CA Application Performance Management

for Oracle WebLogic Server ガイド

リリース 9.5

CA Technologies 製品リファレンス

このドキュメントは、以下の CA Technologies 製品および機能に関するものです。

- CA Application Performance Management （CA APM）
- CA Application Performance Management ChangeDetector （CA APM ChangeDetector）
- CA Application Performance Management ErrorDetector （[assign the value for wed in your book]）
- CA Application Performance Management for CA Database Performance （CA APM for CA Database Performance）
- CA Application Performance Management for CA SiteMinder® （CA APM for CA SiteMinder®）
- CA Application Performance Management for CA SiteMinder® Application Server Agents （CA APM for CA SiteMinder® ASA）
- CA Application Performance Management for IBM CICS Transaction Gateway （CA APM for IBM CICS Transaction Gateway）
- CA Application Performance Management for IBM WebSphere Application Server （CA APM for IBM WebSphere Application Server）
- CA Application Performance Management for IBM WebSphere Distributed Environments （CA APM for IBM WebSphere Distributed Environments）
- CA Application Performance Management for IBM WebSphere MQ （CA APM for IBM WebSphere MQ）
- CA Application Performance Management for IBM WebSphere Portal （CA APM for IBM WebSphere Portal）
- CA Application Performance Management for IBM WebSphere Process Server （CA APM for IBM WebSphere Process Server）
- CA Application Performance Management for IBM z/OS® （CA APM for IBM z/OS®）
- CA Application Performance Management for Microsoft SharePoint （CA APM for Microsoft SharePoint）
- CA Application Performance Management for Oracle Databases （CA APM for Oracle Databases）

- CA Application Performance Management for Oracle Service Bus （CA APM for Oracle Service Bus）
- CA Application Performance Management for Oracle WebLogic Portal （CA APM for Oracle WebLogic Portal）
- CA Application Performance Management for Oracle WebLogic Server （CA APM for Oracle WebLogic Server）
- CA Application Performance Management for SOA （CA APM for SOA）
- CA Application Performance Management for TIBCO BusinessWorks （CA APM for TIBCO BusinessWorks）
- CA Application Performance Management for TIBCO Enterprise Message Service （CA APM for TIBCO Enterprise Message Service）
- CA Application Performance Management for Web Servers （CA APM for Web Servers）
- CA Application Performance Management for webMethods Broker （CA APM for webMethods Broker）
- CA Application Performance Management for webMethods Integration Server （CA APM for webMethods Integration Server）
- CA Application Performance Management Integration for CA CMDB （CA APM Integration for CA CMDB）
- CA Application Performance Management Integration for CA NSM（CA APM Integration for CA NSM）
- CA Application Performance Management LeakHunter （CA APM LeakHunter）
- CA Application Performance Management Transaction Generator （CA APM TG）
- CA Cross-Enterprise Application Performance Management
- CA Customer Experience Manager （CA CEM）
- CA Embedded Entitlements Manager （CA EEM）
- CA eHealth® Performance Manager （CA eHealth）
- CA Insight™ Database Performance Monitor for DB2 for z/OS®
- CA Introscope®
- CA SiteMinder®
- CA Spectrum® Infrastructure Manager （CA Spectrum）

- CA SYSVIEW® Performance Management （CA SYSVIEW）

CA への連絡先

テクニカル サポートの詳細については、弊社テクニカル サポートの Web サイト（http://www.ca.com/jp/support/）をご覧ください。

目次

第 1 章: はじめに

このセクションには、以下のトピックが含まれています。

はじめに

CA APM for Oracle WebLogic Server は Oracle Corporation と共同開発した CA APM 拡張機能です。 この拡張機能は、実運用の WebLogic Server 環境に高度なパフォーマンス管理機能を提供します。

注: このガイドでは、CA APM for Oracle WebLogic Server を「拡張機能」と表現する場合があります。この製品は以前は「PowerPack」と呼ばれていました。

CA APM for Oracle WebLogic Server はクリティカルな WebLogic リソースを監視し、アプリケーション ボトルネックの切り分けを支援し、WebLogic Server とアプリケーションの可用性を向上させます。

このガイドの使用方法

このガイドは、CA APM for Oracle WebLogic Server の設定および使用の手順について説明しています。

注: システム要件については、「*Compatibility Guide*」を参照してください。

高度なレベルでは、このガイドは以下の情報について説明しています。

- CA APM for Oracle WebLogic Server のインストールおよび設定 (P. 11)。
- CA APM for Oracle WebLogic Server の使用 (P. 23)。
- WebLogic Server のメトリック (P. 31)。

第 2 章: 拡張機能のインストールおよび設定

この章では、CA APM for Oracle WebLogic Server を CA APM デプロイ環境にインストールし、設定する方法について説明します。

このセクションには、以下のトピックが含まれています。

開始する前に

CA APM for Oracle WebLogic Server をインストールする前に、以下の準備作業を行います。

次の手順に従ってください:

1. 使用する環境に、WebLogic Server のサポートされているバージョンがインストールされていることを確認します。

   注: サポートされているバージョンの完全なリストについては、「CA APM Compatibility Guide」を参照してください。「*CA APM Compatibility Guide*」には、CA APM コンポーネントのソフトウェアとハードウェアの互換性リストが記載されています。 このガイドは CA サポート から入手できます。

2. CA Introscope® 環境で以下のディレクトリの場所を確認します。
   - **アプリケーション サーバ ホーム ディレクトリ** — アプリケーション サーバのホーム ディレクトリ。これ以降 *<AppServer_Home>* と表記します。
   - **APM ホーム ディレクトリ** — Enterprise Manager サーバの APM のインストール ディレクトリ。これ以降 *<APM_Home>* と表記します。
   - **APM エージェント ディレクトリ** — アプリケーション サーバの APM エージェントのインストール ディレクトリ。これ以降 *<Agent_Home>* と表記します。

3. WebLogic Server 環境に、サポートされている Java 仮想マシン（JVM）のバージョンがインストールされていることを確認します。

   注: CPU の使用率を減らすには、以下のように cacheconnection プロパティを設定します。
   `introscope.agent.sqlagent.cacheConnectionsURLs=true`

