプションの一覧です。

オプション名	説明	必須 / 任意	デフォルト値	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="Super Users"

注： server.properties，webserver.properties または jaas.configuration ファイルに変更を加えた後，Diagnostics Commanding Server を再起動する必要があります。

ライトウェイトなシングル・サインオン（SSO）セキュリティの設定

Diagnostics は，ライトウェイトなシングル・サインオン（SSO）セキュリティのコンシューマです。

TransactionVision から Diagnostics にドリルダウンするときに再度ログインするのを防ぐには，次のプロパティを設定します。

＜ Diagnostics Server のインストール・ディレクトリ＞ /etc/security.properties の lwsso.init.string

このプロパティは，＜ **TVISION_HOME** ＞ **/config/ui/lwssofmconf.xml** ファイルの initString 属性の **crypto** 要素に指定された値と同じ値に設定する必要があります。次に例を示します。

```
<crypto cipherType="symmetricBlockCipher" engineName="AES"
paddingModeName="CBC" keySize="256" encodingMode="Base64Url"
initString="This is a shared secret passphrase" />
```

本項の内容

ライトウェイトな SSO について

以下に，ライトウェイトな SSO（LW-SSO）の重要な動作をまとめます。

- LW-SSO の機密の InitString パラメータ

  ライトウェイトな SSO は，対称型暗号化を使ってトークンを検証および作成します。設定内の initString パラメータは，秘密鍵の初期化に使われます。あるアプリケーションがトークンを作成し，同じ initString パラメータを使用する各アプリケーションがそのトークンを検証します。

  - initString パラメータを設定せずに LW-SSO を使うことはできません。
  - initString パラメータは機密情報であるため，発行，輸送，維持に関しては機密として扱う必要があります。
  - initString は LW-SSO を使って相互に統合しているアプリケーション間でのみ共有する必要があります。
- LW-SSO は，特に必要でない限り無効にする必要があります。

- LW-SSO によって統合されたアプリケーションで使われている認証フレームワークのうち最も弱い認証フレームワークが，すべてのアプリケーションの認証セキュリティのレベルを決定します。強くセキュアな認証フレームワークを使うアプリケーションだけが LW-SSO トークンを発行することが推奨されます（設定のセクションを参照）。

- 対称型暗号化の影響

  LW-SSO は，LW-SSO トークンの発行と検証に対称型暗号化を使用します。したがって，LW-SSO を使用するアプリケーションはすべて，同じ initString を共有するほかのすべてのアプリケーションによって信頼されるトークンを発行できます。

  この潜在的リスクは，initString を共有するアプリケーションが信頼できない場所にあるか，または信頼できない場所でアクセスされる場合に関連します。

- ユーザ・マップ（同期化）

  LW-SSO フレームワークは，統合されているアプリケーション間でのユーザ・マップを保証しません。したがって，統合アプリケーションがユーザ・マップを監視する必要があります。すべての統合されているアプリケーションの中で同じユーザ・レジストリ（LDAP/AD）を共有することが推奨されます。

  ユーザ・マップに失敗した場合，セキュリティ違反や好ましくないアプリケーションの動作を引き起こすことがあります。たとえば，同じユーザ名がさまざまなアプリケーションの異なる実ユーザに割り当てられることなどが考えられます。

- Identity Manager が認証に使われます。

  Identity Manager で保護されていないリソースはすべて，LW-SSO 設定ではセキュアでない URL として設定する必要があります。

ライトウェイトなシングル・サインオンの使用

次のセクションでは，ライトウェイトな SSO を使用するためのガイドラインについて説明します。

- クライアントは，たとえば http://flood.mercury.global:8080/WebApp などのように，ログイン URL では完全修飾名（FQDN）を使ってアプリケーションにアクセスする必要があります。

  LW - SSO は，http://16.59.45.143:8080/WebApp など，IP アドレスを使った URL はサポートできません。

  LW - SSO は，http://flood:8080/WebApp など，ドメインなしの URL はサポートできません。

- アプリケーションは，LW-SSO フレームワーク内で事前に統合されている場合にかぎり，LW-SSO の機能を利用できます。

- マルチ・ドメイン・サポートの制限事項

  - マルチ・ドメイン機能は HTTP 参照元に基づいています。そのため，LW-SSO はあるアプリケーションから別のアプリケーションへのリンクはサポートしますが，両方のアプリケーションが同じドメイン内にある場合を除き，ブラウザ・ウィンドウへの URL の入力をサポートしません。

  - HTTP POST を使った最初のクロス・ドメイン・リンクはサポートしません。

    マルチ・ドメイン機能は，2 番目のアプリケーションへの最初の HTTP POST 要求はサポートしていません（HTTP GET 要求のみサポートしています）。たとえば，アプリケーションに 2 番目のアプリケーションへの HTTP リンクがある場合，HTTP GET 要求はサポートされますが，HTTP FORM 要求はサポートされません。1 回目以降のすべての要求は HTTP POST と HTTP GET のどちらも可能です。

  - LW-SSO トークン・サイズの制限事項

    あるアプリケーションからドメインの異なる別のアプリケーションへ LW-SSO が送信できる情報のサイズは，15 グループ / ロール / 属性に制限されています（各項目は平均 15 文字の長さが可能です）。

➤ マルチ・ドメイン・シナリオにおける，保護されたページ（HTTPS）から保護されていないページ（HTTP）へのリンク

マルチ・ドメイン機能は，保護された（HTTPS）ページから保護されていない（HTTP）ページへリンクする場合はうまく動作しません。これはブラウザの制限であり，保護されたリソースから保護されていないリソースへリンクする場合に参照元のヘッダが送信されないためです。たとえば，http://support.microsoft.com/support/kb/articles/Q178/0/66.ASP を参照してください。

➤ Internet Explorer でのサード・パーティ・クッキーの動作

IE6 では，Microsoft は Platform for Privacy Preferences（P3P）Project をサポートするモジュールが追加されています。つまり，サード・パーティ・ドメインから送信されるクッキーは，インターネット・セキュリティ・ゾーンにおいてデフォルトでブロックされます。これは，セッション・クッキーも同様に IE によってサード・パーティ・クッキーとみなされ，そのためブロックされ，結果として LW - SSO の動作停止を引き起こすことを意味します。

詳細については http://support.microsoft.com/kb/323752/jp-jp を参照してください。考えられる解決策は，起動したアプリケーション（または *.mydomian.com のような DNS ドメイン・サブセット）をコンピュータ上のイントラネット・ゾーンまたは信頼済みゾーンに追加し（IE メニュー ＞［ツール］＞［インターネット オプション］＞［セキュリティ］＞［ローカル イントラネット］＞［サイト］＞［詳細設定］），クッキーを受け付けることです。

重要：重要：LW-SSO セッション・クッキーは，ブロックされるサード・パーティ・アプリケーションによって使われるクッキーの 1 つにすぎません。

➤ Tomcat では JAAS レルムはサポートされません。

➤ Tomcat ディレクトリでの空白の使用はサポートされません。

Tomcat のインストール・パス（フォルダ）に空白が含まれており（たとえば Program Files など），LW-SSO 設定ファイルが common¥classes Tomcat フォルダにある場合，LW-SSO は使用できません。

➤ 負荷分散の設定

LW-SSO と一緒にデプロイされる負荷分散は，スティッキー・セッションを使用するように設定する必要があります。

➤ リバース・プロキシの設定

LW-SSO は対称型リバース・プロキシ仮想ノードを複数のサポートできます。LW-SSO は非対称型リバース・プロキシ仮想ノードを 1 つだけサポートできます。

次のリバース・プロキシ設定が LW-SSO によってサポートされています。

例 1：

```
ProxyPass /App1 http://APP_server/App1
ProxyPass /App2 http://APP_server/App2
...
ProxyPass /AppN http://APP_server/App2N
```

例 2：

```
ProxyPass Node1/App1 http://APP_server/App1
ProxyPass Node1/App2 http://APP_server/App2
...
ProxyPass Node1/AppN http://APP_server/AppN
```

次のリバース・プロキシ設定は LW-SSO によってサポートされていません。

```
ProxyPass Node1/App1 http://APP_server/App1
ProxyPass Node2/App2 http://APP_server/App2
...
ProxyPass NodeN/AppN http://APP_server/AppN
```

重要：LW-SSO トークン有効期限の推奨設定：LW-SSO を使っているアプリケーションごとにトークンの有効期限を設定する必要があります。推奨値は 60 分です。高レベルのセキュリティを必要としないアプリケーションの場合，300 分の値を設定できます。

重要：LW - SSO 統合に含まれているアプリケーションはすべて，最大誤差 15 分で，同じ GMT 時間を使用している必要があります。

重要：マルチ・ドメイン機能では，LW-SSO 統合に含まれているアプリケーションはすべて，異なる DNS ドメインのアプリケーションと統合する必要がある場合は，protectedDomains を設定する必要があります。さらに，設定の lwsso 要素に正しいドメインを追加する必要もあります。

重要：正規化された URL パラメータ機能：ほかのアプリケーションから正規化された URL パラメータとして送信された情報を受信するために，ホスト・アプリケーションは設定内の lwsso 要素で正しいドメインを設定する必要があります。

ライトウェイトなシングル・サインオン・システムの要件

ツール	バージョン
Java:	1.5 以降
HTTP Servlets API	2.1 以降
Internet Explorer	6.0 以降 ブラウザは HTTP セッション・クッキーと HTTP 302 リダイレクト機能が有効になっている必要があります。
FireFox	2.0 以降 ブラウザは HTTP セッション・クッキーと HTTP 302 リダイレクト機能が有効になっている必要があります。
Tomcat 認証	スタンドアロン Tomcat 5.5.28 スタンドアロン Tomcat 5.5.20

ツール	バージョン
Acegi 認証	Acegi 0.9.0 Acegi 1.0.4
Web サービス・エンジン	Axis 1 - 1.4 Axis 2 - 1.2 JAX-WS-RI 2.1.1

付録 C

Diagnostics コンポーネント間での HTTPS 有効化

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。
- コンポーネントの証明書を通信を開始する Diagnostics コンポーネントにコピーする。
- 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 コンポーネントの設定と通信図」の「コンポーネント通信図」が参考になります。

Diagnostics コンポーネントでの着信 HTTPS 通信の有効化

このセクションでは，着信 HTTPS 通信を受信するように Diagnostics Server，Java Probe 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着信できます。

このセクションは，次の項で構成されます。

- 611 ページ「Diagnostics Server の着信 HTTPS 接続用の設定」
- 614 ページ「Java Probe 着信 HTTPS 接続用の設定」
- 617 ページ「Collector の着信 HTTPS 接続用の設定」

Diagnostics Server の着信 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 Server（Commander モード）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。
- Commanding Server で HTTPS 通信を有効にする場合，ポート 8443 を使って Enterprise UI: **https://commanding_server:8443** を実行する必要があります。

