CA Application Performance Management

ChangeDetector ユーザ 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 （CA APM ErrorDetector）
- 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: CA Application Performance Management ChangeDetector 概要

CA Application Performance Management ChangeDetector を使用することにより、CA Introscope はアプリケーション ファイルおよび構成の変更を監視およびレポートできるようになります。 CA APM ChangeDetector は、実運用環境にある Web アプリケーションの変更を検出して、Web アプリケーションのパフォーマンス問題の根本原因となっている変更を特定します。問題の原因となっている変更を特定したら、CA APM ChangeDetector を使用して問題を診断できます。CA APM ChangeDetector により、パフォーマンス問題の原因となっている可能性のあるコード、アプリケーション サーバの構成、および接続されているシステムの構成の変更が明らかになります。

変更は、アプリケーションのさまざまな部分で発生するため、CA APM ChangeDetector は EPAgent（Environment Performance Agent）でもサポートされています。 EPAgent を使用すると、ほぼすべてのソースからアプリケーションのパフォーマンス情報を収集できるため、ユーザ環境に固有の変更データを監視することができます。 EPAgent の詳細については、「CA APM Environment Performance Agent 実装ガイド」を参照してください。

この章では、CA APM ChangeDetector とその CA Introscope 環境での位置付けについて説明し、一般的な使用シナリオを示します。

このセクションには、以下のトピックが含まれています。

このガイドについて

このガイドでは、CA APM ChangeDetector グラフィカル ユーザ インターフェース コンポーネントのインストール、構成、展開、および使用について説明しています。

特に記載のない限り、説明の内容はすべてのプラットフォームに適用されます。 たとえば、.NET プラットフォームのみを対象にするセクションでは、以下のような注意書きがあります。

注: このセクションは .NET プラットフォームにのみ適用されます。

このガイドで使用されるディレクトリの命名規則

このガイドでは、インストール ディレクトリに対して以下の命名規則を使用します。

命名規則

<EM_Home>

説明

Enterprise Manager がインストールされているディレクトリ。 通常は、Program Files ディレクトリの下です。

命名規則

<Workstation_Home>

説明

Workstation がインストールされているディレクトリ。 通常は、Program Files ディレクトリの下です。

命名規則

<Agent_Home>

説明

CA Introscope エージェントがインストールされているディレクトリ。これは通常 CA APM エージェント ディレクトリです。

命名規則

<ProductName_Home>

説明

サードパーティ製の製品またはアプリケーションのインストール ディレクトリ。たとえば、WebLogic を使用している場合は、<WebLogic_Home> がアプリケーション サーバのインストール ディレクトリです。

命名規則

FileName<VersionNumber><Operating System or other identifier>.FileType

説明

特定の識別情報を含むファイル名。

たとえば、Unix オペレーティング システム上で CA APM ChangeDetector 8.1 の tar パッケージからファイルを展開する場合は、以下のファイルをダウンロードします。
ChangeDetector8.1.0.0.unix.tar

ただし、このガイドでは以下のように表記します。
ChangeDetector<バージョン番号>.unix.tar

CA APM ChangeDetector について

CA APM ChangeDetector は、アプリケーション環境における変更の監視に使用できる、CA Introscope の拡張機能セットです。CA APM ChangeDetector は CA Introscope に直接統合され、低いオーバーヘッドでリアルタイムに変更を検出します。 実運用環境で問題が発生した場合、CA APM ChangeDetector を使用することにより、CA Introscope ユーザはアプリケーションの変更とアプリケーションのパフォーマンスの変化を関連付けて、問題の原因となっている変更を切り分けることができます。

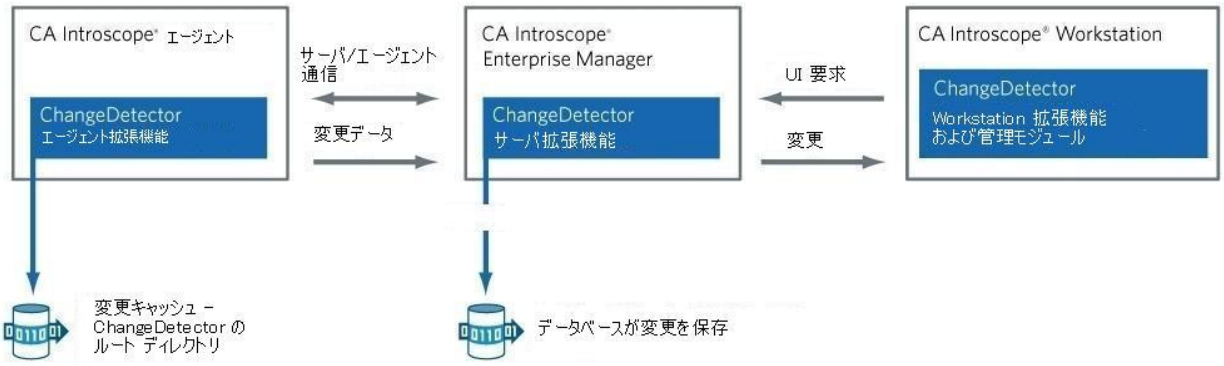
CA APM ChangeDetector は、エージェント内のアプリケーションのコード、構成、および環境の違いを、複数の期間にわたってレポートします。これにより、アプリケーションのインスタンスが、今日と昨日、または今日と先週、のように期間中にどのように異なっているかを把握できます。

CA APM ChangeDetector をインストールすると、以下を表示できます。

- ファイル（テキストおよびバイナリ）、アーカイブ、システム プロパティ、アプリケーション コード、データベース テーブル、特定のデータベース クエリ結果セットなどの変更イベントのグラフィカル ビュー。
- 変更が検出された時刻、変更の性質、テキスト ファイルのバージョンの差異などの、変更イベントの詳細情報。
- 変更データの概要が表示される CA APM ChangeDetector ダッシュボード。
- 変更の階層化ビューおよびヒストリカル ビュー（今日、昨日、先週）。
- アプリケーション内の変更を要約した Change Report。

CA APM ChangeDetector およびお使いの CA Introscope 環境

以下の図に、CA APM ChangeDetector コンポーネントがどのように CA Introscope と対話するかを示します。



CA APM ChangeDetector の使用シナリオ

以下のシナリオでは、CA APM ChangeDetector を使用して変更を検出し、変更を問題と相関させ、変更の内容を確認して、問題の修正方法を決定する様子を描いています。

問題の原因となっている変更の切り分け: 月曜朝の憂うつ

月曜日の朝 9 時、銀行のオンライン バンキング アプリケーションでパフォーマンス問題が発生し始めました。CA Introscope により、ログインとそのほかのクリティカルなトランザクションの応答時間が悪化し、SLA レベルに違反しているとのアラートが IT チームに報告されます。 怒ったユーザからカスタマ サービス デスクに電話がかかり始め、銀行口座にアクセスできないとの苦情が寄せられます。 コールセンターの責任者からこの問題がアプリケーション サポート グループに上げられ、アプリケーション サポート グループはただちに調査を開始します。

「何が変わったというのだ? このアプリケーションは、先週ずっと何の問題もなかったのに。」

アプリケーション サポート チームは、CA Introscope のダッシュボードを見て、パフォーマンスの落ち込みを再確認します。 次に、パフォーマンス グラフを見ます。CA APM ChangeDetector がインストールされているため、変更データが CA APM ChangeDetector のグラフに直接統合されて表示されます。 サポート チームは、グラフに表示された変更の注釈と詳細情報を確認して、現在のパフォーマンス問題が発生する前に、先週末にかけて一連の変更が行われたことに気がつきます。 より詳しくズームインすると、変更は土曜日の夜にかけてアプリケーションのメンテナンス時間中に行われたことがわかります。サポート チームは、CA Introscope のツリービューをクリックしながら移動していき、さまざまなアプリケーション インスタンスのパフォーマンス グラフを検査します。各インスタンスで同じ頃に同様の変更が行われていることが、よく似たグラフからはっきりとわかります。

問題の原因となっている可能性のある変更について、アプリケーション サポート チームはさらに詳細を調査します。 CA Introscope でアプリケーションの各コンポーネントをクリックすると、CA APM ChangeDetector ビューに各コンポーネントで発生した変更が表示されます。 アプリケーション ファイルを確認すると、37 ファイルが変更されており、そのうちの 3 ファイルが構成ファイルです。CA APM ChangeDetector の機能により、チームは、各変更が修正であり、それぞれがメンテナンス時間中に検出されたことを確認できます。 アプリケーション サーバの構成ファイルを 1 つ選択して、サーバにある現在のバージョンと変更前の旧バージョンのファイルを比較します。

原因が明らかになりました。100 に設定されていたデータベース接続プールが、10 に変更されたようです。 この情報を得て、アプリケーション サポート マネージャは開発マネージャに連絡し、その結果、タイプ ミスがあったことがわかりました。また、処理速度の低下は、データベース接続の予期しない変更によるものであることが判明しました。

問題が発生する前の変更検出: 現場作業の検出

突然、CA Introscope のアラートが鳴り出しました。 アプリケーション サポート チームに、Java アプリケーションで変更が発生したことが連絡されます。 この変更は予定外のものであるため（計画されたメンテナンス時間帯でないため）、アプリケーション サポート マネージャはパフォーマンス グラフの調査を開始します。 アプリケーションは今のところ通常どおり動作しているように見えますが、グラフから、数分前に変更がいくつか発生したことがわかります。 変更の上にカーソルを置くと、変更の詳細が表示されます。 システム変数に変更があるようです。

アプリケーション サポート マネージャは、CA Introscope Investigator で調査を開始して、さまざまなアプリケーション コンポーネントに対する変更を探し始めます。 CA APM ChangeDetector では、アプリケーションの変更がツリー表示されます。 アプリケーション サポート マネージャは、さまざまな変更の検査を進めるにつれて、アプリケーション ファイルや構成ファイルでは何も変更されていないことがわかります。APM データベースの構成でも、何も変更されていないようです。 ただし、以前の CA Introscope アラートを確認して、サポート マネージャは、JVM とアプリケーション サーバの 4 つのシステム プロパティが変わったことに気づきます。

検出された変数の変更のそれぞれをクリックすると、その変数名、変更が検出された時刻、および変更された値が表示されます。 見ると、Java ヒープ サイズを決定する変数が低くなっており、アプリケーションの負荷が高まると問題が発生する可能性があることがわかります。 UNIX システム管理者と共に確認すると、日常業務は実行されており、この変更が与える影響の可能性について認識されていないことが判明しました。 変数の変更を元の値に戻して、パフォーマンス問題が発生するのを未然に防ぎました。

第 2 章: CA APM ChangeDetector のインストールと構成

この章では、構成ウィザードを使用して、または手動で XML 構成ファイルを編集して CA APM ChangeDetector データソースを構成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。

CA APM ChangeDetector のインストールと有効化

CA APM ChangeDetector は、エージェントのインストール時にインストールおよび有効化されます。 CA APM ChangeDetector は、スタンドアロン エージェント インストーラまたはサイレント モード インストール オプションを使用してインストールできます。 CA APM ChangeDetector をインストールしたら、構成を行う必要があります。

次の手順に従ってください:

1. エージェントをインストールします。

   エージェントのインストール方法の詳細については、「CA APM .Net Agent 実装ガイド」または「CA APM Java 実装ガイド」を参照してください。CA APM ChangeDetector はデフォルトで有効になっています。

a. CA APM ChangeDetector を有効にするには、*<EM_Home>/config* ディレクトリ内の *IntroscopeEnterpriseManager.properties* ファイルを開き、以下のプロパティを *false* に設定して、Enterprise Manager を再起動します。

```
introscope.changeDetector.disable=false
```

重要: AIX 環境の WebSphere 6.1 または 7.0 と Oracle DB に対して CA APM ChangeDetector を実行する場合、以下のような予期しない例外を受け取ることがあります。

java.security.AccessControlException: Access denied (java.net.SocketPermission hostname:port connect,resolve) at java.security.AccessController.checkPermission(AccessController.java:104)

この例外が発生しないようにするには、*<WebSphere_Home>/properties/server.policy* を修正して Java セキュリティ権限を許可するか、または WebSphere 管理コンソールで［Enable application security］チェック ボックスをオフにします。

2. CA APM ChangeDetector を構成します。

CA APM ChangeDetector を構成する前に、「CA APM ChangeDetector を構成する前に (P. 21)」を参照して、さまざまな構成オプションを把握しておいてください。

注: CA APM ChangeDetector は、Windows （cdnativefile.dll）および Linux （libcdnativefile.so）オペレーティング システム用に変更されたデータと共にファイル所有者のデータをレポートするネイティブ ライブラリをインストールします。 サポートされているプラットフォームのバージョンのリストについては、「*CA APM Compatibility Guide*」を参照してください。

CA APM ChangeDetector を構成する前に

CA APM ChangeDetector を構成する前に、さまざまな構成オプションを理解したうえでお使いの環境に最も適した方法を選択してください。

- CA APM ChangeDetector 構成ウィザードを使用する（P. 23）- この方法では、構成時に表示される一連のページに情報を入力することで CA APM ChangeDetector を構成できます。
- CA APM ChangeDetector 構成ファイルを変更する（P. 36）- この方法は、すべてのプラットフォームで使用できます。 この方法では、手動で XML ファイルを編集することで CA APM ChangeDetector を構成できます。CA APM ChangeDetector には、この方法を選択した場合に参考になるサンプル XML ファイルが付属しています。

また、以下の概念についても理解する必要があります。

- 複数の CA APM ChangeDetector 構成ファイルの使用（P. 21）
- データソースについて（P. 22）

詳細:

サンプル構成ファイル (P. 95)

複数の CA APM ChangeDetector 構成ファイルの使用

単一の CA APM ChangeDetector 構成ファイルを指定する代わりに、複数の構成ファイルが含まれるディレクトリを指定できます。 ディレクトリを指定すると、CA APM ChangeDetector の起動時に、このディレクトリにある有効な構成ファイルがすべて読み込まれます。

CA APM ChangeDetector 構成ディレクトリを指定する方法

- *IntroscopeAgent.profile* を編集して、以下のプロパティを設定します。
  `introscope.changeDetector.profileDir=<ChangeDetector ディレクトリへのパス>`

  このプロパティのデフォルト値は、以下のとおりです。
  `<IntroscopeAgent.profile へのパス>/changeDetector_profiles`

  以下のプロパティに単一のファイルを指定することもできます。
  `introscope.changeDetector.profile`

  この場合、CA APM ChangeDetector には、このファイルと、指定した構成ディレクトリ内のすべてのファイルが読み込まれます。

データソースについて

データソースはリソースのグループで、データソース タイプ（たとえば、ファイル、データベース テーブルのカラム値、またはランタイム クラスのインスタンス）でその種別が定義されます。 リソースは、たとえば、*C:¥Introscope¥Introscope Enterprise Manager.exe*、データベースのカラム名 *buffer_pool* 、または *java.lang.System* のように、名前で識別されます。データソースには数多くのリソースを含めることができます。リソースに数多くの値を含めることもできますが、1 度に 1 つの値のみ可能です。

注: データソース内のリソース名は一意である必要があります。

データソースの定義には、以下の 2 つの方法のいずれかを使用します。

- CA APM ChangeDetector 構成ウィザードの使用 (P. 23)
- CA APM ChangeDetector 構成ファイルの変更 (P. 36)

CA APM ChangeDetector 対応のエージェントでは、構成ファイルで指定したデータソースが、Investigator の［変更］タブに表示されます。変更データを表示する方法の詳細については、「CA APM ChangeDetector データの表示 (P. 67)」を参照してください。

各データソースの定義は、*datasource-type* タグ内に含まれる必要があり、その属性には *name* と *class* があります。 *name* は任意の名前にすることができます。この名前は、後で構成ファイルで参照されます。 *class* 属性は、この種類のデータソースについてデータソース インスタンスの定義を解析するために使用するクラスを定義します。 クラスは、以下のインターフェースを実装する必要があります。

```
com.wily.cd.agent.config.IDataSourceConfig
```

datasource-type エレメントにより、エージェントは使用可能なデータソースの種類を判断します。*datasource-type* エレメントを最大限に活用するには、*datasource-instance* エレメントも定義します。 各エレメントは、エージェントが監視する物理的なデータソースに対応します。

注: データソース インスタンスに、同じ名前のリソースを複数含めることはできません。たとえば、同じデータソース インスタンス内に、完全パスと名前が同じファイルが 2 つある状態や、同じ名前の *javaenv* プロパティがある状態は無効です。 CA APM ChangeDetector 内のデータは、データソース インスタンス別に整理されます。

ChangeDetector 構成ウィザードの使用

CA APM ChangeDetector を構成するには、構成ウィザードを使用します。ただし、*ChangeDetector-config.xml* をアップデートして ChangeDetector を設定することもできます。「サンプル構成ファイル (P. 95)」を参照してください。

構成ウィザードでは、以下の作業を実行できます。

- 構成ウィザードの実行 (P. 23)
- 構成ウィザードによるデータソースの追加 (P. 24)
- 構成ウィザードによるデータソースの変更 (P. 25)
- 構成ウィザードによるデータソースの削除 (P. 26)

注: 複数の CA APM ChangeDetector 構成ファイルを使用できます。 詳細については、「複数の CA APM ChangeDetector 構成ファイルの使用 (P. 21)」を参照してください。

構成ウィザードの実行

構成ウィザードを使用すると、手動で XML を編集する代わりに、グラフィカル ユーザ インターフェースを使用してデータソース プロパティを設定できます。

構成ウィザードを実行する方法

1. CA APM ChangeDetector のインストール後、*<Workstation_Home>/tools* に移動し、コマンド プロンプトから *configwizard.bat* を実行して構成ウィザードを起動します。

   注: *JAVA_HOME* が設定されていない場合は、コマンド プロンプトから *configwizard.bat* を実行するときに、引数として JRE へのパスを指定します。 たとえば、*configwizard.bat s:¥sw¥sun¥jre* のように指定します。

2. 起動したら、以下の作業を行うことができます。
   - 構成ウィザードによるデータソースの追加 (P. 24)
   - 構成ウィザードによるデータソースの変更 (P. 25)
   - 構成ウィザードによるデータソースの削除 (P. 26)

構成ウィザードによるデータソースの追加

構成ウィザードを使用して、以下のデータソースを追加できます。

- ウィザードによるアセンブリ監視の設定 (P. 26)

  注: このセクションは、.NET プラットフォームにのみ適用されます。

- ウィザードによる Java クラス監視の設定 (P. 28)

  注: このセクションは、Java プラットフォームにのみ適用されます。

- ウィザードによるデータベース監視の設定 (P. 30)
- ウィザードによるファイル システム監視の設定 (P. 31)
- ウィザードによる .NET 環境変数監視の設定 (P. 34)

  注: このセクションは、.NET プラットフォームにのみ適用されます。

- ウィザードによる Java システム プロパティ監視の設定 (P. 35)

  注: このセクションは、Java プラットフォームにのみ適用されます。

データソースを追加する方法

1. 構成ウィザードを実行します (P. 23)。
2. 使用しているプラットフォームを選択して、［次へ］をクリックします。
3. ［CA APM ChangeDetector 構成ファイルを新規作成］を選択して、［次へ］をクリックします。

   ウィザードのこのページから、新しい構成ファイルの作成または既存のファイルの変更を行うことができます。

4. オプションを選択して、［次へ］をクリックします。

   作成するデータソースは、ウィザードの［現在のデータソース］セクションに表示されます。これらのデータソースはいつでも編集および削除できます。

   追加したデータソースは、［現在のデータソース］に表示されます。さらに、必要に応じて、さまざまなフィールドやエレメントを編集したり、データソースを削除したりできます。

5. 追加するデータソースの種類を選択して、[次へ]をクリックします。
6. 追加するデータソースの名前を入力して、[次へ]をクリックします。表示されるオプションは、前述の手順で指定したプラットフォームによって異なります。
   - 引き続きすべてのデータソースを構成します（「ウィザードによるデータソースの設定 (P. 26)」を参照してください）。必要な場合は、オプションの CA APM ChangeDetector Workstation および Enterprise Manager プロパティを定義します（「オプションの構成プロパティ (P. 63)」を参照してください）。
7. データソースをすべて追加したら、［名前を付けて保存］をクリックして、XML ファイルを保存します。

構成ウィザードによるデータソースの変更

データソースの追加後に、データソースを変更できます。

注: 旧バージョンからアップグレードする場合は、元のファイルのバックアップを保存してアップグレードするか、または既存のファイルに上書きできます。

データソースを変更する方法

1. 構成ウィザードを実行します (P. 23)。
2. プラットフォームを選択して、［次へ］をクリックします。
3. ［既存の ChangeDetector 構成ファイルを修正］を選択して、［次へ］をクリックします。
4. 変更する構成ファイルを選択します。
5. ウィザードの左ペインでデータソースを選択して、変更します。

   各データソースにはさまざまなフィールドとエレメントがあり、それぞれを変更できます。 変更するには、フィールドを変更するか、エレメントを選択して修正します。
6. 変更を行ったら、［保存］をクリックしてから、［終了］をクリックします。