4. CA APM for Oracle WebLogic Server をインストールするアプリケーション サーバのインスタンスをシャットダウンします。

重要: このインストール手順では、アプリケーションをインスツルメントするために JVM AutoProbe を使用することを前提としています。 JVM AutoProbe の使用方法の詳細については、詳細については、「*CA APM Java Agent 実装ガイド*」または「*CA APM .NET Agent 実装ガイド*」を参照してください。を参照してください。。

CA APM for Oracle WebLogic Server のインストール方法

CA APM for Oracle WebLogic Server のインストールおよび設定には、以下の手順が含まれます。

1. インストーラを実行します (P. 12)。
2. エージェント インストーラを実行します (P. 13)。
3. 管理モジュールをセットアップします (P. 14)。
4. Workstation 拡張機能をセットアップします (P. 15)。
5. JMX メトリック コレクションを有効にします (P. 15)。
6. WebLogic Server で起動クラスを設定します (P. 16)。
7. メトリック エージングから増分カウンタを除外します (P. 17)。
8. （オプション） ChangeDetector を統合します (P. 19)。

インストーラの実行

CA APM インストーラを実行します。

CA APM インストーラは *<APM_Home>*¥examples¥*<Extension_Home>* ディレクトリにファイルを配置します。 ¥examples ディレクトリ内の構成に基づいて、ファイルを正しい場所にコピーします。

CA APM インストーラの実行の詳細については、「*CA APM インストールおよびアップグレード ガイド*」を参照してください。

注: 以下の説明では Microsoft Windows のディレクトリ構文を使用しています。 UNIX など、その他のインストール プラットフォームでは、「¥」を「/」に置き換えます。

インストーラは *<APM_Home>*¥examples¥*<Extension_Home>*¥ ディレクトリに以下のファイルを配置します。

.¥config¥modules¥PPWebLogicManagementModule.jar

管理モジュール。WebLogic 固有のメトリックを表示するダッシュボードおよびアラートが事前設定されています。

.¥ext¥PPWebLogicExtensionPlugins.jar

Enterprise Manager 拡張機能。JMX Aggregate メトリックを有効にします。

.¥ext¥ddtv¥PPWebLogicTypeview.xml

Workstation 拡張機能。Investigator の［App サーバ］ビューを有効にします。

エージェント インストーラの実行

拡張機能のエージェント情報をインストールして初期設定するには、エージェント インストーラを実行します。

詳細については、詳細については、「*CA APM Java Agent 実装ガイド*」または「*CA APM .NET Agent 実装ガイド*」を参照してください。を参照してください。。

注: 以下の説明では Microsoft Windows のディレクトリ構文を使用しています。 UNIX など、その他のインストール プラットフォームでは、「¥」を「/」に置き換えます。

エージェント インストーラは、アプリケーション サーバ コンピュータの *<Agent_Home>¥wily* ディレクトリに以下のファイルを配置します。

ppweblogic.pbd

WebLogic Server 用のアプリケーションのインスツルメントに必要なトレーサが含まれる ProbeBuilder ディレクティブ ファイル。このファイルは *<Agent_Home>/wily/core/config* ディレクトリにあります。

PPWebLogicJMXFilterString.txt

JMX フィルタ文字列が記述されたテキスト ファイル。このファイルは、*<Agent_Home>/common* ディレクトリにあります。

ChangeDetector-config-WebLogic.xml

ChangeDetector の設定ファイル。 このファイルは、*<Agent_Home>/common* ディレクトリにあります。

インストーラは、エージェントと拡張機能のエージェント ファイルに、Enterprise Manager のホストおよびポートの情報を設定します。

エージェントのインストール時に拡張機能を有効にすると、以下が実行されます。

- インストーラは、*ppweblogic.pbd* ファイルを *<Agent_Home>/wily/core/config* ディレクトリまたは *JVM/wily* ディレクトリにインストールします。このディレクトリは、AutoProbe に対応する Web アプリケーション サーバまたは JVM を使用して Java アプリケーションがインスツルメントされているかどうかによって決まります。
- インストーラは、（WebLogic Server ホストの *<Agent_Home>/wily/core/config* ディレクトリにある）*IntroscopeAgent.profile* ファイルの *introscope.autoprobe.directivesFile* プロパティを変更して、新しいディレクティブ ファイル *ppweblogic.pbd* をリストに追加します。 例：
  `introscope.autoprobe.directivesFile=weblogic-full.pbl,ppweblogic.pbd`

管理モジュールのセットアップ

管理モジュールをセットアップして、CA APM for Oracle WebLogic Server ダッシュボードを表示します。

次の手順に従ってください:

1. 管理モジュール ファイル PPWebLogicManagementModule.jar を *<APM_Home>*/config/modules ディレクトリにコピーします。
2. Enterprise Manager を再起動します。

   CA APM は、CA APM for Oracle WebLogic Server ダッシュボードを表示できます。

Workstation 拡張機能のセットアップ

Workstation 拡張機能を設定して、CA APM for Oracle WebLogic Server のデータが Workstation に表示されるようにします。

次の手順に従ってください:

1. PPWebLogicTypeview.xml を *<APM_Home>*/ext/ddtv ディレクトリにコピーします。
2. PPWebLogicExtensionPlugins.jar を *<APM_Home>*/ext ディレクトリにコピーします。
3. Enterprise Manager を再起動します。

JMX メトリック コレクションの有効化

拡張機能を設定して JMX メトリック コレクションを有効にできます。

次の手順に従ってください:

1. IntroscopeAgent.profile ファイルを開き、以下のプロパティの JMX Configuration の見出しの下で以下のように設定します。
   a. introscope.agent.jmx.enable -- このプロパティのコメント化を解除し、値を true に設定します。
   b. introscope.agent.jmx.name.primarykeys -- このプロパティをコメント化します。
   c. introscope.agent.jmx.name.filter -- このプロパティのコメント化を解除します。
2. デフォルト値で指定されたより多くのメトリックを収集するには、PPWebLogicJMXFilterString.txt ファイルで指定された文字列を追加します。 このファイルは、*<Agent_Home>*¥wily¥common の introscope.agent.jmx.name.filter プロパティに配置します。

