

# CA Mediation Manager および CA Mediation Manager for Infrastructure Management

## 管理ガイド

CA Mediation Manager リリース 2.2.2/CA Mediation Manager for  
Infrastructure Management 2.0、リリース 2.2.2



このドキュメント（組み込みヘルプシステムおよび電子的に配布される資料を含む、以下「本ドキュメント」）は、お客様への情報提供のみを目的としたもので、日本 CA 株式会社（以下「CA」）により随時、変更または撤回されることがあります。

CA の事前の書面による承諾を受けずに本ドキュメントの全部または一部を複写、譲渡、開示、変更、複本することはできません。本ドキュメントは、CA が知的財産権を有する機密情報です。ユーザは本ドキュメントを開示したり、

(i) 本ドキュメントが関係する CA ソフトウェアの使用について CA とユーザとの間で別途締結される契約または (ii) CA とユーザとの間で別途締結される機密保持契約により許可された目的以外に、本ドキュメントを使用することはできません。

上記にかかわらず、本ドキュメントで言及されている CA ソフトウェア製品のライセンスを受けたユーザは、社内でユーザおよび従業員が使用する場合に限り、当該ソフトウェアに関連する本ドキュメントのコピーを妥当な部数だけ作成できます。ただし CA のすべての著作権表示およびその説明を当該複製に添付することを条件とします。

本ドキュメントを印刷するまたはコピーを作成する上記の権利は、当該ソフトウェアのライセンスが完全に有効となっている期間内に限定されます。いかなる理由であれ、上記のライセンスが終了した場合には、お客様は本ドキュメントの全部または一部と、それらを複製したコピーのすべてを破棄したことを、CA に文書で証明する責任を負います。

準拠法により認められる限り、CA は本ドキュメントを現状有姿のまま提供し、商品性、特定の使用目的に対する適合性、他者の権利に対して侵害のないことについて、黙示の保証も含めいかなる保証もしません。また、本ドキュメントの使用に起因して、逸失利益、投資損失、業務の中断、営業権の喪失、情報の喪失等、いかなる損害（直接損害か間接損害かを問いません）が発生しても、CA はお客様または第三者に対し責任を負いません。CA がかかる損害の発生の可能性について事前に明示に通告されていた場合も同様とします。

本ドキュメントで参照されているすべてのソフトウェア製品の使用には、該当するライセンス契約が適用され、当該ライセンス契約はこの通知の条件によっていかなる変更も行われません。

本ドキュメントの制作者は CA です。

「制限された権利」のもとでの提供: アメリカ合衆国政府が使用、複製、開示する場合は、FAR Sections 12.212、52.227-14 及び 52.227-19(c)(1)及び(2)、ならびに DFARS Section 252.227-7014(b)(3) または、これらの後継の条項に規定される該当する制限に従うものとします。

Copyright © 2013 CA. All rights reserved. 本書に記載された全ての製品名、サービス名、商号およびロゴは各社のそれぞれの商標またはサービスマークです。

## 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

---

<b>第 1 章: はじめに</b>	<b>7</b>
アーキテクチャ .....	8
<b>第 2 章: 概要</b>	<b>9</b>
コンポーネントの概要 .....	9
主要コンポーネント .....	9
MultiController .....	10
LocalController .....	10
Web .....	11
サブコンポーネント .....	16
Engine .....	16
Presenter .....	16
その他のコンポーネント .....	17
Generic Executor .....	17
Delivery Service .....	17
<b>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</b>	<b>19</b>
システム要件 .....	19
インストールおよびアップグレード .....	20
サービスの開始と停止 .....	20
UNIX .....	20
Windows .....	21
<b>第 4 章: コンポーネントの設定</b>	<b>23</b>
Generic Executor の設定 .....	24
Generic Executor の仕組み .....	26
Generic Executor の設定オプション .....	28
Generic Executor のスタートアップシーケンス .....	28
別の Generic Executor の追加 (UNIX) .....	29
MultiController 設定 .....	30
MultiController の設定オプション .....	33
MultiController の手動による開始と停止 .....	37

---

LocalController 設定 .....	38
LocalController 設定オプション .....	40
LocalController の手動による開始と停止 .....	44
Engine および Presenter の設定 .....	45
フェールオーバー操作 .....	45
MultiController の障害 .....	45
MultiController 通信 .....	46
プライマリ MultiController の障害 .....	47
MultiController プロキシ機能 .....	47
LocalController の障害 .....	48
サブコンポーネントの障害 .....	48
高可用性設定 .....	49
プライマリ MultiController の設定 .....	49
セカンダリ MultiController の設定 .....	49
LocalController の構成 .....	50
ログファイルの設定 .....	50
logging.properties ファイル - コンポーネント別の例 .....	51
ログファイルのクリーンアップの設定 .....	53

## 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 2.2.2) 59

EMS 統合プロファイル .....	59
EMS 統合プロファイルの追加 .....	60
手動での EMS ディスカバリの開始 .....	61
EMS ディスカバリ結果の表示 .....	62
EMS ディスカバリ サービスの開始または停止 .....	63
イベント ルールの追加 .....	64

