CA Mediation Manager および CA Mediation Manager for Infrastructure Management

管理ガイド

CA Mediation Manager リリース 2.2.3 / CA Mediation Manager for Infrastructure Management 2.0、リリース 2.2.3

CA Technologies 製品リファレンス

このマニュアルが参照している CA Technologies の製品は以下のとおりです。

- CA Mediation Manager
- CA Mediation Manager for Infrastructure Management 2.0

CA への連絡先

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

目次

第 1 章: はじめに

このガイドでは、CA Mediation Manager をインストールするためのアーキテクチャ、インストール、前提条件、および要件について説明します。

注: この章の情報は CA Mediation Manager にのみ適用されます。

このセクションには、以下のトピックが含まれています。

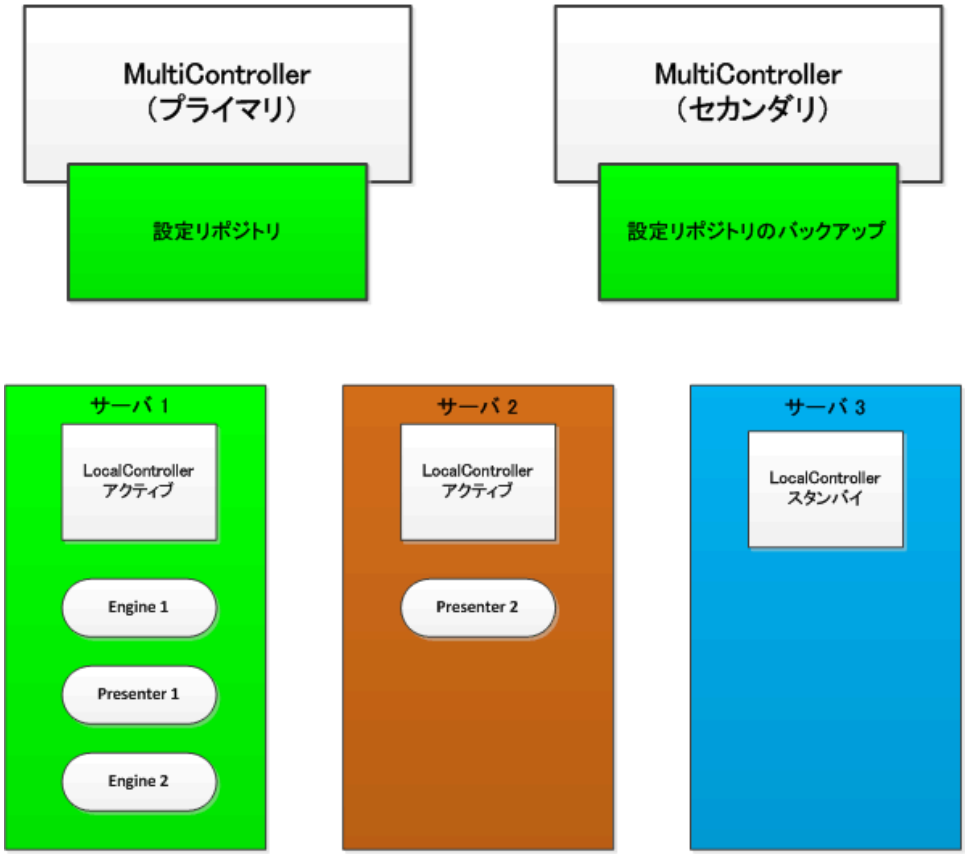
アーキテクチャ (P. 8)

アーキテクチャ

CA Mediation Manager は、2 つの主要コンポーネントおよび 2 つのサブコンポーネントから構成されています。 主要コンポーネントは MultiController（MC）および LocalController（LC）です。

サブコンポーネントは Engine および Presenter です。

以下の図は一般的なアーキテクチャを示しています。



アーキテクチャには、Generic Executor および Delivery Service というコンポーネントも含まれます。 これらのコンポーネントは上の図では表示されていませんが、このガイドの後半で説明します。

第 2 章: 概要

注: この章の情報は CA Mediation Manager にのみ適用されます。

このセクションには、以下のトピックが含まれています。

コンポーネントの概要

CA Mediation Manager インストーラは、MultiController、LocalController、および Web コンポーネントをインストールします。

CA Mediation Manager インストーラは、以下のコンポーネントもインストールします。

- Delivery Service（LocalController の一部）
- Generic Executor

デバイス パックをインストールすると、サブコンポーネントである Engine および Presenter がインストールされます。

注: 特定のデバイス パックのインストールの詳細については、CA Mediation Manager のインストール ディレクトリである CAMM_HOME 下の DpConfig フォルダにある「デバイス パック ガイド」を参照してください。

主要コンポーネント

CA Mediation Manager には、MultiController、LocalController、および Web の 3 つの主要コンポーネントがあります。

MultiController

1 つのクラスタには、最大 2 つの MultiController （プライマリおよびセカンダリ）を展開できます。クラスタあたり少なくとも 1 つの MultiController を展開します。 MultiController はユーザのクラスタ環境で以下のアクションを実行します。

- リモート サーバ上にある LocalController コンポーネントからのハートビート メッセージを監視します。
- クラスタの一元化されたライセンス サーバとして機能します。
- クラスタ内のコンポーネントの集中型設定ファイルを格納します。

LocalController

サブコンポーネント（Engine または Presenter）が存在するクラスタ内の各物理サーバに 1 つの LocalController をインストールします。LocalController は以下のアクションを実行します。

- サーバにインストールされたサブコンポーネント用の通信メカニズムを提供します。
- ローカル サーバ上のサブコンポーネントのハートビート メッセージを監視し、障害が発生した場合、自動的にサブコンポーネントを再起動します。
- 配信サービスを使用して、Engine からの出力を処理します。 このサービスは、XML ドキュメントを圧縮および暗号化された形式でローカルまたはリモートの Presenter に配信します。

Web

Web コンポーネントを使用すると、Web ベース インターフェースからデバイス パックの展開を一元管理できます。 このインターフェースには、以下の情報が表示されます。

- 実行されているデバイス パックのステータス。
- デバイス パックがインストールされている LocalController のステータス。
- プライマリおよびセカンダリ MultiController のステータス。

CA Mediation Manager は以下の 2 つの Web サーバをインストールします。

- プライマリ Web サーバ
- セカンダリ Web サーバ

プライマリ Web サーバはプライマリ MultiController のインストール中にインストールされ、セカンダリ Web サーバはセカンダリ MultiController のインストール中にインストールされます。

- プライマリ Web サーバにアクセスするには、CA Mediation Manager Web UI を起動します。

  `http://<PrimaryMCMachineIP>:<web-port>/tim-web/index.htm`

  <web-port> は CA Mediation Manager のインストール時に設定されたポート番号で、<PrimaryMCMachineIP> は*プライマリ MultiController* システムの IP アドレスまたはホスト名です。

  プライマリ MultiController が応答しない場合、セカンダリ MultiController が自動的にセカンダリ Web サーバを開始します。

  プライマリ MultiController が応答を開始すると、セカンダリ MultiController はセカンダリ Web サーバを停止します。

- セカンダリ Web サーバにアクセスするには、CA Mediation Manager Web UI を起動します。

  `http://<SecondaryMCMachineIP>:<web-port>/tim-web/index.htm`

  <web-port> は CA Mediation Manager のインストール時に設定されたポート番号で、<SecondaryMCMachineIP> は*セカンダリ MultiController* システムの IP アドレスまたはホスト名です。

後続のセクションでは、Web コンポーネントのオプションについて説明します。

デフォルト オプション

以下の情報は、CA Mediation Manager の［Management］タブにあるデフォルト オプションの説明です。

Install or Remove

既存のリポジトリからデバイス パックをインストールまたは削除します。 インストール中に使用されるデフォルト パスは、$CAMM_HOME/MC/repository/device packs です。 インストール中に別のパスを使用した場合、そのデバイス パック パスを参照して選択します。

Upgrade

デバイス パックのバージョンをアップグレードします。

詳細オプション

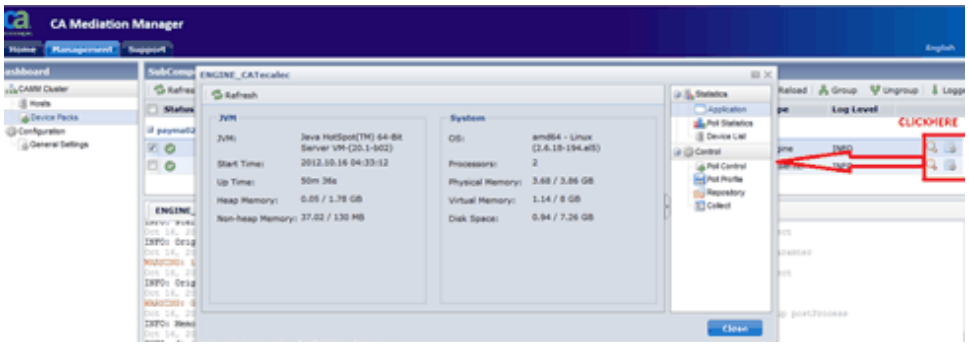
以下の情報は、詳細オプションの説明です。

Statistics

統計を収集し理解するために以下のオプションを提供します。

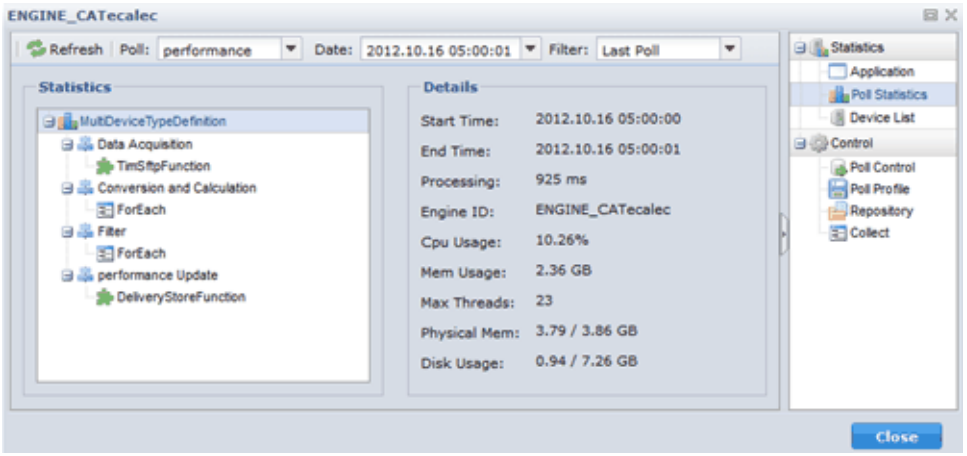
Application

Java Virtual Machine（JVM）および Engine または Presenter が実行されているシステムのパフォーマンス メトリックを提供します。