   注: 推奨されるフィルタ文字列の値によって、WebLogic Server に対する主要な JMX メトリックが選択されます。その他の JMX メトリックを収集するには、推奨されるフィルタ値にカンマで区切った形式でこれらを追加します。

   重要: 文字列では大文字と小文字が区別されます。

3. *IntroscopeAgent.profile* ファイルを保存し、アプリケーションを再起動します。

   注: JMX メトリックは、WebLogic Server 管理コンソールの WLDF コンソール拡張機能のメトリックと一致します。

WebLogic Server での起動クラスの設定

JMX データを取得するには、WebLogic Server で設定された CA Introscope® の起動クラスが必要です。以前に WebLogic Server に APM 起動クラスを設定している場合は、この手順をスキップできます。

次の手順に従ってください:

1. WebLogic 管理コンソールの［Lock & Edit］ボタンをクリックします。
2. 設定するサーバの下の［Environment］ノードを展開し、［Startup & Shutdown］リンクをクリックします。

   既存の起動クラスおよびシャットダウン クラスのテーブルが表示されます。
3. テーブル内の［New］ボタンをクリックします。

   ［Configure a New Startup or Shutdown Class］ダイアログ ボックスが表示されます。
4. ［Class Type］の下にある［Startup Class］オプションをクリックし、［Next］をクリックします。
5. ［Name］フィールドに以下のように入力します。

   ```
   Introscope Startup Class
   ```
6. ［ClassName］フィールドに以下のように入力します。

   ```
   com.wily.introscope.api.weblogic.IntroscopeStartupClass
   ```
7. ［次へ］をクリックします。
8. 選択したターゲットの下で、以下を実行します。
   - 起動クラスを使用するサーバのチェック ボックスをオンにします。
   - アプリケーション サーバ インスタンスがクラスタ化されている場合は、任意のクラスタのチェック ボックスをオンにします。

9. ［Finish］をクリックします。
10. ［Activate Changes］をクリックし、アプリケーション サーバを再起動します。

    起動クラスは WebAppSupport.jar で設定および実装されます。

    重要: WebAppSupport.jar の場所を、アプリケーションの起動クラスパスに追加します。

メトリック エージングからの増分カウンタの除外

エージェントが切断されていると、メトリックはグレー表示または淡色表示されることがあります。 増分および減分カウンタのメトリックは、メトリック エージングの対象にすることはできません。

例: メトリック エージング

Session Count の初期値が 5 であると仮定します。このメトリックが指定された期間内にデータを受信せず、メトリック エージングの対象に含まれている場合は、無効になります。 カウンタは 0 に再初期化されます。 しばらく後にこのメトリックがデータを受信すると、カウンタは 0 から開始されるため、メトリックには不正確なデータが表示されることになります。

増分および減分カウンタがメトリックに含まれると情報が不正確になることがあるため、メトリック エージングには含めないようにしてください。

次の手順に従ってください:

1. *<AgentHome>*¥wily¥core¥config に移動し、IntroscopeAgent.profile ファイルを開きます。
2. introscope.agent.metricAging.metricExclude.ignore プロパティのコメント化を解除し、増分および減分カウンタが含まれるメトリックをメトリック エージングから除外します。

```
introscope.agent.metricAging.metricExclude.ignore.<X+1>=WebLogic|Servlet
Subsystem:Error Response Count
introscope.agent.metricAging.metricExclude.ignore.<X+2>=WebLogic|HTTP
Sessions Subsystem|All Sessions:Session Count
introscope.agent.metricAging.metricExclude.ignore.<X+3>=WebLogic|HTTP
Sessions Subsystem|Cookie Sessions:Session Count
```

```
introscope.agent.metricAging.metricExclude.ignore.<X+4>=WebLogic|HTTP
Sessions Subsystem|File Sessions:Session Count
introscope.agent.metricAging.metricExclude.ignore.<X+5>=WebLogic|HTTP
Sessions Subsystem|JDBC Sessions:Session Count
introscope.agent.metricAging.metricExclude.ignore.<X+6>=WebLogic|HTTP
Sessions Subsystem|Memory Sessions:Session Count
introscope.agent.metricAging.metricExclude.ignore.<X+7>=WebLogic|HTTP
Sessions Subsystem|Replicated Sessions:Session Count
introscope.agent.metricAging.metricExclude.ignore.<X+8>=WebLogic|Clustering|C
hange Event:Count
introscope.agent.metricAging.metricExclude.ignore.<X+9>=WebLogic|Clustering|F
ull State Dump:Count
introscope.agent.metricAging.metricExclude.ignore.<X+10>=WebLogic|Clustering|
Announce:Count
introscope.agent.metricAging.metricExclude.ignore.<X+11>=WebLogic|Clustering|
Peer Gone Listeners:Count
introscope.agent.metricAging.metricExclude.ignore.<X+12>=WebLogic|Clustering|
RJVM Remote Call:Error Count
introscope.agent.metricAging.metricExclude.ignore.<X13>=WebLogic|XML
Subsystem|SAX Parsers:Creation Count
introscope.agent.metricAging.metricExclude.ignore.<X+14>=WebLogic|XML
Subsystem|Document Builder:Creation Count
introscope.agent.metricAging.metricExclude.ignore.<X+15>=WebLogic|XML
Subsystem|SAX Transformer:Creation Count
```

注: このリストを使用して、IntroscopeAgent.profile ファイルのメトリック エージングの除外リストにメトリックを追加することができます。 これを行う場合は、改行を削除してください。 改行は読みやすくする目的でのみ記述されています。

このリストで、X は、IntroscopeAgent.profile ファイルの introscope.agent.metricAging.metricExclude.ignore プロパティに対する既存のシーケンス番号です。 たとえば、IntroscopeAgent.profile ファイルの introscope.agent.metricAging.metricExclude.ignore プロパティの最後のシーケンスが introscope.agent.metricAging.metricExclude.ignore.15=Thread a の場合は、X=15 になります。

オプション: CA APM ChangeDetector の統合

CA APM for Oracle WebLogic Server に CA APM ChangeDetector を統合するには、以下の手順に従います。

ChangeDetector の設定ファイルをインストールした後、CA APM ChangeDetector を使用して、WebLogic Server 用に設定されたフォルダに対する変更を監視します。