# 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:** この章で示される情報は CA Mediation Manager のみに適用されます。

このセクションには、以下のトピックが含まれています。

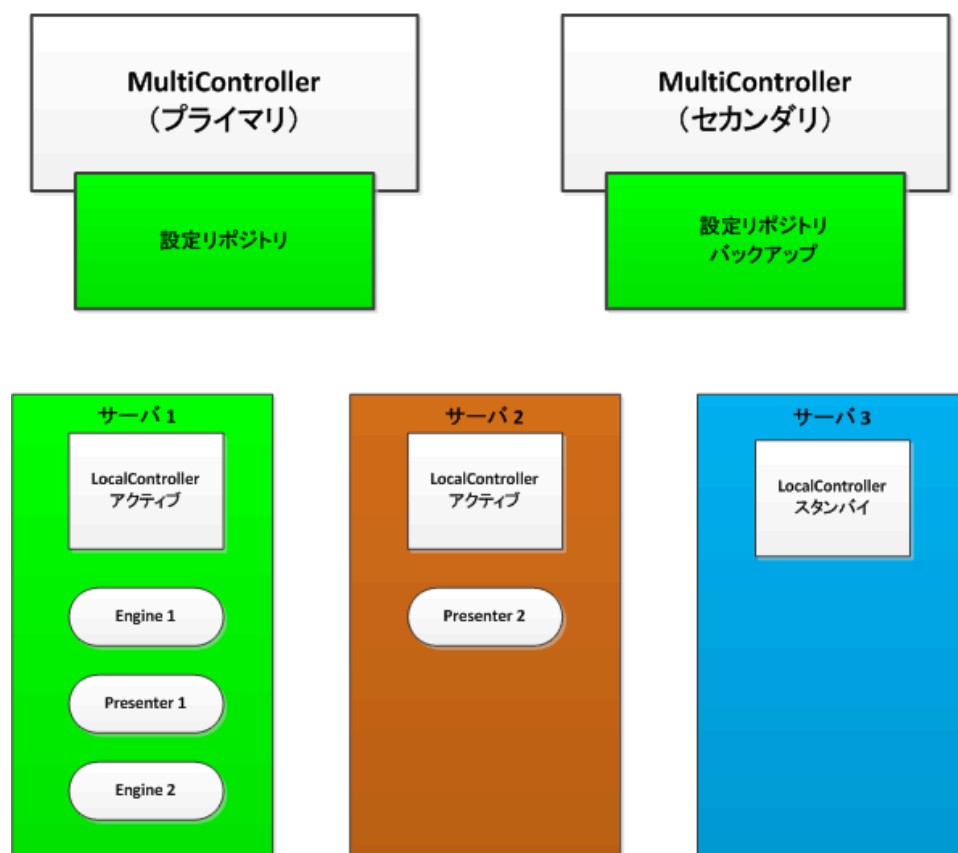
[アーキテクチャ](#) (P. 8)

## 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：この章で示される情報は CA Mediation Manager のみに適用されます。

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](#) (P. 9)

[主要コンポーネント](#) (P. 9)

[サブコンポーネント](#) (P. 16)

[その他のコンポーネント](#) (P. 17)

### コンポーネントの概要

CA Mediation Manager インストール パッケージは、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 インストール パッケージは、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 (GE)

デバイス パックをインストールすると、Engine および Presenter のサブコンポーネント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、CAMM\_HOME 下の DpConfig フォルダで「デバイス パック ガイド」を参照してください。CAMM\_HOME は CA Mediation Manager がインストールされるディレクトリです。

### 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

クラスタ内に、プライマリ 1 つ、セカンダリ 1 つの 2 つまでの **MultiController** 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る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** サブコンポーネントからの出力を処理します。このサービスは、ローカルまたはリモート **Presenter** サブコンポーネントに圧縮して暗号化された形式で XML ドキュメントを配信します。

## Web

Web コンポーネントでは、Web ベース インターフェースを使用してデバイス パック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。
- 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、<PrimaryMCMachineIP> は**プライマリ MultiController** システムの IP アドレス/ホスト名です。

セカンダリ Web サーバは、プライマリ MultiController が応答していない場合にセカンダリ MultiController がアクティブになったときに限り、セカンダリ MultiController によって自動的に起動されます。

セカンダリ Web サーバは、プライマリ MultiController が応答を開始した場合にセカンダリ MultiController がパッシブになったときに限り、セカンダリ MultiController によって自動的に停止されます。

- 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、<SecondaryMCMachineIP> は**セカンダリ MultiController** システムの IP アドレス/ホスト名です。

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

### Default Options

以下の情報は、CA Mediation Manager の [Management] タブから使用可能なデフォルト オプションの説明です。

#### Install/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。デフォルト パスは `$CAMM_HOME/MC/repository/device packs` です。別のパスを使用した場合、デバイス パックのパスを参照して選択できます。