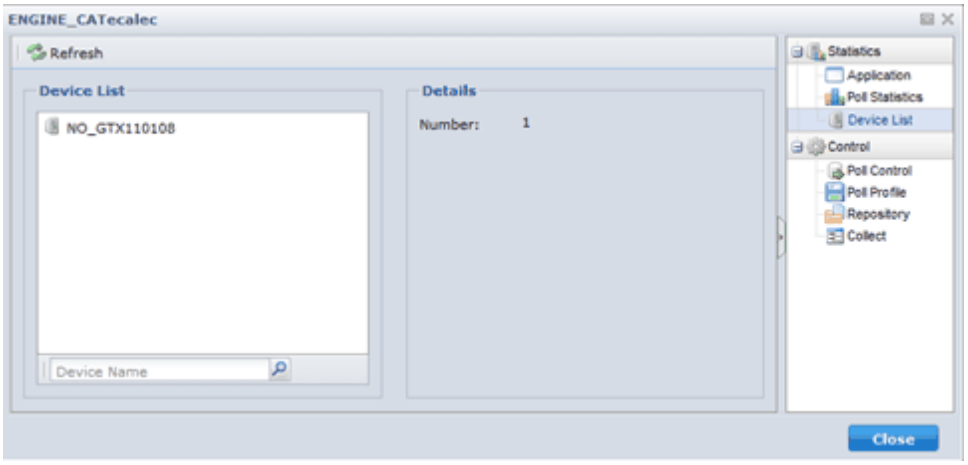
Poll Statistics

インベントリ ポーリングおよびパフォーマンス ポーリングのステータスの情報を提供します。 Engine を再起動して変更を反映させます。



Device List

$CAMM_HOME/repository/work フォルダにあるインベントリ デバイス リスト ファイルを使用してインベントリ ポーリングによって検出されたデバイスのリストを提供します。



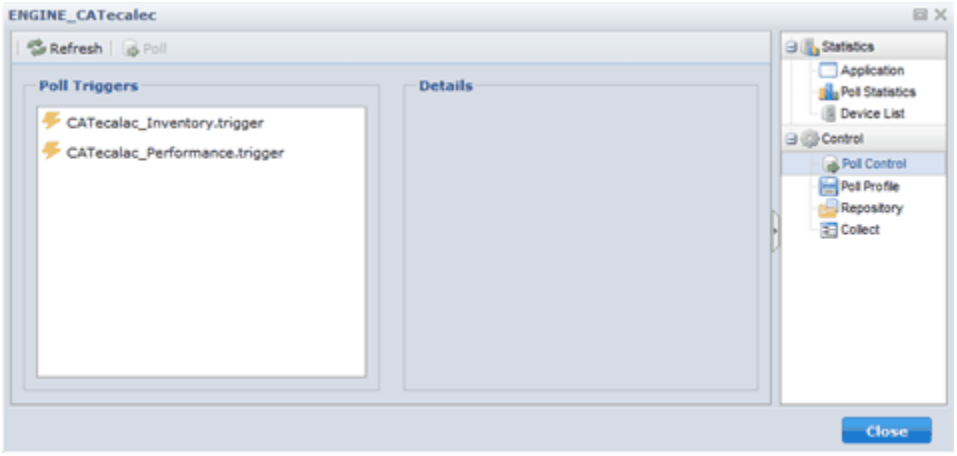
Control

以下の操作を実行できるインターフェースを提供します。

- 手動ポーリングのトリガ
- デバイス パックによって使用される外部グローバル変数値の表示および編集
- デバイス パック ファイルの表示および編集
- ログ ファイルの収集

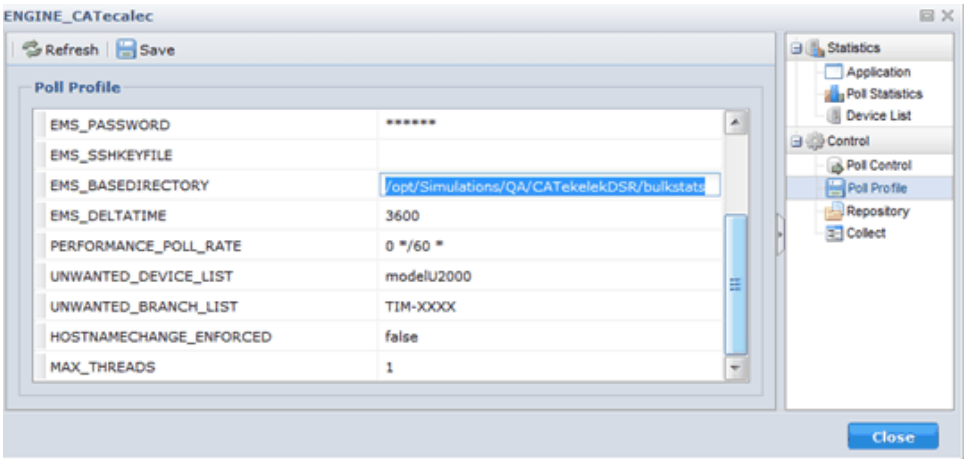
Poll Control

インベントリ ポーリングおよびパフォーマンス ポーリングをトリガします。



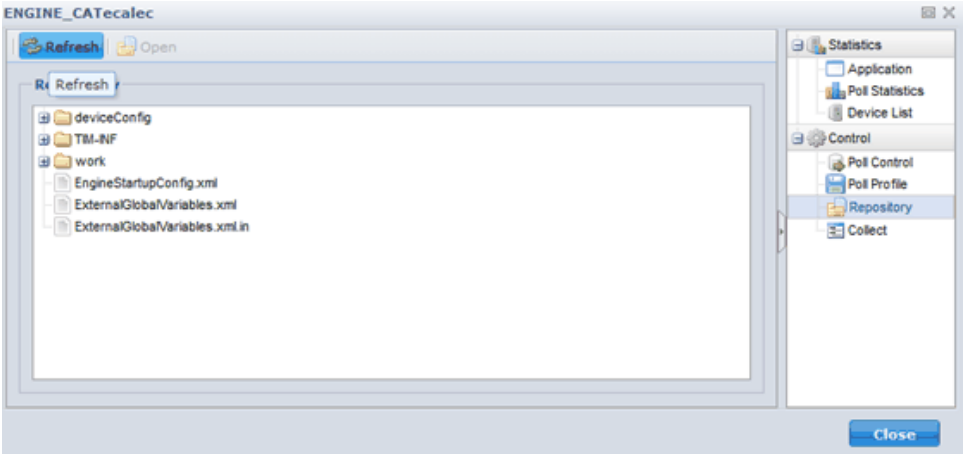
Poll Profile

変更した外部グローバル変数のリストを組み込みます。 Engine または Presenter を再起動して変更を反映させます。



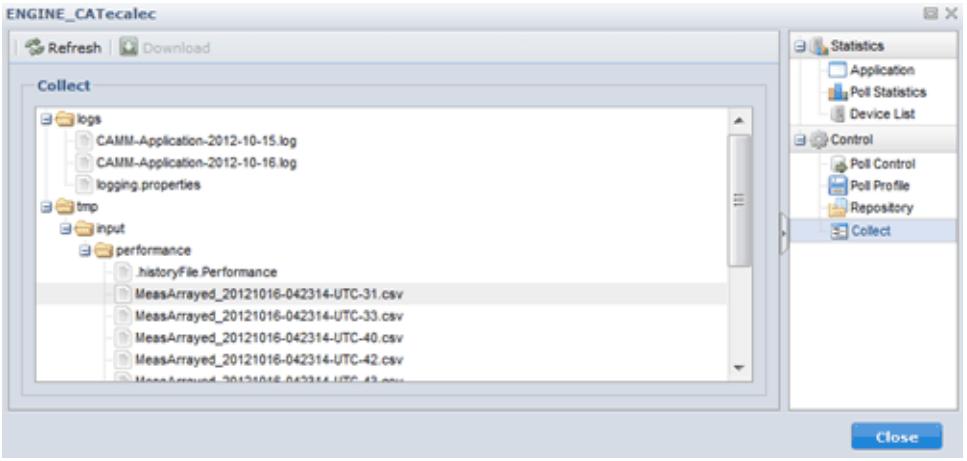
Repository

MultiController 内のデバイス パック コンポーネントのファイルを表示および変更できます。 Engine または Presenter を再起動して変更を反映させます。



Collect

Engine および Presenter のログを収集します。



一般設定

以下のオプションを使用して一般設定を指定します。

Connection

再試行回数および CA Mediation Manager コンポーネント間の接続間隔のデフォルト設定を指定します。

Logging

CA Mediation Manager コンポーネントの更新間隔、バッファ サイズ、最大行数などのログ記録オプションを指定します。

サブコンポーネント

CA Mediation Manager には、Engine と Presenter の 2 つのサブコンポーネントがあります。

Engine

Engine は CA Mediation Manager 内のメインのスレッド ポーリング エンジンです。 アクティブまたはスタンバイ モードで Engine を展開できます。 Engine は以下のアクションを実行します。

- XML、CSV、Telnet、SSH などを使用してデバイスから情報を収集し、CA Mediation Manager 標準の XML ドキュメントへデータを処理します。
- Delivery Service によって処理するために、CA Mediation Manager 標準の XML ドキュメントをキューに展開します。

Presenter

Presenter は、以下のアクションを実行するスレッド プレゼンテーション エンジンです。

- Engine から CA Mediation Manager 標準の XML ドキュメントを受信します。
- CSV、XML、SNMP、DDI などの必要な出力形式にデータをフォーマットします。