注: WebLogic Server を起動するとき、ChangeDetector によって検出されたファイルはすべて初期バージョンとしてマークされます。 確認される初期変更は実際の変更ではなく、監視対象フォルダにすべてのファイルの初期バージョンを追加したことによるものです。 この動作は設計の仕様です。

次の手順に従ってください:

1. *<Agent_Home>*/wily/common ディレクトリに移動します。
2. メトリックを Java アプリケーションに設定するか、JVM に設定するかに応じて、ChangeDetector-config-WebLogic.xml を *<AppServer_Home>*/wily または JVM/wily ディレクトリにコピーします。
3. WebLogic Server コンピュータ上にある IntroscopeAgent.profile ファイルを開き、以下のプロパティ値を設定します。ChangeDetector 設定ファイルの場所を指定します。

   ```
   introscope.changeDetector.profile=<AppServerHome/wily へのパス
   >¥ChangeDetector-config-WebLogic.xml
   ```

   Java 環境の場合の例

   ```
   introscope.changeDetector.profile=C:¥¥bea¥¥wlserver_10.3¥¥wily¥¥ChangeDetecto
   r-config-WebLogic.xml.
   ```

   UNIX 環境の場合の例

   ```
   introscope.changeDetector.profile=/usr/bin/bea/wlserver_10.3/wily/ChangeDetec
   tor-config-WebLogic.xml
   ```

   注: ドメインが複数あり、両方のドメインに ChangeDetector を設定する場合、CA Technologies では IntroscopeAgent.profile で以下のプロパティを有効にすることを推奨します。

   ```
   introscope.changeDetector.agentID=<SampleApplicationName>
   ```

   このプロパティを有効にしないと、Investigator で ChangeDetector のエージェント ID が競合する可能性があり、エージェント ログ ファイルでエラーが発生します。

4. IntroscopeAgent.profile ファイルを保存します。
5. 以下のいずれかの手順を実行して、ChangeDetector の環境を設定します。
   - 起動スクリプトの変更 (P. 20)（推奨）
   - エージェント プロファイルの変更

   これで、ChangeDetector の統合が完了しました。

起動スクリプトの変更

起動スクリプトを変更することにより、ChangeDetector の環境を設定します。

重要: 複数のエージェントに対して同じエージェント プロファイルを使用するには、エージェント プロファイルではなく起動スクリプトを変更します。 これを行わないと、同じエージェント プロファイルを使用するドメイン間で競合する可能性があります。

次の手順に従ってください:

1. CA APM for Oracle WebLogic Server によって監視される WebLogic アプリケーションを起動する .cmd ファイルまたは .bat ファイルに、以下の行を追加します。

   ```
   -DDOMAIN_HOME=<filepath1> -DAPPLICATION_HOME=<filepath2>
   ```

   <filepath1> および *<filepath2>* は、それぞれドメインまたはアプリケーションのホームへのファイル パスです。
2. .cmd ファイルまたは .bat ファイルを保存します。
3. アプリケーションを再起動します。

   注: DOMAIN_HOME プロパティと APPLICATION_HOME プロパティは、CA APM for Oracle WebLogic Server に付属している ChangeDetector の設定ファイルで事前設定されています。 これらの事前設定済みプロパティ以外のプロパティを使用する場合は、必要なプロパティを使用するように ChangeDetector の XML 設定ファイルを変更します。 ChangeDetector および ChangeDetector の XML 設定ファイルを使用する方法の詳細については、「CA APM ChangeDetector ユーザ ガイド」を参照してください。

エージェント プロファイルの変更

エージェント プロファイルを変更することにより、ChangeDetector の環境を設定します。

重要: 複数のエージェントに対して同じエージェント プロファイルを使用するには、エージェント プロファイルではなく起動スクリプトを変更します。 これを行わないと、同じエージェント プロファイルを使用するドメイン間で競合する可能性があります。

次の手順に従ってください:

1. WebLogic Server にある IntroscopeAgent.profile ファイルを開き、ファイルの末尾に以下のプロパティを追加します。

   `DOMAIN_HOME=<filepath1>`

   `APPLICATION_HOME=<filepath2>`

   <filepath1> および *<filepath2>* は、それぞれドメインまたはアプリケーションのホームへのファイル パスです。

   注: DOMAIN_HOME プロパティと APPLICATION_HOME プロパティは、CA APM for Oracle WebLogic Server に付属している ChangeDetector の設定ファイルで事前設定されています。 これらの事前設定済みプロパティ以外のプロパティを使用する場合は、必要なプロパティを使用するように ChangeDetector の XML 設定ファイルを変更します。 ChangeDetector および ChangeDetector の XML 設定ファイルを使用する方法の詳細については、「CA APM ChangeDetector ユーザ ガイド」を参照してください。

2. IntroscopeAgent.profile ファイルを保存します。
3. アプリケーションを再起動します。

拡張機能の削除

拡張機能を削除するには、PBD ファイルと、Enterprise Manager およびエージェントのファイルを手動で削除します。

次の手順に従ってください:

1. *<APM_Home>* から以下のファイルを削除します。
   - .¥config¥modules¥PPWebLogicManagementModule.jar
   - .¥ext¥ddtv¥PPWebLogicTypeview.xml
   - .¥ext¥PPWebLogicExtensionPlugins.jar
2. *wily* ディレクトリのアプリケーション サーバ コンピュータから以下のファイルを削除します。
   - PPWebLogicJMXFilterString.txt
   - ChangeDetector-config-WebLogic.xml
3. IntroscopeAgent.profile の introscope.autoprobe.directives ファイルから ppweblogic.pbd を削除します。
4. エージェントを再起動します。

第 3 章: 拡張機能の使用

この章では、拡張機能に含まれるダッシュボードとアラートについて説明します。

このセクションには、以下のトピックが含まれています。

Workstation での WebLogic Server メトリックの表示

Workstation で WebLogic Server メトリックを表示できます。

次の手順に従ってください:

1. アプリケーションを起動します。
2. Enterprise Manager を起動します。
3. Workstation を起動してログインします。
4. Investigator ツリーを開いて、メトリックを表示します。

   すべての WebLogic 固有のメトリックが Investigator ツリーの［WebLogic］ノードの下に表示されます。 Workstation は、アプリケーションが使用する WebLogic リソースに従ってメトリックを表示します。
5. タブを選択して、Investigator での情報の表示方法を変更します。 利用可能なタブ ビューは、現在 Investigator ツリーで選択されているリソースやメトリックによって変わります。

   注: 標準的なタブ ビューの詳細については、「*CA APM Workstation ユーザ ガイド*」を参照してください。

詳細:

特別なエレメントの表示

［App サーバ］タブ ビューには、拡張機能の特別なエレメントが含まれます。

これらのエレメントを表示するには、Investigator ツリーでエージェントまたは WebLogic ノードを選択します。

［App サーバ］タブ ビューには以下のメトリックが表示されます。

ライブ HTTP セッション数

Cookie、ファイル、メモリ、および JDBC セッションのパフォーマンスに関する情報を表示します。WebLogic Server の場合、ライブ HTTP セッション数は以下のメトリックで決定されます。

```
WebLogic¥|HTTP Sessions Subsystem¥|All Sessions:Session Count
```

Thread Pool Waiting Request Count

実行可能スレッドの可用性に関する情報を表示します。 WebLogic Server の場合、スレッドの可用性は以下のメトリックで決定されます。

```
JMX¥¥|(.*)¥¥|(Type=Ex|Ex)ecuteQueueRuntime(.*):PendingRequestCurrentCount
```

JDBC Connection Pool Waiting Thread Count

プールで解放されている接続の数についての情報を表示します。WebLogic Server の場合、接続の可用性は以下のメトリックで決定されます。

```
JMX¥¥|(.*)¥¥|((Type=JDBC|JDBC)DataSourceRuntime|(Type=JDBC|JDBC)ConnectionPoolRuntime)(.*):WaitingForConnectionCurrentCount
```

EJB Pool Waiting Thread Count

使用可能な EJB インスタンスの数に関する情報を表示します。WebLogic Server の場合、EJB の可用性は以下のメトリックで決定されます。

```
JMX¥¥|(.*)¥¥|(Type=EJB|EJB)PoolRuntime(.*):Waiter(Total|Current)Count
```

WebLogic Server ダッシュボードの表示

CA APM for Oracle WebLogic Server では、WebLogic Server のパフォーマンス メトリックを表示するダッシュボードが事前に設定されています。 これらのダッシュボードは、*WebLogic* から始まる名前によって、インストールされているほかのダッシュボードや標準のダッシュボードと区別されています。

Workstation ダッシュボードに表示されるアラートは、多数のパフォーマンス メトリックに対して定義されているしきい値によって変わります。環境に合わせてこれらのアラートやしきい値をカスタマイズする方法については、「*CA APM Workstation ユーザ ガイド*」を参照してください。

注: Investigator タブ ビュー内のアラートのしきい値はカスタマイズできません。

次の手順に従ってください:

1. Workstation を起動します。
2. ［Workstation］-［新規コンソール］に移動します。

   新しいコンソール ウィンドウが表示されます。
3. 新しいコンソール ウィンドウの上部にあるドロップダウン リストを使用して、任意の WebLogic Server ダッシュボードに移動します。

   注: ダッシュボードの設計、使用、およびナビゲーション方法の詳細については、「*CA APM Workstation ユーザ ガイド*」を参照してください。

WebLogic アプリケーション サーバ - 概要

［WebLogic アプリケーション サーバ - 概要］ダッシュボードは、WebLogic アプリケーション サーバの全般的な稼働状況を表示する高度なダッシュボードです。 この概要ダッシュボードは、［サーバ実行スレッドの可用性］、［JDBC 接続要求の待機数］、［ライブ HTTP セッション数］、および［EJB プール可用性］という 4 つのカテゴリの WebLogic リソースを監視します。 アラート インジケータでは、各リソースのステータスがわかりやすく表示されます。

概要ダッシュボードには、アラート インジケータの意味について説明する運用上の注意事項と修正処置が表示されます。

セカンダリ ダッシュボードのいずれかを使用して、特定のサブシステムの情報を詳しく調べることもできます。[WebLogic アプリケーション サーバ - 概要] ダッシュボードでアラート インジケータをクリックすると、関連するダッシュボードが表示されます。

WebLogic - JDBC 接続プール

[WebLogic - JDBC 接続プール] ダッシュボードには、JDBC 接続のパフォーマンスを示すグラフが表示されます。

[WebLogic - JDBC 接続プール] ダッシュボードには、以下の情報と、グラフ アクティビティについて説明する運用上の注意事項と修正処置が表示されます。

- 接続要求の待機数
- アクティブな接続数
- リークした接続数
- 合計接続数

WebLogic - HTTP セッション

[WebLogic - HTTP セッション] ダッシュボードには、Cookie、ファイル、メモリ、および JDBC セッションに関する HTTP セッション情報を表示するグラフが表示されます。 このダッシュボードには、開いているセッションの数と、セッションを作成するために必要な平均時間に関する情報が表示されます。

[WebLogic - HTTP セッション] ダッシュボードには、以下の情報と、グラフ アクティビティについて説明する運用上の注意事項と修正処置が表示されます。

- ライブ HTTP、Cookie、メモリ セッション
  - ライブ HTTP セッション数
  - Cookie セッション数
  - メモリ セッション数

- ファイル、JDBC、レプリケート セッション
  - ファイル セッション数
  - JDBC セッション数
  - レプリケートされたセッション数

WebLogic - EJB サブシステム

［WebLogic - EJB サブシステム］ダッシュボードには、EJB サブシステムのパフォーマンスを示すグラフが表示されます。

［WebLogic - EJB サブシステム］ダッシュボードには、以下の情報と、グラフ アクティビティについて説明する運用上の注意事項と修正処置が表示されます。

- EJB プール/キャッシュ/機能の平均応答時間
  - エンティティ Bean プール
  - ステートレス Bean プール
  - メッセージ駆動型 Bean プール
  - ステートフル NRU キャッシュ
  - ステートフル LRU キャッシュ
  - ステートフル Bean 読み取り/保存/レプリケート時間