#### 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

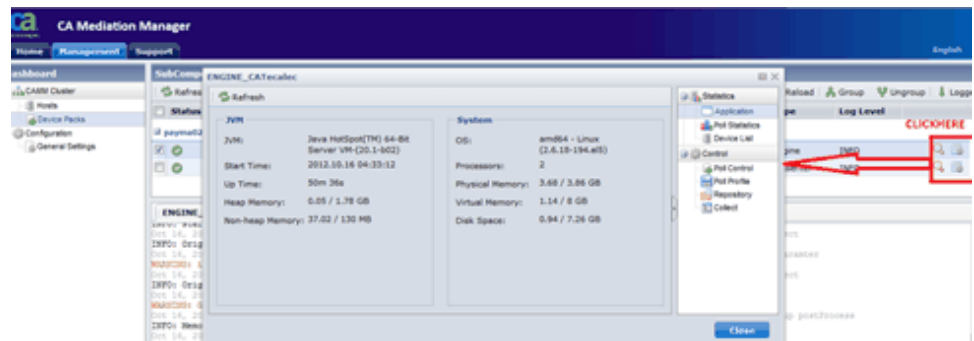
以下の情報は高度なオプションの説明です。

#### 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。

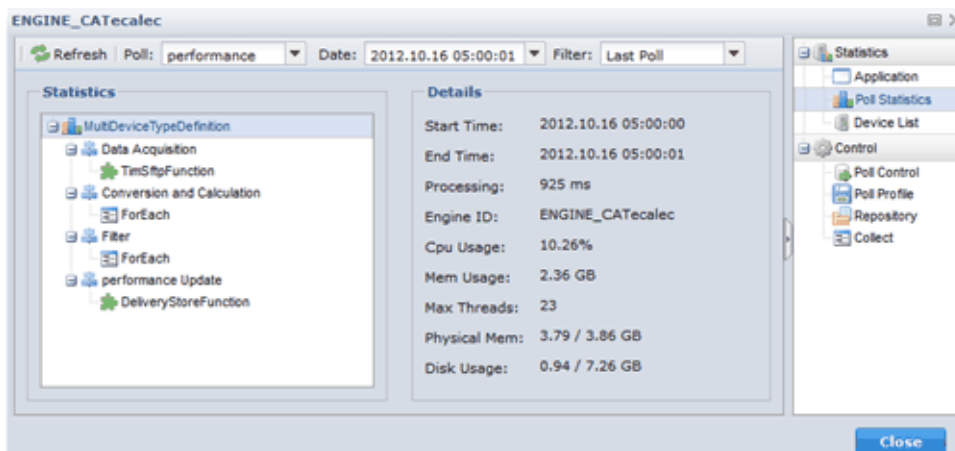
#### Application

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。



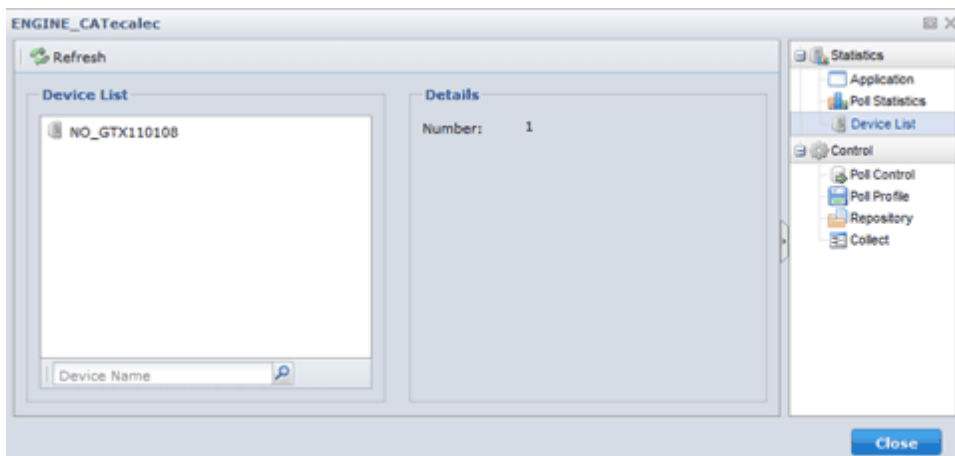
### Poll Statistics

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

\$SCAMM\_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。



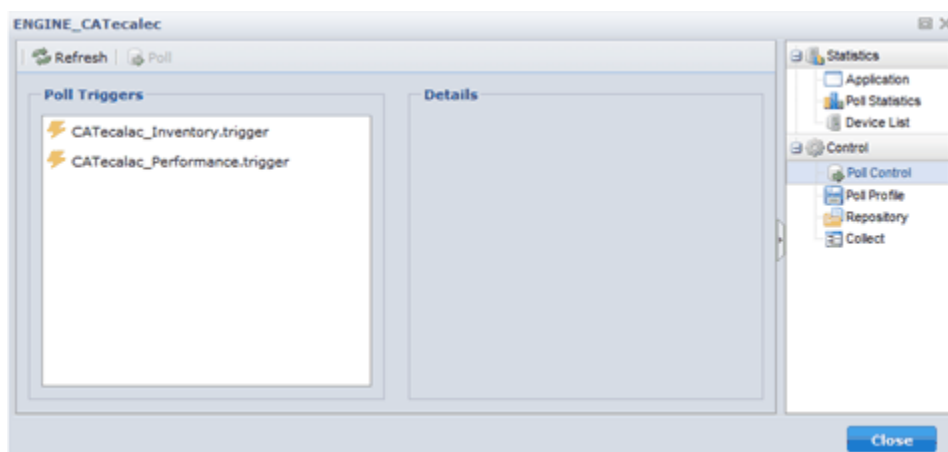
### Control

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

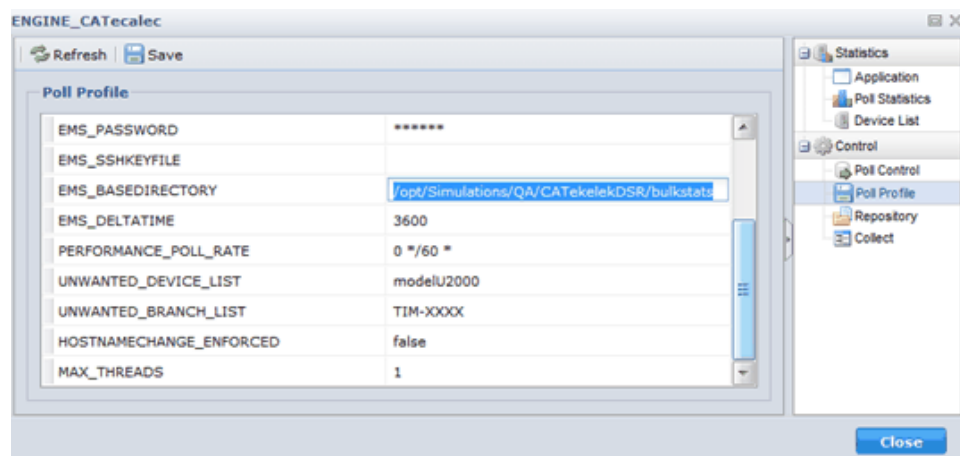
インベントリ ポーリングおよびパフォーマンス ポーリングをトリガします。



### Poll Profile

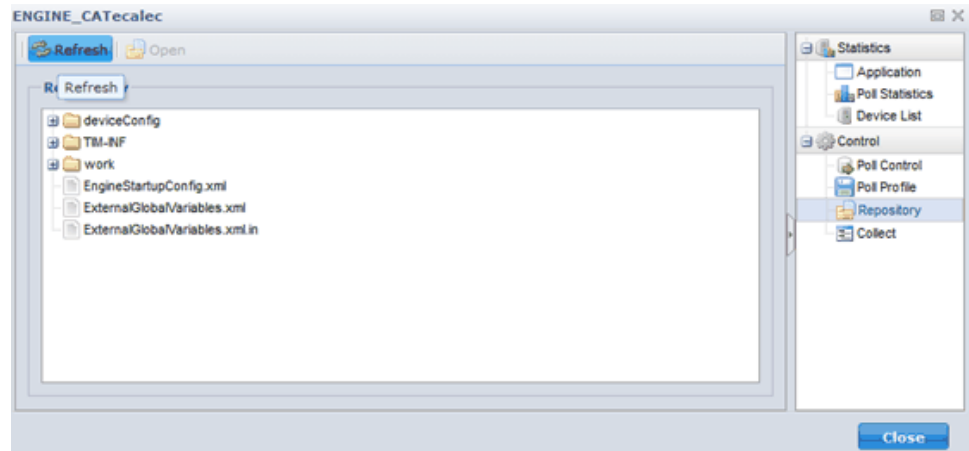
変更した外部グローバル変数のリストを組み込みます。

Engine/Presenter を再起動して変更を反映させます。



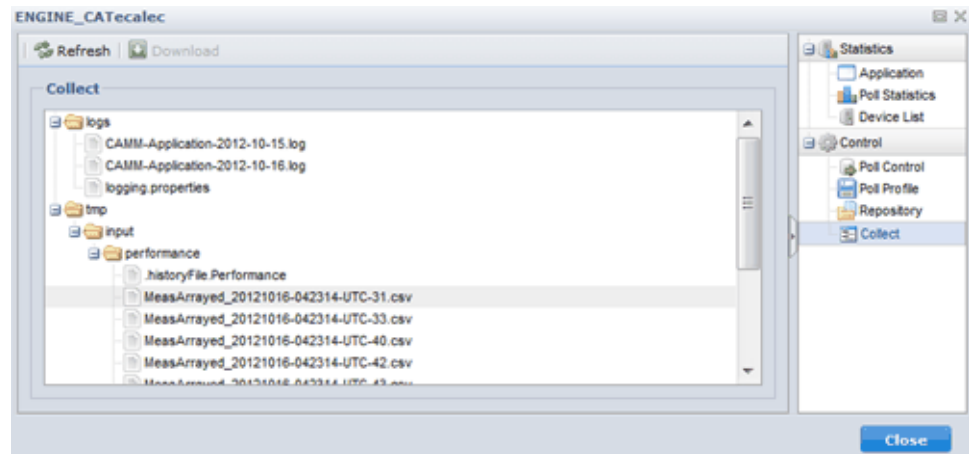
## Repository

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

一般的な設定を指定するために以下のオプションを使用します。