その他のコンポーネント

Generic Executor および Delivery Service は、CA Mediation Manager のその他の 2 コンポーネントです。 Generic Executor はその他のコンポーネントを起動します。 Delivery Service は Presenter に XML ファイル出力を送信します。

Generic Executor

クラスタ内のすべてのコンポーネントは、通信および実行用の共通の機能セットを共有します。Generic Executor は Engine および Presenter サブコンポーネントを開始し、一時ファイルとログ ファイルを消去します。

Generic Executor はシステム起動時に起動し、特定の TCP ポートでリスンします。 MultiController のようなコンポーネントを起動する場合、CA Mediation Manager Control Utility（cammCtrl）は Generic Executor に MultiController XML 設定ファイルを送信します。 Generic Executor は、このデータを受信すると、設定ファイルの情報を使用して MultiController コンポーネントを識別し、起動します。

Delivery Service

Engine はそのポーリング サイクルを完了すると、1 つ以上の CA Mediation Manager 標準の XML ドキュメントをキュー ディレクトリに生成します。 Delivery Service はキュー ディレクトリを単独で監視し、1 つ以上のローカルまたはリモートの Presenter にデータを配信します。

ローカルまたはリモートの Presenter が使用できない場合、Delivery Service は、ローカルまたはリモートの Presenter が使用可能になるまでキューを処理しません。

第 3 章: CA Mediation Manager のインストール、アンインストール、およびアップグレード

注: この章の情報は CA Mediation Manager にのみ適用されます。

このセクションには、以下のトピックが含まれています。

システム要件

CA Mediation Manager は Java Runtime Environment（JRE）バージョン 1.7 以降を必要とします。

以下の表では、サポートされている各オペレーティング システムについて、必要なハードウェアの最低要件を説明します。

オペレーティング システム	アーキテクチャ	CPU	メモリ	ディスク
Solaris 9 または 10	SPARC（64 ビット）	1 x 1.4 GHz	4 GB	18 GB
Linux	x86（64 ビット）	1 x 2 GHz	4 GB	18 GB
Windows 2003	x86（64 ビット）	1 x 2 GHz	4 GB	18 GB
Windows 2008	x86（64 ビット）	1 x 2 GHz	4 GB	18 GB

注: JRE とオペレーティング システム アーキテクチャの整合性を保持します。 たとえば、64 ビットのオペレーティング システムでは、CA Mediation Manager をインストールし実行するために使用する JRE も 64 ビットである必要があります。 JRE の最新のバージョンを使用することを推奨します。これは Java ダウンロード サイトから取得できます。

インストールおよびアップグレード

CA Mediation Manager およびデバイス パックをインストールおよびアップグレードするには、「*CA Mediation Manager インストール ガイド*」を参照してください。

サービスの開始と停止

以下の情報は、UNIX および Windows の CA Mediation Manager でのサービスの開始および停止の説明です。

UNIX

startall または stopall スクリプト、または init.camm スクリプトを実行して、CA Mediation Manager を開始または停止することができます。init.camm スクリプトは、CAMM Home ディレクトリ内の Tools ディレクトリにあります。

システムの起動時およびシャットダウン時に CA Mediation Manager を自動的に開始および停止するには、root または sudo su として以下の init.camm.install スクリプトを実行します。

```
shell# tools/init.camm.install
```

CA Mediation Manager を自動的に開始または停止する設定を削除するには、以下の init.camm.uninstall スクリプトを実行します。

```
shell# tools/init.camm.uninstall
```

Windows

Windows に CA Mediation Manager をインストール中に、Generic Executor および Web コンポーネントは Windows サービスとして登録されます。 サービス名は、CAMM-GE-{user}-{port} および CAMM-tomcat7-8880 です。

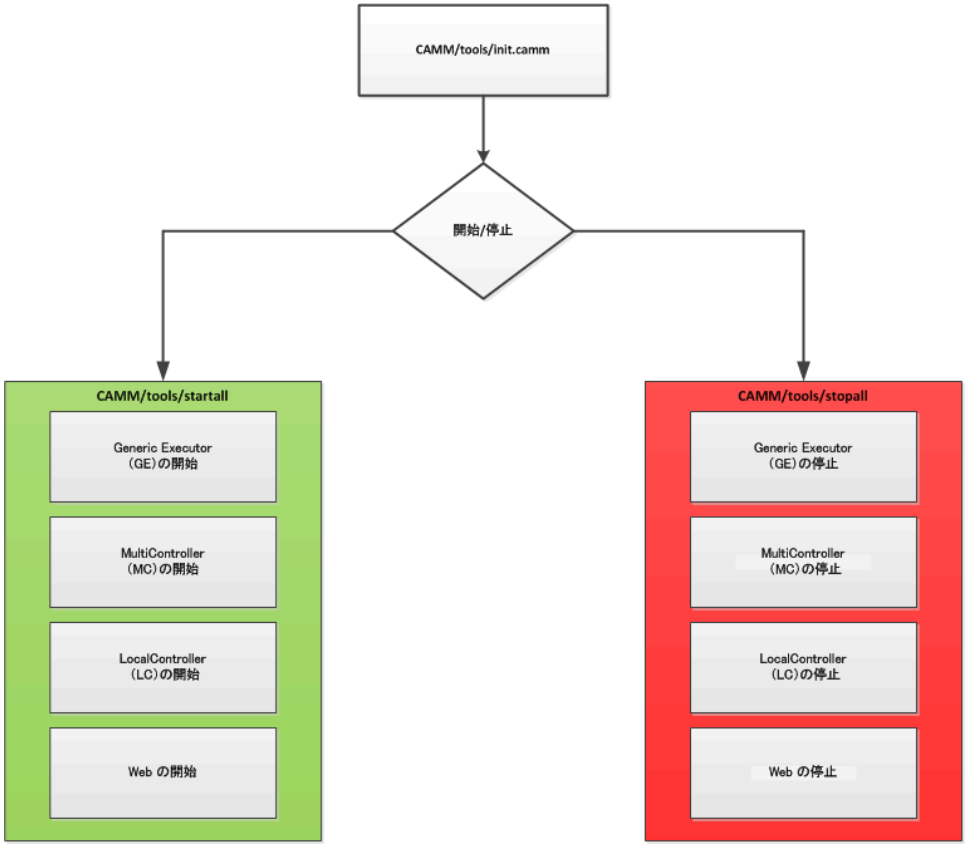
デフォルトでは、これらのサービスは手動で開始されます。 Linux と同様に、Windows では、init.camm.install.bat を実行することで、システムの起動時およびシャットダウン時に CA Mediation Manager を自動的に開始および停止できます。

```
C:/CAMM/tools/init.camm.install.bat
```

CA Mediation Manager を自動的に開始または停止する設定を削除するには、init.camm.uninstall.bat を実行します。

```
C:/CAMM/tools/init.camm.uninstall.bat
```

以下の図では、サービスを開始および停止するプロセス フローを示します。



第 4 章: コンポーネントの設定

以下のセクションでは、CA Mediation Manager でのコンポーネントの設定方法について説明します。

注: この章の情報は CA Mediation Manager にのみ適用されます。

このセクションには、以下のトピックが含まれています。

Generic Executor の設定

Generic Executor は通常初期インストールの後に設定を必要としません。別のユーザ ID でコンポーネントを実行する必要がない限り、サーバごとに必要な Generic Executor は 1 つのみです。

Generic Executor 設定は、CAMM HOME ディレクトリにある GE_<userid> という名前の Generic Executor ディレクトリにインストールされます。<userid> はインストール中に指定したユーザ名です。

Generic Executor ディレクトリには、LocalConfig-ge.xml という名前のファイルが含まれます。

Generic Executor がコンポーネントの役割を引き受ける場合、LocalConfig-ge.xml で設定されているのと同じユーザ ID を使用して実行されます。

UNIX の LocalConfig-ge.xml ファイルの例

```
<?xml version="1.0" encoding="UTF-8"?>
<AppDaemon>
    <Names>
    </Names>
    <Paths>
        <Path name="tim.base">/opt/CA/CAMM</Path>
        <Path name="appHome">${tim.base}/GE_camm</Path>
        <Path name="configBase">${appHome}/tmp</Path>
    </Paths>
    <Binding>
        CA Portal29560</Port>
        <UserId>camm</UserId>
    </Binding>

</AppDaemon>
```

Windows の LocalConfig-ge.xml ファイルの例

Windows ユーザの場合、プライマリ Generic Executor 設定ファイルがわずかに異なります。

```
<?xml version="1.0" encoding="UTF-8"?>
<AppDaemon>
    <Names>
    </Names>
    <Paths>
        <Path name="tim.base">/opt/CA/CAMM</Path>
        <Path name="appHome">${tim.base}/GE_camm</Path>
        <Path name="configBase">${appHome}/tmp</Path>
```

```
    </Paths>
    <Binding>
        CA Portal29560</Port>
        <UserId>camm</UserId>
    </Binding>
    <CompanyItems>
        <Item>
                <Name>MC</Name>
                <Config>${tim.base}/MC/LocalConfig-mc.xml</Config>
                CA Portal29599</Port>
        </Item>
        <Item>
                <Name>LC</Name>
                <Config>${tim.base}/LC/LocalConfig-lc.xml</Config>
                CA Portal29598</Port>
        </Item>
    </CompanyItems>
</AppDaemon>
```

Generic Executor 設定ファイルに追加された <CompanyItems> セクションは、MultiController および LocalController の場所およびサービス ポートを定義します。 <CompanyItems> が定義されると、Generic Executor は 1 分ごとにこれらのアイテムを確認し、それらが実行されていない場合、自動的に開始します。 この設定では、MultiController および LocalController を外部から操作することなく起動することができます。

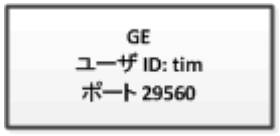
<CompanyItems> 機能は、インストール後はデフォルトで無効になっています。 有効にするには、camm.init.install コマンドを実行します。

注: プライマリ Generic Executor のみが <CompanyItems> セクションを必要とします。他の Generic Executors にはこのセクションを含めないでください。