Diagnostics Server を HTTPS 着信接続用に設定するには

1 次のコマンドを使って，**＜Diagnostics Server 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。

```
＜Diagnostics Server のインストール・ディレクトリ＞/_jvm/bin/keytool -genkey -keystore ＜Diagnostics Server 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 Server 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

- **＜Diagnostics Server のインストール・ディレクトリ＞ を** Diagnostics Server のインストール・ディレクトリのパスに置き換えます。
- **＜Diagnostics Server のホスト名＞** を Diagnostics Server 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。必要に応じて，**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 Server のインストール・ディレクトリ＞/etc/keystore** にキーストアと，Diagnostics Server のホストの **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 Server のインストール・ディレクトリ＞ /_jvm/bin/keytool -export -keystore ＜ Diagnostics Server 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 Server 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

- **＜ Diagnostics Server のインストール・ディレクトリ＞** を Diagnostics Server 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 Server の **＜ Diagnostics Server 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 Server 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 Server のインストール・ディレクトリ＞** を Diagnostics Server 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 Server のインストール・ディレクトリ＞ /_jvm/bin/java -cp
＜ Diagnostics Server 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 Server（Commander モード）では，**＜ Diagnostics Server 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= ＜難読化されたパスワード＞** を設定します。

c **keyPassword= ＜難読化されたパスワード＞** を設定します。

注：＜難読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。例：
keyStorePassword=OBF:1ytc1vu91v2p1y831y7v1v1p1vv11yta

Java Probe 着信 HTTPS 接続用の設定

注：

- Java Probe 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 Probe システムとの HTTPS 通信を有効にする場合，ポート 45000 を使って Probe Profiler UI: **https://probe_system:45000** を実行する必要があります。

Java Probe を着信 HTTPS 接続用に設定するには

1 次のコマンドを使って，**＜ Probe 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。

```
＜ Probe のインストール・ディレクトリ＞ /_jvm/bin/keytool -genkey -keystore ＜ Probe 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= ＜ Probe 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

- **＜ Probe のインストール・ディレクトリ＞** を Java Probe のインストール・ディレクトリのパスに置き換えます。
- **＜ Probe のホスト名＞** を Java Probe 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。必要に応じて，**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，**＜ Probe のインストール・ディレクトリ＞ /etc/keystore** にキーストアと，Java Probe のホストの **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。

```
＜Probe のインストール・ディレクトリ＞/_jvm/bin/keytool -export -keystore ＜Probe 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 ＜Probe のインストール・ディレクトリ＞/etc/＜Probe の証明書名＞.cer
```

このコマンドを使用するには

- **＜Probe のインストール・ディレクトリ＞** を Java Probe 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。
- **＜Probe 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。

  Probe のタイプと Probe 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。例：**Java_probe_＜Probe のホスト名＞**

このコマンドの実行後，Java Probe の **＜Probe のインストール・ディレクトリ＞/etc** ディレクトリに **Java_probe_＜Probe 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 Probe 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 **＜Probe のインストール・ディレクトリ＞** を Java Probe 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。

```
＜ Probe のインストール・ディレクトリ＞ /_jvm/bin/java -cp
＜ Probe 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 **＜ Probe 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= ＜難読化されたパスワード＞** を設定します。

c **keyPassword= ＜難読化されたパスワード＞** を設定します。

注：＜難読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。
例：keyStorePassword=OBF:1ytc1vu91v2p1y831y7v1v1p1vv11yta

Collector の着信 HTTPS 接続用の設定

このセクションでは，着信 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 を着信 HTTPS 接続用に設定するには

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

- **＜ 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。必要に応じて，**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 のホストの **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

- **＜ 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。例：**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= ＜難読化されたパスワード＞** を設定します。

   c **keyPassword= ＜難読化されたパスワード＞** を設定します。

   ＜難読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。
   例：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。

Diagnostics Server（Commander モード）で，HTTPS を使用した Diagnostics Server（Mediator モード）への発信通信を有効にするには

1 Diagnostics Server の **＜Diagnostics Server のインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** から，Diagnostics Server（Commander モード）の **＜Diagnostics Server 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 Server（Commander モード）の **＜ Diagnostics Server 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 Diagnostics Server 通信の場合，Diagnostics Server（Commander モード）の **＜ Diagnostics Server のインストール・ディレクトリ＞ /etc/server.properties** ファイルにある以下のプロパティを更新して，Diagnostics Server の URL を指定します。

commander.url を **https:// ＜ Diagnostics Server Commander のホスト名＞ :8443** に設定します。

Diagnostics Server（Mediator モード）で，HTTPS を使用した Diagnostics Server（Commander モード）への発信通信を有効にするには

1 Diagnostics Server（Commander モード）の **＜ Diagnostics Server のインストール・ディレクトリ＞ /etc/diag_server_commander.cer** から，Diagnostics Server（Mediator モード）の **＜ Diagnostics Server 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 Diagnostics Server（Mediator モード）の **＜ Diagnostics Server 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_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。

3 着信 Diagnostics Server 通信の場合，Diagnostics Server（Mediator モード）の **＜ Diagnostics Server のインストール・ディレクトリ＞ /etc/server.properties** ファイルにある以下のプロパティを更新して，Diagnostics Server の URL を指定します。

commander.url を **https:// ＜ Diagnostics Server Commander 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 Server（Commander または Mediator モード）で，HTTPS を使用した Probe への発信通信を有効にするには

1 各 Probe の **＜ Probe のインストール・ディレクトリ＞ /etc/java_probe_ ＜ Probe のホスト＞ .cer** から Diagnostics Server の **＜ Diagnostics Server のインストール・ディレクトリ＞ /etc/Java_probe_ ＜ Probe のホスト＞ .cer** に証明書ファイルをコピーします。

2 Diagnostics Server の **＜ Diagnostics Server 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 Server（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

1 各 Probe 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 Server の **＜ Diagnostics Server 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 Server の ＜ **Diagnostics Server 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 Probe で，HTTPS を使用した Diagnostics Server（Mediator モード）への発信通信を有効にするには

1 Diagnostics Server（Mediator モード）の **＜ Diagnostics Server のインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** から，Java Probe の **＜ Probe 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 Probe の **＜ Probe 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 Probe 通信の場合，**＜ Probe 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，Diagnostics Server（Mediator モード）の URL を指定します。

registrar.url を **https:// ＜ Diagnostics Server（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 Probe で，HTTPS を使用した Diagnostics Server（Mediator モード）への発信通信を有効にするには

1 Diagnostics Server（Mediator モード）の **＜ Diagnostics Server 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 Probe 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 Probe 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 着信の埋め込み Java Probe 通信の場合，**＜サーバ /Collector のインストール・ディレクトリ＞ /probe/etc/collector.properties** ファイルの以下のプロパティを更新して，Diagnostics Server （Mediator モード）の URL を指定します。**registrar.url** を **https:// ＜ Diagnostics Server（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 Diagnostics Server（Mediator モード）への発信通信を有効にするには

1 Diagnostics Server（Mediator モード）の **＜ Diagnostics Server 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，Diagnostics Server（Mediator モード）の URL を指定します。

registrar.url を **https:// ＜ Diagnostics Server（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 Probe で，HTTPS を使用した Diagnostics Server（Commander モード）への発信通信を有効にするには

1 Diagnostics Server（Mediator モード）の証明書を .NET Probe のホストにコピーします。証明書は，Diagnostics Server（Mediator モード）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。Diagnostics Server（Mediator モード）で HTTPS を受信するように設定する方法については，610 ページ「Diagnostics コンポーネントでの着信 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 Server 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，Diagnostics Server（Mediator モード）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 Server（Commander モード）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。通常，このディレクトリは **＜ Diagnostics Server のインストール・ディレクトリ＞ /etc/diag_server_mediator.cer** です。

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（Mediator Server 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，Probe 設定ファイルを変更する際に使用します。

21 ＜ Probe 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。

Mediator のホストとポートを変更して，
ssl="true": **＜ mediator host=" ＜ Diagnostics Mediator Server のホスト名＞ " port="2612" metricport="8443" ssl="true"/ ＞** を追加します。

22 ＜ Probe 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 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しなければなりません。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。

---

23 IIS を再起動します。IIS の再起動については，214 ページ「IIS の再起動」を参照してください。

Diagnostics Server（Commander モード）との HTTPS 通信が .NET Probe に正常に設定されたことを確認するには

1 .NET アプリケーションを参照して .NET Probe をアクティブにします。

2 システムの状況に .NET Probe ノードが表示されていることを確認します。

Business Availability Center サーバでの HTTPS 通信の有効化

以下は，Business Availability Center に Diagnostics との HTTPS 通信を設定するための手順です。

Diagnostics Server（Commander モード）と Business Availability Center の間で HTTPS 通信を有効にするには

1 Diagnostics Server（Commander モード）のインストール・ディレクトリ **＜Diagnostics Server のインストール・ディレクトリ＞/etc/** から Business Availability Center 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 Availability Center 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 Availability Center 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 Availability Center 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 Availability Center の証明書ファイル **<BAC_certificate_file.cer >** を Diagnostics Server のホストにコピーします。

4 Diagnostics Server 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 Server cacert キーストアにインポートします。