#### 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、および構文の強調表示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。Engine はアクティブ モードまたはスタンバイ モードのいずれかで展開できます。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 ドキュメントヘデータを処理します。
- Delivery Service によって処理するために、キューに CA Mediation Manager 標準の XML ドキュメントを展開します。

### Presenter

Presenter サブコンポーネントは、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 2 つのその他の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

クラスタ内のコンポーネントはすべて、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** (GE) は、必要なコンポーネントの役割を作成して起動し、引き受けます。

**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が起動する必要がある場合、**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 **Presenter** サブコンポーネントまたはリモート **Presenter** サブコンポーネントにデータを配信します。

サブコンポーネントが使用不可能な場合、リモート サブコンポーネントが使用可能になるまで、**Delivery Service** は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: この章で示される情報は CA Mediation Manager のみに適用されます。

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](#) (P. 19)

[インストールおよびアップグレード](#) (P. 20)

[サービスの開始と停止](#) (P. 20)

## 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/stopall スクリプトを使用して、CA Mediation Manager を起動または停止できます。CA Mediation Manager がシステムの起動時に自動的に実行され、システムのシャットダウン時に自動的に停止するように、別のスクリプト `init.camm` が UNIX ホスト `init` スクリプトにインストールされます。

`init` スクリプトをインストールするには、CAMM ホーム ディレクトリで `root` または `sudo su` として以下のコマンドを実行します。

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

CA Mediation Manager は、システムの起動時に自動的に起動し、システムのシャットダウン時に自動的に停止します。

コマンド `init.camm.uninstall` はこの設定を削除します。

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

## Windows

Windows で CA Mediation Manager 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。

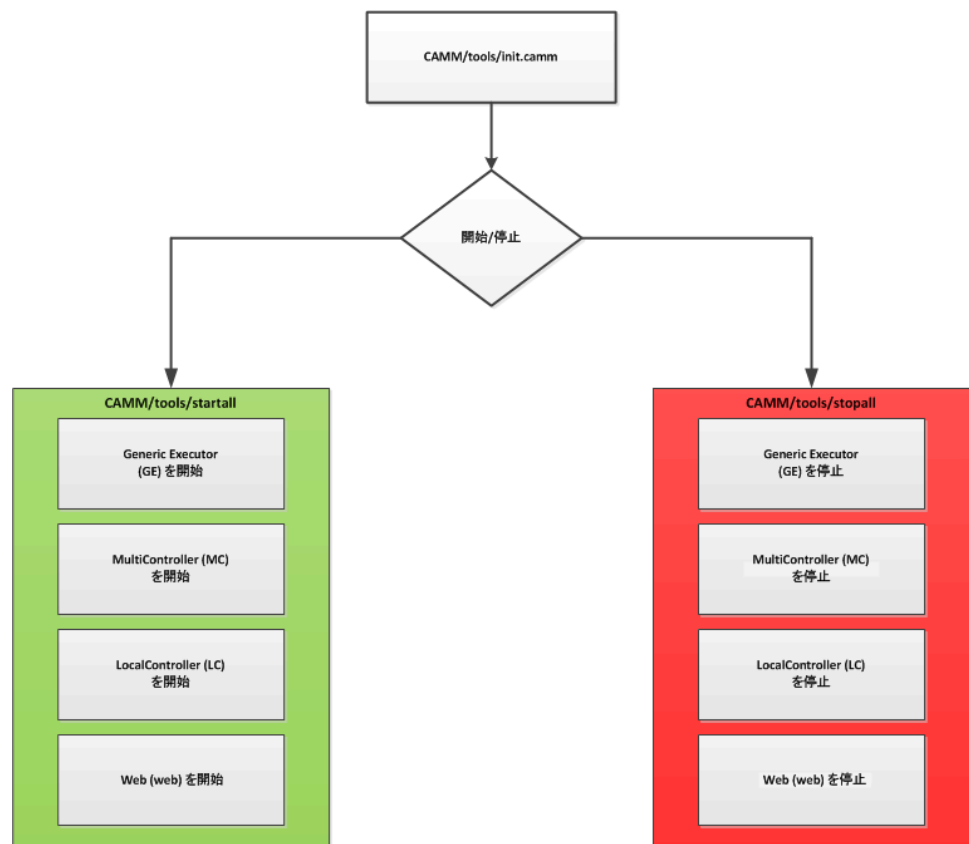
デフォルトでは、これらのサービスは手動で開始されます。Linux と同様、Windows で、CA Mediation Manager がシステムの起動時に自動的に起動し、システムのシャットダウン時に自動的に停止するように、`init.camm.install.bat` を実行できます。