構成ウィザードによるデータソースの削除

データソースの追加後に、データソースを削除できます。

データソースを削除する方法

1. 構成ウィザードを実行します (P. 23)。
2. 使用しているプラットフォームを選択して、［次へ］をクリックします。
3. ［選択したデータソースを削除］を選択して、［次へ］を選択します。
4. データソースを削除する構成ファイルを選択します。
5. ウィザードの左ペインでデータソースを選択して、削除します。
6. 変更をすべて行ったら、［保存］をクリックしてから、［終了］をクリックします。

ウィザードによるデータソースの設定

構成ウィザードを使用すると、以下のデータソースを設定できます。 新しい構成ファイルへのデータソースの追加が完了したら、［名前を付けて保存］をクリックして名前を付けて XML ファイルを保存します。

- ウィザードによるアセンブリ監視の設定 (P. 26)

  注: アセンブリ監視は、.NET プラットフォームにのみ適用されます。
- ウィザードによる Java クラス監視の設定 (P. 28)
- **注**: Java クラス監視は Java プラットフォームにのみ適用されます。
- ウィザードによるデータベース監視の設定 (P. 30)
- ウィザードによる .NET 環境変数監視の設定 (P. 34)

  注: .NET 環境変数監視は .NET プラットフォームにのみ適用されます。
- ウィザードによる Java システム プロパティ監視の設定 (P. 35)

  注: Java システム プロパティは Java プラットフォームにのみ適用されます。

ウィザードによるアセンブリ監視の設定

.NET プラットフォームについては、アセンブリ監視データソースを追加または変更できます。

ウィザードでアセンブリ監視データソースを設定する方法

1. 英数字を使用してデータソース名を入力し、［次へ］をクリックします。

   ウィザードの次のページが表示されます。

2. 以下のプロパティ構成情報を入力し、［次へ］をクリックします。

   - **イタレーションあたりのクラス数** - 定義されているイタレーションごとにロードされるクラス数。 数値が低いと CPU 使用率が下がりますが、スキャン時間が長くなります。
   - **イタレーション間の遅延時間** - イタレーション間でクラスが監視されない時間（秒、分、または時間単位）。 数値が高いと CPU 使用率が下がりますが、スキャン時間が長くなります。
   - **初期待機時間** - クラスが監視されるまでの初期時間（秒、分、または時間単位）。

   ウィザードの次のページが表示されます。

3. 以下の手順に従って、アセンブリおよびクラスの両方に exclude エレメントを追加します。

   a. ［exclude エレメントを新規追加］を選択します。

   b. ［次へ］をクリックします。

   c. 表示されたフィールドに除外パターンを入力します。

   d. 包含パターンを追加するには、［追加］をクリックして、表示されたフィールドにパターンを入力します。 必要に応じてこの手順を繰り返します。

   e. ［次へ］をクリックします。

   f. 上記の手順を繰り返してさらに除外パターンを追加するか、［完了］をクリックします。

   注: 1 番目の exclude エレメントにより、正規表現に一致するアセンブリが監視から除外されます。2 番目の exclude エレメントにより、正規表現に一致するクラスが監視から除外されます。
   除外パターンを使用して、スキャンの対象を絞ります。除外パターンの例外を指定するには、包含パターンを追加します。 たとえば、除外パターンに .* を指定して、包含パターンに java¥.* を指定します。
   除外または包含パターンには正規表現を使用します。 以下に例を示します。
   foo
   .*bar.
   .*
   com¥.wily¥.(.*)

詳細:

構成ウィザードによるデータソースの追加 (P. 24)
構成ウィザードによるデータソースの変更 (P. 25)

ウィザードによる Java クラス監視の設定

Java プラットフォームについては、Java クラス監視データソースを追加または変更できます。

ウィザードで java クラス監視データソースを設定する方法

1. 英数字を使用してデータソース名を入力し、［次へ］をクリックします。

   ウィザードの次のページが表示されます。

2. 以下のプロパティ構成情報を入力し、［次へ］をクリックします。

   - **イタレーションあたりのクラス数** - 数値が低いと CPU 使用率が下がりますが、スキャン時間が長くなります。
   - **イタレーション間の遅延時間** - 数値が高いと CPU 使用率が下がりますが、スキャン時間が長くなります。

   ウィザードの次のページが表示されます。

3. ［exclude エレメントを新規追加］を選択し、［次へ］をクリックします。

4. 表示されたフィールドに除外パターンを入力します。

5. 包含パターンを追加するには、［追加］をクリックして、表示されたフィールドにパターンを入力します。必要に応じてこの手順を繰り返し、［次へ］をクリックします。

6. 上記の手順を繰り返してさらに除外パターンを追加するか、［完了］をクリックします。

   注: 除外パターンを使用して、スキャンの対象を絞ります。除外パターンの例外を指定するには、包含パターンを追加します。 たとえば、除外パターンに .* を指定して、包含パターンに java¥.* を指定します。除外または包含パターンには正規表現を使用します。以下に例を示します。
   foo
   .*bar.
   .*
   com¥.wily¥.(.*)

詳細:

構成ウィザードによるデータソースの追加 (P. 24)
構成ウィザードによるデータソースの変更 (P. 25)

ウィザードによるデータベース監視の設定

ウィザードを使用して、データベース監視データソースを追加または変更できます。

ウィザードでデータベース監視データソースを設定する方法

1. 英数字を使用してデータソース名を入力し、［次へ］をクリックします。

   ウィザードの次のページが表示されます。

2. Java プラットフォームの以下のデータベース情報を入力し、［次へ］をクリックします。

   - **JDBC ドライバ** - このデータベース用のドライバ。 たとえば、oracle.jdbc.driver.OracleDriver です。
   - **JDBC ドライバ クラスパス** - データベース ドライバへのパス（パスを参照することもできます）。
   - **JDBC URL**： データベースへの JDBC URL を入力します。
   - **ユーザ名** - データベースを使用するユーザのユーザ名を入力します。
   - **パスワード** - データベースのパスワードを入力します。 パスワードは構成ファイルで自動的に暗号化されます。
   - **パスワードの確認** - 確認のため、パスワードを再入力します。
   - **スケジュール タイプ** - ［繰り返し］または［起動後］。 変更の少ないデータベース、またはアプリケーションの起動時にチェックのみされるデータベースには、起動後を選択します。
   - **繰り返し間隔** - スケジュールを繰り返しにした場合は、間隔を選択します。 数値が高いと CPU 使用率が下がりますが、スキャン時間が長くなります。
   - **イタレーション間の遅延時間** - 数値が高いと CPU 使用率が下がりますが、スキャン時間が長くなります。

   ウィザードの次のページが表示されます。

3. .NET プラットフォームの以下のデータベース情報を入力し、［次へ］をクリックします。

   - **URL** - データータベースへの URL を入力します。
   - **ユーザ名** - データベースを使用するユーザのユーザ名を入力します。
   - **パスワード** - データベースのパスワードを入力します。 パスワードは構成ファイルで自動的に暗号化されます。
   - **パスワードの確認** - 確認のため、パスワードを再入力します。

   注: .NET プラットフォームでは、ユーザ名とパスワードを URL プロパティ内に指定できますが、何も指定しなくてもかまいません。 ウィザードでは、URL 内のこれらの値はチェックされません。

   - **スケジュール タイプ** - ［繰り返し］または［起動後］。 変更の少ないデータベース、またはアプリケーションの起動時にチェックのみされるデータベースには、起動後を選択します。
   - **繰り返し間隔** - スケジュールを繰り返しにした場合は、間隔を選択します。 数値が高いと CPU 使用率が下がりますが、スキャン時間が長くなります。

   ウィザードの次のページが表示されます。

4. SQL ステートメントを追加するには、［SQL ステートメントを新規追加］を選択して、［次へ］をクリックします。

   上記の手順を繰り返して SQL ステートメントを追加するか、［完了］をクリックします。 例：

   ```
   SELECT name, value FROM v$parameter
   ```

   注: ウィザードでは、SQL ステートメントは検証されません。 データベースに有効な SQL ステートメントを使用してください。

詳細:

構成ウィザードによるデータソースの追加 (P. 24)
構成ウィザードによるデータソースの変更 (P. 25)

ウィザードによるファイル システム監視の設定

ウィザードを使用して、ファイル システム監視データソースを追加または変更できます。

ウィザードでファイル システム監視データソースを設定する方法

1. 英数字を使用してデータ ソース名を入力し、［次へ］をクリックします。

   ウィザードの次のページが表示されます。

2. ［scan-directory エレメントを新規追加］を選択し、［次へ］をクリックします。

3. ディレクトリ名と必須フィールド（［ディレクトリ名］、［再帰］、［ファイル セット］、および［有効］）を入力します。

   - **ディレクトリ名** - CA APM ChangeDetector がスキャンするディレクトリへのパスを入力します。 たとえば、*C:¥¥WebLogic¥¥myApplicationHome* のように入力します。
   - **再帰** - True に設定すると、CA APM ChangeDetector により指定したディレクトリの下のサブフォルダもスキャンされます。
   - **ファイル セット** - スキャンディレクトリに関連付けるファイル セットを選択します。
   - **有効** - テストやその他の目的のため、スキャン ディレクトリを無効にできます。 ただし、スキャン ディレクトリを有効化した後に無効化すると、このスキャン ディレクトリに含まれるファイルは、削除されたファイルとして CA APM ChangeDetector からレポートされます。

4. 除外パターンを追加するには、［以下を新規追加］ をクリックして、パターンを入力します。 必要に応じてこの手順を繰り返します。

5. ［ファイル セットを作成または修正］をクリックして、ファイルセットを追加するか、または既存のファイルセットを編集します。

   - ［ファイル セットを新規追加］を選択して、［次へ］をクリックします。
   - ファイルセット名を入力して、［次へ］をクリックします。

6. ［次へ］をクリックし、上記の手順に従って scan-directory エレメントをさらに追加します。scan-directory エレメントの追加が済んだら、［完了］をクリックします。
7. include/exclude エレメントを入力して、スキャンの対象を絞り込みます。除外パターンの例外を指定するには、包含パターンを追加します。
   - ［exclude エレメントを新規追加］を選択し、［次へ］をクリックします。
   - 表示されたフィールドに除外パターンを入力します。
   - 包含パターンを入力するには、［追加］をクリックして、表示されたフィールドに包含パターンを入力します。 必要に応じてこの手順を繰り返します。
8. 以下の情報を入力して［次へ］をクリックします。
   - **イタレーションあたりのファイル数** - 数値が低いと CPU 使用率が下がりますが、スキャン時間が長くなります。
   - **ファイル イタレーション間の遅延時間** - 数値が高いと CPU 使用率が下がりますが、スキャン時間が長くなります。
   - **アップロードの最大ファイル サイズ** - サーバにアップロードされる ASCII ファイルの最大サイズです。 アップロードされた ASCII ファイルは、CA APM ChangeDetector の Diff View で表示できます。

   注: 以下のアーカイブ プロパティは Java プラットフォームにのみ適用されます。

- **ダイジェストの使用** - ［常に使用］、［使用しない］、または［必要に応じて使用］を選択します。 ダイジェストは、MD5 のようなメッセージ ダイジェストの形式です。 CA APM ChangeDetector は、ハッシュ比較を行い、ファイル実体への変更は無視します。 このオプションは、［常に使用］を選択するとパフォーマンスに影響を与えることがあります。 ［必要に応じて使用］を選択すると、タイムスタンプとファイル サイズが変更されている場合にのみダイジェストが使用されます。
- **アーカイブの内部をスキャン** - *.zip* や *.jar* などのアーカイブ内の個々のファイルをスキャンする場合は、True を選択します。
- **イタレーションあたりのアーカイブ数** - ［アーカイブの内部をスキャン］を選択した場合は、この値を指定します。数値が低いと、スキャンにかかる時間が長くなります。 アーカイブ ファイルは全体でスキャンされるため、アーカイブ内に多数のファイルがある場合は、この値を制限する必要があります。
- **アーカイブ イタレーション間の遅延時間** - 数値が高いと CPU 使用率が下がりますが、スキャン時間が長くなります。

9. ファイル セットを追加し、［このセクションを完了］を選択して［次へ］をクリックします。

   注: 監視対象のファイル システムに対する読み取り権限が必要です。また、ファイル システムがネットワーク上にある場合、CA APM ChangeDetector がファイルを検出するには、ネットワークが正常に機能している必要があります。

詳細:

構成ウィザードによるデータ ソースの追加 (P. 24)
構成ウィザードによるデータ ソースの変更 (P. 25)

ウィザードによる .NET 環境変数監視の設定

.NET プラットフォームについては、環境変数データ ソースを追加または変更できます。

ウィザードでファイル システム監視データソースを設定する方法

1. 英数字を使用してデータ ソース名を入力し、［次へ］をクリックします。
2. ［exclude エレメントを新規追加］を選択して［次へ］をクリックし、exclude エレメントを追加します。

   表示されるフィールドに、除外パターンを入力します。 除外パターンを使用して、スキャンの対象を絞ります。除外パターンの例外を指定するには、包含パターンを追加します。 たとえば、除外パターンに .* を指定して、包含パターンに java¥.* を指定します。

   除外または包含パターンには正規表現を使用します。 以下に例を示します。

   foo

   .*bar.

   (.*)

   java¥.*
3. ［以下を新規追加］を選択して［次へ］をクリックし、包含パターンを追加します。

   表示されるフィールドに、包含パターンを入力します。
4. 必要に応じて、上記の手順を繰り返してさらに除外/包含パターンを追加します。追加が済んだら、［完了］をクリックします。

詳細:

構成ウィザードによるデータ ソースの追加 (P. 24)
構成ウィザードによるデータ ソースの変更 (P. 25)

ウィザードによる Java システム プロパティ監視の設定

Java プラットフォームについては、Java システム プロパティ監視データ ソースを追加または変更できます。

ウィザードで Java システム監視データソースを設定する方法

1. 英数字を使用してデータソース名を入力し、［次へ］をクリックします。
2. ［exclude エレメントを新規追加］を選択して［次へ］をクリックし、exclude エレメントを追加します。

   表示されるフィールドに、除外パターンを入力します。 除外パターンを使用して、スキャンの対象を絞ります。除外パターンの例外を指定するには、包含パターンを追加します。 たとえば、除外パターンに .* を指定して、包含パターンに java¥.* を指定します。

   除外または包含パターンには正規表現を使用します。 以下に例を示します。

   foo

   .*bar.

   (.*)

   java¥.*
3. ［以下を新規追加］を選択して［次へ］をクリックし、包含パターンを追加します。

   表示されるフィールドに、包含パターンを入力します。

詳細:

構成ウィザードによるデータソースの追加 (P. 24)
構成ウィザードによるデータソースの変更 (P. 25)

CA APM ChangeDetector 構成ファイルの変更

XML 構成ファイルを編集して、手動で設定を変更できます。この方法は、すべてのプラットフォームで使用できます。CA APM ChangeDetector には、この方法を選択した場合に参考になるサンプル XML ファイルが付属しています。

詳細:

サンプル構成ファイル (P. 95)

ChangeDetector-config.xml ファイルについて

ChangeDetector-config.xml に基づいてカスタム構成ファイルを作成できます。 このファイルには、CA APM ChangeDetector で監視する変更のタイプを指定します。 CA APM ChangeDetector では、データソースごとに変更がグループ化されます。 この構成ファイルを使用して、以下の構成を変更できます。:

- データベース監視プロパティの手動設定 (P. 38) - *database* データソースは、準拠データベースから変更データを収集する方法を定義します。
- ファイル システム監視プロパティの手動設定 (P. 41) - *file* データソースは、変更の監視対象のファイルを定義します。
- Java クラス監視プロパティの手動設定 (P. 49) - *classmonitor* データソースは、監視対象の Java クラスを定義します。

  注: このセクションは、Java プラットフォームにのみ適用されます。
- Java システム プロパティ監視の手動設定 (P. 50) - *javaenv* データソースは、監視対象の Java プロセス システム プロパティを定義します。

  注: このセクションは、Java プラットフォームにのみ適用されます。
- アセンブリ監視プロパティの手動構成 (P. 51) - .NET 環境用のアセンブリ監視を表す *classmonitor* データソースは、監視対象のアセンブリを CA APM ChangeDetector に指示します。

  注: このセクションは、.NET プラットフォームにのみ適用されます。
- .NET 環境変数監視プロパティの手動構成 (P. 55) - .NET 環境用のシステム監視プロパティを表す *javaenv* データソースは、監視対象のシステム プロパティを CA APM ChangeDetector に指示します。

  注: このセクションは、.NET プラットフォームにのみ適用されます。

重要: 完全ファイル パスを使用してファイルを検索すると、CPU 使用率が急上昇する場合があります。 これを回避するには、完全パス ファイル名のみを使って検索してください。 完全パス ファイル名のみを使用して検索するには、以下のように *fullpath="true"* プロパティを追加して *ChangeDetector-config.xml* ファイルを編集します。

```
<scan-directory recursive="true" name="/opt/Oracle" fileset="default"
enabled="true" fullpath="true" > </scan-directory>
```

構成ファイルでのシステム プロパティまたはエージェント プロパティの使用

構成ファイルでは、XML 属性の値としてシステム プロパティまたはエージェント プロファイル プロパティを使用できます。 これにより、実行時に解決されるプロパティを CA APM ChangeDetector 構成ファイルに使用できます。システム プロパティとエージェント プロファイル プロパティの両方に値が与えられる場合は、システム プロパティが優先されます。

以下に例を示します。

```
<scan-directory name="${APPLICATION_HOME}/bin/" recursive="true"
fileset="default"/>
```

上記の例では、実行時に *${APPLICATION_HOME}* が現在の値に置き換えられます。

注: これらのプロパティの値は、実行時に、有効なシステム プロパティまたはエージェント プロファイル プロパティにマップされる必要があります。

詳細:

サンプル構成ファイル (P. 95)

手動によるデータベース監視プロパティの設定

database データ ソース インスタンスにより、CA APM ChangeDetector が標準準拠のデータベースから変更データを収集する方法が決定されます。使用しているデータベースでサポートされる JDBC または OLEDB ドライバがあることを確認してください。

注: 構成プロパティは、使用しているプラットフォームにより若干異なります。 特定のプラットフォームにのみ適用される属性については、このセクションでそのように注釈されています。

注: このエレメントは、サンプル Java ChangeDetector-config.xml (P. 96) ファイル内に示されているカスタム設定ファイルの例で定義されています。 この例はサンプルのデータであり、ユーザのデータ コンテンツを反映しているとは限りません。 異なるバージョンの CA APM ChangeDetector を使用している場合は、該当するバージョンに適宜置き換えてください。

```
<datasource-instance name="Orcl_on_aserver" type="database" version="8.0"
driver="oracle.jdbc.driver.OracleDriver"
driverClasspath="C:¥¥somePathTo¥¥classes12.zip"
url="jdbc:oracle:thin:@aserver:1521:orcl" username="a3f973777b9d"
password="f478831d9bcd65" isClearText="false" >
SQL Server
SELECT name, value FROM v$parameter
</sql>
<schedule type="repetitive" interval="1" unit="min" />
</datasource-instance>
```

クラスパスの区切り文字として ; または : のいずれかを使用すると、データベース ドライバへのクラスパスを複数指定できます。

データベースに使用するユーザ名とパスワードがある場合は、それらを指定できます。 ただし、ユーザ名とパスワードのどちらか一方のみに値を指定することはできません。

注: セキュリティ上の理由で、ユーザ名とパスワードに指定した値は、自動的に暗号化された値に置換されます。 これらの値を変更するには、まずプロパティ *isClearText="true"* を設定してから、変更を行います。 次にエージェントが実行される際に、*isClearText* プロパティは自動的に *false* にリセットされます。

データベース監視を使用する場合は、すべての *datasource-instance* エレメントで定義する必要のある *name* と *type* 属性のほかに、以下の属性を *datasource-instance* に設定する必要があります。

- *driver* - 使用する JDBC 準拠ドライバのクラス名。 上記の例では Oracle データベースに接続するため、このカスタム構成ファイルには以下のドライバを指定します。
  `oracle.jdbc.driver.OracleDriver`

注: この属性は、Java プラットフォームにのみ適用されます。

- *driverClasspath* - driver 属性で参照されたドライバを含むアーカイブへのパス。プラットフォーム用のクラスパスの区切り文字を使用すると、複数のドライバへのパスを指定できます。

注: この属性は、Java プラットフォームにのみ適用されます。

- *url* - データベースへの接続に使用される JDBC URL。
- *username* - データベースへの接続に使用されるユーザ名。 この値は、エージェントの再起動時に *cleartext* プロパティが *true* に設定された場合でも暗号化されます。
- *password* - データベースへの接続に使用されるパスワード。この値は、エージェントの再起動時に *cleartext* プロパティが *true* に設定された場合でも暗号化されます。
- *isClearText* - ユーザ名とパスワードが暗号化されるかどうかを定義します。 値が *false* の場合は、ユーザ名とパスワードが暗号化され、構成ファイルは暗号化された値で上書きされます。

このプロパティは、セキュリティ上の理由によりデフォルトで *false* に設定されています。*true* に設定した場合、エージェントを再起動するとプロパティは *false* に戻ります。

database タイプの datasource-instance エレメントでは、*sql* と *schedule* の 2 つの最上位エレメントを定義できます。

sql エレメントには、監視する予定のテーブルから情報を収集するために使用する SQL ステートメントを指定します。 *datasource-instance* エレメントのこのエレメントには、SQL エレメントをいくつでも指定できます。