```
＜Diagnostics Server 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 Server のインストール・ディレクトリ＞/JRE/lib/security/cacerts
```

- **＜Diagnostics Server のインストール・ディレクトリ＞** を Diagnostics Server（Commander モード）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 Availability Center 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 Availability Center サーバを Diagnostics Server（Commander モード）の HTTPS ポートに指定します。

a ［**ADMIN**］＞［**Diagnostics**］を選択して，Business Availability Center で Diagnostics の［管理］を開きます。

b ［**Registration**］タブをクリックします。

c Diagnostics Server の詳細セクションを見つけます。

d 適切なフィールドに以下の情報を入力します。

- Diagnostics Server（Commander モード）のホスト名に，Diagnostics Server（Commander モード）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，610 ページ「Diagnostics コンポーネントでの着信 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 Server の Web ポートに 8443 と入力します。

e ［**Submit Configuration**］をクリックします。

Diagnostics コンポーネント別の HTTPS チェックリスト

以下の表は，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	Commanding Server	Mediator Server	Java Probe	Collector	.NET Probe
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
Commanding Server の証明書をコピーする	いいえ	**はい**	いいえ	いいえ	いいえ
Mediator Server の証明書をコピーする	**はい**	いいえ	**はい**	**はい**	**はい**
Java Probe の証明書をコピーする	**はい**	**はい**	いいえ	いいえ	いいえ
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 の enablessl=true，keystorepassword, keypassword を変更する	**はい**	**はい**	**はい**	**はい**	いいえ
security.properties を編集し，Commanding Server 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，Mediator Server 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，Java Probe 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，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
Mediator Server 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**

設定手順	Commanding Server	Mediator Server	Java Probe	Collector	.NET Probe
server.properties を編集して，set 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**

付録 D

System Health Monitor の使用

System Health Monitor を使って，HP Diagnostics コンポーネントの設定を調べたり，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

System Health Monitor の概要

System Health Monitor は，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。ユーザは，問題のあるコンポーネントや，各コンポーネントの負荷，およびコンポーネント間で流れるデータの量を一目で確認することができます。

System Health Monitor component map にドリルダウンすることで，設定，パフォーマンス測定値およびコンポーネントのプロセス・ログに関する詳細を表示することができます。また，System Health Monitor component map に表示されたさまざまな問題に対応するためのトラブルシューティングのヒントも確認できます。多くの場合，System Health Monitor は，Diagnostics デプロイメントおよびそれらをホストするマシンのコンポーネント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。

さらに，System Health Monitor 情報のスナップショットを .xml ファイルでキャプチャし，他者と共有してマップに表示される問題の診断に役立てることもできます。

System Health Monitor へのアクセス

Web ブラウザ，Performance Center，LoadRunner Controller または Business Availability Center から System Health Monitor に直接アクセスできます。

注：System Health Monitor にアクセスするには，**system** 権限が必要です。権限の詳細については，576 ページ「ユーザ権限について」を参照してください。

Web ブラウザから System Health Monitor にアクセスするには

1 ブラウザに次の URL を入力します。

http:// ＜ Diagnostics Server（Commander モード）のホスト＞ : ＜ Diagnostics Server（Commander モード）のポート＞ /registrar/health

Diagnostics Server（Commander モード）のデフォルトのポートは 2006 です。ほかのポートを使うように Diagnostics Server 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。

2 ユーザ名とパスワードを入力します。ユーザ名とパスワードの詳細は，付録 B「ユーザの認証と承認」を参照してください。

ブラウザに System Health Monitor が開きます。

注：Business Availability Center，Performance Center，および LoadRunner からもシステムの状況モニタにアクセスできます。

System Health Monitor へのアクセスの問題を解決する

System Health Monitor にアクセスできない場合，以下を確認してください。

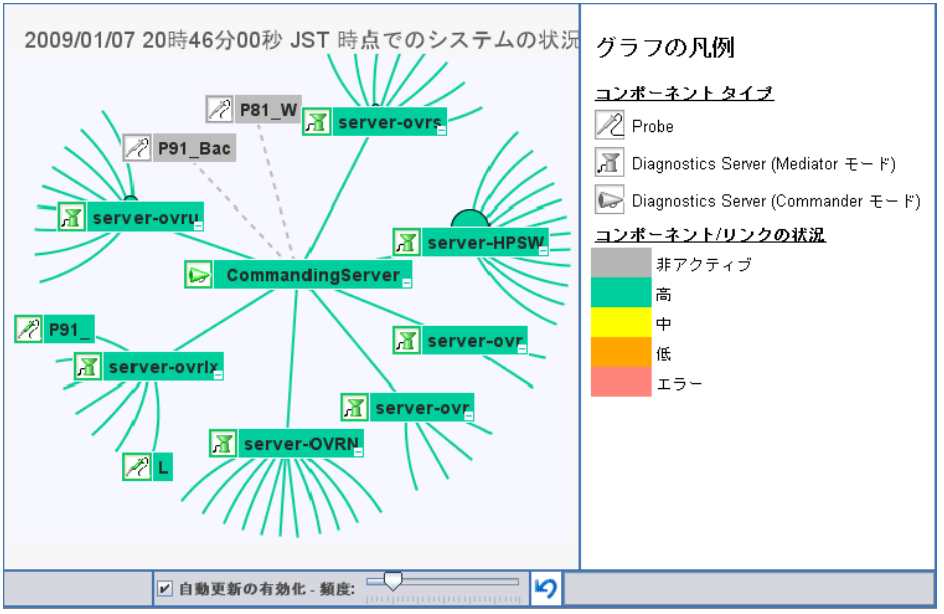
- **Diagnostics Server** ホストのマシン名が正しい。
- **Diagnostics Server** のポート番号が正しい。デフォルトのポートは **2006** です。
- **Diagnostics Server** が正常に起動しており，ホストで実行している。

System Health Monitor について

System Health Monitor は，次の 2 つのペインで構成されます。

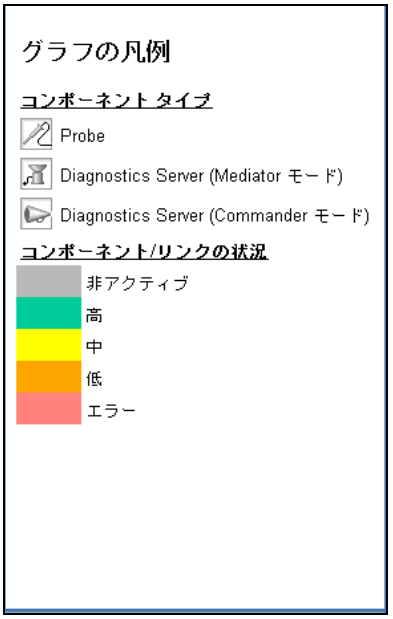
- **グラフの凡例**（右側のペイン）
- **System Health Monitor component map**（左側のペイン）

System Health Monitor が開くと，HP Diagnostics コンポーネントのデプロイメント・マップがグラフの凡例と一緒に表示されます。



グラフの凡例

コンポーネント・マップの右側にあるグラフの判例は，コンポーネント・マップに表示される情報を解釈する上で役立ちます。



システムの状況モニタのコンポーネント・マップ

System Health Monitor component map には，Diagnostics デプロイメントに関する高度な情報，およびコンポーネント実行状況が表示されます。System Health Monitor component map の表示内容は次のとおりです。

- Diagnostics 環境の一部として正常に設定されている各 Diagnostics コンポーネントを表すアイコン。
- コンポーネントの状況を示すコンポーネント・アイコンの色。
- コンポーネントが処理している負荷量を示すインジケータ。
- コンポーネント間の通信状況を示すコンポーネント・アイコンの間のリンク。
  - リンクの色は，リンクの状態を示します。
  - 実線は，データがリンク間を流れていることを示します。
  - 点線は，現在データが流れていないことを示します。

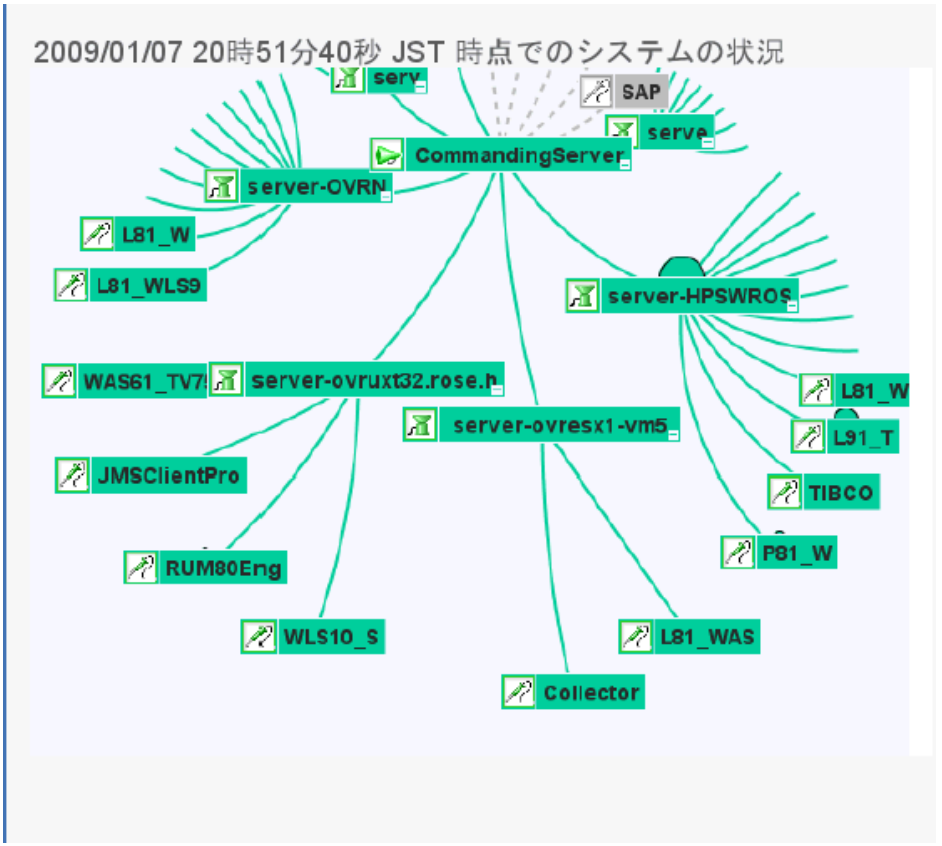
次の Diagnostics デプロイメントの図には，以下のものが表示されています。

1 つの Diagnostics Server が Commander モードでデプロイされています。これは，**CommandingServer** というラベルの付いたアイコンでマップ内に示されます。

複数の Diagnostics Server が Mediator モードでデプロイされています。これらは，**server- ＜ホスト名＞** という名前の付いたアイコンで示されます。

各サーバに報告する Probe が表示されています。Probe は，**＜ Probe 名＞ というアイコンで示されます。**

各サーバに報告する Diagnostics Collector が表示されています。Collector は，**<Collector 名 > というアイコンで示されます。**



コンポーネントの状態とホストの設定ツールチップを表示する

System Health Monitor component map の各コンポーネントのツールチップを表示できます。ツールチップには，基本的な状態情報およびホストの設定情報が表示されます。

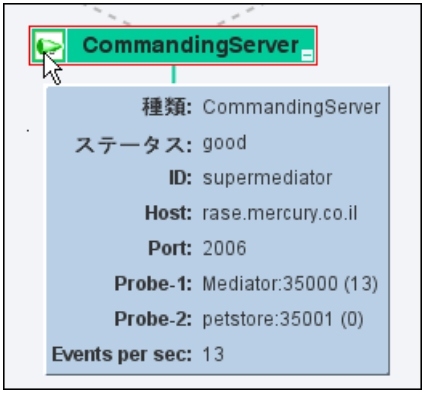
コンポーネントの状態とホストの設定ツールチップを表示するには

ツールチップが表示されるまで，コンポーネント上にマウスを置きます。

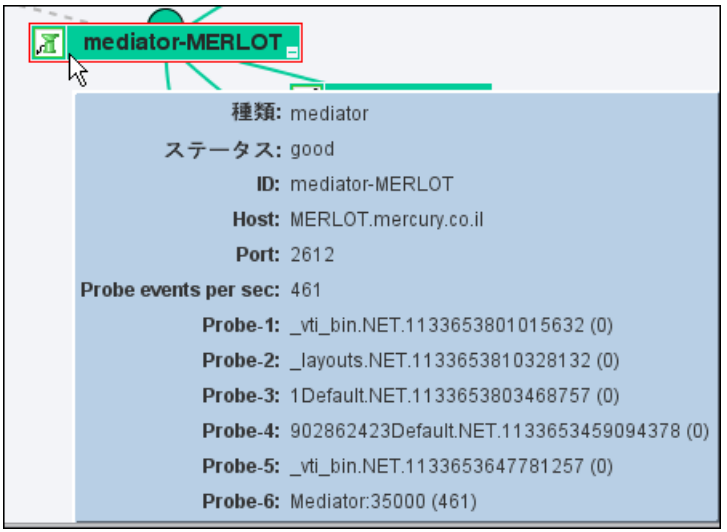
次の例では，ツールチップに，各コンポーネントに関するコンポーネントの状態情報とホストの設定情報が表示されています。

Diagnostics Server コンポーネントのツールチップに表示される情報には，Diagnostics Server と一緒に動作するように設定されている Probe のリストが含まれます。

➤ **Diagnostics Server（Commander モード）**

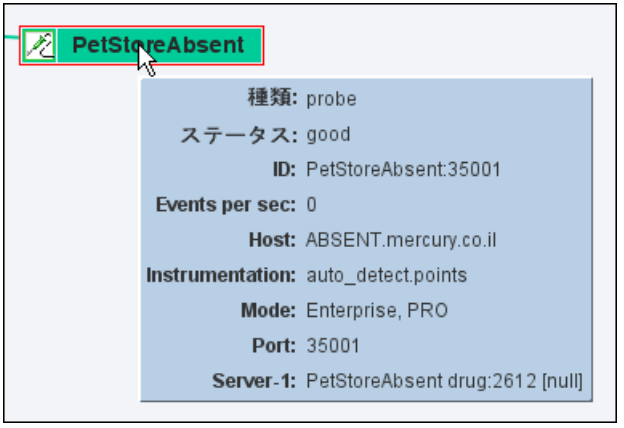


➤ Diagnostics Server（Mediator モード）

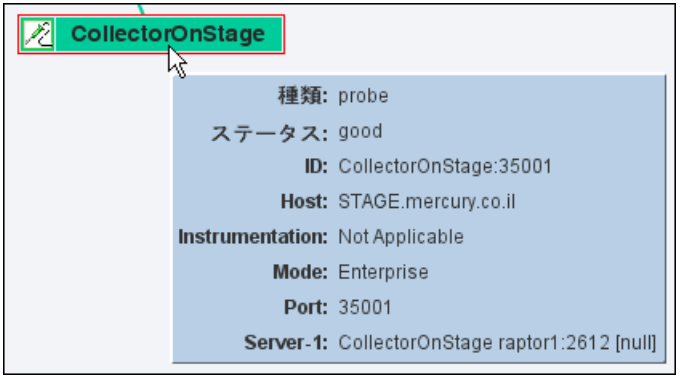


➤ Diagnostics Probe

プローブ・コンポーネントのツールチップに表示される情報には，そのプローブに設定されている動作モードが含まれます（[**Mode**] フィールド）。また，ツールチップには，関連するインストゥルメンテーション・ポイント・ファイルも示されます。



➤ **Diagnostics Collector**



コンポーネントの詳細情報の表示

System Health Monitor component map の各コンポーネントの設定およびパフォーマンスに関する詳細情報を表示できます。これらの情報は，［Componennt Monitor］詳細テーブルに表示されます。

コンポーネントの詳細情報を表示するには

コンポーネントを右クリックして，メニューから関連する項目を選択します。

または，コンポーネント・アイコンをダブルクリックして関連するセクションにスクロールします。

以下の情報を表示できます。

- **コンポーネント測定値情報**：選択したコンポーネントのプロセス測定値を一覧表示します。
- **コンポーネント設定情報**：選択したコンポーネントのコンポーネント設定が一覧表示されます。