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

コマンド `init.camm.uninstall.bat` はこの設定を削除します。

```
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 でコンポーネントをどのように設定するかについて説明します。

**注:** この章で示される情報は CA Mediation Manager のみに適用されます。

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](#) (P. 24)

[MultiController 設定](#) (P. 30)

[LocalController 設定](#) (P. 38)

[フェールオーバー操作](#) (P. 45)

[MultiController の障害](#) (P. 45)

[高可用性設定](#) (P. 49)

[ログ ファイルの設定](#) (P. 50)

## 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 内のディレクトリにインストールされます。設定は、GE\_<userid> という形式を使用して識別されます。<userid> は Generic Executor のインストールで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 で設定されるのと同じユーザ ID を使用して実行します。

### 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。同じサーバ上にあるプライマリ Generic Executor 以外の他の Generic Executor には、このセクションを含めることはできません。

## Generic Executor の仕組み

Generic Executor は、CA Mediation Manager コンポーネントの基礎である再使用可能なエンティティです。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。

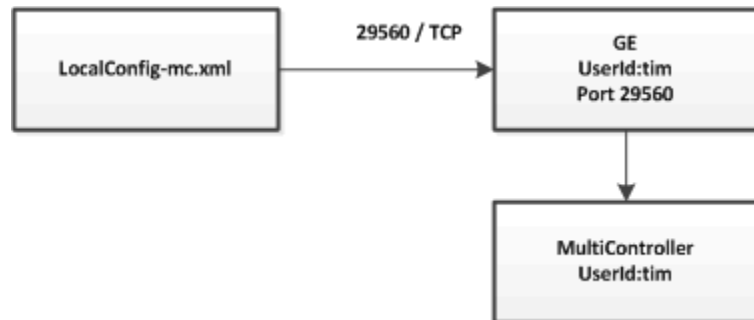
デフォルトでは、LocalConfig-ge.xml ファイルには、CAMM\_USER (CA Mediation Manager をインストールするために使用されたユーザ) のユーザ ID および TCP ポート (デフォルトポートは 29560) が含まれます。



この情報がインストールパッケージによって非表示にされていても、Generic Executor がどのように動作するのかを理解することは重要です。

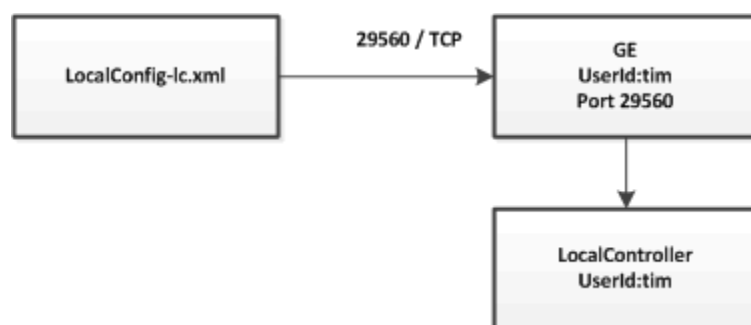
MultiController をこのサーバで実行する必要がある場合、MultiController 用の LocalConfig-mc.xml 設定ファイルおよび Generic Executor 用の TCP ポートが必要です。

以下の図では、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。



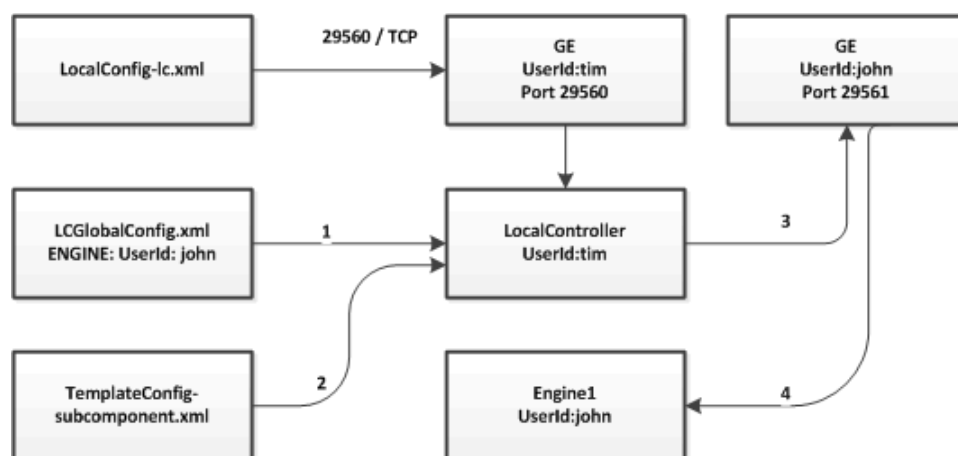
LocalController をこのサーバで実行する必要がある場合、LocalController 用の LocalConfig-lc.xml 設定ファイルおよび Generic Executor 用の TCP ポートのみが必要です。

以下の図では、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。このファイルには、起動する必要があるサブコンポーネントおよびこれらのサブコンポーネントを実行するために使用するユーザ ID が含まれます。

Engine をこの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せられます。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ー ID を指定します。

## Generic Executor のスタートアップ シーケンス

`startall/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 LCGlobalConfig 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 ポートがサブコンポーネントを起動するために使用されます。

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/SCP で情報を取得する必要がある場合、この参照は役立ちます。

## 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/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">CMM-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">CMM-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 コンポーネントの起動に使用されます。



## MultiController の設定オプション

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の説明です。

### 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ーをトリガします。

### Logging

ログ記録オプションを指定できます。

#### LogLevel

出力ログ ファイルで提供されるロギング 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 ファイルで提供されるロギング 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 クラスに必要なロギング 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/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。しかしながら、手動で LC/LocalConfig-lc.xml ファイルを編集できます。

### 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">CMM-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">CMM-Local-Controller</Name>
    <Name name="appShortName">LC</Name>
  </Names>