Generic Executor の仕組み

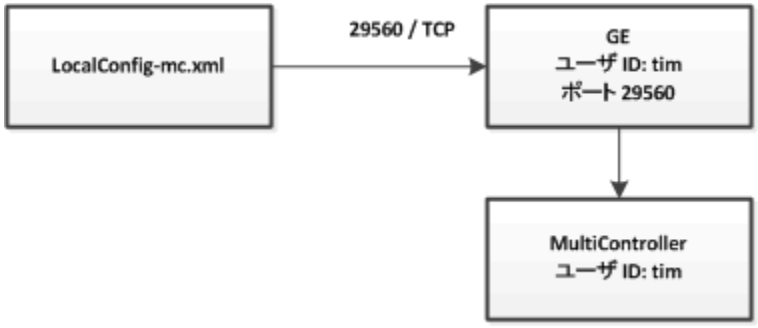
Generic Executor は再利用可能なエンティティであり、CA Mediation Manager コンポーネントの基盤となるものです。サーバが起動すると、少なくとも 1 つの Generic Executor コンポーネントが存在し起動されることが必須です。

デフォルトでは、LocalConfig-ge.xml ファイルには、CAMM_USER のユーザ ID および TCP ポート情報が含まれています。 以下の図は、TCP ポート 29560 上の Generic Executor、および CA Mediation Manager をインストールした CAMM_USER を示しています。



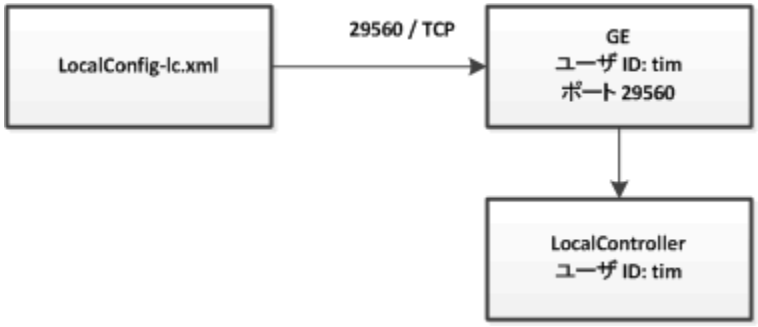
MultiController をサーバで実行する必要がある場合、MultiController 用の LocalConfig-mc.xml 設定ファイルおよび Generic Executor 用の TCP ポートが必要です。

以下の図は、LocalConfig-mc.xml ファイルが TCP ポート 29560 上の Generic Executor に送信され、Generic Executor が MultiController の新しいプロセスを開始する仕組みを示しています。



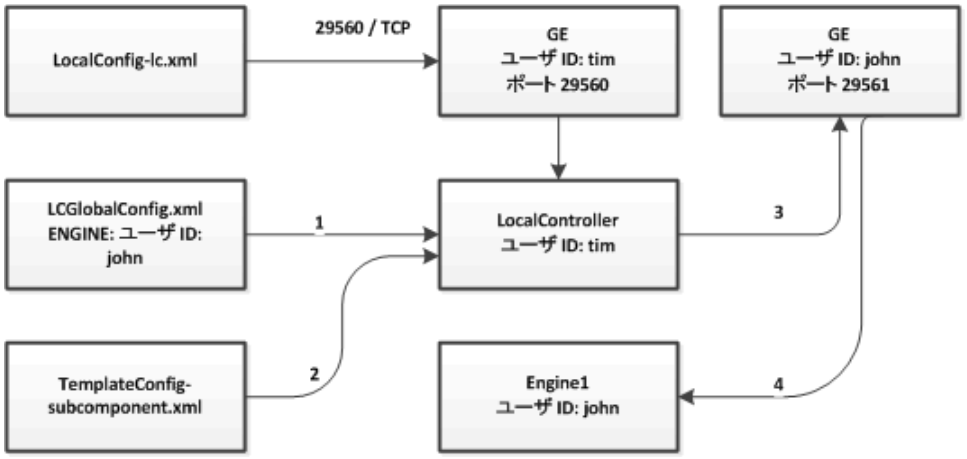
LocalController をサーバで実行する必要がある場合、LocalController 用の LocalConfig-lc.xml 設定ファイルおよび Generic Executor 用の TCP ポートのみが必要です。

以下の図は、LocalConfig-lc.xml ファイルが TCP ポート 29560 上の Generic Executor に送信され、Generic Executor が LocalController の新しいプロセスを開始する仕組みを示しています。



LocalController が起動すると、リポジトリから LCGlobalConfig.xml ファイルを読み取ります。 LCGlobalConfig.xml ファイルには、サブコンポーネントおよびポート情報が含まれています。

Engine をサーバで実行する必要がある場合、LocalController は LCGlobalConfig.xml ファイルを使用して、どのユーザ ID として実行する必要があるかを決定します。 LocalController ディレクトリで、LocalController は ExecutorMap.xml ファイルを検索します。 このアクションによって、LocalController は必要なユーザ ID として実行されている Generic Executor を見つけることができます。TemplateConfig-subcomponent.xml ファイルは、LCGlobalConfig.xml ファイルからの特定の設定と組み合わされます。 この組み合わせは、Generic Executor の TCP ポートに送信されます。



Generic Executor の設定オプション

Paths tim.base、appHome、および configBase は必須で、インストーラによって設定されます。

tim.base

CA Mediation Manager のベース ディレクトリを指定します。

appHome

この CA Mediation Manager Generic Executor のベース ディレクトリを指定します。

configBase

Generic Executor が現在実行しているコンポーネントの設定を格納するための一時ディレクトリを指定します。

Port

Generic Executor がリスンする TCP ポートを指定します。

デフォルト： TCP 29560

UserId

Generic Executor プロセスおよびそのサブプロセスを所有するユーザ ID を指定します。

Generic Executor のスタートアップ シーケンス

startall または startall.bat スクリプトを使用して、手動で Generic Executor プロセス（または Windows 上のサービス）を開始できます。 ただし、camm.init.install が実行される場合、Generic Executor はシステムの起動時に自動的に起動します。

別の Generic Executor の追加（UNIX）

コンポーネントを別のユーザ ID として実行する必要がある場合は、別の Generic Executor を追加します。

次の手順に従ってください：

1. 新しいユーザを追加のユーザ ID を必要とする LocalController に追加します。新しいユーザは、CAMM_HOME ディレクトリを所有するのと同じグループのメンバである必要があります。

   注: 任意のスタンバイ LocalControllers にこの手順を繰り返します。

2. 新しいユーザとしてログインします。
3. 以下のコマンドを実行して新しい Generic Executor を追加します。

   ```
   shell# /opt/CA/CAMM/tools/camm.ge.install
   ```

   Generic Executor が追加されます。

   注: ExecutorMap.xml という名前の LocalController ディレクトリ内の設定ファイルを変更して、必要なユーザ ID と TCP ポートの間のマップを提供します。

Engine サブコンポーネントが起動されると、LocalController はそのコンポーネントの LCGobalConfig で指定されたユーザ ID を参照します。ユーザ ID がデフォルト ユーザ ID でない場合、ExecutorMap が参照され、代替 Generic Executor の TCP ポートによりサブコンポーネントが起動されます。

この参照は、Engine がユーザまたはホスト ベースの RSA キー認証を使用して SFTP または SCP により情報を取得する場合に役立ちます。

MultiController 設定

MultiController はクラスタの最も重要な部分です。 一元化されたライセンス ファイルが含まれ、クラスタ内のコンポーネントとのハートビートを維持します。

クラスタ内の LocalControllers、Engines、および Presenter の設定も MultiController のリポジトリにあります。

MultiController は、TCP ポート番号 29599 上のクラスタ メンバからのハートビート操作をリスンします。

GUI インストール中に、基本的な MultiController をインストールするのに必要な必須オプションのみを設定できます。 ただし、MC または LocalConfig-mc.xml ファイルは手動で編集できます。

サンプル LocalConfig-mc.xml ファイル（基本的な設定）

```
<?xml version="1.0" encoding="UTF-8"?>
<LocalConfig>
      <Description>Configuration for Multi Controller</Description>
      <Names>
            <Name name="mainClass">com.torokina.tim.mc.Main</Name>
            <Name name="appName">CAMM-Multi-Controller</Name>
            <Name name="appShortName">MC</Name>
            <Name name="primaryMcAddress">127.0.0.1</Name>
            <Name name="secondaryMcAddress"></Name>
            <Name name="primaryMcPort">29599</Name>
            <Name name="secondaryMcPort">-1</Name><
            <Name name="myMode">primary</Name>
            <Name name="myAddress">127.0.0.1</Name>
            <Name name="mcPort">29599</Name>
            <Name name="otherMcAddress"></Name>
            <Name name="otherMcPort">-1</Name>
            <Name name="heartbeatFrequency">15</Name>
            <Name name="heartbeatTimeout">180</Name>
            <Name name="repositoryFrequency">15</Name>
      </Names>
      <Paths>
            <Path name="license">${tim.base}/license.lic</Path>
      </Paths>
<LocalConfig>
```

サンプル LocalConfig-mc.xml ファイル（非表示ログおよびクリーンアップ設定）

```
    <Logging>
        <LogLevel>INFO</LogLevel>
        <LogDirectory>${logbase}</LogDirectory>
```

```
        <ObjectLogging>
          <ObjectToLog>
            <ObjectName>com.torokina.tim.config</ObjectName>
            <ObjectLogLevel>TRACE</ObjectLogLevel>
          </ObjectToLog>
        </ObjectLogging>
    </Logging>
    <CleanUps>
        <CleanUp>
            <CleanUpName>clean-temporary-directory</CleanUpName>
            <CleanUpAction>delete</CleanUpAction>
            <CleanUpTarget>${tmp}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>3d</ParameterValue>
            </Parameter>
        </CleanUp>
        <CleanUp>
            <CleanUpName>archive-log-directory</CleanUpName>
            <CleanUpAction>archive</CleanUpAction>
            <CleanUpTarget>${logbase}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>3d</ParameterValue>
            </Parameter>
        </CleanUp>
        <CleanUp>
            <CleanUpName>clean-log-directory</CleanUpName>
            <CleanUpAction>delete</CleanUpAction>
            <CleanUpTarget>${logbase}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>7d</ParameterValue>
            </Parameter>
        </CleanUp>
    </CleanUps>
```