WebLogic - JMS サブシステム

［WebLogic - JMS サブシステム］ダッシュボードには、JMS メッセージング システムのパフォーマンスを示すグラフが表示されます。

［WebLogic - JMS サブシステム］ダッシュボードには、以下の情報が表示されます。

- JMS アクティビティ
  - JMS 送受信のメッセージ時間
  - JMS 送受信の秒あたりのメッセージ率
  - JMS 読み取り/書き込みの現在のバイト数
  - JMS 読み取り/書き込みの累積バイト数

- JMS キュー、JMS トピック
  - JMS キュー/トピックの現在のコンシューマ数
  - JMS キュー/トピックの現在のメッセージ数
  - JMS キュー/トピックの現在のバイト数
- JMS キュー/トピックの累積メッセージ数

WebLogic - JTA サブシステム

［WebLogic - JTA サブシステム］ダッシュボードには、JTA Java Transaction API （トランザクション マネージャ）のトランザクションのパフォーマンスを示すグラフが表示されます。

［WebLogic - JTA サブシステム］ダッシュボードには、以下の情報が表示されます。

- 合計トランザクション数
- コミット数
- 中止数
- トランザクション ロールバック/ヒューリスティック
  - ロールバック合計数
  - ロールバック タイムアウト数
  - ヒューリスティック合計数

WebLogic - セキュリティ

［WebLogic - セキュリティ］ダッシュボードには、アプリケーションのセキュリティ機能に関する情報を示すグラフが表示されます。

［WebLogic - セキュリティ］ダッシュボードには、以下の情報が表示されます。

- ユーザ認証機能の平均応答時間
  - 基本ユーザ形式認証
  - ユーザの認証
  - 実行ユーザ

- 他のセキュリティ機能の平均応答時間
  - アクセス チェック
  - ロールの取得
  - 結果の裁決
  - プリンシパルの検証

付録 A: WebLogic Server メトリック

この付録では、CA APM for Oracle WebLogic Server のメトリックについて説明します。

このセクションには、以下のトピックが含まれています。

EJB サブシステム

Workstation では、以下のサブノードの下に WebLogic EJB サブシステムのメトリックが表示されます。

- ステートフル メトリック
- NRU キャッシュ メトリック
- LRU キャッシュ メトリック
- ステートレス メトリック
- エンティティ キャッシュ
- メッセージ駆動型
- エンティティ

ステートフル メトリック

Investigator ツリーの［WebLogic］-［EJB Subsystem］-［Stateful］ノードの下には、以下のメトリックがあります。

File Serialization:Average Response Time (ms)

ファイルをシリアル化する平均時間。

File Deserialization: Average Response Time (ms)

ファイルを逆シリアル化する平均時間。

Replicate Bean:Average Response Time (ms)

Bean をレプリケートする平均時間。

NRU キャッシュ メトリック

Investigator ツリーの［WebLogic］-［EJB Subsystem］-［NRU Cache］ノードの下には、最近使用されていないキャッシュに対する以下のメトリックがあります。

Get Beans from Pool:Average Response Time (ms)

NRU キャッシュ プールから Bean を取得する平均時間。

Get Beans from Pool:Average Responses Per Second

NRU キャッシュ プールから Bean を取得する頻度。

Return Beans to Pool:Average Response Time (ms)

NRU キャッシュ プールから Bean を返す平均時間。

Return Beans to Pool: Average Responses Per Second

NRU キャッシュ プールから Bean を返す頻度。

LRU キャッシュ メトリック

Investigator ツリーの［WebLogic］-［EJB Subsystem］-［LRU Cache］ノードの下には、最も長い間使われていないキャッシュに対する以下のメトリックがあります。

Get Beans from Pool:Average Response Time (ms)

LRU キャッシュ プールから Bean を取得する平均時間。

Get Beans from Pool:Average Responses Per Second

LRU キャッシュ プールから Bean を取得する頻度。

Return Beans to Pool:Average Response Time (ms)

LRU キャッシュ プールから Bean を返す平均時間。

Return Beans to Pool:Average Responses Per Second

LRU キャッシュ プールから Bean を返す頻度。

ステートレス メトリック

Investigator ツリーの［WebLogic］-［EJB Subsystem］-［Stateless］ノードの下には、以下のメトリックがあります。

Get Bean:Average Response Time (ms)

ステートレス Bean を取得する平均時間。

Get Bean:Average Responses per Second

ステートレス Bean を取得する頻度。

Return Bean:Average Response Time (ms)

ステートレス Bean を返す平均時間。

Return Bean:Average Responses per Second

ステートレス Bean を返す頻度。

エンティティ キャッシュ メトリック

Investigator ツリーの［WebLogic］-［EJB Subsystem］-［Entity Cache］ノードの下には、以下のメトリックがあります。

Get Bean:Average Response Time (ms)

エンティティ LRU キャッシュから Bean を取得する平均時間。

Get Bean:Average Responses per Second

エンティティ LRU キャッシュから Bean を取得する頻度。

Return Bean:Average Response Time (ms)

エンティティ LRU キャッシュから Bean を返す平均時間。

Return Bean:Average Responses per Second

エンティティ LRU キャッシュから Bean を返す頻度。

Message-Driven メトリック

Investigator ツリーの［WebLogic］-［EJB Subsystem］-［Message-driven］ノードの下には、以下のメトリックがあります。

Get Bean:Average Response Time (ms)

メッセージ駆動型 Bean を取得する平均時間。

Get Bean:Average Responses per Second

メッセージ駆動型 Bean を取得する頻度。

エンティティ メトリック

Investigator ツリーの［WebLogic］-［EJB Subsystem］-［Entity］ノードの下には、以下のメトリックがあります。

Get Bean:Average Response Time (ms)

エンティティ Bean を取得する平均時間。

Get Bean:Average Responses per Second

エンティティ Bean を取得する頻度。

サーブレット サブシステム メトリック

Investigator ツリーの［WebLogic］-［Servlet Subsystem］ノードの下には、以下のメトリックがあります。

Error Response Count

累積エラー数。

Proxy Services:Average Response Time (ms)