```

```
<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 クラスに解析される引数を指定します。

### 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 ファイルで提供されるロギング 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 ファイルで提供されるロギング 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 クラスに必要なロギング 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。

## フェールオーバー操作

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

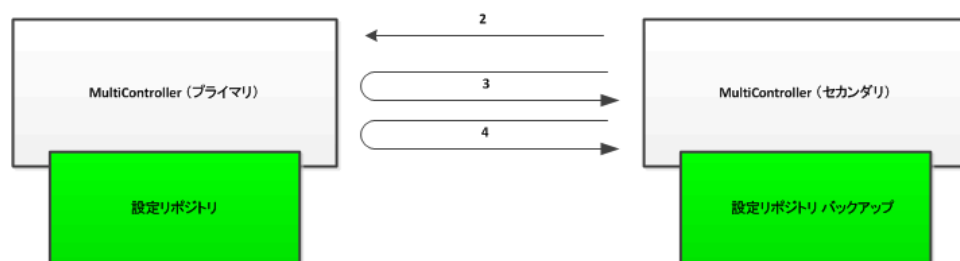
## MultiController の障害

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。

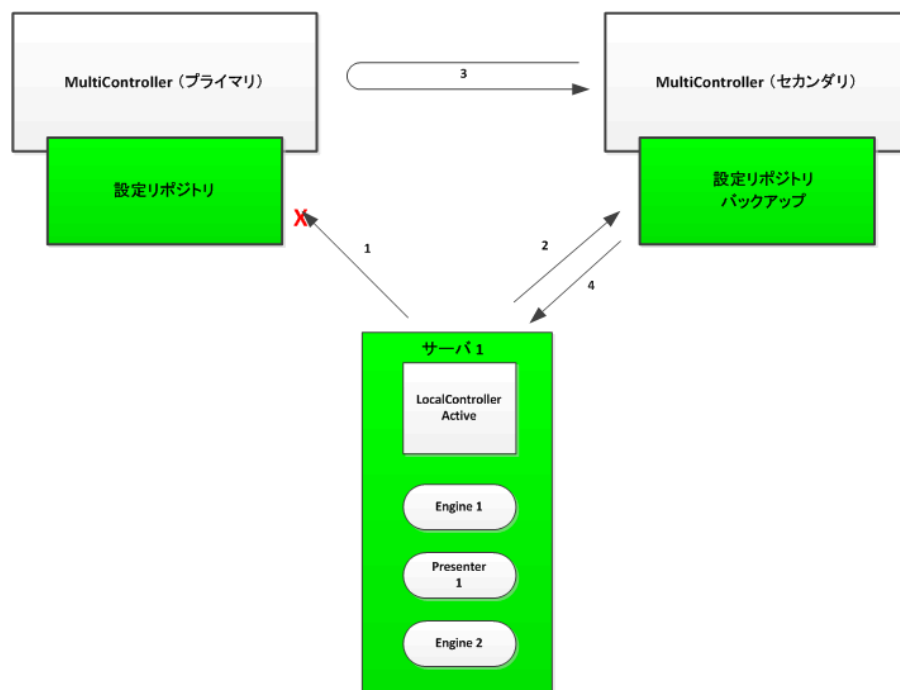
セカンダリが使用不可能になった場合に気づく Watchdog がプライマリにあります。セカンダリが使用不可能になると、アラートがログファイルに作成されます。



1. MultiController セカンダリが開始します。
2. MultiController セカンダリは、最初のハートビートをプライマリに送信し、アライブとして登録します。
3. MultiController セカンダリは、ハートビートをプライマリに定期的に送信します。
4. MultiController セカンダリは、リポジトリを定期的に同期します。

## 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、ハートビートをプライマリに送信し、失敗しました。
2. LocalController は、ハートビートをセカンダリ MultiController に送信します。
3. セカンダリ MultiController では、プライマリがまだ使用可能であると認識されているので、プライマリ MultiController にリクエストをプロキシします。
4. プライマリは、セカンダリ MultiController を通じて LocalController に対してハートビートの応答を返します。

## MultiController プロキシ機能

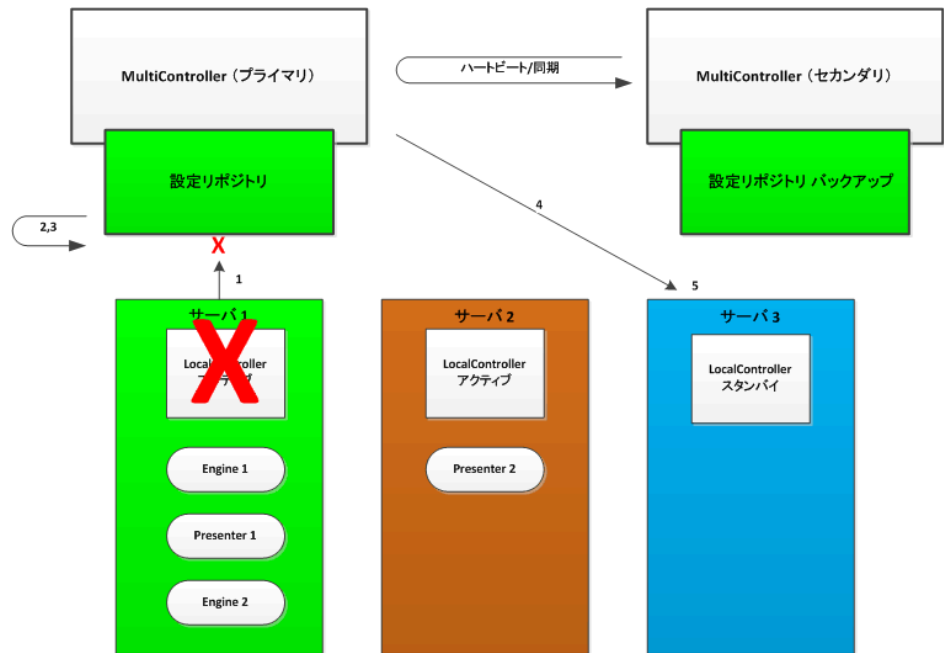
あるネットワーク状況の下では、LocalController は直接プライマリ MultiController と通信できません。このため、セカンダリ MultiController には、プロキシ機能があり、ハートビートまたはネーム サービス リクエストのプライマリ MultiController への送信を試みます。

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ーをトリガします。

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



1. MultiController で、タイムアウト期間を超えてハートビートを受信していないことが認識されました。
2. MultiController は、すべての Engine サブコンポーネントに対する設定リポジトリをサーバ 1 からサーバ 3 に移動します。
3. サーバ 1 にインストールされた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します。
4. MultiController は、サーバ 3 の LocalController にメッセージを送信し、その設定を再起動します。
5. サーバ 3 上の LocalController がアクティブになります。

## 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-Generi  
c-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lement 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/PRESENTER**) :  
TemplateConfig-subcomponent.xml

注: TemplateConfig-subcomponent.xml ファイルは <cammm.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>CMM-.*%.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 2.2.2)

---

注: この章で示される情報は CA Mediation Manager for Infrastructure Management 2.0 のみに適用されます。

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](#) (P. 59)

[イベントルールの追加](#) (P. 64)

### 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）。

以下の考慮事項を覚えておきます。