注: SQL ステートメントは 2 カラムのみです。1 番目の列の値は、定義した SQL エレメントの数に関係なく一意（主キー）である必要があります。*resultset* の 1 番目のカラムに NULL 値は許可されません。2 番目のカラムに NULL 値がある場合は、行が存在しないものとして処理されます（*resultset* にこの行が存在しない）。

schedule エレメントでは、データベース監視が変更をスキャンする頻度を定義します。このエレメントでは、*type* 属性を指定する必要があります。この属性では、以下の値が有効です。

- *repetitive - type* 属性の値に *repetitive* を指定する場合は、*interval* および *unit* 属性を定義する必要があります。*interval* 属性には整数値を指定します。 *unit* 属性には、*minute*、*hour*、または *sec* のうちの 1 つを指定します。この 2 つの属性により、CA APM ChangeDetector がデータベースの変更をスキャンする頻度が決定されます。 前述の例では、CA APM ChangeDetector は毎分スキャンします。10 秒ごとにスキャンさせるには、*interval* の値に *10* を、*unit* の値に *sec* を指定します。
- *post-startup - type* 属性の値が *post-startup* の場合は、その他の属性を定義する必要はありません。 *type* 属性にこの値を使用すると、CA APM ChangeDetector はエージェントの起動後に 1 度だけ、指定した SQL ステートメントをスキャンします。

手動によるファイル システム監視プロパティの設定

CA APM ChangeDetector のファイル監視システムでは、I/O オーバーヘッドに対するさまざまなニーズと許容度を抱える組織であっても、ファイル変更の監視を制御できます。 このため、変更が発生したときから検出されるまで、処理と I/O コストおよび時間のバランスを保つことができます。

注: ファイル監視プロパティは大文字と小文字を区別します。 たとえば、監視対象のディレクトリ名が *i18n* である場合に、*scan-directory name* プロパティを *I18N* に設定すると、CA APM ChangeDetector は監視対象のディレクトリを特定できません。

注: このエレメントは、サンプル Java ChangeDetector-config.xml (P. 96) ファイル内に示されているカスタム設定ファイルの例で定義されています。この例はサンプルのデータであり、ユーザのデータ コンテンツを反映しているとは限りません。 異なるバージョンの CA APM ChangeDetector を使用している場合は、該当するバージョンに適宜置き換えてください。

```
<datasource-instance name="C_Drive" type="file" version="8.0">
      <!-- datasource-instance specific XML here -->
      <property name="explodeArchiveFiles" value="true" />
      <!-- Accepted units are hour, min, sec -->
      <property name="delayBetweenIterations" value="1" unit="sec" />
      <property name="filesPerIteration" value="1" />
      <property name="delayBetweenArchiveIterations" value="10" unit="sec" />
      <property name="archiveFilesPerIteration" value="1" />
      <!-- Accepted units are bytes, KBytes, MBytes -->
      <property name="maxFileSizeToUpload" value="4" unit="KB" />
      <property name="useDigest" value="needed" />
      <fileset name="default">
         <exclude pattern="(.*)¥.err" />
         <exclude pattern="(.*)¥.log" />
         <exclude pattern="(.*)¥.lok" />
         <exclude pattern="(.*)¥.tlog" />
         <exclude pattern="(.*)¥.log0(.*)" />
      </fileset>
      <fileset name="NoCode">
         <include-fileset name="default" />
         <exclude pattern="(.*)¥.jar" >
             <include pattern="(.*)wily(.*)¥.jar" />
         </exclude>
         <exclude pattern="(.*)¥.zip" />
      </fileset>
      <!--  typically, the wily agent is installed in the "wily" directory. -->
      <scan-directory name="wily" recursive="true"
         fileset="default">
         <exclude name="data" />
      </scan-directory>
      <!-- scan the java home directory, as specified in the java.home system property
-->
      <scan-directory recursive="true" fileset="default"
         name="${java.home}" enabled="false">
         <exclude name="lib/zi" />
      </scan-directory>
      <!--  directories to be scanned in a typical jboss installation -->
      <scan-directory name="." recursive="true" fileset="default"
         enabled="false">
         <exclude name="log" />
         <exclude name="tmp" />
      </scan-directory>
      <!--  test scan directory -->
      <scan-directory recursive="true" fileset="NoCode" name="."
         enabled="true" />
      <!--  directories to be scanned in a typical WebSphere 5.0ee installation -->
      <!--  basically exclude the following dirs:
         _uninst, _uninstPME, BRBeans, classes, installableApps, logs, temp, tranlog,
wstemp -->
```

```
    <scan-directory recursive="true" name="." fileset="default"
       enabled="false">
       <exclude name="_uninst" />
       <exclude name="_uninstPME" />
       <exclude name="logs" />
       <exclude name="temp" />
       <exclude name="tranlog" />
       <exclude name="wstemp" />
    </scan-directory>
</datasource-instance>
```

file データソース インスタンス タイプのエレメントの定義

file タイプの datasource-instance には、以下の 3 つの最上位エレメントを定義できます。

- property（P. 43）
- fileset（P. 46）
- scan-directory（P. 48）

datasource-instance の *file* タイプは、構成ファイルの先頭でファイル システム監視として定義されます。

ファイル変更監視システムは、*file* データソース インスタンスの設定で指定されたすべてのファイルを順にスキャンします。CPU は、作業単位で収集タスクに割り当てられますが、収集タスクの終了後に開放されて他のタスクに戻ります。

property エレメント

各 *property* エレメントには、*name* 属性と *value* 属性が必要です。 ここで定義されている property の一部には、*unit* 属性も使用されています。

注: .NET プラットフォームではアーカイブがサポートされていないため、アーカイブに関するプロパティはすべて Java 専用として定義できます。

このタイプの datasource-instance では、以下の property を使用できます。

- *explodeArchiveFiles*

  このプロパティは、Java エージェントを使用する場合にのみ適用可能です。 これは、.NET エージェントが動作する環境とは互換性がありません。

  このプロパティの value 属性には、*true* または *false* を指定できます。

  value に *true* を指定すると、CA APM ChangeDetector によってアーカイブ ファイルの内容がレポートされます。 CA APM ChangeDetector が監視しているアーカイブ ファイル内で変更が発生すると、少なくとも 2 つの変更イベントが送信されます。1 つはアーカイブ ファイルの変更について、もう 1 つはアーカイブ ファイル内のファイルの変更についてです。 変更されたアーカイブ ファイル内のファイルもアーカイブ ファイルである場合、CA APM ChangeDetector はそのアーカイブ ファイルも開いて、ネストされたアーカイブ内のファイルの変更イベントも送信します（アーカイブのネストが続く限り、この動作が続きます）。

  アーカイブとしてサポートされるのは、ZIP および GZIP ファイル形式のみです（たとえば、*zip*、*gzip*、*jar*、*ear*、*war*、*rar*、*sar*）。 CA APM ChangeDetector では、*tar* ファイルはサポートされていません。

  explodeArchiveFiles の value に false を指定すると、CA APM ChangeDetector によってアーカイブの内容は調査されません。 アーカイブへの変更は、アーカイブ自体の修正、追加、削除としてレポートされます。

  デフォルト値は *false* です。

- *delayBetweenIterations*

  このプロパティの value には整数を指定し、unit には *sec*、*min*、または *hour* *で単位を指定できます。*

  このプロパティでは、ファイル キューへの各イタレーション間のスリープ時間を定義します。このプロパティは、*filesPerIteration* プロパティと関連します。

  デフォルト値は 3 秒です。

- *filesPerIteration*

  このプロパティの value には整数を指定し、*delayBetweenIterations* プロパティで定義された作業単位でのファイル数を定義します。

  このプロパティでは、スキャンされるファイルの数を指定します。ファイルのスキャン後に、CPU が解放されます。 filesPerIteration に 10 を指定した場合、CA APM ChangeDetector は 10 個のファイルの変更をスキャンしてから、*delayBetweenIterations* プロパティに定義されている間スリープ状態になります。 スリープ時間の終了後に次の 10 ファイルをスキャンして、またスリープ状態に戻る、という動作を繰り返します。

  デフォルト値は 5 です。

- *delayBetweenArchiveIterations*

  このプロパティの value には整数を指定し、unit には *sec*、*min*、または *hour で単位を指定できます。*

  このプロパティでは、アーカイブ キューでのスリープ時間を制御します。CA APM ChangeDetector がファイルをスキャンして、そのファイルがアーカイブであることがわかると、CA APM ChangeDetector はそのアーカイブをアーカイブ キューに置き、その内容を検査できるようにします。

  このプロパティは、*explodeArchiveFiles* が *true* に設定されている場合にのみ使用できます。その他の場合、アーカイブ ファイルは通常のファイルとして処理されます。

  デフォルト値は 10 秒です。

- *archiveFilesPerIteration*

  filesPerIteration プロパティと同じ属性を使用できます。*filesPerIteration* プロパティと同様に、*archiveFilesPerIteration* はアーカイブ キュー用であり、イタレーションごとに内容がスキャンされるアーカイブの数を制御します。このプロパティは、*explodeArchiveFiles* が *true* に設定されている場合にのみ使用できます。その他の場合、アーカイブ ファイルは通常のファイルとして処理されます。

  デフォルト値は 1 です。

  注: アーカイブの内容はすべて一度にアップロードされます。*filesPerIteration* の値は適用されません。 ただし、スキャン中のアーカイブ内にアーカイブが見つかった場合、そのネストされたアーカイブはアーカイブ キューに追加されます。

- *maxFileSizeToUpload*

  このプロパティの value には整数を指定し、unit には *B*、*KB*、または *MB* で単位を指定できます。 このプロパティでは、内容がサーバに送られるファイルの最大サイズを定義します。 現在、このプロパティで定義されたファイル サイズ以下の ASCII ファイルのみが送られます。

  デフォルト値は 50 KB です。

- *useDigest*

  このプロパティでは、変更の検出にメッセージ ダイジェスト（MD5 など）を使用する方法を定義します。ダイジェストを使用すると、CA APM ChangeDetector は、ハッシュ比較を行い、ファイル実体への変更は無視します。 このプロパティには以下の value を指定できます。

  - never： ダイジェストは使用されません
  - always： 常にダイジェスト比較が行われます
  - needed： タイムスタンプとファイル サイズが変更されている場合にのみダイジェストが使用されます（デフォルト）

fileset エレメント

fileset エレメントの例を以下に示します。

```
<fileset name="default">
         <exclude pattern="(.*)¥.err" />
         <exclude pattern="(.*)¥.log" />
         <exclude pattern="(.*)¥.lok" />
         <exclude pattern="(.*)¥.tlog" />
         <exclude pattern="(.*)¥.log0(.*)" />
</fileset>
<fileset name="NoCode">
         <include-fileset name="default" />
         <exclude pattern="(.*)¥.jar" >
             <include pattern="(.*)wily(.*)" />
         </exclude>
         <exclude pattern="(.*)¥.zip" />
</fileset>
<fileset name="all">
         <exclude pattern="(.*)>
             <include pattern="(.*) />
         </exclude>
</fileset>
```

ファイルセットでは、スキャン対象とする、または除外するファイルのパターンを定義します。 fileset エレメントでは、以下のサブエレメントを使用できます。

- *exclude*
- *include-fileset*

exclude エレメントには、属性 *pattern* を定義する必要があります。 *exclude* エレメントには、サブノードとして *include* エレメントを含めることができます。*include* エレメントには、属性 *pattern* を定義する必要があります。 *include* エレメントは、*exclude* エレメントを無効にする目的で使用します。 上記の例では、ファイルセット *NoCode* について、*.jar* で終わるファイル名は、名前に *wily* があるファイルを除いてすべて除外されます。

include-fileset エレメントには、属性 *name* を定義する必要があります。 *include-fileset* エレメントを使用する場合は、含まれるファイルセットを事前に定義する必要があります。 上記の例では、ファイルセット *NoCode* にファイルセット *default* が含まれています。 これら 2 つのファイルセットの順序を逆にして、default ファイルセットを 2 つ目に定義すると、設定は無効になります。 ファイルセットで *include-fileset* エレメントを使用すると、参照されるファイルセットの包含および除外パターンはマスタ エレメントの一部になります。 上記の例では、ファイルセット「*NoCode*」に除外パターン **.jar*、**.zip*、**.err*、**.log*、**.lok*、などが含まれます。

fileset エレメント内での *exclude* および *include-fileset* エレメントの順序は関係ありません。 ファイルは、定義された除外パターンに一致しない場合、スキャン対象になります。定義された除外パターンに一致する場合、そのファイルは、除外パターン内に定義された包含パターンに一致するかどうか確認されます。 包含パターンに一致する場合、そのファイルはスキャン対象になります。 包含パターンに一致しない場合、そのファイルはスキャン対象から外されます。

scan-directory エレメント

scan-directory エレメントでは、スキャン対象となるディレクトリを定義します。 *scan-directory* エレメントの属性は、*name*、*recursive*、*fileset*、および *enabled* です。

- *name* 属性では、スキャン対象となるディレクトリを定義します。 必須です。 たとえば、*c:¥¥test* または *C:/test* のように指定します。
- *recursive* 属性には、*true* または *false* を value に指定します。この属性では、ファイル システム監視が、検出するディレクトリのスキャンを繰り返すかどうかを定義します。 デフォルト値は *true* です。
- fileset 属性には、このデータソース インスタンスで使用するファイル セットを定義します。 指定するファイルセットは、構成ファイルでこのエレメントより前に定義されている必要があります。 これは必須の属性です。
- *enabled* 属性には、*true* または *false* を value に指定します。 *scan-directory* エレメントを有効化または無効化します。

scan-directory エレメントには、子の *exclude* エレメントを使用できます。 *exclude* エレメントには、*name* 属性が必要です。 *name* 属性は、ディレクトリ にマップされる必要があります。

注: name 属性に指定する値には正規表現を使用できません。 実際のディレクトリにマップする文字列を指定する必要があります。 パターンに基づいて除外するには、*fileset* エレメントの指定内容を修正します。

以下に、Sample Java ChangeDetector-config.xml ファイル (P. 96)に定義されているこのエレメントの例を示します。

```
<scan-directory name="." recursive="true" fileset="default"
   enabled="false">
   <exclude name="log" />
   <exclude name="tmp" />
</scan-directory>
```

手動による Java クラス監視プロパティの設定

注: このセクションは、Java プラットフォームにのみ適用されます。

classmonitor データ ソース インスタンスにより、CA APM ChangeDetector が監視する Java クラスが決定されます。 クラス名とそのクラスローダをリソース名として使用します。クラス定義に対応するバイトアレイがリソースの値として使用されます。

このタイプの *datasource* インスタンスは、1 つの CA APM ChangeDetector インスタンスにつき 1 つしか使用できません。

注: Java はロードオンデマンドで動作するため、CA APM ChangeDetector では、クラスが実行中のバイナリの一部でなくなったのか、またはまだロードされていないのかを判断できません。このため、*classmonitor* データ ソースでは、削除による変更が生成されません。

注: このエレメントは、サンプル Java ChangeDetector-config.xml (P. 96) ファイル内に示されているカスタム設定ファイルの例で定義されています。この例はサンプルのデータであり、ユーザのデータ コンテンツを反映しているとは限りません。 異なるバージョンの CA APM ChangeDetector を使用している場合は、該当するバージョンに適宜置き換えてください。

```
<datasource-instance name="Java class monitor" type="classmonitor"  version="8.0">
      <property name="delayBetweenIterations" value="2" unit="sec"/>
      <property name="classesPerIteration" value="100" />
      <exclude pattern="foo" />
      <exclude pattern=".*bar.*">
         <include pattern="hello" />
         <include pattern=".*world.*" />
      </exclude>
   </datasource-instance>
```

exclude エレメントの構文は、Java システム プロパティ監視と同じです。

詳細:

手動による Java システム プロパティ監視の設定 (P. 50)

classmonitor データソース インスタンス タイプのエレメントの定義

注: このセクションは、Java プラットフォームにのみ適用されます。

classmonitor タイプのデータソース インスタンスには、2 つの property エレメントを定義できます。これらの各 *property* エレメントには、以下のように *name* 属性と *value* 属性が必要です。

- *delayBetweenIterations*

  このプロパティの value には整数を指定し、unit には *sec*、*min*、または hour で単位を指定できます。

  このプロパティでは、クラス キューへの各イタレーション間のスリープ時間を定義します。このプロパティは、*classesPerIteration* プロパティと関連します。

  デフォルト値は 2 秒です。

- *classesPerIteration*

  このプロパティの value には整数を指定し、*delayBetweenIterations* プロパティで定義された作業単位でのクラス数を定義します。

  このプロパティでは、スキャンされるクラスの数を指定します。クラスのスキャン後に、CPU が解放されます。classesPerIteration に 10 を指定した場合、CA APM ChangeDetector は 10 個のクラスの変更をスキャンしてから、*delayBetweenIterations* プロパティに定義されている間スリープ状態になります。 スリープ時間の終了後に次の 10 クラスをスキャンして、またスリープ状態に戻る、という動作を繰り返します。

  デフォルト値は 100 です。

手動による Java システム プロパティ監視の設定

注: このセクションは、Java プラットフォームにのみ適用されます。

javaenv データソース インスタンスにより、CA APM ChangeDetector が監視する Java システム プロパティが決定されます。CA APM ChangeDetector は、プロセスの再起動間の Java システム プロパティの変更のみを監視します。実行中の Java プログラムによって引き起こされた実行時の変更は監視しません。このタイプの変更の収集は、起動時に一度だけ行われます。 プロパティ名がリソース名として使用され、プロパティの値がこれらのリソースの値を構成します。

注: このエレメントは、サンプル Java ChangeDetector-config.xml (P. 96) ファイル内に示されているカスタム設定ファイルの例で定義されています。この例はサンプルのデータであり、ユーザのデータ コンテンツを反映しているとは限りません。 異なるバージョンの CA APM ChangeDetector を使用している場合は、該当するバージョンに適宜置き換えてください。

```
<datasource-instance name="Java system properties" type="javaenv"  version="8.0">
      <exclude pattern="foo" />
      <exclude pattern=".*bar.*">
         <include pattern="hello" />
         <include pattern=".*world.*" />
      </exclude>
   </datasource-instance>
```

javaenv データ ソース内では、*exclude* エレメントのみを最上位エレメントとして定義できます。 *exclude* エレメントには、 孫エレメントを使用できます。

exclude エレメントを使用して、CA Introscope に表示させないプロパティを除外します。 *exclude* エレメントには、*pattern* 属性を定義する必要があります。 たとえば、CA Introscope に *foo* プロパティを表示させないようにするには、上記の例のように *exclude* エレメントを使用します。

exclude エレメントには、*include* サブノードをいくつでも定義できます。*exclude* エレメントと同様に、*include* エレメントには *pattern* 属性を定義する必要があります。 *include* サブノードを使用すると、*exclude* エレメントの動作を無効にすることができます。 上記の例に示すように、*.*bar.** に一致するプロパティは、*hello* または *.*world.** に一致するプロパティを除いてすべて除外できます。

手動によるアセンブリ監視プロパティの設定

注: このセクションは、.NET プラットフォームにのみ適用されます。

.NET 環境では、*classmonitor* データ ソースはアセンブリ監視を表します。このデータ ソースにより、CA APM ChangeDetector が監視するアセンブリが決定されます。

アセンブリ監視は、一度に 1 つのメソッドをロードします。メソッドには、以下のメタデータが含まれます。

- アセンブリ名
- バージョン
- クラス
- メソッド
- メソッド シグネチャ

同じ名前を持つアセンブリのバージョン間で、メタデータの変更が監視できます。 たとえば、*cd_sample.dll 1.0.0* および *cd_sample.dll version 1.0.1* 内のクラスは、バージョンが異なる同じメタデータとして扱われます。 このデータへの変更が監視され、Workstation Investigator に表示されます。ただし、アセンブリの名前が変わると、そのアセンブリは新しいリソースとして扱われるため、その中のクラスは新しいリソースとなり、追加イベントとして扱われます。

注: このエレメントは、サンプル .NET ChangeDetectorDotnet-config.xml (P. 102) ファイル内に示されているカスタム設定ファイルの例で定義されています。 この例はサンプルのデータであり、ユーザのデータ コンテンツを反映しているとは限りません。 異なるバージョンの CA APM ChangeDetector を使用している場合は、該当するバージョンに適宜置き換えてください。

```
<datasource-instance name="Assembly Monitor" type="classmonitor" version="8.0">

    <property name="initialWaitTime" value="30" unit="sec" />
    <property name="delayBetweenIterations" value="2" unit="min" />
    <property name="classesPerIteration" value="5" />
    <excludeassembly pattern=".¥mscorlib¥.dll"/>
    <excludeassembly pattern=".¥System¥.dll"/>
    <excludeassembly pattern=".¥System¥.Xml¥.dll"/>
    <excludeassembly pattern=".¥System¥.Web¥.dll"/>
    <excludeassembly pattern=".¥System¥.Configuration¥.dll"/>
    <excludeassembly pattern=".¥wily¥.."/>
    <excludeassembly pattern=".¥Microsoft¥.JScript¥.dll"/>
    <excludeassembly pattern=".¥VJSharpCodeProvider¥.dll"/>
    <excludeassembly pattern=".¥System¥.Data¥.dll"/>
    <excludeassembly pattern=".¥Oracle¥.DataAccess¥.dll"/>
    <excludeassembly pattern=".¥System¥.Web¥.Mobile¥.dll"/>
    <excludeassembly pattern=".¥System¥.ServiceModel¥.dll"/>
    <excludeassembly pattern=".¥SMDiagnostics¥.dll"/>
    <excludeassembly pattern=".¥System¥.Drawing¥.dll"/>
    <excludeassembly pattern=".¥System¥.Web¥.RegularExpressions¥.dll"/>
    <excludeassembly pattern=".¥Microsoft¥.VisualBasic¥.dll"/>
```

```
    <excludeassembly pattern=".¥CppCodeProvider¥.dll"/>
    <excludeassembly pattern=".¥System¥.EnterpriseServices¥.dll"/>
    <excludeassembly pattern=".¥System¥.Transactions¥.dll"/>

    <exclude pattern="com¥.wily¥.(.*)"/>

</datasource-instance>
```