- **コンポーネント・ログ情報**：選択したコンポーネントのログ・メッセージが一覧表示されます。
- **コンポーネント・ステータス情報**：コンポーネントのパフォーマンス・ステータスに関する詳細情報が必要に応じて表示されます。
- **コンポーネントのその他の情報**：コンポーネントに関するその他の情報が表示されます。

コンポーネント測定値情報

次の図は，Diagnostics Server（Commander モード）のコンポーネント測定値情報の部分を示します。

Component Monitor Update As Of 2007/09/03 12時54分06秒 JST

Metric	
Events since launch	<unknown>
Last update (commander time)	Wed Dec 07 04:22:35 PST 2005
Up time	1 day(s) 0 hour(s) 4 minute(s) 53 second(s)
WDE bytes relayed	166836
WDE bytes relayed since launch	92396096
WDE samples failed	75
WDE samples failed since launch	58870
WDE samples relayed	171
WDE samples relayed since launch	94676

コンポーネント設定情報

次の図は，.NET Probe のコンポーネント設定情報を示します。

Component Monitor Current As Of 2009/01/07 21時04分00秒 JST

Configuration	
Host	cinderella.asiapacific.hpqcorp.net
IP Address	15.39.60.205
Install dir	C:\MercuryDiagnostics\.NET Probe\
Instrumentation	ASP.NET.points,MSPetShop.points
Lightweight Memory Diagnostics	OFF
Mode	Enterprise,PRO
Port	35000
Probe Group	Default
System	CINDERELLA (Microsoft Windows NT 5.2.3790.0)
VM	.NET 1.1.4322.2407
Version	8.0.25.450

コンポーネント・ログ情報

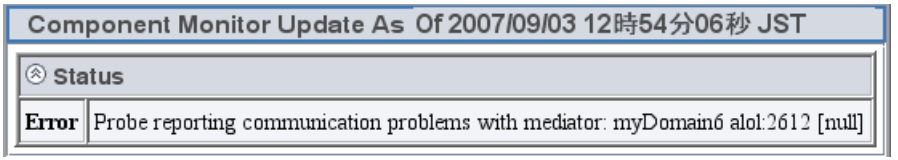
次の図は，Diagnostics Server（Mediator モード）のコンポーネント・ログ情報を示します。

Component Monitor Update As Of 2007/09/03 12時54分06秒 JST

Log	
Wed Dec 07 04:06:30 PST 2005	WARNING: real user fragment 31289557 timed out
Wed Dec 07 03:09:02 PST 2005	WARNING: real user fragment 32930633 timed out
Wed Dec 07 03:08:49 PST 2005	WARNING: real user fragment 32687513 timed out
Wed Dec 07 01:08:38 PST 2005	WARNING: real user fragment 8834978 timed out
Tue Dec 06 23:08:41 PST 2005	WARNING: real user fragment 12640068 timed out
Tue Dec 06 05:51:44 PST 2005	SEVERE: out of order com.mercury.diagnostics.server.persistence.impl.D
Tue Dec 06 05:51:44 PST 2005	SEVERE: out of order com.mercury.diagnostics.server.persistence.impl.D
Tue Dec 06 05:46:01 PST 2005	SEVERE: out of order com.mercury.diagnostics.server.persistence.impl.D

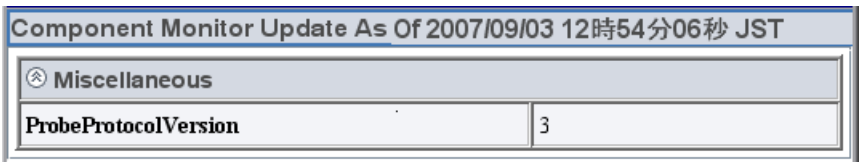
コンポーネント・ステータス情報

次の図は，Java Probe のコンポーネント・ステータス情報の例です。この情報は，コンポーネントのステータスが「**高**」よりも低くなったときに表示されます。



コンポーネントのその他の情報

次の図は，Java Probe のコンポーネントのその他の情報を示します。



トラブルシューティングのヒントの表示

System Health Monitor component map には，各コンポーネントに関するトラブルシューティング情報が表示されます。

Diagnostics コンポーネントのトラブルシューティングのヒントを表示するには

関連するコンポーネントを右クリックし，［**トラブルシューティングのヒントを表示 ...**］を選択します。選択したコンポーネントのトラブルシューティング情報が表示されます。

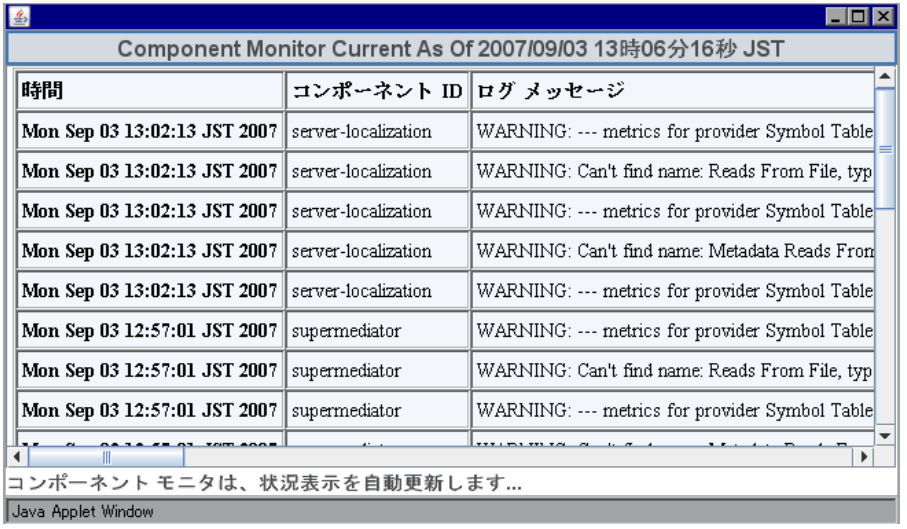
調べている症状に関する情報に目をとおし，トラブルシューティングのヒントに推奨する解決策があるかどうかを確認します。

システム全体のログ情報の表示

画面に，System Health Monitor のすべてのコンポーネントに関するログ・メッセージを表示できます。

すべてのコンポーネントに関するログ情報を表示するには

System Health Monitor component map の任意の場所（コンポーネント・アイコン以外）を右クリックして［**ログ履歴の表示**］を選択します。すべてのコンポーネントの活動に関するログが表示されます。



System Health Monitor 画面のカスタマイズ

System Health Monitor の情報の表示方法をカスタマイズすることができます。

システムの状況モニタのコンポーネント・マップ表示の操作

System Health Monitor component map で，ツリーの領域の表示，展開した分岐または畳んだ分岐の表示，画像の全体像の変更を操作できます。

システムの状況モニタのコンポーネント・マップ表示を操作するには

➤ コンポーネントをクリックしてマップの中央に移動します。

- コンポーネント・マップ内の任意の場所をクリックして，マップの強調部分を変更します。コンポーネントは，マウスをクリックした場所が中心になります。
- コンポーネント・マップの任意の場所をクリック・アンド・ドラッグして，そこを中心にコンポーネントを移動および回転します。
- コンポーネント・アイコンの右下角に表示される展開（+）または折り畳み（-）記号をクリックして，サブツリーの分岐を展開したり折り畳んだりします。
- キーボードの ALT キーを押しながらコンポーネント・マップにドラッグして，各分岐間の隙間を広げたり縮めたりします。

グラフの凡例の表示

System Health Monitor にグラフの凡例を表示するかどうかを指定することができます。

グラフの凡例を表示するには

- グラフの凡例が表示されていない場合，System Health Monitor component map の任意の場所（コンポーネント・アイコン以外）を右クリックして［**グラフの凡例を表示する**］を選択します。

グラフの凡例を非表示にするには

- グラフの凡例が表示されている場合，System Health Monitor component map の任意の場所（コンポーネント・アイコン以外）を右クリックして［**グラフの凡例を非表示する**］を選択します。

負荷の表示

System Health Monitor の負荷の表示方法をカスタマイズすることができます。負荷インジケータをコンポーネント・アイコンの後ろに円として表示するか，コンポーネント・アイコンの下にバー・スケールとして表示するかを選択できます。

負荷を円で表示するには

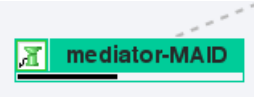
- 負荷がバー・スケールとして表示されている場合，System Health Monitor component map の任意の場所（コンポーネント・アイコン以外）を右クリックして［**円を使って負荷を表示する**］を選択します。

次の例で，コンポーネントの後ろの円は負荷の量を表します。

負荷をバー・スケールで表示するには

➤ 負荷が円で表示されている場合，System Health Monitor component map の任意の場所（コンポーネント・アイコン以外）を右クリックして［**スケールを使って負荷を表示する**］を選択します。

次の例で，コンポーネントの後ろのバー・スケールは負荷の量を表します。

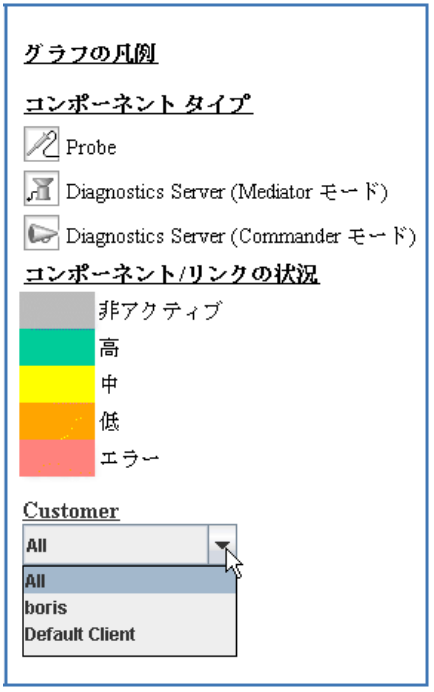


システムの状況モニタのコンポーネント・マップのカスタマごとのフィルタリング

複数のカスタマが関係する Diagnostics in an Software as a Service（SaaS）環境で Diagnostics を実行している場合，選択したカスタマの情報だけがグラフに表示されるように System Health Monitor に表示される情報をフィルタリングすることができます。

デフォルトで，System Health Monitor にはすべてのカスタマのコンポーネントが表示されます。

System Health Monitor に表示される Diagnostics デプロイメントに複数のカスタマが示されている場合，［グラフの凡例］ボックスにカスタマのリストが表示されます。



System Health Monitor に表示するコンポーネントを持つカスタマを選択するか，［**すべて**］を選択してすべてのカスタマのコンポーネントを一度に表示します。

System Health Monitor の更新頻度の制御

デフォルトで，System Health Monitor は，表示される情報を自動的に更新するように設定されています。自動更新頻度を変更したり，自動更新機能を完全に無効にすることができます。また，いつでも手動更新を要求することができます。

System Health Monitor の下部にあるコントロールを使って，表示される情報の更新を次のように制御することができます。

自動更新機能を無効にするには

➤［**自動 - 更新の有効化**］チェック・ボックスの選択を解除します。

自動更新機能を有効にするには

➤［**自動 - 更新の有効化**］チェック・ボックスを選択します。

自動更新頻度を調整するには

➤［**頻度**］スライド・コントロールを右にスライドさせると自動更新頻度が低くなり，左にスライドさせると頻度が高くなります。

表示を手動で更新するには

➤ スライド・バーの右側にある［**更新**］ボタンをクリックします。

System Health Monitor のスナップショットの作成および使用

システムの状況モニタのスナップショットを撮って他者と情報を共有することができます。また，他者が撮ったシステムの状況モニタのスナップショットをインポートして，他者のシステムの情報を確認することもできます。

System Health Monitor のスナップショットのエクスポート

System Health Monitor に表示された情報を .xml 形式のファイルにエクスポートして，情報を他者と共有することができます。これは，特に問題の診断の助けが必要なときに便利です。System Health.xml ファイルの情報は，.xml 形式で表示したり，System Health Monitor の作業コピーにインポートできます。

注：Diagnostics Server（Commander モード）ホストへのアクセス権を持つユーザは，URL がわかっている場合に Web ブラウザを使って System Health Monitor を直接表示することができます。

System Health Monitor のスナップショットを .xml ファイルにエクスポートするには

1 Web ブラウザに次の URL を入力します。

http:// ＜ Diagnostics Server（Commander モード）のホスト＞ : ＜ Diagnostics Server（Commander モード）のポート＞ /registrar/xml

デフォルトのポートは **2006** です。ほかのポートを使うように Diagnostics Server 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。

2 Web ブラウザの［**名前を付けて保存**］メニュー・オプションを使って，Web ページをファイルに保存します。ファイルには，スナップショットを保存した理由がわかりやすい名前を付けてください。たとえば，Diagnostics Server（Mediator モード）のパフォーマンスが低下しているために System Health Monitor のスナップショットを撮影した場合，ファイルに次のような名前を付けると便利です。