プロキシ サービスの平均時間。

HTTP Sessions

Investigator ツリーの［WebLogic］-［HTTP Sessions Subsystem］ノードの下には、以下のメトリックがあります。 メトリックは、Cookie Sessions、File Sessions、JDBC Sessions、Memory Sessions、および Replicated Sessions のサブノードで整理されます。

All Sessions:Session Count

開いている HTTP セッションの総数。

以下のセッションは Cookie セッションです。

Session Count

開いている Cookie セッションの数。

Create Session:Average Response Time (ms)

セッションを作成する平均時間。

以下のセッションはファイル セッションです。

Session Count

開いているファイル セッションの数。

Create Session:Average Response Time (ms)

セッションを作成する平均時間。

以下のセッションは JDBC セッションです。

Session Count

開いている JDBC セッションの数。

Create Session:Average Response Time (ms)

セッションを作成する平均時間。

以下のセッションはメモリ セッションです。

Session Count

開いているメモリ セッションの数。

Create Session:Average Response Time (ms)

セッションを作成する平均時間。

以下のセッションはレプリケート セッションです。

Session Count

開いているレプリケート セッションの数。

Create Session:Average Response Time (ms)

セッションを作成する平均時間。

JMS サブシステム

Investigator ツリーの［WebLogic］-［JMS Subsystem］ノードの下には、以下のメトリックがあります。

Send Message:Average Response Time (ms)

JMS メッセージを送信する平均時間。

Send Message:Average Responses Per Second

JMS メッセージを送信するレート。

Receive Message:Average Response Time (ms)

JMS メッセージを受信する平均時間。

Receive Message:Average Responses Per Second

JMS メッセージを受信するレート。

Producer Send Response:Average Response Time (ms)

メッセージ プロデューサが応答を送信する平均時間。

Producer Send Response:Average Responses Per Second

メッセージ プロデューサが応答を送信するレート。

分散デスティネーション メトリック

Investigator ツリーの［WebLogic］-［JMS Subsystem］-［Distributed Destination］サブノードの下には、以下のメトリックがあります。

Add Rate response:Average Response Time (ms)

分散デスティネーションが応答を追加する平均時間。

Add Rate response:Average Responses Per Second

分散デスティネーションがレート応答を追加するレート。

Remove Rate Response:Average Response Time (ms)

配信先が応答を削除する平均時間。

Remove Rate Response:Average Responses Per Second

分散デスティネーションがレート応答を削除するレート。

セキュリティ サブシステム

Investigator ツリーの［WebLogic］-［Security Subsystem］ノードの下には、以下のメトリックがあります。

Basic User Form Authentication: Average Response Time (ms)

基本ユーザ形式認証を確認する平均時間。

Authenticate Users: Average Response Time (ms)

ユーザを認証する平均時間。

Access Checks: Average Response Time (ms)

アクセスをチェックする平均時間。

Get Roles: Average Response Time (ms)

リソースと件名を指定してロールを取得する平均時間。

Adjudicate Results: Average Response Time (ms)

結果を裁決する平均時間。

Validate Principal: Average Response Time (ms)

プリンシパルを検証する平均時間。

Run as Users: Average Response Time (ms)

特定のユーザとしての実行に対する要求の平均時間。

クラスタ化

Investigator ツリーの［WebLogic］-［Clustering］ノードの下には、以下のメトリックがあります。

Change Event:Count

クラスタ変更イベントがすべてのリスナに対して発生した回数。

Change Event:Average Responses Per Second

クラスタ変更イベントがすべてのリスナに対して発生したレート。

Announce:Count

リモート クラスタから受信したアナウンスメントの数。

Announce:Average Response Time (ms)

リモート クラスタからのアナウンスメントを処理する平均時間。

Announce:Average Responses Per Second

リモート クラスタからアナウンスメントを受信するレート。

Full State Dump:Count

すべてのリモート クラスタに送信されたフル ダンプの数。

Full State Dump:Average Responses Per Second

すべてのリモート クラスタに送信されたフル ダンプのレート。

NAK Processing:Average Response Time (ms)

リモート クラスタからの NAK を処理する平均時間。

NAK Processing:Average Responses Per Second

リモート クラスタから NAK を受信するレート。

Peer Gone Listeners:Count

RJVM に対するピア消失リスナの数。

Peer Gone Listeners|Add:Average Response Time (ms)

ピア消失リスナを追加する平均時間。

Peer Gone Listeners|Add:Average Responses Per Second

ピア消失リスナを追加するレート。

RJVM Remote Call:Error Count

RJVM リモート呼び出しのエラーの数。

XML サブシステム

Investigator ツリーの［WebLogic］-［XML Subsystem］ノードの下には、以下のメトリックがあります。

SAX Parsers:Creation Count

作成された SAX パーサの数。

Document Builder:Creation Count

作成されたドキュメント ビルダ パーサの数。

SAX Transformer:Creation Count

作成された SAX トランスフォーマの数。

JMX メトリック

Investigator ツリーの［JMX］ノードの下には、以下のメトリックがあります。形式は、JMX¥|(.*)¥|(Type=)?<*MBean Name*>:<*Attribute Name*> です。