- [EMS 統合プロファイル] オプションは、デバイス パックのインストール後、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 2 回以上同じ EMS 統合プロファイルを追加することは推奨されません。

## EMS 統合プロフィールの追加

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 統合プロフィール**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 統合プロフィール**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）にそのプロフィールが配置されます。グローバル領域のプロフィー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 統合プロフィール**を特定のテナントのみにアクセス可能にするには、そのプロフィールを作成する *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、手動でテナントを **CA Infrastructure Management** と同期するか、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 統合プロフィール**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 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** ユーザ インターフェイスで、[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 統合プロフィール**] をクリックします。

[**EMS 統合プロフィール**] 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 統合プロフィールが作成されます。

[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。
- 手動で EMS 統合プロフィールを開始した場合。

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 統合プロフィー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するために使用されます。ディスカバリを開始するために手動で EMS 統合プロフィールを開始できます。

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 ユーザ インターフェイスで、[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 統合プロフィール] をクリックします。

[EMS 統合プロフィール] 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 統合プロフィール**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 統合プロフィール**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。

ディスクバリが開始されます。選択したディスクバリ 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 統合プロフィール] ページに戻ります。

ディスクバリが 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、ディスクバリは、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クバリ プロファイルの状態が 10 分以内に更新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。ディスクバリ 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、選択したディスクバリ 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** 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** 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 ユーザ インターフェイスで、[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 統合プロフィール] をクリックします。

[EMS 統合プロフィール] 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 統合プロフィール インスタンスを選択し、[履歴] ボタンをクリックします。

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イスと各インターフェイ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 統合プロフィー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 ユーザ インターフェイスで、[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。