これらのエレメントの構文を以下に示します。

- *excludeassembly*

  <!-- アセンブリを除外 -->

  <excludeassembly pattern=".¥mscorlib¥.dll"/>

  <excludeassembly pattern=".¥System¥.dll"/>

  <excludeassembly pattern=".¥System¥.Xml¥.dll"/>

  <excludeassembly pattern=".¥System¥.Web¥.dll"/>

  <excludeassembly pattern=".¥System¥.Configuration¥.dll"/>

  <excludeassembly pattern=".¥wily¥.."/>

  <excludeassembly pattern=".¥Microsoft¥.JScript¥.dll"/>

  <excludeassembly pattern=".¥VJSharpCodeProvider¥.dll"/>

  <excludeassembly pattern=".¥System¥.Data¥.dll"/>

  <excludeassembly pattern=".¥Oracle¥.DataAccess¥.dll"/>

  <excludeassembly pattern=".¥System¥.Web¥.Mobile¥.dll"/>

  <excludeassembly pattern=".¥System¥.ServiceModel¥.dll"/>

  <excludeassembly pattern=".¥SMDiagnostics¥.dll"/>

  <excludeassembly pattern=".¥System¥.Drawing¥.dll"/>

  <excludeassembly pattern=".¥System¥.Web¥.RegularExpressions¥.dll"/>

  <excludeassembly pattern=".¥Microsoft¥.VisualBasic¥.dll"/>

  <excludeassembly pattern=".¥CppCodeProvider¥.dll"/>

  <excludeassembly pattern=".¥System¥.EnterpriseServices¥.dll"/>

  <excludeassembly pattern=".¥System¥.Transactions¥.dll"/>

- exclude

  <exclude pattern="com¥.wily¥.(.)"/>*

- include

  <excludeassembly pattern=".¥System¥.dll"/>

  <exclude pattern="abc¥.xyz¥.(.)">*

  <include pattern="abc¥.xyz¥.asdf¥.(.)"/> </exclude>*

classmonitor タイプのデータ ソース インスタンスには、プロパティ エレメントを定義できます。これらの各 *property* エレメントには、以下のように *name* 属性と *value* 属性が必要です。

- *initialWaitTime*

  このプロパティの value には整数を指定し、unit には sec、min、または hour で単位を指定できます。

  このプロパティでは、エージェントが、アセンブリが最後にロードされてからクラスのスキャンを開始するまで待機する時間を定義します。

- *delayBetweenIterations*

  このプロパティの value には整数を指定し、unit には sec、min、または hour で単位を指定できます。

  このプロパティでは、クラス キューへの各イタレーション間のスリープ時間を定義します。このプロパティは、classesPerIteration プロパティと関連します。

  デフォルト値は 2 秒です。

- *classesPerIteration*

  このプロパティの value には整数を指定し、delayBetweenIterations プロパティで定義された作業単位でのクラス数を定義します。

  このプロパティでは、スキャンされるクラスの数を指定します。クラスのスキャン後に、CPU が解放されます。classesPerIteration に 10 を指定した場合、CA APM ChangeDetector は 10 個のクラスの変更をスキャンしてから、delayBetweenIterations プロパティに定義されている間スリープ状態になります。 スリープ時間の終了後に次の 10 クラスをスキャンして、またスリープ状態に戻る、という動作を繰り返します。

  デフォルト値は 100 です。

手動による .NET 環境変数監視プロパティの設定

注: このセクションは、.NET プラットフォームにのみ適用されます。

.NET 環境変数監視データソース（javaenv）インスタンスにより、CA APM ChangeDetector が監視する環境変数が決定されます。 CA APM ChangeDetector は、プロセスの再起動時の環境変数の変更のみを監視します。実行中のアプリケーションによって引き起こされた実行時の変更は監視しません。このタイプの変更の収集は、起動時に 1 回のみ行われます。変数名がリソース名として使用され、変数の値がこれらのリソースの値を構成します。

注: このエレメントは、サンプル .NET ChangeDetectorDotnet-config.xml (P. 102) ファイル内に示されているカスタム設定ファイルの例で定義されています。 この例はサンプルのデータであり、ユーザのデータ コンテンツを反映しているとは限りません。 異なるバージョンの CA APM ChangeDetector を使用している場合は、該当するバージョンに適宜置き換えてください。

```
<datasource-instance name="System Properties" type="javaenv"  version="8.0">
      <exclude pattern="foo" />
      <exclude pattern=".*bar.*">
         <include pattern="hello" />
         <include pattern=".*world.*" />
      </exclude>
   </datasource-instance>
```

javaenv データソース内では、exclude エレメントのみを最上位エレメントとして定義できます。 exclude エレメントには、孫エレメントを使用できます。

exclude エレメントを使用して、CA Introscope に表示させないプロパティを除外します。 exclude エレメントには、pattern 属性を定義する必要があります。 たとえば、CA Introscope に foo プロパティを表示させないようにするには、上記の例のように exclude エレメントを使用します。

exclude エレメントには、include サブノードをいくつでも定義できます。exclude エレメントと同様に、include エレメントには pattern 属性を定義する必要があります。 include サブノードを使用すると、exclude エレメントの動作を無効にすることができます。 上記の例に示すように、.*bar.* に一致するプロパティは、hello または .*world.* に一致するプロパティを除いてすべて除外できます。

既存の ChangeDetector-config.xml ファイルの更新

CA APM ChangeDetector をアップグレードすると、構成ファイルは自動的に新しい形式に更新されます。

注: 更新する既存の構成ファイルを、元の名前 *.bak* で保存するには、構成ファイルおよび現在のディレクトリへの書き込み権限が必要です。

CA APM ChangeDetector により、既存の構成ファイルの名前が変更されて、更新後のファイルに置き換えられます。 既存のファイルの名前が変更できない場合、またはディレクトリに書き込みできない場合は、新しい更新構成ファイルは一時ディレクトリに保存されて、既存のファイルを新しいファイルで上書きするように求められます。

エージェント構成ファイルの変更

IntroscopeAgent.profile を変更して、CA APM ChangeDetector エージェント拡張 ID、CA APM ChangeDetector 構成ファイルのパス、およびフェールオーバ メカニズムを指定します。 これらの変更は必ずしも必要なく、自動 ID 割り当てを選択することも可能です。

ChangeDetector エージェント ID と構成ファイルへのパスの設定（オプション）

1. ChangeDetector 対応の各エージェント インスタンスで、*IntroscopeAgent.profile* に、*IntroscopeAgentBackup.profile* のような覚えやすい名前を付けて、別のディレクトリにバックアップします。
2. *IntroscopeAgent.profile* を編集して、以下のプロパティを設定します。

   introscope.changeDetector.agentID=<ChangeDetector agent ID>

   agentID プロパティに使用できるのは、英数字と、特殊文字であるアンダースコア（_）およびハイフン（-）のみです。

   この ID は、CA APM ChangeDetector エージェントのグローバル ID に対応し、Enterprise Manager または Enterprise Manager クラスタに接続しているすべてのエージェントで一意である必要があります。

3. ChangeDetector 対応の各エージェント インスタンスで *introscopeAgent.profile* を編集して、以下のようにファイル名を含めた構成ファイルのパスを設定します。

   introscope.changeDetector.profile=<ChangeDetector へのパス-config.xml>

   サンプル構成ファイルの名前を変更しないで使用している場合、ファイル名は *ChangeDetector-config.xml* です。 名前を変更した場合は、その名前を指定します。

   パス名の入力には、絶対パス名と 2 つの円記号を使用します。 たとえば、WebLogic アプリケーション サーバを参照する場合、プロパティ値に入力するパスは以下のようになります。
   <ProductName_Home>¥¥<Agent_Home>.

   注: 各エージェント用の構成ファイルにそれぞれパスを確実に指定してください。 *IntroscopeAgent.profile* から複数のエージェントを実行する場合は、コマンド ラインでプロパティを定義します（たとえば、アプリケーション サーバの起動スクリプトを使用）。 これにより、CA Introscope は CA APM ChangeDetector エージェント拡張構成を区別できます。 新しい IntroscopeAgent プロファイルの作成方法については、「CA APM Java Agent 実装ガイド」を参照してください。

詳細:

ChangeDetector エージェント ID の命名オプション (P. 60)

負荷分散環境での CA APM ChangeDetector の構成

CA APM ChangeDetector エージェント拡張は、負荷分散されるエージェント環境で動作するように構成できます。 このためには、CA APM ChangeDetector エージェント拡張対応のエージェントをマネージャ オブ マネージャ（MOM）に接続するように構成します。 MOM は、weight プロパティ（*introscope.enterprisemanager.clustering.login.em1.weight*）の構成に応じて、エージェントの負荷を適切なコレクタに分散させます。

CD で負荷分散を行うように設定する方法

- 以下のプロパティを設定して、ChangeDetector エージェントを MOM に接続します。

```
introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=<MOM Host>
introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=5001
introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
```

- 以下のプロパティを設定して、MOM での負荷分散を設定します。

（コレクタ 1）

```
introscope.enterprisemanager.clustering.login.em1.host=sqw32vserv12
introscope.enterprisemanager.clustering.login.em1.port=5001
introscope.enterprisemanager.clustering.login.em1.publickey=internal/server/EM.public
introscope.enterprisemanager.clustering.login.em1.weight=50
```

（コレクタ 2）

```
introscope.enterprisemanager.clustering.login.em2.host=sqw32vserv10
introscope.enterprisemanager.clustering.login.em2.port=5002
introscope.enterprisemanager.clustering.login.em2.publickey=internal/server/EM.public
introscope.enterprisemanager.clustering.login.em2.weight=50
```

エージェントのプロパティの詳細については、「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。Enterprise Manager プロパティの詳細については、「CA APM 設定および管理ガイド」を参照してください。

CA APM ChangeDetector でのエージェント フェールオーバ メカニズムの構成

CA APM ChangeDetector エージェント拡張がエージェント フェールオーバ メカニズムと組み合わせて機能するようにインストールされている場合は、以下の手順に従ってプライマリおよびフォールバックの Enterprise Manager コレクタを構成します。

フェールオーバ時に、CA APM ChangeDetector エージェント拡張は、フォールバック Enterprise Manager として指定された Enterprise Manager に変更データを送信し始めます。 ただし、フォールバック Enterprise Manager の変更データベース（changes.db）がプライマリ Enterprise Manager と同期されないときには、CA APM ChangeDetector はシャットダウンされます。 CA Introscope エージェントの実行は継続され、CA APM ChangeDetector はプライマリ（デフォルト） Enterprise Manager が復帰するまでプライマリ Enterprise Manager への送信を再試行します。

注: CA Introscope エージェントが直接 MOM に接続されていると、変更はレポートされません。

CD で Agent フェールオーバ メカニズムを設定する方法

IntroscopeAgent.profile で、以下のプロパティを設定します。

- CA APM ChangeDetector を有効にするエージェントにプライマリ Enterprise Manager コレクタ（MOM ではない）を指定します。 以下のプロパティを設定します。

  ```
  introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=<primary EM collector host>
  introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=<primary EM collector port>
  introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
  ```

- MOM Enterprise Manager をフォールバック Enterprise Manager として指定します。 以下のプロパティを設定します。

  ```
  introscope.agent.enterprisemanager.transport.tcp.host.FALLBACK =<MOM EM collector host>
  introscope.agent.enterprisemanager.transport.tcp.port.FALLBACK =<MOM EM collector port>
  introscope.agent.enterprisemanager.transport.tcp.socketfactory.FALLBACK=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
  ```

- 以下のプロパティを設定して、接続順と再試行間隔を指定します。
  ```
  introscope.agent.enterprisemanager.connectionorder=DEFAULT, FALLBACK
  introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds=
  <failover retry time in seconds>
  ```

エージェントのプロパティの詳細については、「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。

.NET における個別の構成ファイルでの複数アプリケーションの実行

.NET の IIS で複数のアプリケーションを実行している場合に、各アプリケーションからの変更が個別にレポートされるようにするには、カンマで区切ったリストで各アプリケーション用の設定フォルダを特定する必要があります。 各アプリケーションに、独自の ChangeDetector 設定フォルダが必要です。このフォルダに、各アプリケーション用の ChangeDetector 設定 XML を置きます。 パスの最後のフォルダ名は、アプリケーションの名前（AppDomain FriendlyName）の一部に一致する必要があります。

たとえば、個別の設定フォルダを使用して BalloonShop および Petshop の変更を監視する場合は、*introscope.changeDetector.profileDir* を以下のように設定します。

```
introscope.changeDetector.profileDir=S:¥sw¥CA_Wily¥wily_dotnet¥Introscope<Version
Number>¥wily¥CD-config¥balloonshop,S:¥sw¥CA_Wily¥wily_dotnet¥Introscope<VersionNu
mber>¥wily¥CD-config¥petshop
```

CA APM ChangeDetector の無効化

CA APM ChangeDetector を無効にする方法

以下のプロパティを false に設定します。

```
introscope.changeDetector.enable=false
```

ChangeDetector エージェント ID の命名オプション

ChangeDetector エージェント ID は、インストールする各 CA APM ChangeDetector エージェント拡張で一意である必要があります。 CA Introscope エージェントが 1 つのみの単純な環境では、CA APM ChangeDetector が自動的に割り当てる ID を使用してかまいません。

もっと複雑な環境では、別の方法で一意の ChangeDetector エージェント ID を取得できます。 以下の状況に応じて、選択肢が異なります。

- CA APM ChangeDetector 対応エージェントが複数あるかどうか
- 各エージェントが固有のプロファイル ファイル（IntroscopeAgent.profile）を持っているか、または、すべてのエージェントが同じファイルを使用するか
- CA APM ChangeDetector エージェント拡張に自動 ID 生成を使用したいかどうか
- CA APM ChangeDetector エージェント拡張の識別に、Java システム プロパティ、.NET 環境変数監視、またはエージェント プロパティを使用するかどうか

注: 同一の CA Introscope エージェントで実行される複数の CA APM ChangeDetector エージェント拡張が同じ ID を持っている場合、2 番目に起動されるインスタンスはシャットオフされてエラー メッセージが出力されます。

以下の表に、推奨される選択肢の一部を要約します。

環境	推奨されるエージェント ID の設定
各 CA APM ChangeDetector 対応エージェントが、固有のディレクトリにある固有のエージェント プロファイルを使用する。	■ CA APM ChangeDetector の自動 ID 割り当てを使用します。この場合は、何もする必要はありません。
複数の CA APM ChangeDetector 対応エージェントが同じエージェント プロファイルを使用する。	■ Java 起動コマンドで、-D パラメータを使用して *introscope.changeDetector.agentID* プロパティを指定します。 ■ Java コマンド ラインまたは *IntroscoptAgent.profile* で、使用可能な Java システム プロパティ、.NET 環境変数監視、またはエージェント プロファイルのプロパティに基づく式を使用して、実行時に解決します。

詳細:

-D Java システム パラメータによる ID の割り当て (P. 62)
Java システム プロパティまたはエージェント プロファイル プロパティに基づく ID の割り当て (P. 62)

自動 ID 割り当ての使用

各 CA APM ChangeDetector 対応エージェントに 1 つの IntroscopeAgent.profile を使用している場合にのみ、自動 ID 割り当てを使用できます。

この場合は、Java システム コマンド、または *IntroscopeAgent.profile* ファイルでプロパティ *introscope.changeDetector.agentID* が定義されないときに、CA APM ChangeDetector によりこのプロパティに一意の値が生成されます。この値は、CA APM ChangeDetector ルート ディレクトリの「.id」ファイルに保存されます。 このファイルを削除または変更しないでください。

注: CA APM ChangeDetector のルート ディレクトリ（change_detector）は、デフォルトで *IntroscopeAgent.profile* ファイルがあるディレクトリに作成されます。 ただし、プロパティ *introscope.changeDetector.rootDir* で別のルート ディレクトリを指定できます。

-D Java システム パラメータによる ID の割り当て

CA APM ChangeDetector エージェント拡張の ID は、Java コマンド ラインでも定義できます。 Java コマンド ラインで定義するには、以下の例で示すように、-D パラメータと *introscope.changeDetector.agentID* プロパティを一緒にして使用します。

```
java -Dintroscope.changeDetector.agentID=<value>
```

Java システム プロパティまたはエージェント プロファイル プロパティに基づく ID の割り当て

ChangeDetector エージェント ID は、Java システム プロパティまたはエージェント プロファイル プロパティに基づいて実行時に動的に割り当てることができます。 この方法は、環境が複雑で、すでに別のプロパティを使用してさまざまなプロセスを区別している場合に使用します。

別のプロパティに基づいて ChangeDetector エージェント ID を割り当てる方法（オプション 1）

- *IntroscopeAgent.profile* ファイルで、以下の例のようにエージェント ID プロパティの式を入力します。prefix と suffix には、単純な定数文字列を指定します。

```
introscope.changeDetector.agentID=<prefix>${any Java system property}<suffix>
introscope.changeDetector.agentID=<prefix>${any Agent profile property}<suffix>
```

  prefix または suffix のどちらか、または両方を使用できます。

別のプロパティに基づいて ChangeDetector エージェント ID を割り当てる方法（オプション 2）

- Java 起動コマンドで、以下の例のように、上記に似た式を使用します。prefix と suffix には、単純な定数文字列を指定します。

```
java -Dintroscope.changeDetector.agentID=<prefix>${any Java system property}<suffix>
java -Dintroscope.changeDetector.agentID=<prefix>${any Agent profile property}<suffix>
```

  prefix または suffix のどちらか、または両方を使用できます。

オプションの構成プロパティ

CA APM ChangeDetector には、CA Introscope エージェント、Workstation、および Enterprise Manager の構成ファイルで設定できるオプションのプロパティがあります。

オプションのエージェント プロパティ

以下のプロパティを、*IntroscopeAgent.profile* に設定します。

- *introscope.changeDetector.rootDir*

  CA APM ChangeDetector のルート ディレクトリの場所を指定します。このディレクトリがまだ存在しない場合は、作成される場所を指定します。 指定しない場合、ルート ディレクトリはデフォルトで *<Agent_Home>* になります。

  注: CA APM ChangeDetector は、ルート ディレクトリを使用して、通常の処理に必要なファイルを作成します。このディレクトリは削除しないでください。ルート ディレクトリの場所を定義するプロパティはオプションです。

- *introscope.changeDetector.compressEntries.enable*

  この値を false に設定すると、データ圧縮が無効になります。データ圧縮では、CA APM ChangeDetector データ バッファを圧縮できます。 これは、起動時にメモリ消費が発生する場合に役立ちます。 デフォルト値は *true* です。 *false* に設定した場合、その設定を有効にするにはアプリケーションの再起動が必要です。

- *introscope.changeDetector.compressEntries.batchSize*

  このプロパティは、*introscope.changeDetector.compressEntries.enable* プロパティと一緒に使用します。 このプロパティは、圧縮ジョブのバッチ サイズを定義します。 デフォルト値は 1000 です。

- *introscope.changeDetector.isengardStartupWaitTimeInSec*

  このプロパティは、デフォルトで有効になっています。 このプロパティでは、エージェントが起動してから Enterprise Manager への接続を試行するまでの待ち時間を秒単位で指定できます。デフォルト値は 15 です。

- *introscope.changeDetector.waitTimeBetweenReconnectInSec*

  このプロパティは、デフォルトで有効になっています。 このプロパティでは、エージェントが Enterprise Manager への再接続を試行するまでの待ち時間を秒単位で指定できます。 デフォルト値は 10 です。

オプションの Workstation プロパティ

以下のプロパティは、*IntroscopeWorkstation.properties* で設定できます。

- *introscope.changeDetector.defaultDataWindowValue=<value>*

  Investigator でライブ データを表示する期間を指定します。このプロパティは、ライブ データを表示する時間単位を指定する *defaultDataWindowUnit* プロパティと共に使用します。 デフォルト値は 1 です。

- *introscope.changeDetector.defaultDataWindowUnit*=<value>

  Investigator でライブ データを表示する時間単位を指定します。このプロパティは、 *defaultDataWindowValue* プロパティと共に使用します。 このプロパティの有効な値は、*seconds*、*minutes*、*hours*、*days*、*weeks*、および *months* です。 デフォルト値は *weeks* です。 days（日）または months（月）を指定した場合に使用される実際の開始日は、ロケールによって異なります。たとえば、週が日曜に始まるロケールと、月曜に始まるロケールがあります。