```
sys_health_mediator_yyyymmdd.xml
```

System Health Monitor のスナップショットのインポート

他者から System Health Monitor の .xml スナップショットが送られてきたときに，既存の System Health Monitor で表示することができます。

System Health Monitor のスナップショットをインポートするには

1 システムの状況のスナップショット・ファイルを Diagnostics Server（Commander モード）のホスト・マシンのディレクトリにコピーします。

2 Web ブラウザに次の URL を入力します。

http:// ＜ Diagnostics Server（Commander モード）のホスト＞ : ＜ Diagnostics Server（Commander モード）のポート＞ /registrar/health?xml= ＜ XML ファイル＞

＜ XML ファイル＞ は，Diagnostics Server（Commander モード）ホスト・マシンの .xml ファイルへのパスです。

たとえば，**sys_health_mediator_yyyymmdd.xml** を Diagnostics Server（Commander モード）ホスト・マシンのハード・ドライブ（C:）に jake という名前でコピーした場合，URL は次のようになります。

http://jake:2006/registrar/health?xml=c:¥sys_health_mediator_yyyymmdd.xml

＜ Diagnostics Server（Commander モード）のデフォルトのポート＞は 2006 です。ほかのポートを使うように Diagnostics Server 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。

Web ブラウザに System Health Monitor が表示されている場合，情報は .xml ファイルからインポートされたものです。

付録 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。1 つ目は，各ユーザが作成したカスタム・ビューで構成されます。2 つ目は，Probe によって収集され，Diagnostics Server によって集計される Diagnostics データで構成されます。このデータは，Diagnostics Server の時系列のデータベースに保存されます。各 Diagnostics Server では，報告を行う Probe によって収集されるデータを保存します。また，Diagnostics Server（Commander モード）は，AD と AM の両方の仮想トランザクションのデータおよびアプリケーション測定値を保存します。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は，以下で説明します。

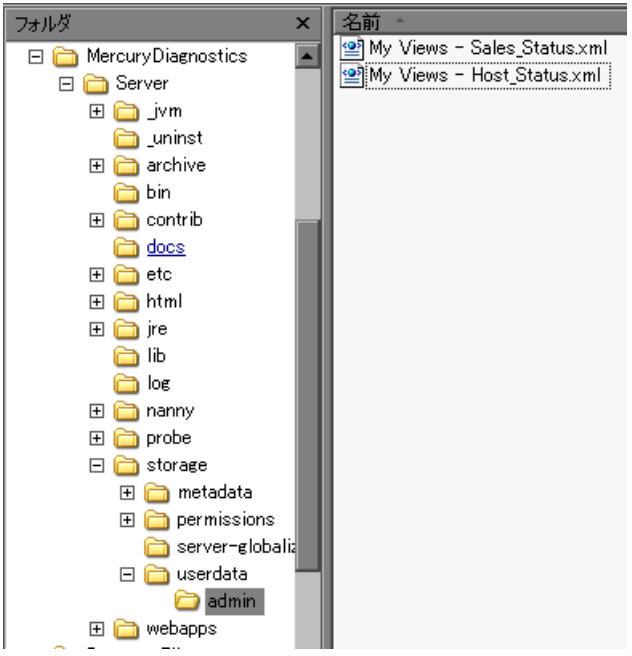
カスタム・ビュー・データ

Diagnostics ユーザは，『**HP Diagnostics User's Guide**』（英語版）の「Customizing Diagnostics Views」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 Server（Commander モード）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 Server（Commander モード）のホストの **＜Diagnostics Server 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 という 2 つのカスタム・ビューを作成した場合，それら 2 つのビューは，次の例のように，Diagnostics Server（Commander モード）の **＜Diagnostics Server のインストール・ディレクトリ＞/storage/userdata/Default Client/admin** ディレクトリに個別の XML ファイルとして保存されます。



パフォーマンス履歴データ

Diagnostics は，Diagnostics Server（Mediator モード）の時系列のデータベースにパフォーマンス履歴データを保存します。Diagnostics Server に，報告を行う数多くの Probe 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。各アプリケーションの収集されたデータ量はサイズが異なりますが，監視している各仮想マシンに約 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，656 ページ「データの保存サイズとデータ保存」を参照してください。

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，Diagnostics Server（Mediator モード）の **＜Diagnostics Server 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，Diagnostics Server（Mediator モード）のホストの名前です。
- **＜カスタマ名＞** は，Diagnostics Server（Mediator モード）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。

注：

- SaaS カスタマでない限り，カスタマ名は必ず**デフォルトのクライアント**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。

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 5 種類のファイルに保存されます。

記号テーブル・ファイル

記号テーブルには，ほかのデータ・ファイルの小型で高速なデータ・エンコードの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 Server 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。

期間ごとのパフォーマンス・サマリ・ファイル

Diagnostics ビューで，データを年 1 回，年 4 回，月 1 回，週 1 回，毎日，6 時間ごと，1 時間ごと，20 分ごと，5 分ごとに表示することができます。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。たとえば，1 週間の間に，次のサマリ・ファイルが作成されます。

- 週 1 回サマリ・ファイル 1 つ
- 毎日サマリ・ファイル 7 つ
- 6 時間ごとサマリ・ファイル 28 個
- 1 時間ごとサマリ・ファイル 168 個
- 20 分ごとサマリ・ファイル 504 個
- 5 分ごとサマリ・ファイル 2,016 個

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，2006 年 11 月 23 日の深夜（GMT）に始まるデータを含むサマリ・ファイルには，**2006_11_23_0.summary** という名前が付けられます。

サマリ・ファイルは，含まれるデータの期間に基づいてサブディレクトリに保存されます。サブディレクトリは，**Years/1Y**，**Years/3M**，**Years/Years**，**Months/1M**，**Weeks/7d**，**Days/1d**，**Days/6h**，**Hours/1h**，**Hours/20m**，**Hours/5m** です。たとえば，毎日サマリ・ファイルの 1 つは，**＜ Diagnostics Server 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＞ _ /Days/1d** サブディレクトリに保存されます。

サマリ・ファイルのデータは，Diagnostics ビューに表示されるグラフとテーブルを作成するのに使われます。

長期間ごとのパフォーマンス・トレンド・ファイル

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。長期間は，年 1 回，月 1 回，週 1 回，毎日，日数，時数です。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，2006 年 11 月 23 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，**Hours/1h/2006_11_23_0.trend** という名前が付けられます。

トレンド・ファイルは，ユーザが UI でグラフにする要素を選択したときにアクセスします。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だけが読み取られます。

長期間ごとのインスタンス・ツリー・ファイル

インスタンス・ツリー・ファイルは，トレンド・ファイルに似ています。長期間でそれぞれ収集したインスタンス呼び出しツリーの対応するダンプがあります。例：**Hours/1h/2005_11_23_0.tree**

インスタンス・ツリー・ファイルは，ユーザが Diagnostics UI 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します。

データの保存サイズとデータ保存

Diagnostics では，データ圧縮および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・ファイルの圧縮

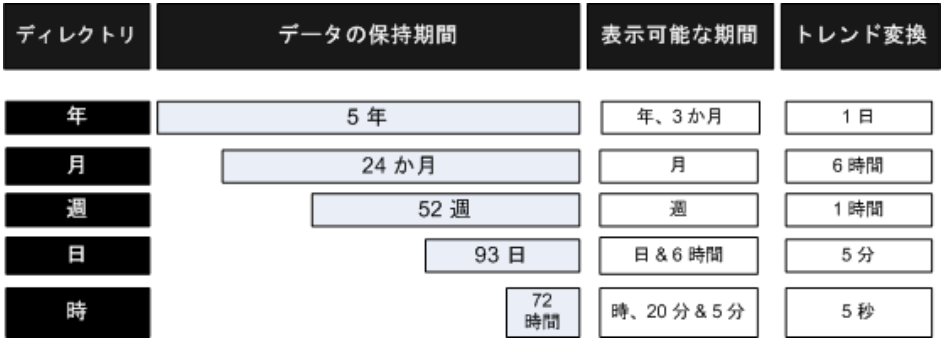
Diagnostics パフォーマンス・データは，さまざまな期間を示す複数のファイルに保存されます。ただし，Diagnostics では，新しい診断データだけを**現在の**期間を示すファイルに**書き込み**ます。つまり，アプリケーションのほとんどの Diagnostics データ・ファイルは，読み取り専用ファイルとして保存可能です。

未圧縮のデータ・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，.zip 拡張子が付いています。

データ・ファイルの保存

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，診断データ保存ポリシーに基づいて保存される各層のデータと一緒に層に保存されます。低解像度データ・ポイントの層のデータは，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。高解像度データ・ポイントの層のデータは，パフォーマンス診断などの活動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より，測定値は，Diagnostics で調査したそれぞれのマシンに対して約 3 GB 使用することになります。

次の図は，一般的な Diagnostics データ保存ポリシーを示します。



注：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。657 ページ「データ保存および削除ポリシー」を参照してください。

データ保存および削除ポリシー

TSDB で使われる削除メカニズムは，次のように機能します。

- スナップショットの一部であるデータは，不要なインシデントが生じるために削除されません。
- 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，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を自動的に削除します。**＜Diagnostics Server 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。
- Probe の名前が変更され，6 か月間（デフォルト）Probe からデータが受信されなければ，Diagnostics は自動的に Probe とそのデータをデータベースから削除します。

  削除間隔は，**persistency.major.4.total.length** パラメータで時間数を増やすことによって，**server.properties** で変更できます（サーバの再起動が必要です）。デフォルト値は 6 か月に設定されています（4320 時間）。

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。また，スナップショットの削除時に，その期間は削除されます。同時に複数のスナップショットを作成するために，各スナップショットでは，新しい保存エントリを作成します。インシデント・エントリは，削除処理ができないためにマージされません。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。1 週間後に Commanding Server に接続されなかった場合，Commanding Server が永久にオフラインになった場合に，分散サーバでアンバウンドな増加を避けるために，保存リストに統合されずに削除が実行されます。

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。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。存在し，期間が保存されていない各トレンド・ファイルでは，トレンドに関連付けられているすべてのファイルが含まれる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。

Diagnostics ビューのデータ・ファイル保存およびトレンド・データ

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 Server 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 秒期間の生データ・ポイントを統合して作成されたものです。

インストール前のデータ管理に関する留意点

大規模な Diagnostics Server のインストールおよび設定を準備する際，パフォーマンス調整について以下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 Server 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 Server 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活動で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 Server 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 Server 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 Server 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。

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 Server では，HTTP If-Modified-Since および Request-Range ヘッダ（「reget」）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ができます。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 には，**＜ Diagnostics Server のインストール・ディレクトリ /bin/remote-backup.sh** に保存されているリモート・バックアップ・スクリプトが一緒にインストールされています。リモート・バックアップ・スクリプトは，wget ユーティリティを使用する（http://www.gnu.org/software/wget/wget.html）UNIX スクリプトで，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/）を使って，このスクリプトを実行できます。