これらのフィールドを、LocalConfig-mc.xml ファイルに正しい XML 構造で指定すると、デフォルト コンテンツが上書きされます。 たとえば、以下の設定はデフォルト ログ レベルを FINEST に変更します。

デフォルト ログ レベルの FINEST への変更

```
<?xml version="1.0" encoding="UTF-8"?>
<LocalConfig>
      ... ...
      <Logging>
        <LogLevel>FINEST</LogLevel>
        <LogDirectory>${logbase}</LogDirectory>
      </Logging>
```

```
      ... ...
</LocalConfig>
```

サンプル LocalConfig-mc.xml（MultiController ランタイム）ファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<Runtime>
   <Names>
      <Name name="mainClass">com.torokina.tim.mc.Main</Name>
      <Name name="appName">CAMM-Multi-Controller</Name>
      <Name name="appShortName">MC</Name>
      <Name name="primaryMcAddress">127.0.0.1</Name>
      <Name name="secondaryMcAddress"/>
      <Name name="primaryMcPort">29599</Name>
      <Name name="secondaryMcPort">-1</Name>
      <Name name="myMode">primary</Name>
      <Name name="myAddress">127.0.0.1</Name>
      <Name name="mcPort">29599</Name>
      <Name name="otherMcAddress"/>
      <Name name="otherMcPort">-1</Name>
      <Name name="heartbeatFrequency">15</Name>
      <Name name="heartbeatTimeout">180</Name>
      <Name name="repositoryFrequency">15</Name>
      <Name name="lcPort">29598</Name>
      <Name name="manageable">469</Name>
   </Names>
   <Paths>
      <Path name="license">${tim.base}/license.lic</Path>
      <Path name="apphome">${tim.base}/${appShortName}</Path>
      <Path name="runtimeConfig">${apphome}/runtime.xml</Path>
      <Path name="tmp">${apphome}/tmp</Path>
      <Path name="logbase">${apphome}/logs</Path>
      <Path name="basedir">${tim.base}</Path>
   </Paths>
</Runtime>
```

runtime.xml ファイルは非表示設定とマージされ、MultiController コンポーネントを起動します。

MultiController 設定オプション

以下の情報は、MultiController を設定するために使用できるオプションを説明します。

Path

以下のアイテムのパス情報を指定できます。

basedir

CA Mediation Manager ベース ディレクトリを指定します。

apphome

CA Mediation Manager MultiController アプリケーション ホーム ディレクトリを指定します。

tmp

CA Mediation Manager MultiController 一時ファイル ディレクトリを指定します。

logbase

CA Mediation Manager MultiController ログ ディレクトリを指定します。

runtimeConfig

Generic Executor によって提供される CA Mediation Manager MultiController ランタイム XML 設定を指定します。

Java

Java を使用するためのオプションを指定できます。

CommandPath

MultiController を開始するために Generic Executor が呼び出す Java 実行プログラムへのフル パスを指定します。

ClassPath/JarBase

ClassPath に追加する 1 つ以上のエントリを使用して、リストを作成できます。

Options

Java に解析されるコマンド ライン オプションを指定します。

Environment

MultiController コンポーネントを実行する環境変数を指定します。

MainClass/Class

実行する Java メイン クラスを指定します。

MainClass/Args

Java クラスに解析された引数を指定します。

Runtime

ランタイム オプションを指定できます。

Binding/BindAddress

MultiController コンポーネントがバインドする IP アドレスを指定します。

値： IP アドレス 0.0.0.0 をすべての MultiController コンポーネントに使用します。 2 つ以上の IP アドレスについては、カンマ区切りリストを使用します。

MultiController は TCP ポート 29599 にバインドします。

Binding/MyAddress

MultiController がそれ自体を識別するために使用する IP アドレスを指定します。

注: IP アドレスはこのホスト上の有効な IP アドレスである必要があります。

MultiControllerConfig/Mode

MultiController のオペレーティング モードを指定します。

値： Primary または Secondary のいずれかを指定します。

MultiControllerConfig/MCAddresses/Other

クラスタ内の他の MultiController の IP アドレスを指定します。

MultiControllerConfig/Heartbeat/ParameterName == frequency

他の MultiController に送信されるハートビート メッセージの頻度を指定します。

MultiControllerConfig/Heartbeat/ParameterName == timeout

この MultiController が LocalController コンポーネントからのハートビートを待つ期間を指定します。 MultiController が 180 秒間 LocalController からハートビートを受信しない場合、MultiController は最初の使用可能な LocalController へのフェールオーバをトリガします。

Logging

ログ記録オプションを指定できます。

LogLevel

出力ログ ファイルのロギング レベルを指定します。

値： DEBUG、TRACE、INFO、WARNING または ERROR を指定します。

LogDirectory

出力ログ ファイルのロギング レベルを指定します。

値： DEBUG、TRACE、INFO、WARNING または ERROR を指定します。

ObjectLogging/ObjectToLog/ObjectName

ログ記録を有効にする Java クラス名を指定します。

ObjectLogging/ObjectToLog/ObjectLogLevel

Java クラスのロギング レベルを指定します。

Cleanup

クリーンアップ オプションを指定できます。

CleanUpName

クリーンアップのわかりやすい名前を指定します。

CleanUpAction

クリーンアップ アクションを指定します。

値： Delete または Archive を指定します。

CleanupTarget

クリーンアップするディレクトリを指定します。

値： ${camm.variable} などの CA Mediation Manager 変数を使用して指定できます。

Parameter/ParameterName – Parameter/ParameterValue

<n><unit> の形式で期限切れになるパラメータおよびその有効期限を指定します。 例：

10d =10 日

10h = 10 時間

10m = 10 分

MultiController の手動による開始と停止

init.camm スクリプトは自動的に MultiController プロセスを開始します。cammCtrl ユーティリティを使用して、MultiController コンポーネントを個別に手動で停止または開始できます。

次の手順に従ってください：

1. CAMM_USER としてログインし、CAMM ホーム ディレクトリに移動します。
2. 以下のコマンドを実行して、MultiController を開始します。

   ```
   /opt/CA/CAMM# tools/startall -c mc
   ```

3. 以下のコマンドを実行して、MultiController を停止します。

   ```
   /opt/CA/CAMM # tools/stopall –c mc
   ```

LocalController 設定

LocalController はクラスタで各サーバにインストールされる重要なサービスです。

LocalController は以下の重要な機能を実行します。

- ローカル サーバで実行されるサブコンポーネントとリモート MultiController 間の通信を円滑にします。
- Engine および Presenter のパフォーマンスおよび可用性を監視し、障害が発生したコンポーネントを再起動します。
- クラスタ メンバからのハートビート操作をリスンします。デフォルトでは、LocalController はポート 29598 または TCP でリスンします。
- MultiController にハートビート情報を送信します。
- ローカル Engine および Presenter を開始、停止、および再起動します。

GUI インストール中に、基本的な LocalController をインストールするのに必要な必須オプションのみを設定できます。 ただし、LC または LocalConfig-lc.xml ファイルは手動で編集できます。

サンプル LocalConfig-lc.xml ファイル（基本的な設定）

```
<LocalConfig>
      <Names>
            <Name name="mainClass">com.torokina.tim.lc.Main</Name>
            <Name name="primaryMcAddress">127.0.0.1</Name>
            <Name name="secondaryMcAddress"></Name>
            <Name name="primaryMcPort">29599</Name>
            <Name name="secondaryMcPort">-1</Name>
            <Name name="myAddress">127.0.0.1</Name>
            <Name name="appName">CAMM-Local-Controller</Name>
            <Name name="appShortName">LC</Name>
            <Name name="lcPort">29598</Name>
            <Name name="heartbeatFrequency">15</Name>
            <Name name="heartbeatTimeout">180</Name>
      </Names>
      <Paths>
            <Path name="dsLocalConfig">${basedir}/DS/LocalConfig-ds.xml</Path>
      </Paths>
</LocalConfig>
```

サンプル LocalConfig-lc.xml ファイル（非表示ログおよびクリーンアップ設定）

```
    <Logging>
        <LogLevel>INFO</LogLevel>
        <LogDirectory>${logbase}</LogDirectory>
```

```
        <ObjectLogging>
          <ObjectToLog>
            <ObjectName>com.torokina.tim.config</ObjectName>
            <ObjectLogLevel>TRACE</ObjectLogLevel>
        </ObjectToLog>
        </ObjectLogging>
    </Logging>
    <CleanUps>
        <CleanUp>
            <CleanUpName>clean-temporary-directory</CleanUpName>
            <CleanUpAction>delete</CleanUpAction>
            <CleanUpTarget>${tmp}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>3d</ParameterValue>
            </Parameter>
        </CleanUp>
        <CleanUp>
            <CleanUpName>archive-log-directory</CleanUpName>
            <CleanUpAction>archive</CleanUpAction>
            <CleanUpTarget>${logbase}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>3d</ParameterValue>
            </Parameter>
        </CleanUp>
        <CleanUp>
            <CleanUpName>clean-log-directory</CleanUpName>
            <CleanUpAction>delete</CleanUpAction>
            <CleanUpTarget>${logbase}</CleanUpTarget>
            <Parameter>
                <ParameterName>expire</ParameterName>
                <ParameterValue>7d</ParameterValue>
            </Parameter>
        </CleanUp>
    </CleanUps>
```

サンプル LocalConfig-lc.xml ファイル（LocalController ランタイム）

```
<?xml version="1.0" encoding="UTF-8"?>
<Runtime>
   <Names>
      <Name name="mainClass">com.torokina.tim.lc.Main</Name>
      <Name name="primaryMcAddress">127.0.0.1</Name>
      <Name name="secondaryMcAddress"/>
      <Name name="primaryMcPort">29599</Name>
      <Name name="secondaryMcPort">-1</Name>
      <Name name="myAddress">127.0.0.1</Name>
      <Name name="appName">CAMM-Local-Controller</Name>
      <Name name="appShortName">LC</Name>
```