- *introscope.changeDetector.useCh*angeTime=false

  デフォルトで、CA APM ChangeDetector はファイル変更の検出時刻を表示しますが、変更時刻は表示しません。 検出時刻と変更時刻の両方を表示するには、このプロパティを *true* に設定します。 このプロパティは、ファイル変更にのみ適用されます。システム プロパティ、環境変数、データベース、または Java クラスの変更には適用されません。

EPAgent プラグインからの CA APM ChangeDetector データ送信の構成

注: この属性は、Java プラットフォームにのみ適用されます。

EPAgent プラグインまたは EPA 拡張用に CA APM ChangeDetector を使用する場合は、EPAgent を構成します。

CA APM ChangeDetector は、STDOUT に出力された XML を使用してデータを Enterprise Manager に送信します。このため、EPAgent データは、以下のように CA APM ChangeDetector で扱える形式にする必要があります。

```
<changeData dataSource="dataSource name">
<resource name="resource1" value="resource1 value"/>
<resource name="resource2" value="resource2 value"/>
</changeData>
```

注: XML を STDOUT に出力する場合、データは 1 行にする必要があります（上記のサンプル XML は読みやすいように複数行になっています）。STDOUT に 1 行で出力すると、以下のようになります。

<changeData dataSource="dataSource name"><resource name="resource1" value="resource1 value"/><resource name="resource2" value="resource2 value"/></changeData>

各 changeData エレメントには、dataSource 属性が定義されている必要があります。CA Introscope Workstation で CA APM ChangeDetector のデータを表示するときに、この属性に定義された名前が表示されます。各 changeData エレメントは、特定のデータ ソースにリンクしています。 複数のデータ ソース上の CA APM ChangeDetector のデータをレポートする場合は、それぞれの changeData エレメントを別の行に出力する必要があります。

changeData エレメントには、リソース エレメントをいくつでも含めることができます。 各リソース エレメントには、属性を 2 つ定義する必要があります。1 つ目の name 属性は、レポート対象のリソース名に対応し、2 つ目の value 属性は、 レポート対象のリソースの現在値に対応します。

changeData エレメントを出力するたびに、すべての既知のリソースの現在の状態を含める必要があります。

- 同じデータソースで、前回 changeData エレメントを出力したときに含まれていなかったリソースを含めると、そのリソースはシステムに追加されたと判断されます。
- リソースの value 属性が、2 回のイタレーション間で変化した場合、そのリソースは変更されたと判断されます。
- 同じデータソースで、前回 changeData エレメントを出力したときに含まれていたリソースを含めないと、そのリソースはシステムから削除されたと判断されます。

オプションの Enterprise Manager プロパティ

以下のプロパティは、*IntroscopeEnterpriseManager.properties* で設定できます。

- *introscope.changeDetector.storage.purge.maxDataAgeInDays*=<value>

  変更が保存される最大経過期間を日数で指定します。この期間が過ぎると、変更はデータ ストアから削除されます。 デフォルト値は 90 です。

- *introscope.changeDetector.storage.purge.offsetHour*=<value>

  パージが有効にされている場合に、パージが行われる時刻を指定します。指定しない場合は、デフォルトで 4（午前 4 時）に設定されます。

- *introscope.changeDetector.storage.perst.dbfile*=<path to storage file>

  ChangeDetector データに使用されるストレージ ファイルの場所を指定します。相対パスと完全パスのどちらでも指定できます。デフォルトは、*{$EM_INSTALL}/data/changes.db* です。

- *introscope.changeDetector.jdbc.maxNumRows*=<value>

  SQL クエリで処理できる最大行数を指定します。実行されたクエリは、Enterprise Manager にレポートされたすべての変更を行形式で返します。変更は *changes.db* ファイルからレポートされます。デフォルト値は 10000 です。

第 3 章: CA APM ChangeDetector データの表示

CA APM ChangeDetector は CA Introscope に直接統合されるため、CA Introscope でアプリケーション環境の変更を簡単に監視できるようになります。

CA APM ChangeDetector をインストールすると、CA Introscope に以下の変更が表示されます。

- CA Introscope コンソールでは、CA APM ChangeDetector ダッシュボードに変更データの概要が表示されます。
- Workstation では、［変更］タブに変更データが階層化されたツリー ビューまたはテーブル ビューで表示されます。
- Workstation では、変更ビューアでオプションを設定して、表示する変更を指定できます。
- CA Introscope メトリック グラフおよびダッシュボードでは、注釈により変更イベントが識別できます。
- CA Introscope のレポートでは、変更履歴を確認できます。

注: すべての CA Introscope エージェントが Enterprise Manager に接続する前に MOM に接続した場合は、変更の数と CA Introscope エージェントの数が誤って Workstation に表示されます。この問題を修正するには、数分間、別のノードをクリックするか、別の Investigator を開きます。

このセクションには、以下のトピックが含まれています。

CA Introscope での変更データの表示 (P. 67)
グラフおよびレポートでの変更データの表示 (P. 82)

CA Introscope での変更データの表示

CA APM ChangeDetector をインストールして構成するときには、監視する CA Introscope エージェントを特定し、CA APM ChangeDetector が収集する変更データのタイプを指定します。 また、Workstation で変更データを表示してレポートを作成できます。

Workstation では、以下のように変更データを表示できます。

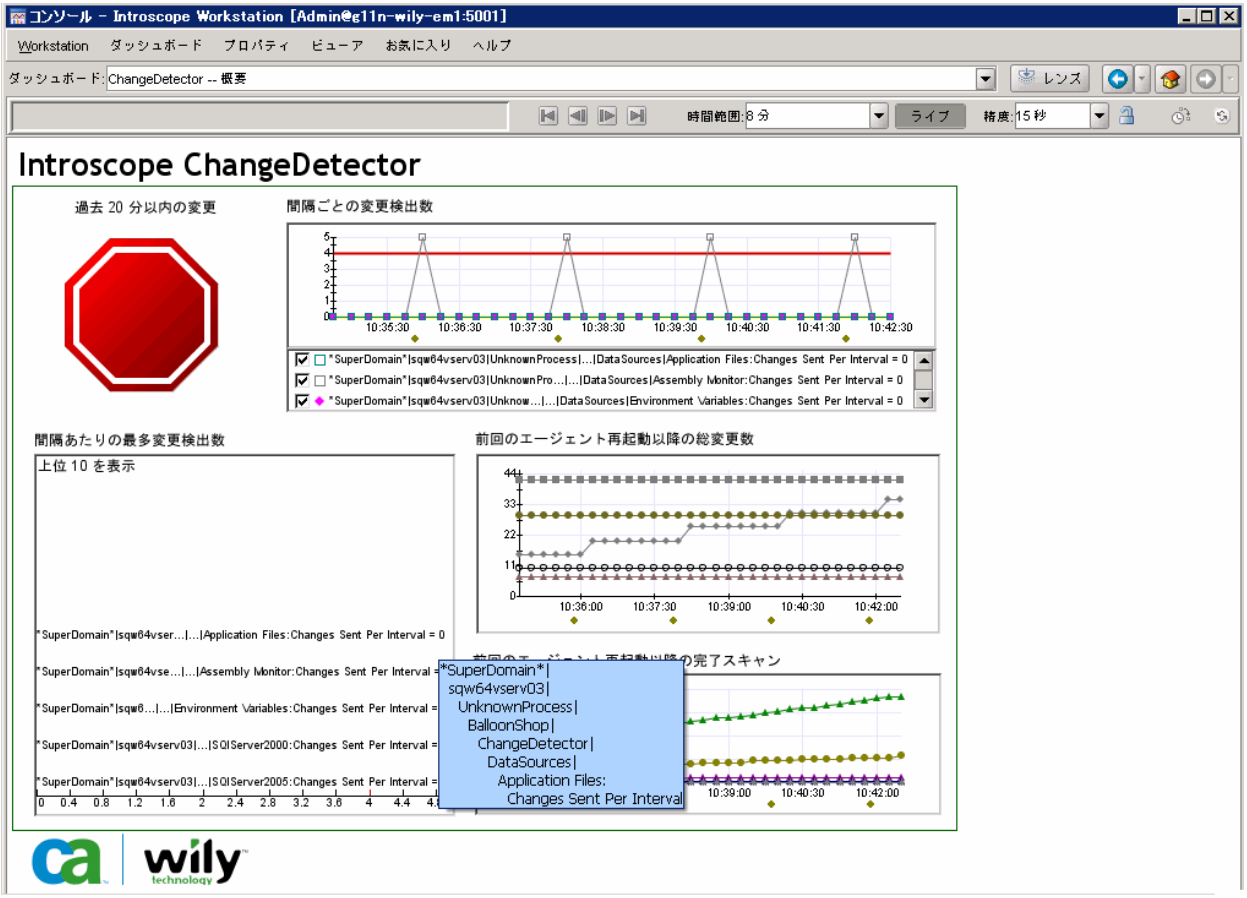
- CA APM ChangeDetector（69P. ）ダッシュボード
- Investigator のツリー ビュー
- Investigator のテーブル ビュー テーブル ビューには、ツリー ビューで選択したリソースの情報が別の形式で表示されます。

Investigator の変更テーブルには、［変更ビューア ウィンドウの起動］ボタンがあります。 Change Viewer には、表示する変更データをさらに細かく制御するためのオプションがあります。

注: すべての CA Introscope エージェントが Enterprise Manager に接続する前に MOM に接続した場合は、変更の数と CA Introscope エージェントの数が誤って Workstation に表示されます。この問題を修正するには、数分間、別のノードをクリックするか、別の Investigator を開きます。

CA APM ChangeDetector ダッシュボードでの変更データの表示

CA Introscope コンソールでは、CA APM ChangeDetector ダッシュボードに変更データの概要が表示されます。



ChangeDetector ダッシュボードには、変更データが以下のように表示されます。

- アラート インジケータには、過去 20 分間の変更が表示されます。

  注: CA APM ChangeDetector は、過去 20 分間に変更があったかどうかを確認します。ただし、これは過去 20 分間の累積値ではありません。たとえば、過去 20 分間に変更が 10 存在する場合でも、これらの変更が別の期間に送信された場合は、黄色のインジケータになります。インジケータが赤色になるのは、1 つの期間で 5 つ以上の変更がある場合のみです。

- CA Introscope エージェントが最後に再起動されてからの、間隔ごとの変更検出数がグラフに表示されます。
- CA Introscope エージェントが最後に再起動されてから完了したスキャン数がグラフに表示されます。

CA APM ChangeDetector を開く操作

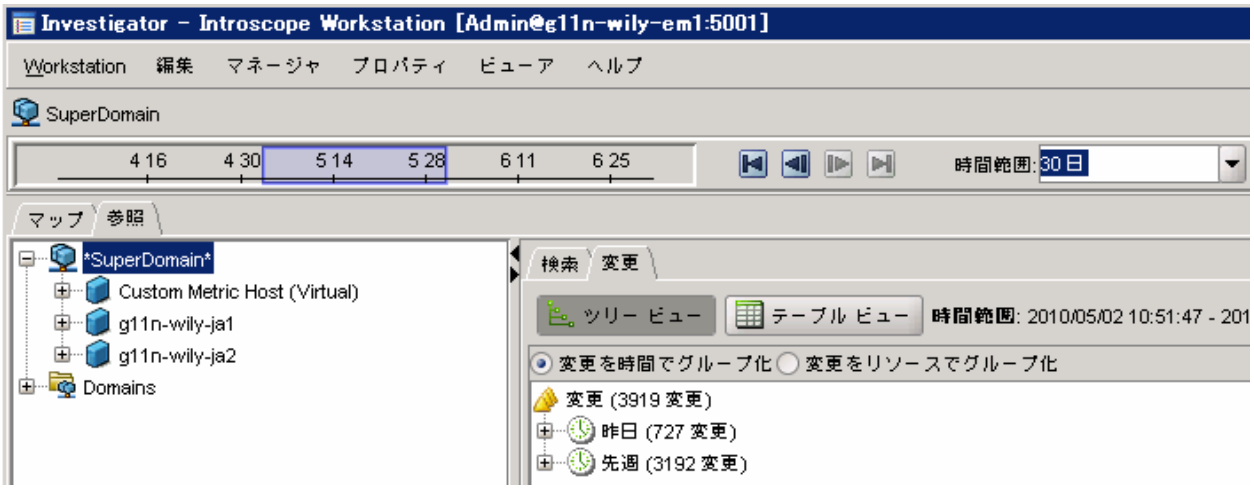
CA APM ChangeDetector をインストールすると、CA Introscope Workstation に［変更］タブが追加されます。

CA APM ChangeDetector を開く方法

- 新しい CA Introscope Investigator を開き、［変更］タブをクリックします。 CA APM ChangeDetector がツリー ビュー モードで開きます。

ツリー ビューでの変更データの表示

Investigator の CA APM ChangeDetector のツリー ビューには、ルート ノードの［*変更*］に変更データが階層化され、さらに日付と時間でグループ化されて表示されます。 ツリー ビューは、デフォルトの Investigator ビューです。



CA Introscope Investigator のツリーで CA Introscope エージェントの ChangeDetector ノードをクリックするか、または CA Introscope エージェントの ChangeDetector データ ソースのいずれかをクリックすると、特定のエージェントの変更データも表示できます。

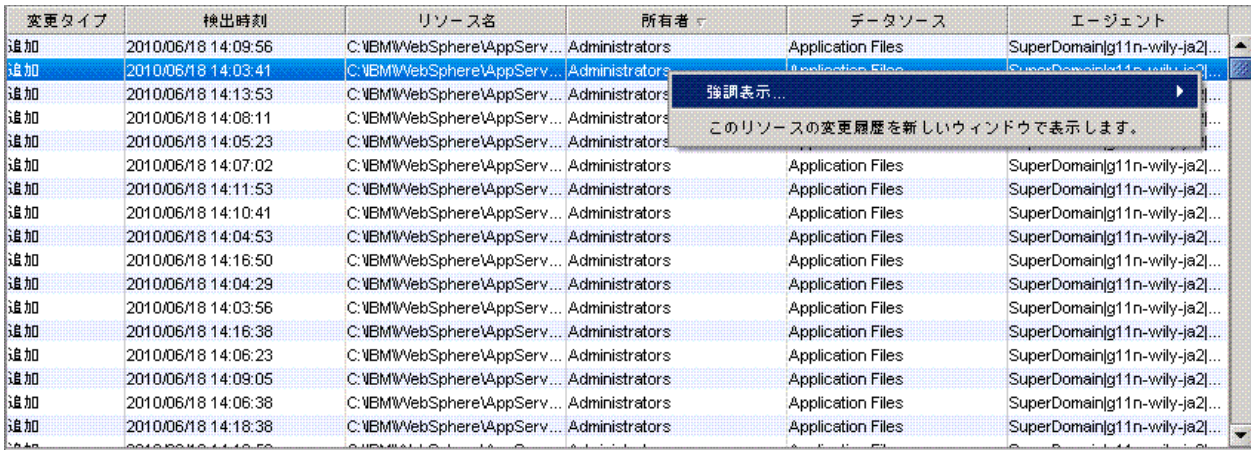
注: ChangeDetector データ ソースがシャット オフされると、［一般］タブのメトリック データは表示されなくなりますが、［変更］タブの変更データはそのまま表示されます。

Introscope の［変更］タブには、先週の変更データと、セッション中の変更のライブ データが表示されます。 別の対象期間を選択すると、その対象期間の変更データが表示されます。

ChangeDetector 構成パラメータを使用して、デフォルトの期間を変更して変更データを表示できます。詳細については、13 ページの「*ChangeDetector のインストールと設定*」を参照してください。

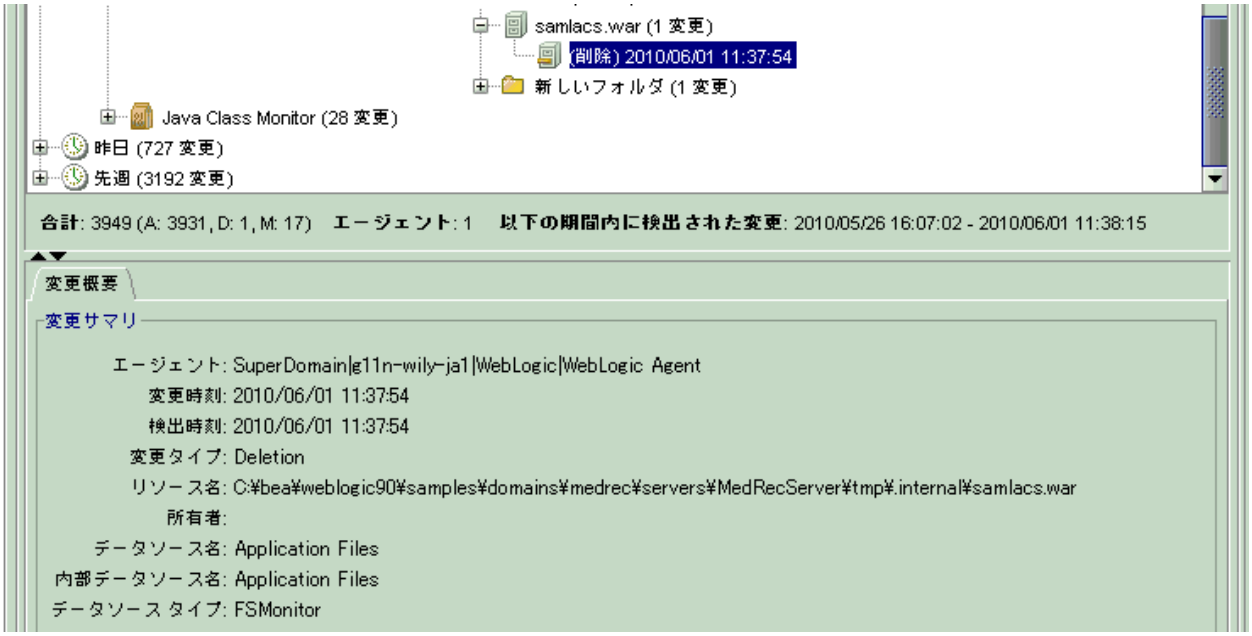
ツリー ビューの変更データは、時間ごと、またはリソースごとにグループ化できます。

いくつかの変更があるノードをクリックすると、下部のペインに、そのノードの変更イベントがすべてテーブル形式で一覧表示されます。

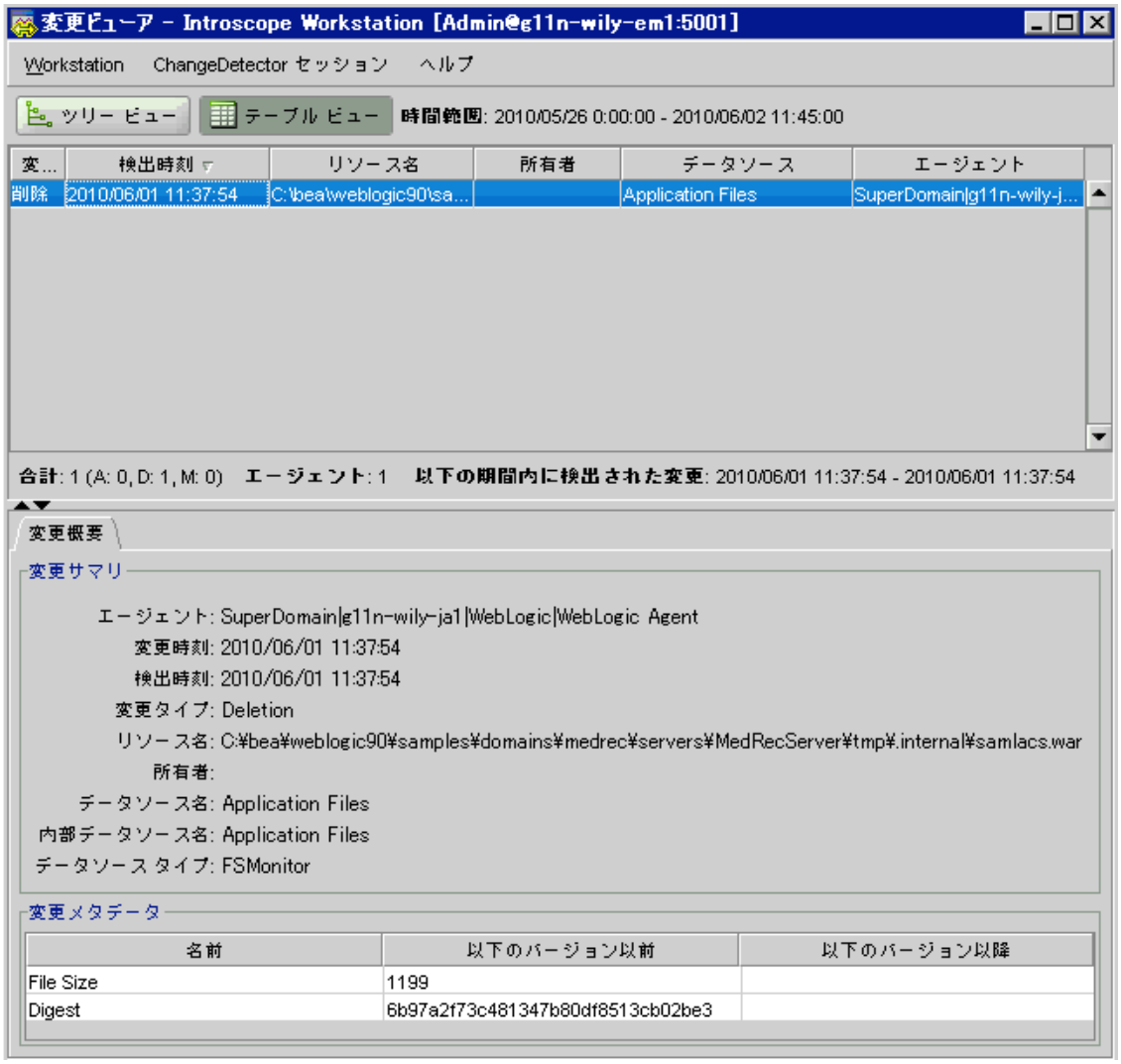


下部のペインの変更イベントを右クリックすると、さらに別の表示オプションが選択できます。

- ［強調表示］を選択すると、テーブルが強調表示されて特定の種類の変更がさらに見やすくなります（「テーブルの強調表示の使用（P. 81）」を参照）。
- ［この変更を親ビューで選択］をクリックすると、その変更のツリービューが開くと共に、［変更概要］タブにその特定の変更データが表示されます。