注：Diagnostics Server 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

パラメータ	説明
-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（オプション）デフォルト：2006
-r	（RHTTP バックアップの）読み取り元の Diagnostics Server（Mediator モード）の ID（オプション）

たとえば，Dragonfly マシンで実行している Diagnostics Server 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/**

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 Server 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

1 クリーンな Diagnostics Server をインストールします。インストールが完了すると，Diagnostics Server は自動的に起動します。

2 Diagnostics Server 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 Server 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 Server から **＜Diagnostics Server 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 Server 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 Server のホスト名が変更されている場合，Diagnostics Server 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 Server 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 Server 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 Server のインストール・ディレクトリ＞/archive/mediator-oldhost** を **＜Diagnostics Server 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 Server 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 Server は Probe 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

HP Diagnostics 7.50 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「Diagnostics およびその他の HP ソフトウェア製品のアップグレード」を参照してください。

付録 F

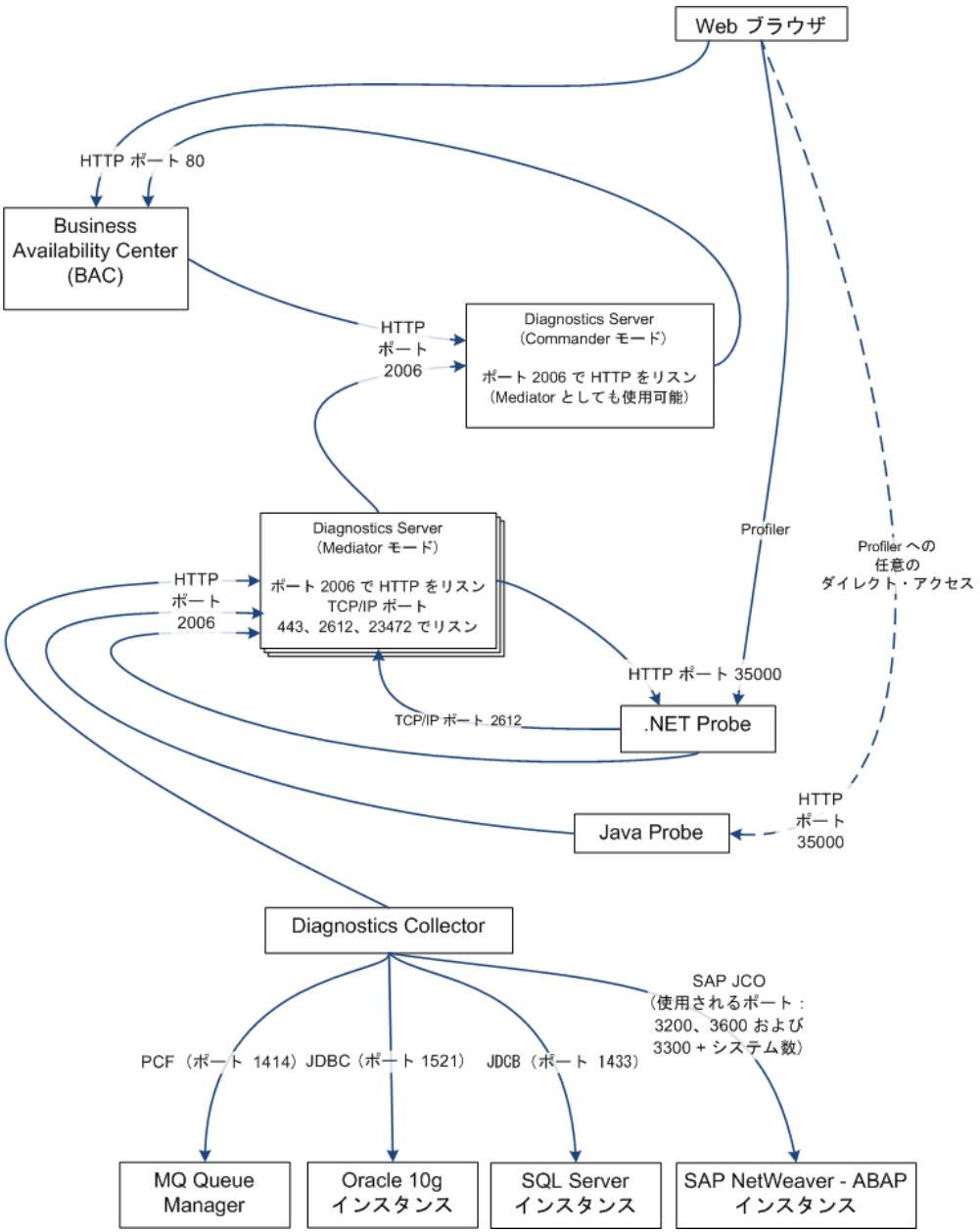
Diagnostics コンポーネントの設定と通信図

Diagnostics コンポーネントをインストールおよび設定する際に役立つコンポーネントと通信図を示します。

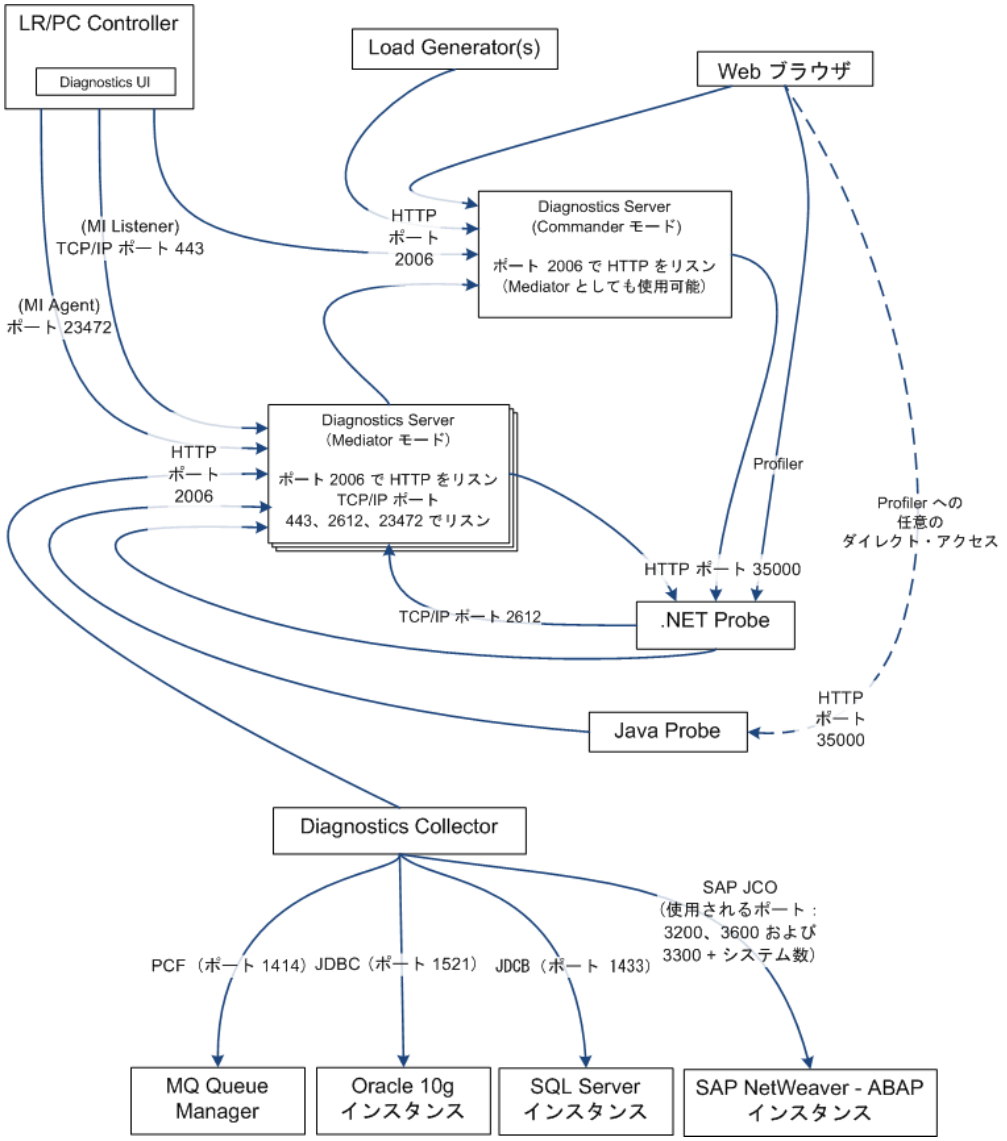
本章の内容

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 Availability Center との通信



LoadRunner および Performance Center との通信



付録 G

Diagnostics およびその他の HP ソフトウェア製品のアップグレード

Diagnostics およびその他の HP ソフトウェア製品の現在のデプロイメントをアップグレードして，Diagnostics 8.00 に含まれるコンポーネントを使用する方法について説明します。

本章の内容

製品のアップグレード方針の概要

以下の手順は，現在使用している HP ソフトウェア製品をアップグレードして，HP Diagnostics 8.00 の新機能を活用する方法を理解する上で役に立ちます。

ここに記載されている情報について質問がある場合は，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。

アップグレードに関する全般的な推奨事項

通常，以下の推奨事項は，Diagnostics の以前のバージョンから，または Diagnostics と連動しない製品のバージョンからアップグレードする際に適用されます。

- コンポーネントの以前のバージョンで使用していたものと同じホストに最新バージョンのコンポーネントをインストールする前に，ホストが新しいコンポーネントのシステム要件を満たしていることを確認します。
- Probe をアップグレードする前に，Diagnostics Server をアップグレードする必要があります。
- Business Availability Center または Peformance 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 と以前の Diagnostics バージョンとの互換性

Diagnostics Server のバージョン 8.00 は，次に示す以前のバージョンの Probe と連携して動作します。

- Java Probe 7.0, 7.50
- .NET Probe 7.0，7.50
- Collector 7.0，7.50

Diagnostics コンポーネントの Diagnostics 8.00 へのアップグレード

ここでは，既存の Diagnostics コンポーネントを Diagnostics 8.00 にアップグレードするための手順について説明します。

本項の内容

Diagnostics Server のアップグレード

このセクションでは，Diagnostics Server を 7.0/7.50 から 8.00 にアップグレードする手順を示します。

注：

- Diagnostics Server 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 Server 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 Server では，同じ Diagnostics バージョンを実行する必要があります。
- 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。

重要：アップグレード中，キーストアは JRE と一緒に変更されます。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 Server 7.0/7.50 から Diagnostics Server 8.00 にアップグレードするには

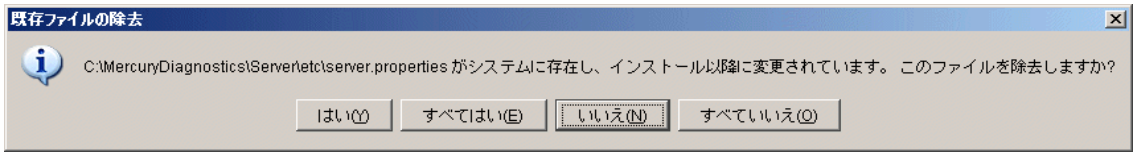
1 現在の Diagnostics Server をシャットダウンします。

2 現在の Diagnostics Server ディレクトリのバックアップ・コピーを作成します。デフォルトのディレクトリは Windows では C:¥MercuryDiagnostics¥Server，UNIX では /opt/MercuryDiagnostics/Server ですが，Server 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可能性もあります。

アップグレード手順では現在の Diagnostics Server をアンインストールしなければならないため，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 etc.old ディレクトリ（Windows の場合は C:¥MercuryDiagnostics¥Server¥etc.old，UNIX の場合は /opt/MercuryDiagnostics/Server/etc.old）がすでに存在する場合は，削除します。

4 現在の Diagnostics Server 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必ず保存してください。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 Server のアンインストールに関する詳細については，第 2 章「Diagnostics Server のインストール」を参照してください。

5 新しい Diagnostics Server を，以前のバージョンの Diagnostics Server 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 Server に対して，以前の Diagnostics Server 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。

6 Windows でインストールする場合は，Diagnostics Server を停止します。

Windows で，インストーラが完了すると，Diagnostics Server は自動的に起動します。UNIX では，Server が自動的に起動しないため，停止する必要はありません。

7 インストールにより，以前の Diagnostics Server のデータである **¥etc.old** ディレクトリが作成されています。インストーラが完了したら，**¥etc** ディレクトリと **¥etc.old** 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することができます。この場合，diff/merge ツールを使うと便利です。

¥etc.old 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，それらの相違が失われないようにします。以下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 Server 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ります。
server.properties	タイムアウト / 除外設定
thresholds.configuration	すべての変更を更新します

8 システムが LoadRunner または Performance Center と統合されている場合，**¥etc.old** 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。

9 Diagnostics Server（Commander モード）をアップグレードしている場合，**¥etc.old** ディレクトリから新しい **¥etc** ディレクトリに **DiagnosticsServer.lic** ファイルをコピーします。

10 Diagnostics Server を起動します。

11 Diagnostics UI にアクセスする前に，ブラウザのキャッシュを消去して，ブラウザを再起動します。

12 システムの状況グラフでバージョンをチェックして，アップグレードした Diagnostics Server が実行していることを確認できます。
URL http:// ＜ Commanding サーバ＞ : ＜ポート＞ /registrar/health にアクセスします。Diagnostics Server アイコンをダブルクリックします。アップグレードが正常に完了し，Diagnostics Server が再起動すると，［設定］のバージョンが 8.00.x.x になります。

13 Diagnostics Server 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：以前のバージョンの Diagnostics で作成したカスタム・ビューを Diagnostics 8.00 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 Probe のアップグレード

注：Diagnostics Server は下位互換性がないため，接続されている Probe をアップグレードする前に Diagnostics Server をアップグレードする必要があります。

Java Probe 7.0/7.50 を 8.00 にアップグレードするには

注：指示に従って起動スクリプトを更新して新しい Probe を起動し，アプリケーションを再起動するまで，新しい Probe 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 現在のプローブのインストール・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。

リリース 7.50 以降では，Java Probe と TransactionVision Sensor の機能は **Diagnostics/TransactionVision Java Agent** に結合されています。Agent の新しい構造により，7.50 より前のリリースで使われていたディレクトリは使用できません。

インストール時には以下の点を確認してください。

- Java Agent を Diagnostics Server と一緒に動作する Diagnostics Probe として設定するか，スタンドアロンの Diagnostics Profiler として設定する。必要であれば，Java Agent を TransactionVision Sensor として設定することもできます。
- Agent 名には，以前の Probe によって使用された Probe 名と同じ名前を使用する。

- ➤ Agent グループ名には，以前の Probe によって使用されたグループ名と同じ名前を使用する。
- ➤ メディエータ・サーバ名とポートには，以前の Probe によって使用された情報と同じ情報を使用する。

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 Probe によってキャプチャされた測定値と確実に適合します。

2 インストーラは，新しいインストール・ディレクトリに **＜ Probe 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，＜ Probe のインストール・ディレクトリ ＞ のデフォルトのディレクトリは変更されています。デフォルトの場所は，Windows の場合は C:¥MercuryDiagnostics¥JavaAgent¥DiagnosticsAgent，UNIX の場合は /opt/MercuryDiagnostics/JavaAgent/DiagnosticsAgent です。

3 新しいプローブの **¥etc** ディレクトリと以前のプローブの **¥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できます。

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 **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 **＜ Probe 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 ー console**（UNIX の場合は **.sh**）コマンドを実行します。