```
      <Name name="lcPort">29598</Name>
      <Name name="heartbeatFrequency">15</Name>
      <Name name="heartbeatTimeout">180</Name>
      <Name name="mcPort">29599</Name>
      <Name name="manageable">996</Name>
   </Names>
   <Paths>
      <Path name="dsLocalConfig">${basedir}/DS/LocalConfig-ds.xml</Path>
      <Path name="apphome">${tim.base}/${appShortName}</Path>
      <Path name="runtimeConfig">${apphome}/runtime.xml</Path>
      <Path name="tmp">${apphome}/tmp</Path>
      <Path name="logbase">${apphome}/logs</Path>
      <Path name="basedir">${tim.base}</Path>
   </Paths>
</Runtime>
```

LocalController 設定オプション

以下の情報は、LocalController を設定するために使用できるオプションを説明します。

Path

以下のアイテムのパス情報を指定できます。

basedir

CA Mediation Manager ベース ディレクトリを指定します。

apphome

CA Mediation Manager LocalController アプリケーション ホーム ディレクトリを指定します。

tmp

CA Mediation Manager LocalController 一時ファイル ディレクトリを指定します。

logbase

CA Mediation Manager LocalController ログ ディレクトリを指定します。

runtimeConfig

Generic Executor によって提供される CA Mediation Manager LocalController ランタイム XML 設定ファイルを指定します。

Java

Java を使用するためのオプションを指定できます。

CommandPath

LocalController を開始するために Generic Executor が呼び出す Java 実行プログラムへのフル パスを指定します。

ClassPath/JarBase

ClassPath に追加する 1 つ以上のエントリを使用して、リストを作成できます。

Options

Java に解析されるコマンド ライン オプションを指定します。

Environment

MultiController コンポーネントを実行する環境変数を指定します。

MainClass/Class

実行する Java メイン クラスを指定します。

MainClass/Args

Java クラスに解析された引数を指定します。

Runtime

ランタイム オプションを指定できます。

Binding/BindAddress

LocalController コンポーネントがバインドする IP アドレスを指定します。

値： IP アドレス 0.0.0.0 をすべての LocalController コンポーネントに使用します。 2 つ以上の IP アドレスについては、カンマ区切りリストを使用します。

Binding/MyAddress

LocalController がそれ自体を識別するために使用する IP アドレスを指定します。

注: この IP アドレスはこのホストで有効な IP アドレスである必要があります。

Binding/BindPort

LocalController がそれ自体を識別するために使用する IP アドレスを指定します。

注: この IP アドレスはこのホストで有効な IP アドレスである必要があります。 デフォルトでは、LocalController は TCP ポート 29598 にバインドします。

LocalControllerConfig/Mode

LocalController のオペレーティング モードを指定します。

値： Active または Standby のいずれかを指定します。

LocalControllerConfig/MCAddresses/Primary

クラスタのプライマリ MultiController の IP アドレスを指定します。

LocalControllerConfig/MCAddresses/Secondary

クラスタのセカンダリ MultiController の IP アドレスを指定します。

LocalControllerConfig/Heartbeat/ParameterName == frequency

MultiController に送信されるハートビート メッセージの頻度を指定します。

LocalControllerConfig/Heartbeat/ParameterName == timeout

この LocalController がサブコンポーネント（Engine および Presenter）からのハートビートを待つ時間を指定します。LocalController が 180 秒間サブコンポーネントからハートビートを受信しない場合、LocalController は再起動します。

Logging

ログ記録オプションを指定できます。

LogLevel

出力ログ ファイルのロギング レベルを指定します。

値： DEBUG、TRACE、INFO、WARNING または ERROR を指定します。

LogDirectory

出力ログ ファイルのロギング レベルを指定します。

値： DEBUG、TRACE、INFO、WARNING または ERROR を指定します。

ObjectLogging/ObjectToLog/ObjectName

ログ記録を有効にする Java クラス名を指定します。

ObjectLogging/ObjectToLog/ObjectLogLevel

Java クラスのロギング レベルを指定します。

Cleanup

クリーンアップ オプションを指定できます。

CleanUpName

クリーンアップのわかりやすい名前を指定します。

CleanUpAction

クリーンアップ アクションを指定します。

値： Delete または Archive を指定します。

CleanupTarget

クリーンアップするディレクトリを指定します。

値： ${camm.variable} などの CA Mediation Manager 変数を使用して指定できます。

Parameter/ParameterName – Parameter/ParameterValue

<n><unit> の形式で期限切れになるパラメータおよびその有効期限を指定します。 例：

10d =10 日

10h = 10 時間

10m = 10 分

LocalController の手動による開始と停止

init.camm スクリプトは自動的に LocalController プロセスを開始します。cammCtrl ユーティリティを使用して、LocalController コンポーネントを個別に手動で停止または開始できます。

次の手順に従ってください：

1. CAMM_USER としてログインし、CAMM ホーム ディレクトリに移動します。
2. 以下のコマンドを実行して、LocalController を開始します。

   ```
   /opt/CA/CAMM# tools/startall –c lc
   ```

3. 以下のコマンドを実行して、LocalController を停止します。

   ```
   /opt/CA/CAMM # tools/stopall –c lc
   ```

Engine および Presenter の設定

Engine および Presenter のインストールおよび設定は、各デバイス パックに付属の Device Pack インストール プログラムを使用して行います。

フェールオーバ操作

MultiController は 2 つのモード（プライマリとセカンダリ）のいずれかで動作します。

MultiController には 3 つの主な通信機能があります。

- ハートビート
- ネーム サービス ルックアップ
- リポジトリ同期

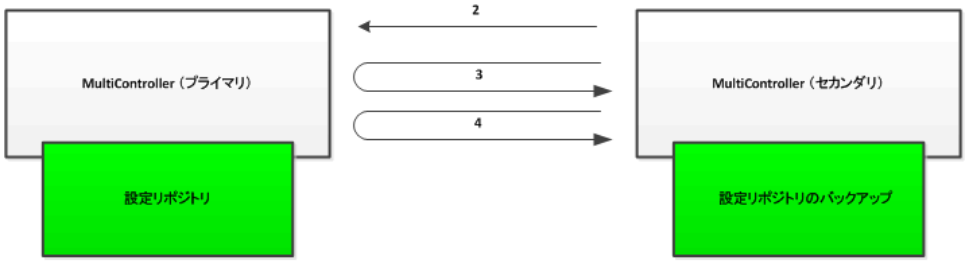
MultiController の障害

以下の情報は、CA Mediation Manager での MultiController の障害の管理方法について説明します。

MultiController 通信

以下の図では、セカンダリ MultiController とプライマリの通信方法を示します。

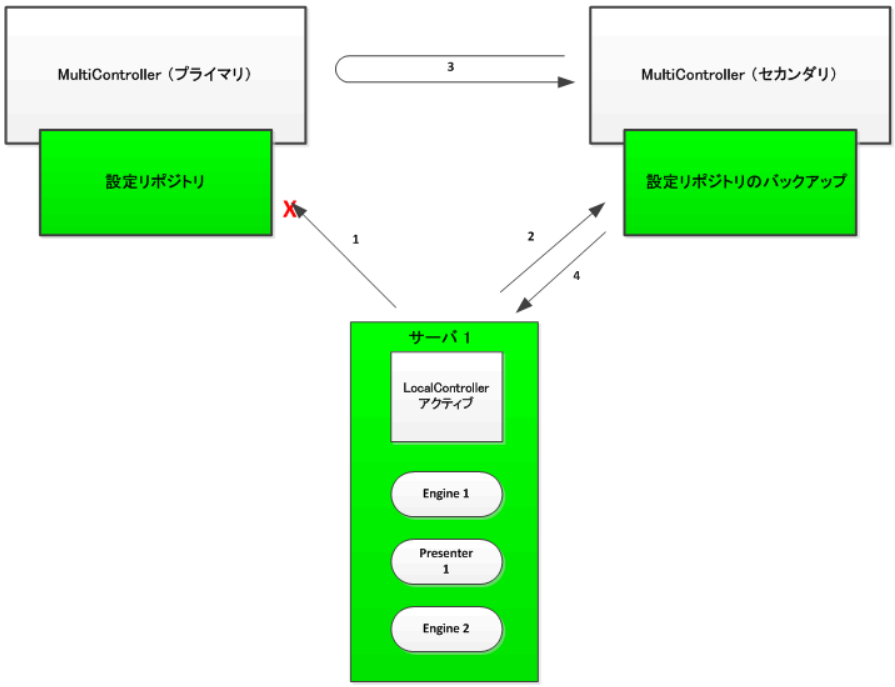
セカンダリが使用不可能になった場合に気づく Watchdog がプライマリにあります。 セカンダリが使用不可能になると、アラートがログ ファイルに作成されます。



1. セカンダリ MultiController が起動する。
2. セカンダリ MultiController は最初のハートビートをプライマリに送信し、アクティブであることを通知する。
3. セカンダリ MultiController は定期的にハートビートをプライマリに送信する。
4. セカンダリ MultiController は定期的にリポジトリを同期する。

プライマリ MultiController の障害

以下の図では、プライマリ MultiController の障害がセカンダリによってどのように処理されるかを示します。 このプロセス中に、プライマリ MultiController が再度使用可能になるまで、セカンダリ MultiController はハートビート、ネーム サービスおよび設定リポジトリ リクエストを引き継ぎます。



1. LocalController はプライマリにハートビートを送信し、失敗する。
2. LocalController はセカンダリ MultiController にハートビートを送信する。
3. セカンダリ MultiController はプライマリが使用可能であると確認し、プライマリ MultiController に要求のプロキシを行う。
4. プライマリは、セカンダリ MultiController を介してハートビート を LocalController に返す。

MultiController プロキシ機能

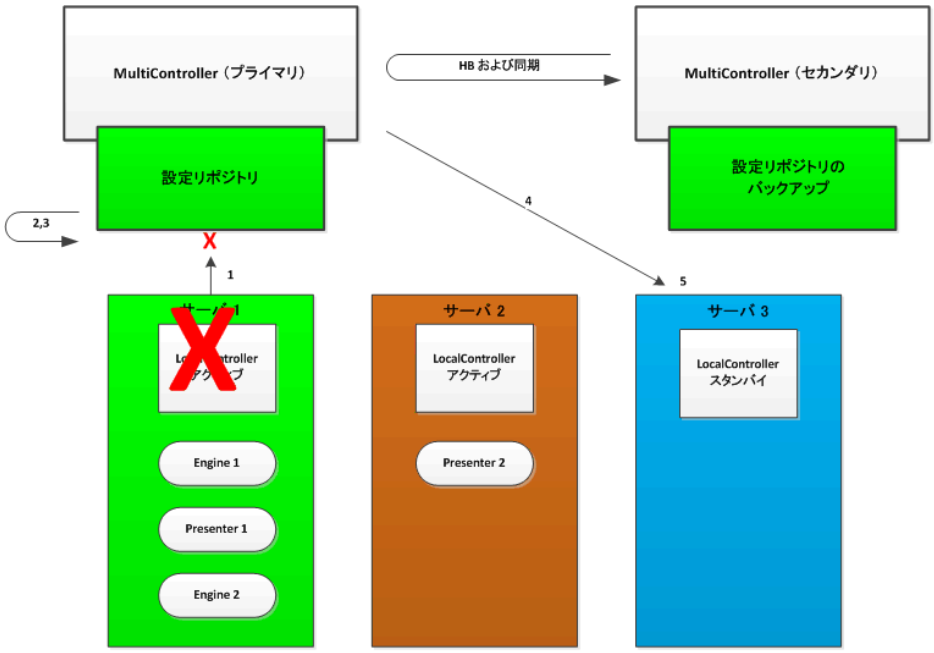
特定のネットワークでは、LocalController はプライマリ MultiController と直接通信できません。 このため、セカンダリ MultiController にはプロキシ機能があり、プライマリ MultiController へのハートビートの送信を試行します。

ただし、セカンダリ MultiController に送信された設定リポジトリ リクエストは、設定リポジトリが同期されるように、セカンダリ MultiController によって処理されます。

LocalController の障害

LocalController コンポーネントはすべて MultiController にハートビート情報を送信します。 アクティブな MultiController が LocalController からハートビートを受信しなかったことを検出した場合、それは抜けているハートビートをログ記録します。 この時間がタイムアウト期間を超えている場合、MultiController は以下の図で説明しているプロセスを使用して、最初の使用可能なスタンバイ LocalController へのフェールオーバをトリガします。

フェールバックは手動プロセスです。 フェールオーバを行うには、Server 3 上の LocalController で障害を強制的に発生させます。



1. MultiController はタイムアウト期間までにハートビートを受信しなかったことを検出する。
2. MultiController はすべての Engine サブコンポーネント用の設定リポジトリをサーバ 1 からサーバ 3 に移動する。
3. サーバ 1 にインストール済みのサブコンポーネントがない場合、MultiController はサーバ 1 の設定をスタンバイにマークする。
4. MultiController は、サーバ 3 の LocalControllerにメッセージを送信し、再起動して設定を再度読み込むように指示する。
5. サーバ 3 の LocalController がアクティブになる。

サブコンポーネントの障害

LocalController コンポーネントはその実行しているサブコンポーネント（Engine および Presenter）を監視します。 サブコンポーネントに障害が発生した場合、サブコンポーネントはすぐに再起動されます。

高可用性設定

以下のセクションでは、高可用性用の CA Mediation Manager を設定する方法について説明します。

プライマリ MultiController の設定

プライマリ MultiController のインストール中に、高可用性用のプライマリ MultiController を設定します。

次の手順に従ってください：

1. ［MultiController Configuration］パネルで、［*Will the other MC exist in the Cluster*］チェック ボックスをオンにします。
2. プライマリ MultiController IP アドレスを提供し、［Next］をクリックします。
3. ［Other MultiController Configuration］パネルで、セカンダリ MultiController の IP アドレスを提供します。
4. ［Next］をクリックし、インストールを完了します。

セカンダリ MultiController の設定

セカンダリ MultiController インストール中に、高可用性用のセカンダリ MultiController を設定します。

次の手順に従ってください：

1. ［MultiController Configuration］パネルで、［MultiController］ドロップダウンから［Secondary］を選択します。
2. セカンダリ MultiController IP アドレスを提供します。
3. ［*Will the other MC exist in the Cluster*］チェック ボックスをオンにし、［Next］をクリックします。
4. ［Other MultiController Configuration］パネルで、プライマリ MultiController の IP アドレスを提供します。
5. ［Next］をクリックし、インストールを完了します。

LocalController の構成

LocalController のインストール中に、高可用性用の LocalController を設定します。

次の手順に従ってください：

1. ［LocalController Configuration］パネルで、プライマリ MultiController およびセカンダリ MultiController の IP アドレスを提供します。
2. ［Next］をクリックし、インストールを完了します。
3. 各 LocalController のインストールでこれらの手順を繰り返します。

ログ ファイルの設定

以下のセクションでは、CA Mediation Manager でログ ファイルを設定する方法について説明します。

すべての CA Mediation Manager コンポーネントのログ プロパティは、logging.properties ファイルを使用して設定します。このファイルは、コンポーネントの起動時にデフォルト ログ ディレクトリに作成されます。logging.properties ファイルは、デフォルト ログ ディレクトリにログ ファイルを生成するようにあらかじめ設定されています。 しかし、logging.properties ファイルを編集することにより、別のディレクトリにログ ファイルを転送できます。logging.properties ファイルを変更したら、コンポーネントを再起動して、変更したログ プロパティをロードします。すべてのログ（アプリケーション ログおよび STD-ERROR/STD-OUTPUT ログ）は、logging.properties ファイルに指定されているディレクトリに生成されます。

logging.properties ファイル - コンポーネント別の例

以下の例では、各 logging.properties ファイルについて説明します。

MultiController の logging.properties ファイル例