- ［このリソースの変更履歴を新しいウィンドウで表示します］をクリックすると、別のウィンドウにテーブル ビューが開いて、選択したリソースの変更履歴が表示されます。



[変更概要]タブでの情報の表示

ツリーで変更を 1 つ選択すると、必ず［変更概要］タブが開きます。［変更概要］タブには、常に［変更サマリ］が表示されます。 さらに、［変更メタデータ］および［変更詳細］パネルが表示される場合もあります。

- **変更サマリ**

  変更を 1 つ選択すると［変更概要］タブに表示されます。 変更サマリには、変更に関する基本情報が表示されます。表示される情報は、エージェント ID、変更時刻（ファイルのみ。以下の注記を参照）、検出時刻、変更の種類、リソースとデータソースの名前、およびデータソースの種類です。

- **変更メタデータ**

  変更データにメタデータが含まれる場合に［変更概要］タブに表示されます。たとえば、ファイルの変更の場合、メタデータには、Last Time Modified と File Size が含まれます。

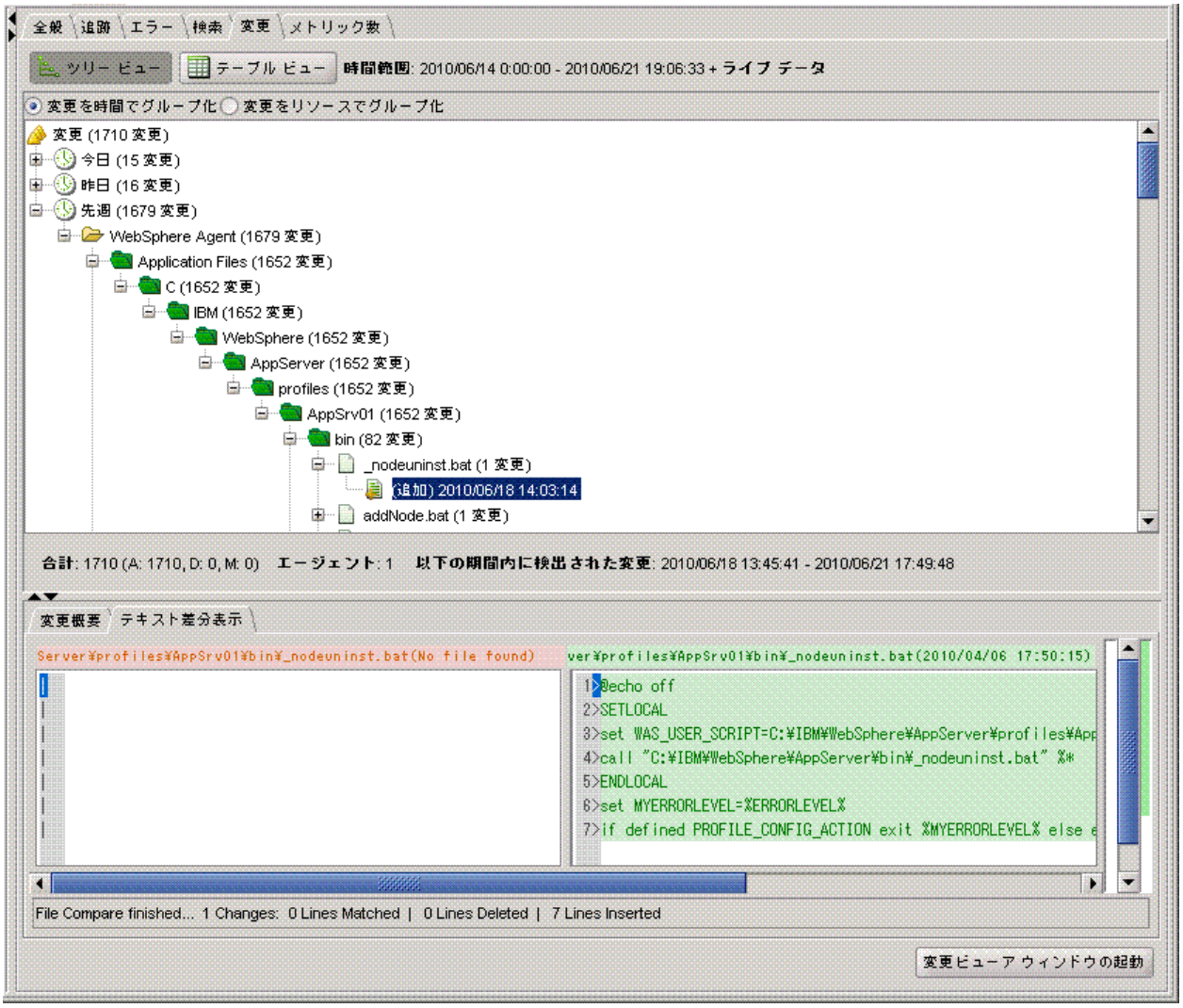
- **変更詳細**

  変更の詳細を［以下のバージョン以前］/［以下のバージョン移行］テーブルに表示できる場合にのみ［変更概要］タブに表示されます。たとえば、環境プロパティ、データベース属性、および Java クラスなどの種類のデータの場合です。

注: ファイルの場合、オプションの Workstation のプロパティを設定すると、CA APM ChangeDetector は変更の検出時刻とファイルの最終変更時刻を区別します（「オプションの構成プロパティ (P. 63)」を参照）。 CA APM ChangeDetector 対応のエージェントを初めて起動した場合、ファイルの最終変更時刻がライブ タイム期間（デフォルトで 7 日）よりも前だと、ライブ モードでファイルの追加イベントが表示されないことがあります。 ただし、［時間範囲］ドロップダウン リストから別の対象期間を選択すると、現在の時間範囲で表示されていないデータにアクセスできます。
一部のインスタンスで、「検出時刻」と「変更時刻」の時間差が、スキャンの完了までにかかる時間よりも大きい場合は、CA APM ChangeDetector がスキャン完了後しばらくしてからファイルの変更を検出した可能性があります。これには、さまざまな理由があります。たとえば、既存のファイルを上書きしてファイルの名前が変更されたとします。 CA APM ChangeDetector は、上書きされたファイルの変更イベントとしてこの状況を判断しますが、このファイルの最終変更タイムスタンプは、名前を変更する前のタイムスタンプのままです（この動作は、オペレーティング システムによって異なります）。このため、このような状況が発生します。また、一部のオペレーティング システムでは、システム ユーティリティを使用して、ファイルの最終変更タイムスタンプを変更できます。これによっても、同じ状況が発生します。
変更時刻を表示するように CA APM ChangeDetector を構成するには、「オプションの構成プロパティ (P. 63)」を参照してください。

[テキスト差分表示]タブでの情報の表示

ChangeDetector の［テキスト差分表示］タブを使用して、テキスト ファイルの内容の差異を特定できます。



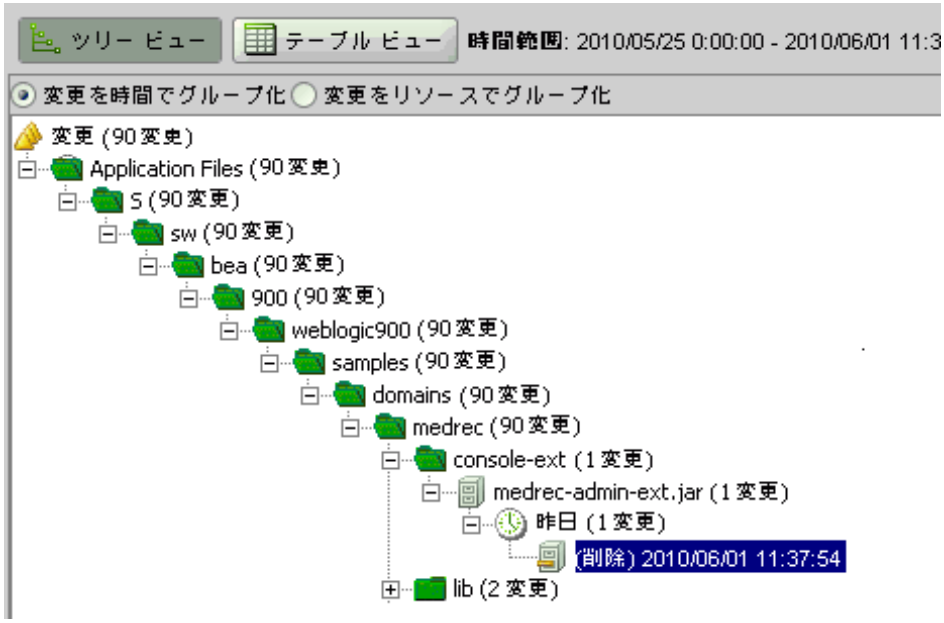
［テキスト差分表示］タブは、以下の変更を選択すると、ChangeDetector のツリー ビューまたはテーブル ビューに表示されます。

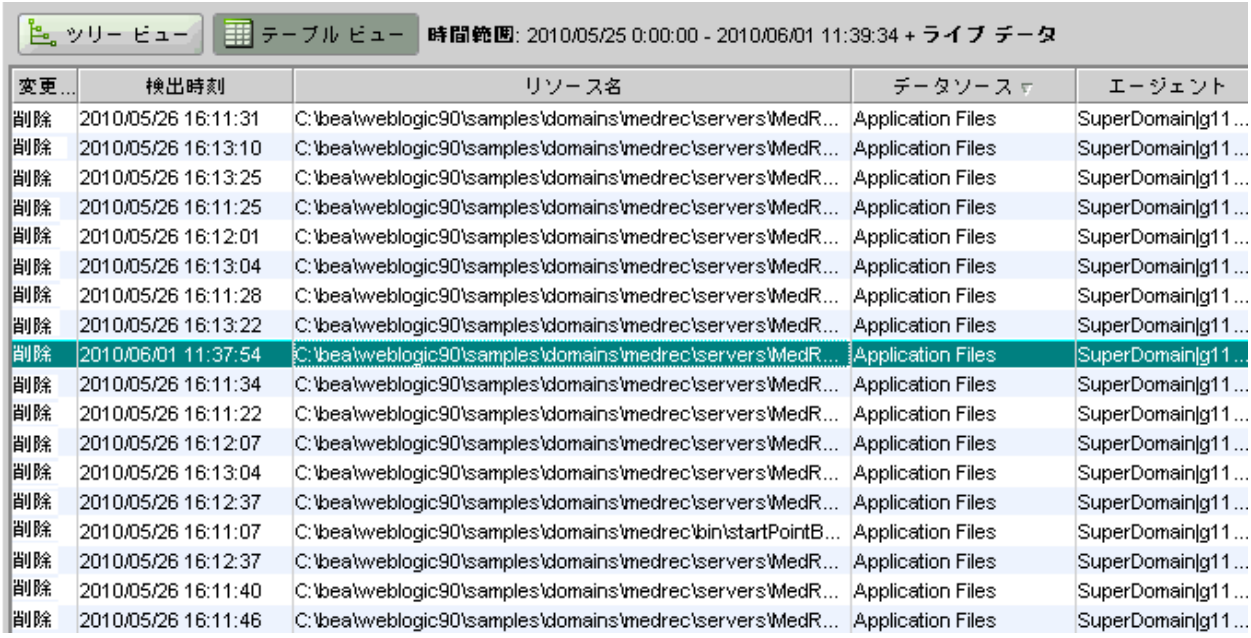
- テキスト ファイル（構成ファイルなど）、および
- 構成ファイル（*ChangeDetector-config.xml*）に定義されている *maxFileSizeToUpload* プロパティの値よりもサイズが小さいファイル

サイズが maxFileSizeToUpload よりも大きいテキスト ファイルは、Workstation の［テキスト差分表示］に表示されません。

テーブル ビューでの変更データの表示

Investigator でリソースを選択すると、そのリソースの変更データをツリー ビューまたはテーブル ビューで表示できます。 ツリー ビューで変更イベント（削除や修正など）を選択すると、選択したイベントの詳細を示す行がテーブル ビューで強調表示されます。



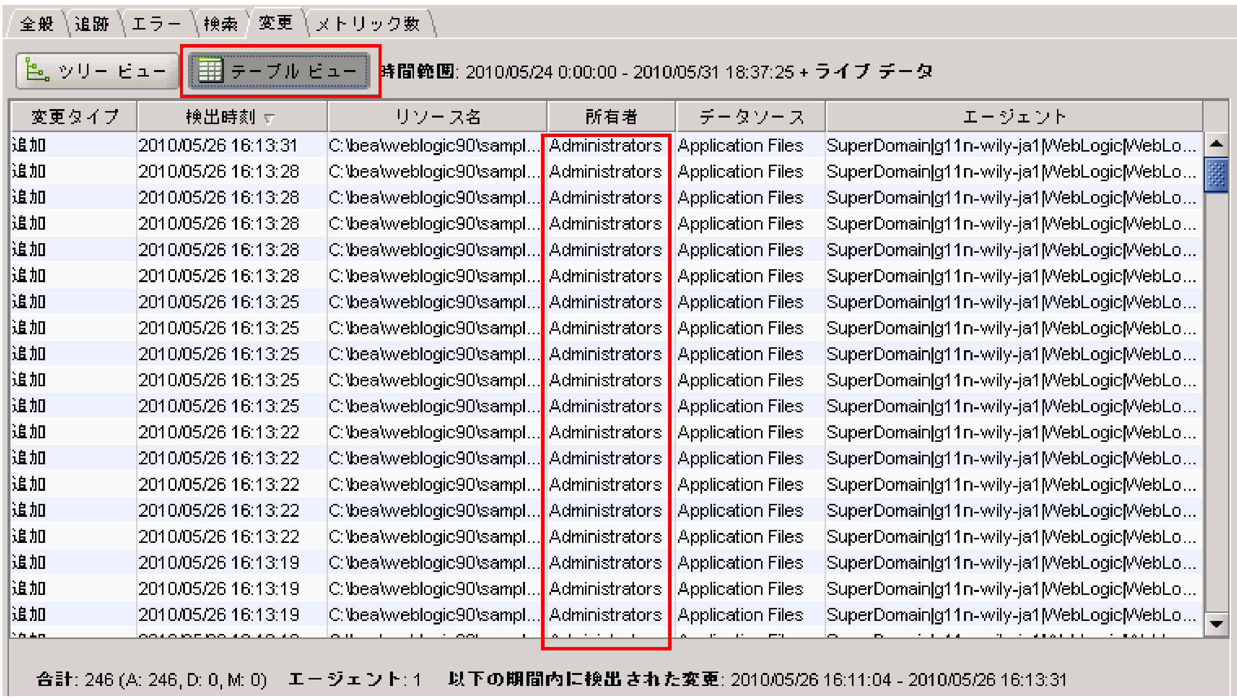

ツリー ビュー　テーブル ビュー　時間範囲: 2010/05/25 0:00:00 - 2010/06/01 11:39:34 + ライブ データ

変更...	検出時刻	リソース名	データソース	エージェント
削除	2010/05/26 16:11:31	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:13:10	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:13:25	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:11:25	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:12:01	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:13:04	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:11:28	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:13:22	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/06/01 11:37:54	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:11:34	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:11:22	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:12:07	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:13:04	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:12:37	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:11:07	C:\bea\weblogic90\samples\domains\medrec\bin\startPointB...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:12:37	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:11:40	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...
削除	2010/05/26 16:11:46	C:\bea\weblogic90\samples\domains\medrec\servers\MedR...	Application Files	SuperDomain\g11...

テーブル ビューに表示されたデータは、変更の種類、検出時刻、リソース名、データ ソース、およびエージェント ID など、さまざまな軸で並べ替えられます。

テーブル ビューでの所有者データの表示

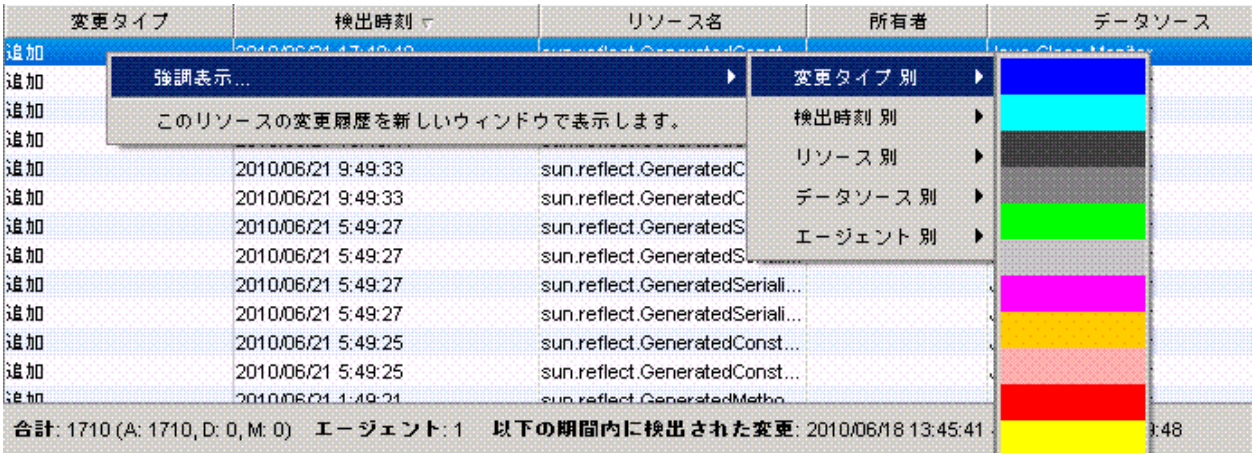
CA APM ChangeDetector は、ファイル システム監視データソースに対して変更を行ったユーザを識別します。 変更を行ったユーザは、［テーブル ビュー］でファイル システム データソースを表示して、［所有者］列のユーザ識別情報を表示するとわかります。



その他のデータソースを表示する場合、［テーブル ビュー］の［所有者］列にはデータが表示されません。

テーブルの強調表示の使用

特定の種類の変更データをわかりやすく表示するために、テーブルビュー内のイベントを強調表示できます。強調表示では、変更の種類、検出時刻、リソース、データソース、またはエージェントごとに色を変更できます。



強調表示を適用する方法

1. 強調表示する特性を持つイベント（たとえば、特定のリソースのイベント、または修正イベントなど）を選択します。
2. イベントを右クリックして、[強調表示]を選択し、[変更タイプ別]、[検出時刻別]、[リソース別]、[データソース別]、または[エージェント別]の中から強調表示する特性を選択します。

   [変更時刻別]を選択した場合は、時間範囲も指定する必要があります。
3. 強調表示の色を選択します。

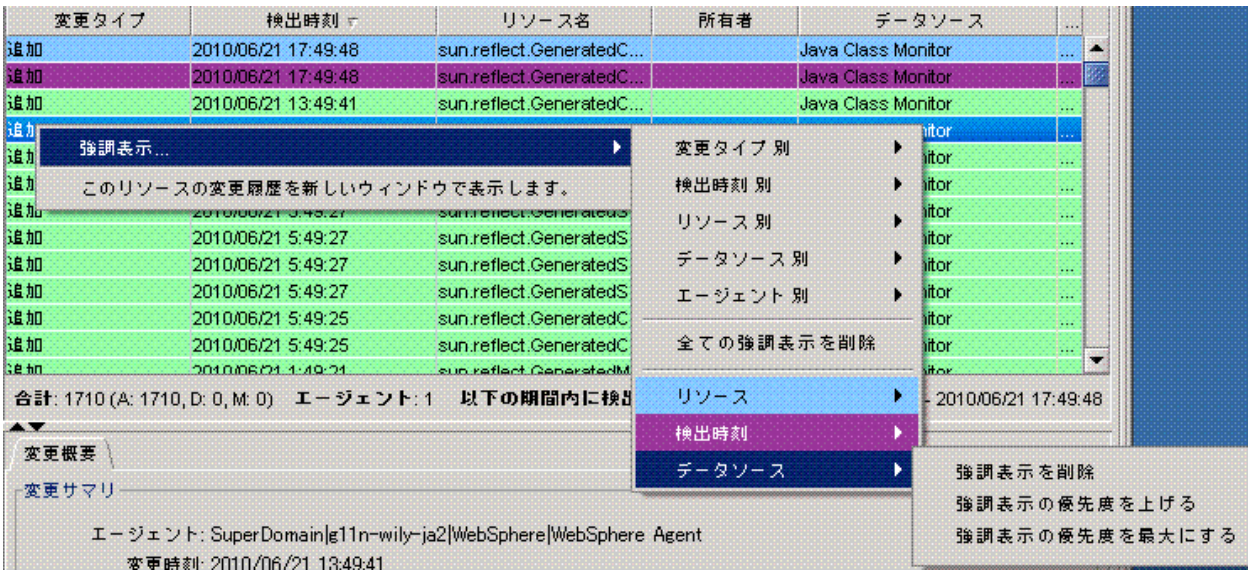
   これにより、選択した特性を持つイベントがすべて強調表示されます。

強調表示の動作

- 強調表示は、定義した順序が優先されて適用され、重複しません。 たとえば、変更タイプを対象にして赤い強調表示を定義した場合、その変更タイプのイベントはすべて赤く強調表示されます。 次に、リソースを対象にして黄色い強調表示を定義すると、赤く強調表示されたイベントの色は変わりません。 黄色の強調表示は、残りのイベントが、選択したリソースに一致した場合に適用されます。