ServerRuntime: OpenSocketsCurrentCount

サーバで現在開いているソケットの数。

ServerRuntime: SocketsOpenedTotalCount

サーバで開かれたソケットの総数。

ExecuteQueueRuntime: ExecuteThreadCurrentIdleCount

キューに割り当てられているアイドル スレッドの数。

ExecuteQueueRuntime: PendingRequestCurrentCount

キューで待機している要求の数。

ExecuteQueueRuntime: ServicedRequestTotalCount

このキューが処理する要求の数。

JDBCDataSourceRuntime: ActiveConnectionsCurrentCount

このデータ ソースで現在使用中の JDBC 接続の数。

JDBCDataSourceRuntime: ConnectionsTotalCount

展開された時間からこのデータ ソースで作成された JDBC 接続の総数。

JDBCDataSourceRuntime: LeakedConnectionCount

リークされた JDBC 接続の数。

JDBCDataSourceRuntime: WaitingForConnectionCurrentCount

JDBC 接続を待機している接続要求の数。

JDBCDataSourceRuntime: NumAvailable

このデータ ソースで現在利用可能な JDBC 接続の数。

EJBCacheRuntime: ActivationCount

この EJB ホームのアクティブ化されている Bean の総数。

EJBCacheRuntime: CacheAccessCount

このキャッシュの Bean にアクセスした試行の総数。

EJBCacheRuntime: CachedBeansCurrentCount

この EJB ホームの EJB キャッシュにある Bean の現在の数。

EJBCacheRuntime: CacheHitCount

成功したキャッシュ アクセスの試行の数。

EJBLockingRuntime: TimeoutTotalCount

Bean に対するロックを待機してタイムアウトになったスレッドの総数。

EJBPoolRuntime: BeansInUseCount

このプールの現在使用されている Bean インスタンスの数。

EJBPoolRuntime: IdleBeansCount

このプールの現在使用されていない Bean インスタンスの数。

EJBPoolRuntime: TimeoutTotalCount

このプールの使用可能な Bean を待機してタイムアウトになったスレッドの総数。

EJBPoolRuntime: WaiterCurrentCount

未使用のプールの使用可能な Bean インスタンスを現在待機しているスレッドの数。

EJBTransactionRuntime: TransactionCommittedTotalCount

この EJB に対してコミットされたトランザクションの総数。

JMSRuntime: ConnectionsTotalCount

前回のリセット以降に、WebLogic Server に対して作成された JMS 接続の総数。

JMSRuntime: JMSServersCurrentCount

この WebLogic Server インスタンスに展開された JMS サーバの総数。

JMSPooledConnectionRuntime: NumAvailable

現在使用されていないプール内の使用可能な JMS セッションの数。

JMSDestinationRuntime: BytesReceivedCount

前回のリセット以降に、この宛先で受信したバイト数。

JMSDestinationRuntime: ConsumersTotalCount

前回のリセット以降に、この宛先にアクセスしたコンシューマの総数。

JMSDestinationRuntime: MessagesReceivedCount

前回のリセット以降に、この宛先で受信したメッセージの数。

JMSDestinationRuntime: BytesCurrentCount

この宛先に現在格納されているバイト数。

JMSDestinationRuntime: ConsumersCurrentCount

この宛先に現在アクセスしているコンシューマの数。

JMSDestinationRuntime: MessagesCurrentCount

この宛先の現在のメッセージの数。

JMSDurableSubscriberRuntime: MessagesReceivedCount

前回のリセット以降に、この恒久サブスクライバが受信したメッセージの数。

JMSDurableSubscriberRuntime: BytesCurrentCount

この恒久サブスクライバが受信したバイト数。

JMSDurableSubscriberRuntime: MessagesCurrentCount

この恒久サブスクライバで引き続き利用可能なメッセージの数。

JMSServerRuntime: BytesReceivedCount

前回のリセット以降に、この JMS サーバが受信したバイトの総数。

JMSServerRuntime: BytesCurrentCount

この JMS サーバに現在格納されているバイト数。

JMSConnectionRuntime: SessionsCurrentCount

この接続の現在のセッション数。

TransactionNameRuntime: TransactionAbandonedTotalCount

前回のリセット以降に、中止されたトランザクションの総数。

TransactionNameRuntime: TransactionCommittedTotalCount

前回のリセット以降に、コミットされたトランザクションの総数。

TransactionNameRuntime: TransactionHeuristicsTotalCount

前回のリセット以降に、ヒューリスティック状態で完了したトランザクションの総数。

TransactionNameRuntime: TransactionRolledBackTotalCount

前回のリセット以降に、ロール バックされたトランザクションの総数。

TransactionNameRuntime: TransactionRolledBackTimeoutTotalCount

前回のリセット以降に、タイムアウトになったためにロール バックされたトランザクションの総数。

TransactionNameRuntime: TransactionTotalCount

前回のリセット以降に、処理（コミット/ロール バック/ヒューリスティック）されたトランザクションの総数。

TransactionResourceRuntime: TransactionCommittedTotalCount

前回のリセット以降に、コミットされたトランザクションの総数。

TransactionResourceRuntime: TransactionHeuristicsTotalCount

前回のリセット以降に、ヒューリスティック状態で完了したトランザクションの総数。

TransactionResourceRuntime: TransactionRolledBackTotalCount

前回のリセット以降に、ロール バックされたトランザクションの総数。

TransactionResourceRuntime: TransactionRolledBackTimeoutTotalCount

前回のリセット以降に、タイムアウトになったためにロール バックされたトランザクションの総数。

TransactionResourceRuntime: TransactionTotalCount

前回のリセット以降に、処理（コミット/ロール バック/ヒューリスティック）されたトランザクションの総数。

JTARuntime: TransactionAbandonedTotalCount

前回のリセット以降に、中止されたトランザクションの総数。

JTARuntime: TransactionCommittedTotalCount

前回のリセット以降に、コミットされたトランザクションの総数。

JTARuntime: TransactionHeuristicsTotalCount

前回のリセット以降に、ヒューリスティック状態で完了したトランザクションの総数。

JTARuntime: TransactionRolledBackTotalCount

前回のリセット以降に、ロール バックされたトランザクションの総数。

JTARuntime: TransactionRolledBackTimeoutTotalCount

前回のリセット以降に、タイムアウトになったためにロール バックされたトランザクションの総数。

JTARuntime: TransactionTotalCount

前回のリセット以降に、処理（コミット/ロール バック/ヒューリスティック）されたトランザクションの総数。

Server: IdleConnectionTimeout

HTTP セッション タイムアウトの現在の値。

JMX Aggregate メトリック

Investigator ツリーの［WebLogic］-［JMX Aggregate］ノードの下には、以下のメトリックがあります。

Thread Pool: Waiting Request Count

未使用のプールの使用可能なインスタンスを現在要求しているスレッドの総数。

JDBC Connection Pool: Waiting Thread Count

未使用のプールの使用可能な接続インスタンスを現在待機しているスレッドの総数。

EJB Pool: Waiting Thread Count

未使用のプールの使用可能な Bean インスタンスを現在待機しているスレッドの総数。