```
runPropertyScanner ー console ー diffOnly yes ー Source1 ..¥..¥..¥etc ー
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
```

以前のプローブの **¥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プローブの **¥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。以下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 Server にコピーする設定プロパティ
auto_detect.points	アップグレード方法は以下の手順を参照してください。
capture.properties	深さとレイテンシの除外
inst.properties	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。
dispatcher.properties	➤ **minimum.sql.latency** - これは，Diagnostics Mediator プロパティ **server.properties/sql.latency.trim** に関連付けて設定される新しいプロパティです。 ➤ 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 7.0 からアップグレードする場合は，アプリケーションの起動スクリプトを更新して，アップグレードしたプローブ・インストールを指定します。これには，-javaagent または -Xbootclasspath が含まれます。

7.50 からアップグレードする場合は，既存の JavaAgent ディレクトリの名前を JavaAgent_7.5 に変更し，新しい Agent のディレクトリの名前を JavaAgent に変更します。こうすることで，アプリケーションの起動スクリプトを更新せずに済みます。

5 承認時に，古い Probe 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プローブ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消去して，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 システムの状況でバージョンをチェックして，アップグレードした Java Probe が実行しており，正しく登録されていることを確認できます。システムの状況 http:// ＜ Commanding サーバ＞ : ＜ポート＞ /registrar/health にアクセスして，Probe アイコンをダブルクリックします。アップグレードが正常に完了し，Java Probe が起動すると，［設定］のバージョンが 8.00.x.x になります。

9 すべてのアプリケーションが 8.00 にマイグレートされ，すべて正しく作動していれば，以前のバージョンの Java Probe をアンインストールできます。

.NET Probe のアップグレード

注： Diagnostics Server は下位互換性がないため，接続されている .NET Probe をアップグレードする前に Diagnostics Server をアップグレードする必要があります。

注： 7.50 以降の .NET Agent には SOAP Extension Handler が含まれており，.NET Agent のインストール時に，SOAP を使った Web アプリケーションの再起動が必要になることがあります。

.NET Probe のアップグレード・プロセスは，アップグレードしている Probe のバージョンによって異なります。

.NET Probe 7.0/7.50 を 8.00 にアップグレードするには

1 新しい Diagnostics Probe for .NET をインストールします。

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

a 現在の .NET Probe 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 Probe 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 システムの状況でバージョンをチェックして，アップグレードした .NET Probe が実行しており，正しく登録されていることを確認できます。
http:// ＜ Commanding サーバ＞ : ＜ポート＞ /registrar/health にアクセスして，Probe アイコンをダブルクリックします。アップグレードが正常に完了し，.NET Probe が起動すると，［設定］のバージョンが 8.00.x.x になります。

Diagnostics Collector のアップグレード

注：Diagnostics Server 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 Server をアップグレードする必要があります。

Diagnostics Collector 7.0/7.50 を 8.00 にアップグレードするには

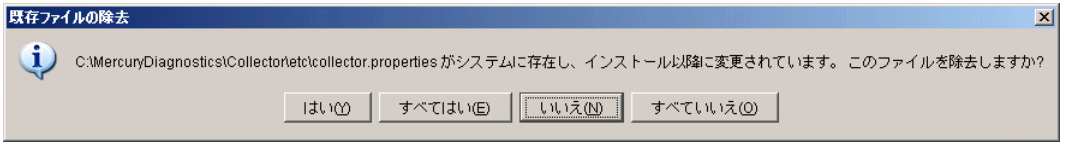
1 アップグレードする Diagnostics Collector を停止またはシャットダウンします。

2 現在の Collector のインストールのディレクトリをバックアップします。

デフォルトでは，このディレクトリは Windows では C:¥MercuryDiagnostics¥Collector，UNIX では /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しなければならないため，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必ず保存してください。たとえば，Windows の場合は次のプロンプトが表示されます。



Collector アンインストールに関する詳細については，97 ページ「Diagnostics Collector のアンインストール」を参照してください。

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測定値と適合するように，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 インストーラが完了したら，**¥etc** ディレクトリと古い **¥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（この場合，diff/merge ツールを使うと便利です）。

古い **¥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 **¥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 Server 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。 sqlserver-config.xml ファイルから databaseName 属性を削除する必要があります。Collector 8.00 では，databaseName は自動的に検出されます。

プロパティ・ファイル	新しい Diagnostics Server にコピーする設定プロパティ
r3config.xml	Collector が R3 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 システムの状況グラフで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。http:// ＜ Commanding サーバ＞：＜ポート＞ /registrar/health にアクセスして，Collector アイコンをダブルクリックします。アップグレードが正常に完了し，Collector が起動すると，［設定］のバージョンが 8.00.x.x になります。

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

次の表は，Diagnostics 8.00 コンポーネントとほかの HP ソフトウェア製品の互換性を示します。

バージョン互換性の最新情報については，Diagnostics の製品可用性マトリックス（http://support.openview.hp.com/sc/support_matrices.jsp）を参照してください。

HP ソフトウェア製品	Diagnostics 8.00 と互換性のあるバージョン
LoadRunner	9.10, 9.50
Performance Center	9.10, 9.50
Business Availability Center	7.0, 7.50, 8.00
TransactionVision	8.00

付録 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

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 **/var/sadm/pkg/IS*** および **/var/sadm/pkg/MERQ** を削除します。

Java Probe が正常に作動しない

Java Probe が正常に作動しない場合，インストール・プロセス中に，**＜ Probe 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，JRE Instrumenter を実行して作成します。詳細については，第 6 章「JRE Instrumenter の実行」を参照してください。

Java Probe の過度なスロットリング

Probe のログを見直して，スロットリングがいつ開始して，いつ停止するのかを確認することができます。予想よりもスロットリングの実行時間が長い場合，Probe のスレッド設定が，アプリケーション・サーバに設定されているスレッドの数と同期していることを確認する必要があります。アプリケーション・サーバに，Probe で処理できる数を上回るスレッドがある場合，スロットリングはただちに作動し，解除されないことがあります。

Probe を調整してスロットリングを低減する方法については，393 ページ「Probe のスロットリングの制御」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ゥ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 ＜ **Probe 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 Probe 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，一定の期間に（デフォルトで 10m）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。この警告はほとんど発行されず，トランザクションがタイムアウトになったときだけ発行されます。この警告の後，UI にトランザクション・データが表示されるはずです。ELT の詳細は，367 ページ「Diagnostics Server 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，BAC または MMS 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 Availability Center または SaaS で BPM が実行していること，および BPM が実行中であることを確認します。
2. BPM コンソールのプロファイルのステータスを確認します。

緊急時のリザーブ・バッファの消費

症状：

一部の Diagnostics データが消失し，次のエラーが Probe ログ・ファイルに記録されます。

"Emergency reserve buffers exhausted. Application Threads will begin dropping events."

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，アプリケーションのスレッド数が多すぎるか，アプリケーションが過剰にインストゥルメントされていることを示します。

解決方法：

アプリケーションが大量のスレッドを使用している場合は，**capture.properties** への次の変更を解決策として試してみることができます。

- 次のプロパティでバッファ数を 2 倍にする

  maximum.private.buffer.count

  gentle.reserve.buffer.count

  hard.reserve.buffer.count

- 次のプロパティでバッファのサイズを 50% 減らす

  maximum.buffer.size

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 Probe 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 Probe 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 Sensor 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 Sensor 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 Sensor のみ）間の違いを報告する Property Scanner レポート
- Probe または Sensor 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

1. **＜ Diagnostics Probe のインストール・ディレクトリ＞ ¥contrib¥ JASMUtilities¥Snapins** に移動します。
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。

付録 I

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。選択するオプションをすべて選択したら，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。
- 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。

付録 J

正規表現の使用

キャプチャ・ポイント・ファイルに各 Probe インスタンスのインストゥ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することができます。

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。

デフォルトで，Diagnostics では，ピリオド（.），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。

本章の内容

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

円マーク文字の使用

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（690 ページ「1 文字の検索」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。

例：

- 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 1 文字を検索するように指示します（¥n を除く）。次に例を示します。