- 強調表示の優先順位はすべて変更できます。
- 各強調表示は個別に削除することも、すべて一度に削除することもできます。

  以下の例の［強調表示］メニューを参照してください。 3 つの強調表示が定義されています (リソース、検出時刻、およびデータ ソース) 。

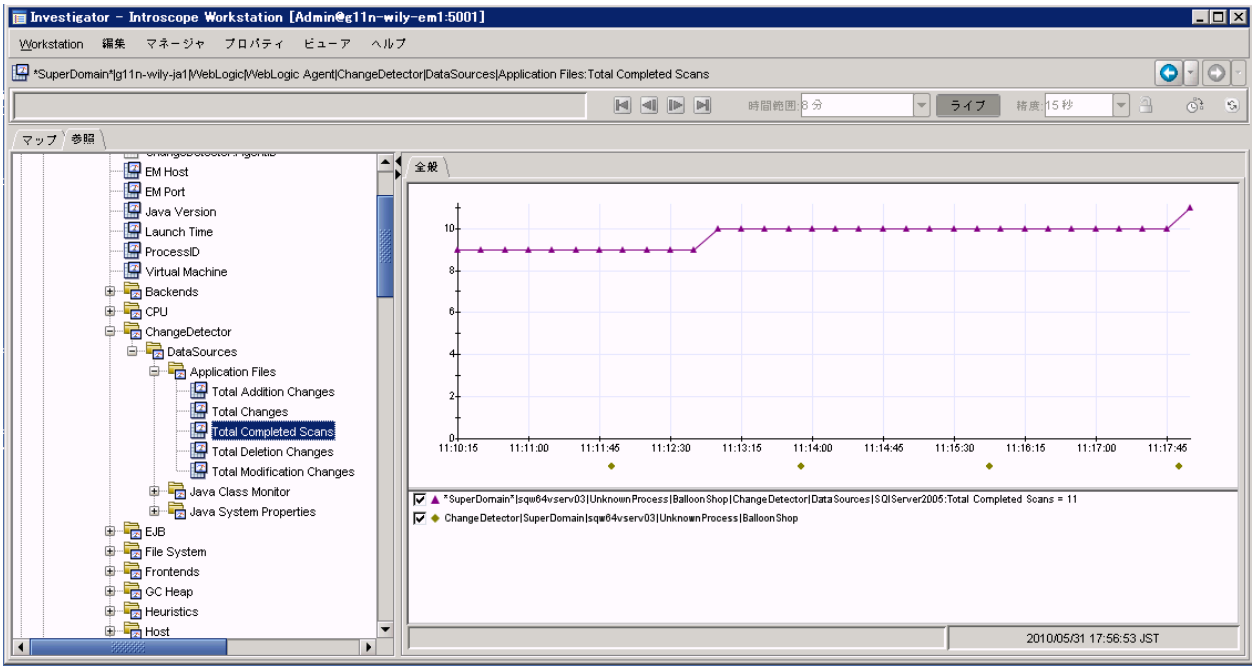


グラフおよびレポートでの変更データの表示

CA APM ChangeDetector は、変更データに関する情報を CA Introscope のグラフに統合します。また、過去 24 時間にシステムで発生したすべての変更を表示するレポート テンプレートを提供します。

統合された CA APM ChangeDetector グラフ

CA APM ChangeDetector は、変更データに関する情報を CA Introscope のグラフに統合します。変更データは、以下の図に示すように、グラフの X 軸の下とヒントにグラフの注釈として表示されます。



CA APM ChangeDetector のグラフ注釈の表示

CA APM ChangeDetector のグラフ注釈には、凡例に対応する記号と色が使用されるため、変更の起源を簡単に特定できます。以下に、ChangeDetector のグラフ注釈の機能の一部を示します。

- 注釈の上にマウス ポインタを置くと、ヒントが開いて詳細情報が表示されます。

  複数のグラフ注釈の位置が近すぎて識別できなくなる可能性があるため、CA APM ChangeDetector では最大で 5 つの注釈を積み上げてヒントに表示し、同時に読めるようにします。 注釈が 6 つ以上ある場合は、ウィンドウの上部に注釈の総数を示すメッセージが表示されて、最初の 5 つの注釈のみが表示されます。

- グラフ注釈をダブルクリックすると、Change Viewer が開かれて、注釈の両側にある 2 本のグリッド線の間で発生した変更が一覧表示され、注釈自体に対応するイベントがハイライト表示されます。
- グラフに表示される注釈は、グラフに現れているメトリックによって異なります。たとえば、グラフ内に CA Introscope エージェント A、B、および C からのメトリックが表示されているときに、エージェント A と B のみが CA APM ChangeDetector 対応である場合は、エージェント A と B からの、指定期間内の変更の注釈のみが表示されます。
- 指定期間内に変更がない場合は、変更データが表示されず、グラフ最下部の変更の注釈も表示されません。

CA Introscope グラフの詳細については、「CA APM Workstation ユーザ ガイド」を参照してください。

表示されるグラフ注釈の指定

グラフに表示される注釈を指定できます。 たとえば、特定のデータ ソースまたはリソースからの注釈のみを表示したい場合や、ある CA Introscope エージェントから収集された変更を、別の CA Introscope エージェントから収集されたメトリックと相関させたい場合に、この機能は便利です。

表示される変更は、以下の 2 種類の方法で指定できます。

- Investigator で、［プロパティ］>［グラフ注釈オプション］を選択します。
- グラフを右クリックして、［グラフ注釈オプション］を選択します。

グラフの注釈を指定する方法

1. ［グラフ注釈オプション］ダイアログ ボックスを開きます。
2. ［デフォルト オプションを使用］または［カスタム オプションを使用］を選択します。

   デフォルト設定

   グラフにメトリックが表示されている CA APM ChangeDetector 対応エージェントのグラフ注釈のみを表示します。

   たとえば、グラフ内に CA Introscope エージェント A、B、および C からのメトリックが表示されているときに、エージェント A と B のみが CA APM ChangeDetector 対応である場合は、エージェント A と B からの、指定期間内の変更の注釈のみが表示されます。エージェント D からの変更は、エージェント D が CA APM ChangeDetector 対応であるかどうかに関係なく、グラフには表示されません。これは、エージェント D からのメトリックがグラフに含まれていないからです。

   カスタム設定

   グラフ注釈のラベルを付ける CA APM ChangeDetector コンポーネントを選択します。

   注: 注釈オプションの変更は、現在のグラフにのみ適用されます。 グラフから他の場所に移動すると、変更は維持されません。

組み込み CA APM ChangeDetector レポートの実行

CA APM ChangeDetector には、過去 24 時間にシステムで発生したすべての変更が表示されるレポート テンプレートがあります。 このレポートは CA Introscope エージェント別にグループ化され、エージェントごとのサマリとレポート サマリがあります。

*過去 24 時間での変更*レポートを実行する方法

1. ［Workstation］-［レポートを生成］を 選択して、［レポート テンプレートを選択］ダイアログ ボックスを開きます。

   リストに、*［過去 24 時間での変更］*レポート テンプレートが含まれています。

2. ［過去 24 時間での変更］を選択し、［選択］をクリックして、［レポートを生成］ダイアログ ボックスを開きます。

3. レポートの開始および終了日を指定します。

   注: レポートの対象時間は、レポートを生成する Workstation のタイムゾーンに応じて計算されます。

4. CA APM ChangeDetector 対応のエージェントのリストからエージェントを選択するか、別のエージェント式を指定して、テンプレートのエージェント式を上書きします。

5. ［プレビューを生成］をクリックします。

   プレビューにレポートのタイトル ページが表示されます。

6. プレビューの各ボタンを使用して、レポートの出力を操作し、レポートを保存します。

CA Introscope レポートへの CA APM ChangeDetector エレメントの追加

組み込みの*［過去 24 時間での変更］*レポート テンプレートのほかに、CA APM ChangeDetector には CA Introscope のレポート機能も組み込まれているため、CA APM ChangeDetector エレメントを追加することによってカスタム レポートを作成できます。

カスタム CA APM ChangeDetector レポートを作成する方法

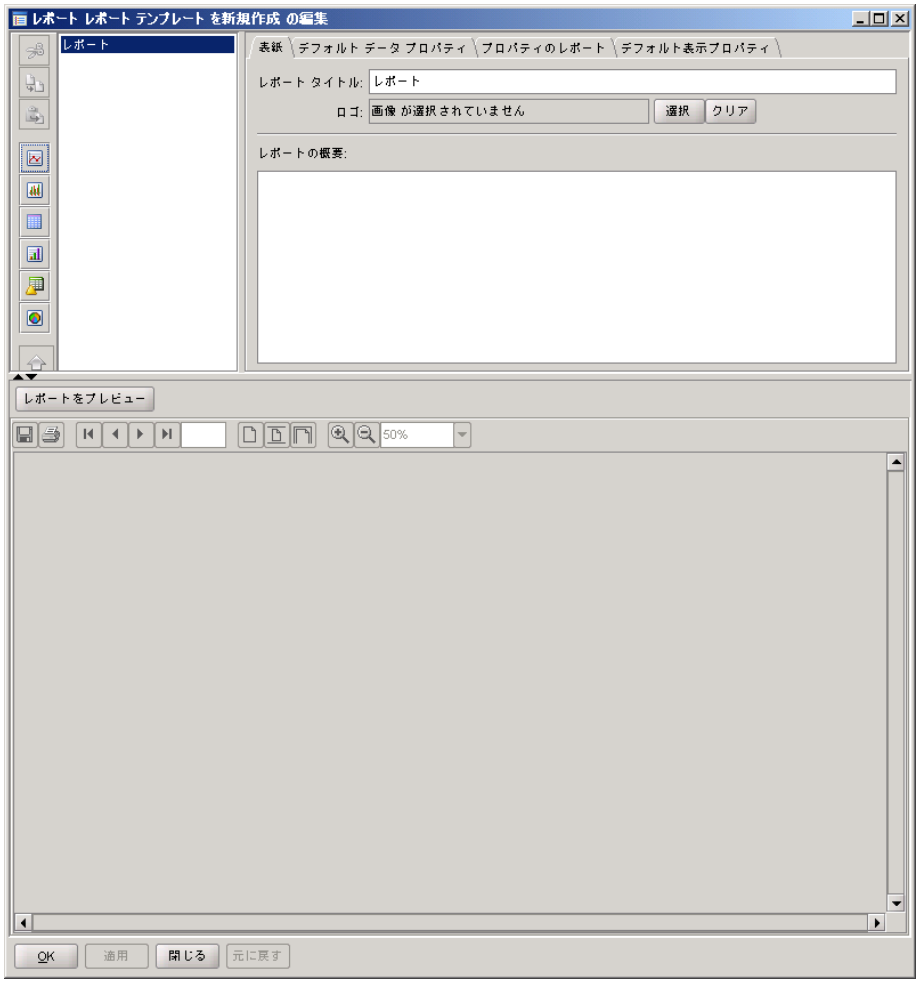
1. Workstation の Investigator で、［ファイル］-［新規管理モジュール エディタ］を選択します。
2. ［エレメント］-［レポート テンプレートを新規作成］を選択します。
3. 新しいレポート テンプレートの名前を入力し、［強制的に一意にする］をクリックして、レポートの名前が一意になるようにします。

   ［強制的に一意にする］を選択すると、ユーザが入力した名前が一意でない場合は、CA Introscope により番号が追加されて一意の名前に変更されます。追加される番号は、作成後のレポート テンプレートを 管理モジュール エディタに表示したときに表示されます。［強制的に一意にする］をオンにしなかった場合、同じレポート テンプレート名が存在するときは、CA Introscope でエラー メッセージが表示され、レポートは作成されません。
4. レポートに含める管理モジュールを選択して、［OK］をクリックします。

   ［レポート テンプレートを新規作成］ダイアログ ボックスが表示されます。
5. ［アクティブ］チェック ボックスをクリックして、レポート テンプレートをアクティブにし、CA Introscope コンソール、Investigator、および管理モジュール エディタのレポート テンプレートのリストに表示されるようにします。

6. ［テンプレート エディタを開く］をクリックします。

   レポート テンプレートが開き、レポートの目次でで選択できるオプションが表示されます。



注: レポートの期間中に発生した変更を示す方法として、変更データ用のレポート オプションのみを追加することも、そのオプションと一緒にメトリック グラフを追加することもできます。

7. ［表紙］タブをクリックしてレポートの目的を入力します。

   レポート タイトル

   生成されるレポートのタイトルを入力します。タイトルは、目次と共にタイトル ページに表示されます。

   ロゴ

   レポートと関連付けるロゴを選択します。

   レポートの概要

   レポートの概要を説明するテキストを入力します。

8. ［デフォルト データ プロパティ］タブをクリックして、プロパティを指定します。

   デフォルトでは、データ プロパティは変更できません。ただし、［変更時間を使用］をオフにして、ファイルの実際の変更時刻ではなく、変更検出数を表示するようにできます。

   デフォルトの対象期間を変更するには、［テンプレートのデフォルト時間範囲］以下の次のフィールドを変更します。

   開始時刻と終了時刻

   対象期間を指定する場合は、特定の開始日および終了日、または「*24 時間*」などの期間を指定できます。

   レポートの対象期間は、以下のいずれかの方法で指定できます。

   a. 特定の開始および終了日時を入力するか、カレンダ アイコンをクリックして開始および終了日を選択します。

   b. ［開始時刻］を空白にし、［継続時間］および［単位］パラメータを使用してレポートの実行期間を指定します。

   c. ［終了時刻］を空白にし、［継続時間］および［単位］パラメータを使用してレポートの実行期間を指定します。

   d. ［終了時刻］に「現在」と入力し、［継続時間］および［単位］パラメータを使用して、過去の履歴をどれだけさかのぼってレポートするかを指定します。

継続時間

レポートを実行する期間を指定する数字を入力します。 この数値は、［単位］との組み合わせで機能します。たとえば、［単位］が［*時間*］の場合は、［継続時間］フィールドに「*24*」と入力できます。

注:［継続時間］および［単位］パラメータと［開始時刻］および［終了時刻］を組み合わせた場合の機能については、開始時刻と終了時刻の説明を参照してください。

単位

ドロップダウン リストから時間の単位を選択します。設定値は、［分］、［時間］、［日］、または［週］です。

9. ［表示プロパティ］タブをクリックし、レポート生成後のレポートのグラフおよびテーブルの表示を決定するプロパティを設定します。

並べ替え基準

この行をクリックしてリストを開き、［タイムスタンプ］、［データソース］、［リソース］、または［変更タイプ］から選択します。 レポート内の変更はすべて、最初はエージェント ID ごとにグループ化されます。［並べ替え基準］フィールドでは、各エージェント内での 2 番目のグループ化を指定します。

並べ替え順序

この行をクリックしてリストを開き、［昇順］または［降順］を選択します。

最大行数

レポートに表示する最大行数を入力します。

レポートの作成と生成の詳細については、「CA APM Workstation ユーザ ガイド」を参照してください。

第 4 章: CA APM ChangeDetector メトリック

CA APM ChangeDetector は、Enterprise Manager および CA Introscope エージェント用のサポータビリティ メトリックをレポートします。

このセクションには、以下のトピックが含まれています。

Enterprise Manager 用の CA APM ChangeDetector サポータビリティ メトリック

Enterprise Manager 用のサポータビリティ メトリックは、*Enterprise Manager|ChangeDetector|DataStore* ノードの下の仮想エージェント下に配置できます。

メトリックの詳細については、以下のトピックを参照してください。

Avg Time For Insertion (ms)

1 つの CA APM ChangeDetector データ ポインを Enterprise Manager 側データベースに挿入するのにかかる、ミリ秒単位の平均時間。

Datastore Used (%)

CA APM ChangeDetector データを格納するために割り当てられたデータベース スペースのパーセント。 値が 100 パーセントに到達し、さらに別のデータ ポイントが挿入される場合、データベースはより多くのスペースを必要とします。

Number of Insertions

データベースに挿入された CA APM ChangeDetector 変更の数。

Number of known agents

Enterprise Manager がデータをレポートする、CA APM ChangeDetector 対応の CA Introscope エージェントの数。 この値はユーザ権限およびエージェント接続性に応じて異なることがあります。

Number of Changes (database)

現在データベース内に格納されている CA APM ChangeDetector 変更の数。

Size of Datastore

データ ストアのバイト単位のサイズ。

注: *Avg Time For Insertion (ms)* メトリックと *Number of Insertions* メトリックの値は、Enterprise Manager の寿命においてのみ有効です。Enterprise Manager が再起動されると、これらのメトリックはゼロで初期化されます。

変更の数（CA Introscope）

各 CA Introscope エージェントから送信された CA APM ChangeDetector 変更の数。

CA Introscope 用の CA APM ChangeDetector サポータビリティ メトリック

CA APM ChangeDetector は、スキャンが完了するか、データ ポイントが送信されると、構成時に定義されたすべてのデータ ソースに対する CA Introscope エージェントのエージェント サポータビリティ メトリックをレポートします。すべてのメトリック合計は、CA Introscope エージェントの寿命においてのみ有効です。これは、CA Introscope エージェントが再起動されると、すべてのカウンタがゼロで初期化されるためです。

エージェント サポータビリティ メトリックは、Workstation Investigator ツリーの ChangeDetector|DataSources ノードで表示できます。

Changes Sent Per Interval

CA Introscope によって間隔ごとに送信された変更の数。間隔ごとにカウンタはゼロで初期化されます。

Total Addition Changes

このデータ ソースに関して CA Introscope によって送信された追加変更の総数。

Total Completed Scans

このデータ ソースに関して完了したスキャンの数。

Total Changes

このデータ ソースに関してエージェントによって送信された変更の総数。

Total Deletion Changes

このデータ ソースに関して CA Introscope によって送信された削除変更の総数。

Total Modification Changes

このデータ ソースに関して CA Introscope によって送信された修正変更の総数。

付録 A: サンプル構成ファイル

CA APM ChangeDetector をインストールすると、サンプルの標準構成ファイルがインストールされます。

構成ウィザードを使用すると、使用環境に合わせて簡単に CA APM ChangeDetector を構成 (P. 19)できます。

環境に合わせて構成ファイルを変更する必要がある場合は、*ChangeDetector-config.xml* または *ChangeDetector-configDotnet.xml* に基づいてカスタムの CA APM ChangeDetector 構成を作成します。