```
com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/CAMM/MC/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-Multi-Controller-
```

LocalController の logging.properties ファイル例

```
com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/CAMM/LC/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-Local-Controller-
```

配信システムの logging.properties ファイル例

```
com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/CAMM/DS/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-Delivery-System-
```

ENGINE_CAMM の logging.properties ファイル例

```
com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/CAMM/COMPONENTS/ENGINE_CAMM/logs

handlers=com.torokina.common.logging.apache.FileHandler
```

```
com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-ENGINE_
CAMM-
```

Generic Executor の logging.properties ファイル例

デフォルトでは、Generic Executor はそのログ ディレクトリで logging.properties ファイルを作成しません。 ログはすべて ~GE/ログ ディレクトリで生成されます。 以下の例では、logging.properties ファイルをログ ディレクトリに作成して、Generic Executor ログをリダイレクトする方法を示します（Windows プラットフォーム以外）。

```
#Properties for Logger

#Tue May 07 04:08:45 EDT 2013

com.torokina.common.logging.apache.FileHandler.directory=/opt/CA/C
AMM/GE/logs

handlers=com.torokina.common.logging.apache.FileHandler

com.torokina.common.logging.apache.FileHandler.level=INFO

.level=INFO

com.torokina.common.logging.apache.FileHandler.prefix=CAMM-Generic
-Executor-
```

ログ ファイルのクリーンアップの設定

デフォルトでは、各コンポーネントのクリーンアップ アクションは *logbase* ディレクトリで実行されるように設定されています。 ログ ファイルが別のディレクトリにリダイレクトされる場合は、*Archive* または *Delete* アクションが正常に行われるようにクリーンアップ設定を変更します。

設定ファイル内の *LocalConfig* XML エレメントでクリーンアップ アクションを定義します。 すべての CAMM コンポーネントについては、それぞれの設定ファイルでクリーンアップ アクションを定義します。

- **MultiController：** LocalConfig-mc.xml
- **LocalController：** LocalConfig-lc.xml
- **Delivery Service：** LocalConfig-ds.xml
- **サブコンポーネント（ENGINE または PRESENTER）：** TemplateConfig-subcomponent.xml

  注: TemplateConfig-subcomponent.xml ファイルは <camm.base>/LC ディレクトリにあります。

以下の例では、2 つのサンプル クリーンアップ設定について説明します。

例: Delivery Service コンポーネント用のクリーンアップ設定ファイル（*Delete* アクション）

```
<LocalConfig>

<Description>Configuration for Delivery Module</Description>

       …

       …

       <CleanUps>

              <!-- SAMPLE  DELETE  ACTION -->

              <CleanUp>

                     <CleanUpName>Delete</CleanUpName>

                     <CleanUpAction>delete</CleanUpAction>


       <CleanUpTarget>${apphome}/.local</CleanUpTarget> <!--
Directory Name -->

                     <Parameter>

                            <ParameterName>expire</ParameterName>

                            <ParameterValue>7d</ParameterValue> <!--
1y0m3d1h -->

                     </Parameter>

                     <Parameter>


       <ParameterName>includeDir</ParameterName>


       <ParameterValue>true</ParameterValue><!-- true/false -->

                     </Parameter>

                     <Parameter>


       <ParameterName>recursive</ParameterName>


       <ParameterValue>true</ParameterValue><!-- true/false -->
```

```
                </Parameter>
                <Parameter>
                        <ParameterName>match</ParameterName>
<ParameterValue>^[¥d]+¥.xml$</ParameterValue>
                        <!-- Regular Pattern -->
                </Parameter>
        </CleanUp>
```

例: Delivery Service コンポーネント用のクリーンアップ設定ファイル(*Archive* アクション)

```
<!-- SAMPLE  ARCHIVE  ACTION -->
<CleanUp>
<CleanUpName>Archive</CleanUpName>
<CleanUpAction>archive</CleanUpAction>
<CleanUpTarget>${logbase}</CleanUpTarget> <!--
Directory Name -->
<Parameter>
<ParameterName>expire</ParameterName>
<ParameterValue>7d</ParameterValue> <!--
1y0m3d1h -->
</Parameter>
<Parameter>
<ParameterName>includeDir</ParameterName>
<ParameterValue>true</ParameterValue><!-- true/false -->
</Parameter>
<Parameter>
<ParameterName>recursive</ParameterName>
<ParameterValue>true</ParameterValue><!-- true/false -->
</Parameter>
<Parameter>
<ParameterName>match</ParameterName>
<ParameterValue>CAMM-.*¥.log</ParameterValue>
<!-- Regular Pattern -->
</Parameter>
<Parameter>
```