```
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，692 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，692 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，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。次に例を示します。

```
[^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。次に例を示します。

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。次に例を示します。

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，直前の文字がゼロ個以上のものを検索するように指示します。次に例を示します。

```
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。次に例を示します。

```
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。

例：

```
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 **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。次に例を示します。

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。

例：

¥w* は，Diagnostics に，英数字 － **A-Z**，**a-z**，**0-9** および下線（**_**）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。

例：

¥w{3} は，Diagnostics に，英数字 － **A-Z**，**a-z**，**0-9** および下線（**_**）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。

例：

¥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，'.' および '*' 文字を結合して，任意の文字がゼロ個以上あるものを検索することができます（¥n 以外）。

例：

```
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。次に例を示します。

```
[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)
```

付録 K

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 Availability Center 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。

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することができます。

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 の表示方法を指定するには，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 Server や同じ Diagnostics Server へのほかのユーザ・アクセスには影響しません。

Diagnostics UI を特定の言語で表示するには

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・マシンのデフォルト言語が使われます。

たとえば，ローカル・マシンのデフォルト言語が英語で，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/jp-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 を表示する言語を選択します。

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。

付録 L

データのエクスポート

Diagnostics で収集した測定データを，サード・パーティのデータベースへ直接アーカイブして法的な目的で必要に応じて保存したり，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，651 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，Commanding Diagnostics Server からアクセスできる SQL Server 2005，SQL Server 2008，または Oracle 10g データベースを使用できます。

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証明書を指定する必要があります。

タスク 2：エクスポートする測定値の決定

いくつかの方法で，エクスポートする測定値を制御できます。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。そのグループから特定の測定値を除外したり，特定の測定値だけを含むように指定したりできます。

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 Probe グループのすべての Probe の測定値。

- /probegroup/probe/fragment

  すべての Probe グループと Probe のすべてのサーバ要求の測定値。

- /probegroup/index[name='rollup_fragment']/fragment

  すべての Probe グループの Probe によってロールアップされたサーバ要求の測定値。

- /probegroup/probe/index[name='services']/service

  すべての Probe グループと Probe 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

  すべてのホストの測定値（さまざまなシステム測定値）。

- /txn

  すべての BPM トランザクションの測定値。

注：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。

次の表は，測定値とそのカテゴリを示します。

これらの測定値の詳細については，『**Diagnostics User's Guide**』（英語版）の「Appendix A」を参照してください。

カテゴリ	測定値
Classes	Classes Currently Loaded Classes Loaded Classes Unloaded
Dynamic Caching	Caching Current Cache Size Caching Max Cache Size

カテゴリ	測定値
EJB	EJB Activates EJB Activation Time EJB Committed Transactions / sec EJB Concurrent Active Methods EJB Concurrent Live Beans EJB Create Time EJB Creates EJB Drain Size EJB Drains From Pool EJB Frees EJB Gets Found EJB Gets From Pool EJB Instantiates EJB Load Time EJB Loads EJB Passivates EJB Passivation Time EJB Passive Beans EJB Pools Size EJB Ready Beans EJB Remote Time EJB Removes EJB Response Time EJB Returns Discarded EJB Returns To Pool EJB Rolled Back Transactions / sec EJB Store Time EJB Stores EJB Timed Out Transactions / sec EJB Total Method Calls EJB-Cache Access / sec EJB-Cache Beans Cached EJB-Cache Get Failures / sec EJB-Pool Access / sec EJB-Pool Available Beans EJB-Pool Beans in Use EJB-Pool Current Waiters EJB-Pool Get Failures / sec EJB-Pool Get Timeouts / sec

カテゴリ	測定値
Execute Queues	Execute Queues Idle Threads Execute Queues Pending Requests Execute Queues Requests / sec Execute Queues Total Threads
GC	GC Collections/sec GC Time Spent in Collections
Http Status	5xx-6xx
J2C Connections	J2C Connection Handles J2C Connection Released J2C Connections Allocated J2C Connections Closed J2C Connections Created
JDBC	JDBC Connections Created/sec JDBC Create Connection Delay JDBC Current Capacity JDBC Execute Statement JDBC Leaked Connections JDBC Reconnect Failures JDBC Requests Waiting for Connection JDBC Statement Cache Accesses / sec JDBC Statement Cache Hits / sec JDBC Statement Cache Size JDBC Total Connections Opened JDBC Wait Seconds High

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測定値を指定します。

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 Server 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。

また，頻度の回復期間も指定できます。回復期間は，Commanding Diagnostics Server 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 Commanding Diagnostics Server が利用できなくなる場合にかぎり使われます。この値は，Commanding Diagnostics Serve 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 Commanding Diagnostics Server 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，Commanding Diagnostics Server 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測定値はすでにより大きなバケットに集計されているため，返される測定値には意味がありません。

しかし，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，Commander モードで実行する Diagnostics Server の **＜ Diagnostics Server 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 Server 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 Server 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，92 ページ「パスワードの難読化」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。デフォルトでは 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，デフォルト値に調節する必要があります。

6 エクスポートする測定値セットごとに，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なければなりません。

<frequency>：クエリを実行する頻度を指定する文字列値。オプションは，1h, 5m および 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 Probe グループ内のすべての Probe 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 Probe グループの 1 時間ごとのロールアップ・フラグメントのレイテンシの測定値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>
```

7 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，付録 J「正規表現の使用」を参照してください。

8 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"/>
```

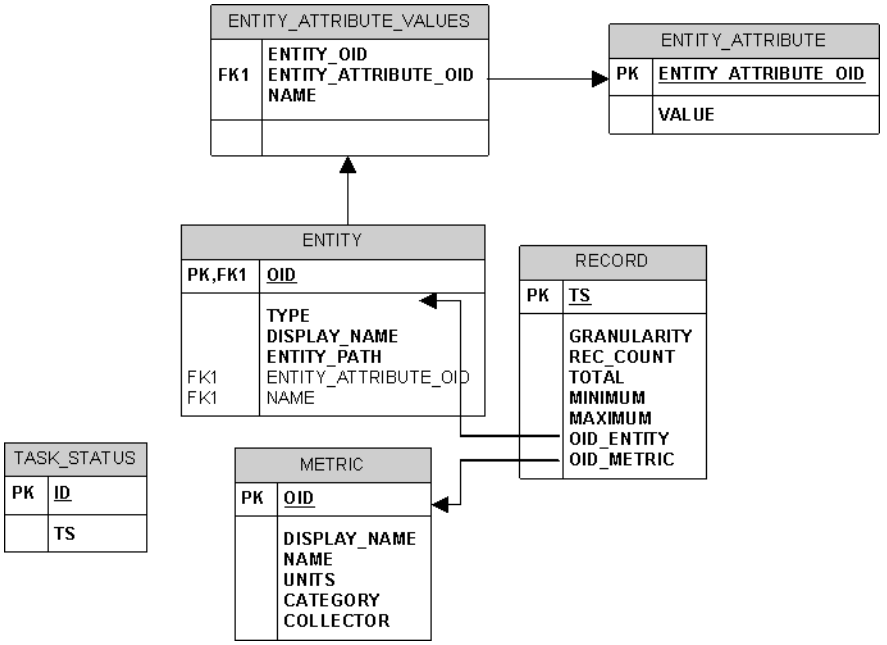
9 **data-export-config.xml** ファイルへの変更を保存します。

タスク 5：データ・エクスポート操作の監視

対応する Commanding Diagnostics Server 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。Commanding Diagnostics Server 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 Server 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。

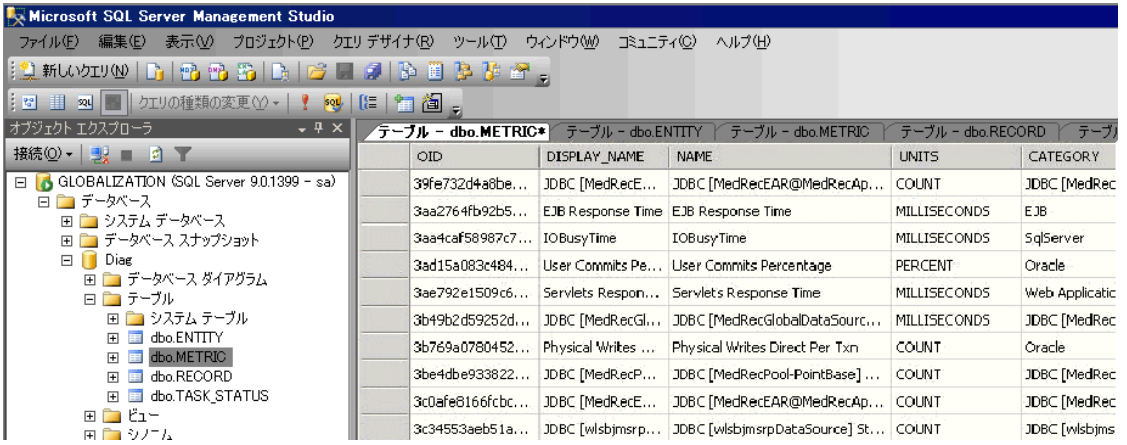
3 予定されているクエリが実行すると，クエリの結果は次のようにデータベース・テーブルに保存されます。

- Probe，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。

索引