この章では、カスタム構成の作成方法について詳しく説明し、カスタム構成ファイルの例を示します。

このセクションには、以下のトピックが含まれています。

サンプル Java ChangeDetector-config.xml ファイル

以下に、Java 用のカスタム CA APM ChangeDetector 構成の例を示します。

注: 以下の XML 例はサンプル データです。 現在のサンプル Java ChangeDetector-config.xml ファイルの内容が示されることがあります。また、以前のバージョンの CA APM ChangeDetector の内容が示されることもあります。

```
<change-detector>
<!--
#################################################################
# Introscope ChangeDetector Configuration
#
# CA Wily Introscope(R) ChangeDetector Version 8.1
# Copyright (c) 2008 CA. All Rights Reserved.
# Introscope(R) is a registered trademark of CA.
#################################################################
-->

    <!-- ======================================================== -->
    <!-- FILE CHANGE MONITORING                                   -->
    <!-- ======================================================== -->
    <!--
    スキャン ディレクトリの構成
    修正が必要な唯一の構成エレメントは、
    <scan-directory> エレメントの「name」属性です。これは、
    この <datasource-instance> エレメントの最下部の近くにあります。
    この「name」属性には、ChangeDetector でスキャンする
    ディレクトリへのパスを指定する必要があります。 一般的に、
    スキャンするディレクトリは、アプリケーション サーバの Web アプリケーション
    がデプロイされているディレクトリと構成ディレクトリです。

    ファイル タイプのカスタマイズ
    デフォルトで、以下の <fileset> エレメントで指定されたファイル タイプが
    現在の作業ディレクトリでスキャンされます。
    これらのファイル タイプでは、変更された場合にアプリケーションのパフォーマンスに
    影響を与える可能性が高いエレメントを表します。
    特殊なファイル タイプをスキャンする必要がない限り、
    これらのエレメントを修正する必要はありません。 -->
        <datasource-instance name="Application Files" type="file" version="8.1">

                <property name="explodeArchiveFiles" value="false" />

                <!-- Accepted units are hour, min, sec -->
                <property name="delayBetweenIterations" value="3" unit="sec" />

                <property name="filesPerIteration" value="5" />
```

```
                <property name="delayBetweenArchiveIterations" value="10"
unit="sec" />

                <property name="archiveFilesPerIteration" value="1" />

                <!-- Accepted units are bytes, KBytes, MBytes -->
                <property name="maxFileSizeToUpload" value="50" unit="KB" />

                <property name="useDigest" value="needed" />

                <!-- Web アプリケーションの設定に使用される一般的なファイルをスキャン -->
                <fileset name="config">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)¥.xml$"/>
                                <include pattern="(.*)¥.cmd$"/>
                                <include pattern="(.*)¥.sh$"/>
                                <include pattern="(.*)¥.bat$"/>
                        </exclude>
                </fileset>

                <!-- Web アプリケーションのコードが含まれる一般的なファイルをスキャン -->
                <fileset name="webElements">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)¥.jsp$"/>
                                <include pattern="(.*)¥.cfm$"/>
                                <include pattern="(.*)¥.js$"/>
                                <include pattern="(.*)¥.html$"/>
                        </exclude>
                </fileset>

                <!-- Java アーカイブをスキャン -->
                <fileset name="archives">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)¥.zip$"/>
                                <include pattern="(.*)¥.jar$"/>
                                <include pattern="(.*)¥.ear$"/>
                                <include pattern="(.*)¥.war$"/>
                        </exclude>
                </fileset>

                <!-- config、webElements、および archive filesets で定義されたファイル
タイプのみをスキャン-->
                <fileset name="default">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)¥.xml$" />
                                <include pattern="(.*)¥.cmd$" />
                                <include pattern="(.*)¥.sh$" />
                                <include pattern="(.*)¥.bat$" />
                                <include pattern="(.*)¥.jsp$" />
```

```
                                <include pattern="(.*)¥.cfm$" />
                                <include pattern="(.*)¥.js$" />
                                <include pattern="(.*)¥.html$" />
                                <include pattern="(.*)¥.zip$" />
                                <include pattern="(.*)¥.jar$" />
                                <include pattern="(.*)¥.ear$" />
                                <include pattern="(.*)¥.war$" />
                        </exclude>
                </fileset>

                <!-- config および webElements で定義されたファイル タイプのみをスキャン
-->
                <fileset name="defaultNoArchives">
                        <exclude pattern="(.*)">
                                <include pattern="(.*)¥.xml$" />
                                <include pattern="(.*)¥.cmd$" />
                                <include pattern="(.*)¥.sh$" />
                                <include pattern="(.*)¥.bat$" />
                                <include pattern="(.*)¥.jsp$" />
                                <include pattern="(.*)¥.cfm$" />
                                <include pattern="(.*)¥.js$" />
                                <include pattern="(.*)¥.html$" />
                        </exclude>
                </fileset>

                <!-- ログ ファイル以外をすべてスキャン -->
                <fileset name="noLogs">
                        <exclude pattern="(.*)¥.err$" />
                        <exclude pattern="(.*)¥.log$" />
                        <exclude pattern="(.*)¥.lok$" />
                        <exclude pattern="(.*)¥.tlog$" />
                        <exclude pattern="(.*)¥.log0(.*)" />
                </fileset>

        <!--
        name 属性の値を変更して、監視対象のディレクトリを指定
        注: すぐに使用する場合には、「default」ファイルセットを使用します。 Java アーカイブ
ファイルを監視対象に含めない場合は、
        代わりに「defaultNoArchive」ファイルセットを使用します。 または、目的のファイルを
監視するように
        ファイルセット エレメントをカスタマイズして、下の scan-directory エレメントを置き換
えるか、新しく追加します。
        Java システム プロパティまたは Introscope エージェント プロパティの値も name 属性
        （またはその他の属性）の値に使用できます（たとえば、
name="${MY_APP_HOME}/filesICareAbout/"） -->
                <scan-directory recursive="true" name="." fileset="default"
                        enabled="true">
                </scan-directory>
        </datasource-instance>
```

```
    <!-- ========================================================= -->
    <!-- DATABASE CHANGE MONITORING                                -->
    <!-- ========================================================= -->
    <!--
    以下の DB 監視用のサンプル設定では、Oracle v$parameter テーブルの
    name/value ペアが 10 分ごとにスキャンされます。

    注: このデータソース インスタンスはデフォルトでコメント化されています。
    これは、環境に固有の接続パラメータが
    必要なためです。 使用しているデータベース設定に
    固有の値を入力してください。 -->
    <!--
    <datasource-instance name="Oracle DB" type="database"
driver="oracle.jdbc.driver.OracleDriver"
driverClasspath="C:¥¥somePathTo¥¥yourOracleDriver.zip"
                        url="jdbc:oracle:thin:@yourdbserver:1521:orcl"
username="username"
                        password="password" version="8.1">
                SQL Server
                        SELECT name, value FROM v$parameter
                </sql>
                <schedule type="repetitive" interval="10" unit="min"/>
        </datasource-instance>
        -->


    <!-- ========================================================= -->
    <!-- JAVA SYSTEM PROPERTIES MONITORING                         -->
    <!-- ========================================================= -->
    <!--
        デフォルトで、すべてのプロパティがシステム プロパティ監視に
    含まれ、監視されます。

    プロパティを除外するには、除外ノードを追加する必要があります。 すべてを除外するのではなく、
一部を除外するには、
    特定の exclude エレメント内に include エレメントを
    ネストさせる必要があります。 以下の例では、
    「java」で始まるプロパティ以外のプロパティがすべて除外されます。
                <exclude pattern=".*">
                        <include pattern="java¥.*"/>
                </exclude> -->
        <datasource-instance name="Java System Properties" type="javaenv"
version="8.1">
        </datasource-instance>


    <!-- ========================================================= -->
```

```
    <!-- JAVA CLASS MONITOR                                                 -->
    <!-- ================================================================= -->
    <!--
    デフォルトで、アプリケーション サーバのクラスは、以下の exclude エレメントを使用して除外
されます。
    exclude エレメントの pattern 属性では、パッケージ名も含め、
    クラス名に一致させる正規表現を定義します。

    すべてを除外するのではなく一部を除外するには、特定の exclude エレメント内に、
    include エレメントをネストさせる必要があります。  以下の例では、
    Java パッケージ内のクラス以外のクラスがすべて除外されます。
                <exclude pattern=".*">
                        <include pattern="java¥.*"/>
                </exclude>

        注: ChangeDetector では現在、クラス監視データソースのインスタンスを JVM につき
                        1 つのみサポートしています。

        注:     いずれの除外パターンにも一致しないクラスはすべて、
                        Java クラス監視に含まれて監視されます。

        -->
        <datasource-instance name="Java Class Monitor" type="classmonitor"
version="8.1">
                <!-- Accepted units are hour, min, sec -->
                <property name="delayBetweenIterations" value="2" unit="sec" />
                <property name="classesPerIteration" value="100" />

                <!-- Wily からのクラスを除外 -->
                <exclude pattern="com¥.wily¥.(.*)"/>

                <!--
                        以下は、ある種のアプリケーション サーバのクラスをスキップさせます。
                        これらのクラスを含めると、変更されることが
                        ほとんどないクラスが大量に追跡されて、
                        アプリケーションのパフォーマンスに影響を与えることがあります。
                -->

                <!-- BEA のクラスを除外 -->
                <exclude pattern="weblogic¥.(.*)"/>
                <exclude pattern="com¥.bea¥.(.*)"/>

                <!-- IBM のクラスを除外 -->
                <exclude pattern="com¥.ibm¥.(.*)">
                        <include pattern="com¥.ibm¥.(.*)jdbc(.*)"/>
                </exclude>

                <!-- SAP のクラスを除外 -->
                <exclude pattern="com¥.sap¥.(.*)"/>
```

```
                <!-- Oracle のクラスを除外 -->
                <exclude pattern="oracle¥.*">
                        <include pattern="oracle¥.jdbc¥.(.*)"/>
                </exclude>

                <!-- Sun のクラスを除外 -->
                <exclude pattern="com¥.sun¥.enterprise¥.(.*)"/>
        </datasource-instance>

        <!-- ========================================================== -->
    <!-- CONFIGURATION PROPERTIES - datasource-type は修正しない -->
    <!-- ========================================================== -->
        <datasource-type name="file"
class="com.wily.rave.agent.ds.file.FileDataSourceConfig" />
        <datasource-type name="database"
class="com.wily.rave.agent.ds.db.DBDataSourceConfig" />
        <datasource-type name="javaenv"
class="com.wily.rave.agent.ds.sysprop.SysPropDataSourceConfig" />
        <datasource-type name="classmonitor"
class="com.wily.rave.agent.ds.classmonitor.RuntimeClassMonitorConfig" />

</change-detector>
```

サンプル .NET ChangeDetectorDotnet-config.xml ファイル

以下に、.NET 用のカスタム CA APM ChangeDetector 構成の例を示します。

注: 以下の XML 例はサンプル データです。 現在のサンプル .NET ChangeDetectorDotnet-config.xml ファイルの内容が示されることがあります。また、以前のバージョンの CA APM ChangeDetector の内容が示されることもあります。

```
<change-detector>
<!--
############################################################
# Introscope ChangeDetector Configuration
#
# CA Wily Introscope(R) ChangeDetector Version 8.1
# Copyright (c) 2008 CA. All Rights Reserved.
# Introscope(R) is a registered trademark of CA.
############################################################
-->
    <!-- ======================================================= -->
    <!-- ENVIRONMENT VARIABLES MONITORING                        -->
    <!-- ======================================================= -->
    <!--
        デフォルトでは、すべての環境変数が環境変数監視によって
        含まれ、監視されます。

    変数を除外するには、除外ノードを追加する必要があります。 すべてを除外するのではなく、一部を除外するには、
    特定の exclude エレメント内に include エレメントを
    ネストさせる必要があります。  以下の例では、
    「windows」で始まるプロパティ以外のプロパティがすべて除外されます。
                <exclude pattern=".*">
                       <include pattern="windows¥.*"/>
                </exclude> -->

     <datasource-instance name="Environment Variables" type="javaenv"
version="8.1">
        <exclude pattern="foo" />
        <exclude pattern=".*bar.*">
          <include pattern="hello" />
          <include pattern=".*world.*" />
        </exclude>
     </datasource-instance>

    <!-- ======================================================= -->
    <!-- FILE CHANGE MONITORING                                  -->
    <!-- ======================================================= -->
    <!--
    スキャン ディレクトリの構成
```

```
修正が必要な唯一の構成エレメントは、
<scan-directory> エレメントの「name」属性です。これは、
この <datasource-instance> エレメントの最下部の近くにあります。
この「name」属性には、ChangeDetector でスキャンする
ディレクトリへのパスを指定する必要があります。 一般的に、
スキャンするディレクトリは、アプリケーション サーバの Web アプリケーション
がデプロイされているディレクトリと構成ディレクトリです。

ファイル タイプのカスタマイズ
デフォルトで、以下の <fileset> エレメントで指定されたファイル タイプが
現在の作業ディレクトリでスキャンされます。
これらのファイル タイプでは、変更された場合にアプリケーションのパフォーマンスに
影響を与える可能性が高いエレメントを表します。
特殊なファイル タイプをスキャンする必要がない限り、
これらのエレメントを修正する必要はありません。 -->
    <datasource-instance name="Application Files" type="file" version="8.1">

        <property name="explodeArchiveFiles" value="false" />

        <!-- Accepted units are hour, min, sec -->
        <property name="delayBetweenIterations" value="3" unit="sec" />

        <property name="filesPerIteration" value="5" />

        <property name="delayBetweenArchiveIterations" value="10"
unit="sec" />

        <property name="archiveFilesPerIteration" value="1" />

        <!-- Accepted units are bytes, KBytes, MBytes -->
        <property name="maxFileSizeToUpload" value="50" unit="KB" />

        <property name="useDigest" value="needed" />

        <!-- Web アプリケーションの設定に使用される一般的なファイルをスキャン -->
        <fileset name="config">
                <exclude pattern="(.*)">
                        <include pattern="(.*)¥.xml$"/>
                        <include pattern="(.*)¥.cmd$"/>
                        <include pattern="(.*)¥.sh$"/>
                        <include pattern="(.*)¥.bat$"/>
                </exclude>
        </fileset>

        <!-- Web アプリケーションのコードが含まれる一般的なファイルをスキャン -->
        <fileset name="webElements">
                <exclude pattern="(.*)">
                        <include pattern="(.*)¥.asp$"/>
                        <include pattern="(.*)¥.asm$"/>
```

```
                        <include pattern="(.*)¥.js$"/>
                        <include pattern="(.*)¥.html$"/>
                </exclude>
        </fileset>

        <!-- アーカイブをスキャン -->
        <fileset name="archives">
                <exclude pattern="(.*)">
                        <include pattern="(.*)¥.zip$"/>
                </exclude>
        </fileset>

        <!-- config、webElements、および archive filesets で定義されたファイル
タイプのみをスキャン-->
        <fileset name="default">
                <exclude pattern="(.*)">
                        <include pattern="(.*)¥.xml$" />
                        <include pattern="(.*)¥.cmd$" />
                        <include pattern="(.*)¥.sh$" />
                        <include pattern="(.*)¥.bat$" />
                        <include pattern="(.*)¥.asp$" />
                        <include pattern="(.*)¥.asm$" />
                        <include pattern="(.*)¥.js$" />
                        <include pattern="(.*)¥.html$" />
                        <include pattern="(.*)¥.zip$" />
                        <include pattern="(.*)¥.profile$" />
                </exclude>
        </fileset>

        <!-- config および webElements で定義されたファイル タイプのみをスキャン
-->
        <fileset name="defaultNoArchives">
                <exclude pattern="(.*)">
                        <include pattern="(.*)¥.xml$" />
                        <include pattern="(.*)¥.cmd$" />
                        <include pattern="(.*)¥.sh$" />
                        <include pattern="(.*)¥.bat$" />
                        <include pattern="(.*)¥.asp$" />
                        <include pattern="(.*)¥.asm$" />
                        <include pattern="(.*)¥.js$" />
                        <include pattern="(.*)¥.html$" />
                </exclude>
        </fileset>

        <!-- ログ ファイル以外をすべてスキャン -->
        <fileset name="noLogs">
                <exclude pattern="(.*)¥.err$" />
                <exclude pattern="(.*)¥.log$" />
                <exclude pattern="(.*)¥.lok$" />
```

```
                        <exclude pattern="(.*)¥.tlog$" />
                        <exclude pattern="(.*)¥.log0(.*)" />
                </fileset>

        <!--
        scan-directory:
        name 属性の値を変更して、監視対象のディレクトリを指定
        注: すぐに使用する場合には、「default」ファイルセットを使用します。 または、目的のファイルを監視するように
        fileset エレメントをカスタマイズして、下の scan-directory エレメントを置き換えるか、
        新しく追加します。 環境変数プロパティまたは Introscope エージェント プロパティの値も name 属性
        （またはその他の属性）の値に使用できます（例：
name="${MY_APP_HOME}/filesICareAbout/"） -->
        単一のドット（.）を指定した場合、それはアプリケーション（エージェントではない）の作業ディレクトリを基準とした相対パスを意味します。
        または、フル パスが必要になります。 以下の例では、通常エージェントのインストールに使用される Introscope8.1 フォルダを
        エージェントのインストール
        -->

                <scan-directory recursive="true" name="C:/Program Files/CA
Wily/Introscope8.1/wily" fileset="default"
                        enabled="true">
                </scan-directory>
        </datasource-instance>


    <!-- ========================================================== -->
    <!-- ASSEMBLY MONITOR                                           -->
    <!-- ========================================================== -->
    <!--
    デフォルトでは、Windows システム クラスは以下の exclude エレメントを使用して除外されます。
    exclude エレメントの pattern 属性では、パッケージ名も含め、
    クラス名に一致させる正規表現を定義します。

    すべてを除外するのではなく一部を除外するには、特定の exclude エレメント内に、
    include エレメントをネストさせる必要があります。 以下の例では、
    ネームスペース内の java クラス以外のクラスがすべて除外されます。
                <exclude pattern=".*">
                        <include pattern="system¥.*"/>
                </exclude>

    注: ChangeDetector では現在、アセンブリ監視データソースのインスタンスを
                1 つのみサポートしています。

    注:     いずれの除外パターンにも一致しないクラスはすべて、
                アセンブリ監視に含まれて監視されます。
```

```
        -->



        <datasource-instance name="Assembly Monitor" type="classmonitor"
version="8.1">

                <!-- スキャンが開始されるまでの、起動時およびアセンブリ ロード時の初期待機
時間 -->
                 <!-- Accepted units are hour, min, sec -->
                <property name="initialWaitTime" value="30" unit="sec" />

                 <!-- Accepted units are hour, min, sec -->
                <property name="delayBetweenIterations" value="2" unit="min" />
                <property name="classesPerIteration" value="5" />

                <!--
                        以下では、システム アセンブリからのクラスをスキップします。
                        これらのクラスを含めると、変更されることが
                        ほとんどないクラスが大量に追跡されて、
                        アプリケーションのパフォーマンスに影響を与えることがあります。
                        監視する必要のないアセンブリは、
                        <excludeassembly> タグを使用して追加します。
                -->

                <!-- アセンブリを除外 -->
                <excludeassembly pattern=".¥¥mscorlib¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.Xml¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.Web¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.Configuration¥.dll"/>
                <excludeassembly pattern=".¥¥wily¥.."/>
                <excludeassembly pattern=".¥¥Microsoft¥.JScript¥.dll"/>
                <excludeassembly pattern=".¥¥VJSharpCodeProvider¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.Data¥.dll"/>
                <excludeassembly pattern=".¥¥Oracle¥.DataAccess¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.Web¥.Mobile¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.ServiceModel¥.dll"/>
                <excludeassembly pattern=".¥¥SMDiagnostics¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.Drawing¥.dll"/>
                <excludeassembly
pattern=".¥¥System¥.Web¥.RegularExpressions¥.dll"/>
                <excludeassembly pattern=".¥¥Microsoft¥.VisualBasic¥.dll"/>
                <excludeassembly pattern=".¥¥CppCodeProvider¥.dll"/>
                <excludeassembly pattern=".¥¥System¥.EnterpriseServices¥.dll"/>
                <excludeassembly pattern=".¥System¥.Transactions¥.dll"/>

                <!-- Wily からのクラスを除外 -->
```

```
                <!-- 除外するクラスを除外パターンに追加します -->
                <exclude pattern="com¥.wily¥.(.*)"/>

        </datasource-instance>


    <!-- ========================================================= -->
    <!-- DATABASE CHANGE MONITORING                                -->
    <!-- ========================================================= -->
    <!--
    以下の DB 監視用のサンプル設定では、Oracle v$parameter テーブルの
    name/value ペアが 10 分ごとにスキャンされます。

    注: このデータソース インスタンスはデフォルトでコメント化されています。
    これは、環境に固有の接続パラメータが
    必要なためです。 使用しているデータベース設定に
    固有の値を入力してください。 -->
<!--
       <datasource-instance name="SQl Server DB" type="database"
                            url="Provider=SQLOLEDB;Data Source=localhost;
Integrated Security=SSPI;Initial Catalog=northwind" version="8.1">
                SQL Server
                        SELECT name, value FROM sampletable
                </sql>
                <schedule type="repetitive" interval="10" unit="min"/>
        </datasource-instance> -->


        <!-- ========================================================= -->
        <!-- CONFIGURATION PROPERTIES - datasource-type は修正しない -->
        <!-- ========================================================= -->
        <datasource-type name="javaenv"
class="com.wily.rave.agent.ds.sysprop.SysPropDataSourceConfig" />
        <datasource-type name="file"
class="com.wily.rave.agent.ds.file.FileDataSourceConfig" />
        <datasource-type name="classmonitor"
class="com.wily.rave.agent.ds.classmonitor.RuntimeAssemblyMonitorConfig" />
        <datasource-type name="database"
class="com.wily.rave.agent.ds.db.DBDataSourceConfig" />

</change-detector>
```

付録 B: FAQ

構成ファイルで maxFileSizeToUpload プロパティを 4MB に設定したにもかかわらず、CA APM ChangeDetector で 9MB のアーカイブ ファイルが検出され、そのアーカイブ ファイルへの変更がレポートされました。これはバグですか?

これはバグではありません。*maxFileSizeToUpload* プロパティは、テキスト ファイルにのみ適用され、アーカイブ ファイルには適用されません。 CA APM ChangeDetector では、アーカイブ ファイル全体の中身をアップロードするのではなく、アーカイブ ファイルを展開して各ファイルを個別に扱います。

アプリケーションで実行されているコードからクラスを削除しましたが、CA APM ChangeDetector では削除が検出されませんでした。なぜですか?

CA APM ChangeDetector では、コード内の追加と変更のみが検出されます。コードの削除は検出されません。

CA APM ChangeDetector で、監視対象のファイル システム内のファイルが 0 と表示されます。

監視対象のファイル システムに読み取り権限がない可能性があります。または、ファイル システムがネットワーク フォルダにある場合は、ネットワークがダウンしている可能性があります。