```
        <ParameterName>achiveHome</ParameterName>

        <ParameterValue>${logbase}</ParameterValue> <!-- folder path
-->
                    </Parameter>
                    <Parameter>

        <ParameterName>achivePrefix</ParameterName>

        <ParameterValue>Archive-</ParameterValue> <!-- prefix string
-->
                    </Parameter>
                    <Parameter>

        <ParameterName>achiveSuffix</ParameterName>
                            <ParameterValue>.zip</ParameterValue>
<!-- suffix string -->
                    </Parameter>
                </CleanUp>
        </CleanUps>
<LocalConfig>
```

コンポーネントの設定ファイルには任意のクリーンアップ アクションを入れることができます。 Generic Executor はクリーンアップ アクションをすべて実行します。 Generic Executor は {camm.base}/GE_<User>/cleanup ディレクトリにクリーンアップ ファイルを格納します。ここで camm.base は CA Mediation Manager インストール ディレクトリです。 コンポーネントの設定ファイルを変更した場合、変更後の設定が有効になるように、関連するコンポーネントを再起動します。

第 5 章: EMS プロファイルの使用（CA Mediation Manager for Infrastructure Management 2.0、リリース 2.2.3）

注: この章の情報は CA Mediation Manager for Infrastructure Management 2.0 にのみ適用されます。

このセクションには、以下のトピックが含まれています。

EMS 統合プロファイル

EMS 統合プロファイルは、EMS インベントリ ディスカバリが Data Aggregator 環境でどのように動作するかを指定します。

EMS 統合プロファイルで、ステータス、データ コレクタ、デバイス パック、EMS IP および Backup EMS IP を指定します。 作成する各 EMS 統合プロファイルにつき 1 つの IP ドメインを指定します。 指定する IP ドメインはターゲット EMS サーバ用です。それは複数のデバイス（通常は一度に 1,000 のデバイス）を管理します。 Data Aggregator は、EMS サーバからのインベントリ データを絶えず同時に処理します （SNMP または ICMP を使用する場合、ポーリングは、デバイス単位で行われます）。

注:

- デバイス パックをインストールすると、EMS Integration Profiles オプションがユーザ インターフェースに表示されます。
- 同じ Data Collector に対して、同じ EMS 統合プロファイルを 2 回以上追加しないでください。

EMS 統合プロファイルの追加

EMS インベントリ ディスカバリが Data Aggregator 環境でどのように動作するかを指定するために EMS 統合プロファイルを作成できます。

注: このタスクを実行するには、管理者としてログインする必要があります。

まず範囲をテナントに設定せずに、EMS 統合プロファイルを作成すると、グローバル 領域（すべてのテナントによってアクセス可能）にそのプロファイルが配置されます。 グローバル領域のプロファイルを使用してディスカバリを実行すると、範囲をテナントに設定したかどうかに関係なく、すべてのユーザがディスカバリ結果を確認できます。

そのため、EMS 統合プロファイルを特定のテナントのみにアクセス可能にするには、そのプロファイルを作成する*前に*、範囲をテナントに設定します。 範囲をテナントに設定した後、テナント インジケータがページの右上に表示されます。 その後、テナントを CA Infrastructure Management と手動で同期するか、または自動同期が発生するまで待機できます。 テナントが Data Aggregator と同期されるまで、EMS 統合プロファイルを作成できません。

注: 範囲をテナントに設定し、テナントを同期する詳細については、「CA Performance Center の統合方法 管理者ガイド」を参照してください。

次の手順に従ってください:

1. CA Performance Center の統合方法 ユーザ インターフェースで、［管理］-［データ ソース設定］を選択し、Data Aggregator データ ソースをクリックします。
2. ［監視設定］メニューから［EMS 統合プロファイル］をクリックします。

   ［EMS 統合プロファイル］ページが開き、利用可能なディスカバリ プロファイルのリストが表示されます。
3. ［新規］をクリックします。

   ［EMS ディスカバリ プロファイルの追加］ダイアログ ボックスが表示されます。

4. フィールドに、必要な情報を入力します。 表示される設定フィールドは、選択するデバイス パックによって異なります。 デバイス パックにはそれぞれ設定する一意のグローバル変数があります。

   注: 各製品の一意のグローバル変数の詳細については、CA Support サイトを参照してください。

5. ［保存］をクリックします。

   EMS 統合プロファイルが作成されます。

   ［保存］をクリックし、［有効］オプションが選択されている場合でも、インベントリ ディスカバリは自動的に実行されません。以下のいずれかの条件を満たした場合のみ、インベントリ ディスカバリが実行されます。

   - インベントリ ポーリング スケジュールに到達した場合。
   - EMS 統合プロファイルが手動で開始された場合。

詳細情報:

EMS ディスカバリ結果の表示 (P. 62)
手動での EMS ディスカバリの開始 (P. 61)

手動での EMS ディスカバリの開始

EMS 統合プロファイルはネットワーク内のデバイスおよびそれらのコンポーネントを検出します。 ディスカバリを開始するために手動で EMS 統合プロファイルを開始できます。

注: または、インベントリ ポーリング スケジュールに到達し、ディスカバリが自動的に開始されるまで、待機できます。

次の手順に従ってください:

1. CA Performance Center の統合方法 ユーザ インターフェースで、［管理］-［データ ソース設定］を選択し、Data Aggregator データ ソースをクリックします。
2. ［監視設定］メニューから［EMS 統合プロファイル］をクリックします。

   ［EMS 統合プロファイル］ページが開き、利用可能なディスカバリ プロファイルのリストが表示されます。

3. ディスカバリを実行する EMS 統合プロファイルを 1 つ以上選択し、［開始］をクリックします。

   注: ディスカバリを実行できるのは、「準備完了」ステータスの EMS 統合プロファイルのみです。

   確認ダイアログ ボックスが表示されます。

4. ［はい］をクリックします。

   ディスカバリが開始されます。選択したディスカバリ プロファイルの［ステータス］列は「開始」を示します。

   確認ダイアログ ボックスが表示されます。

5. ［OK］をクリックします。

   デバイスおよびそれらの関連するすべてのインターフェースが検出され、ポーリングが開始されます。 ［EMS 統合プロファイル］ページに戻ります。

   ディスカバリが 10 分を超えてハングアップする場合は Data Aggregator はそれを停止します。 Data Aggregator では、ディスカバリは、新しいデバイスが 10 分以内に検出されず、さらに選択されたディスカバリ プロファイルの状態が 10 分以内に変更されなかった場合に、ハングアップしていると見なされます。 ディスカバリ インスタンス アイテムで監査イベントが生成されます。デバイスが正常に検出されなかった場合、選択したディスカバリ プロファイルの［ステータス］列は「失敗」を示します。 少なくとも 1 つのデバイスは正常に検出されたが、一部のデバイスが検出されなかった場合、［ステータス］列は「部分的な失敗」を示します。

   検出されたデバイスおよびコンポーネントは、CA Performance Center の統合方法 との同期に最長 5 分間かかります。同期が完了すると、検出されたデバイスおよびコンポーネントが CA Performance Center の統合方法 内の［インベントリ］タブ内に表示されます。

詳細情報:

EMS ディスカバリ結果の表示 (P. 62)

EMS ディスカバリ結果の表示

検出されたすべての管理可能な EMS デバイスの数のサマリを表示できます。

次の手順に従ってください:

1. CA Performance Center の統合方法 ユーザ インターフェースで、［管理］-［データ ソース設定］を選択し、Data Aggregator データ ソースをクリックします。
2. ［監視設定］メニューから［EMS 統合プロファイル］をクリックします。

   ［EMS 統合プロファイル］ページが開き、利用可能なディスカバリ プロファイルのリストが表示されます。
3. ディスカバリ結果を表示する EMS 統合プロファイル インスタンスを選択し、［履歴］をクリックします。

   EMS 履歴結果は以下のように表示されます。

   - デバイス テーブルには、監視対象デバイスと各デバイスの作成時間が表示されます。
   - エレメント テーブルには、監視対象インターフェースと各インターフェースの作成時間が表示されます。

EMS ディスカバリ サービスの開始または停止

「開始」サービスは EMS サーバ上のディスカバリがインベントリを連続的に確認するために使用されます。 ディスカバリはスケジュールできますが、必要に応じて手動でサービスを開始、停止、再起動できます。 たとえば、EMS サーバがダウンし、再起動した後、またはデバイス パック インストールをアップグレードした後、ディスカバリを再起動できます。

サービスを停止すると、すべての非アクティブ データ ポーリングは削除されますが、アクティブなデータ ポーリングが完了するまで中断なしで待機します。 EMS ファイルは削除されません。 また、このアクションにより、サービスが停止状態である限り、すべての新しいポーリングを実行できなくなります。再度開始するまで、サービスは停止状態のままです。

Data Collector の再起動は、EMS 統合プロファイルのステータスに影響しません。

注: このタスクを実行するには、管理者としてログインする必要があります。

次の手順に従ってください:

1. CA Performance Center の統合方法 ユーザ インターフェースで、［管理］-［データ ソース設定］を選択し、Data Aggregator データ ソースをクリックします。
2. ［監視設定］メニューから［EMS 統合プロファイル］をクリックします。

   ［EMS 統合プロファイル］ページが開き、利用可能なディスカバリ プロファイルのリストが表示されます。
3. プロファイルを選択し、［開始］または［停止］をクリックします。

   確認ダイアログ ボックスが表示されます。
4. 操作を確認するために［はい］をクリックします。

   サービスはユーザの選択に応じて、開始するか停止します。 このサービスは、それを手動で変更するまで、開始状態または停止状態のままです。

イベント ルールの追加

イベント ルールは Data Aggregator 監視プロファイル ダイアログ ボックスを使用して追加できます。

注: 詳細については、「Data Aggregator 管理者ガイド」またはオンライン ヘルプを参照してください。