Oracle Endeca Commerce

管理者ガイド

バージョン**3.1.1 · 2012**年**12**月

ORACLE

ENDECA

目次

Copyright and disclaimer

Oracle Endeca Commerceソリューションにより、企業は、パーソナライズされた一貫性のある顧客購買時体験を、すべてのチャネル (オンライン、インストア、モバイルまたはソーシャル) にわたって提供できます。Oracle Endeca Commerceソリューションは、顧客が自社と取引する時間や場所に関係なく、適切な顧客に対する適切なコンテンツを提供、分析およびターゲティングし、顧客のクリックを誘導してビジネス成果をもたらします。

Oracle Endeca Commerceは、顧客が店頭を動的に閲覧し、希望に沿って適切なアイテムをすばやく見つけるための最も有効な手段です。業界をリードするファセット検索とGuided Navigationのソリューションであるを使用すると、企業は、顧客の検索エクスペリエンスの各ステップで各顧客を容易に誘導できます。Oracle Endeca Commerceの中核であるMDEX Engine(TM)は、高いパフォーマンスの閲覧や検出のために特に設計されたハイブリッドな検索-分析データベースです。Endeca Content Acquisition System (CAS) は、構造化データと非構造化コンテンツの両方を多様なソース・システムからMDEX Engineに取得するための、一連の拡張可能なメカニズムを備えています。Endecaアセンブラは任意のリソースからコンテンツを動的に収集し、MDEX Engineからの結果とシームレスに結合します。

Oracle Endeca Experience Managerは単一の柔軟なソリューションで、ユーザーはコンテンツリッチなクロスチャネルの顧客エクスペリエンスを創出、提供および管理できます。また、技術者ではないビジネス・ユーザーは、ターゲット指定されたユーザー中心のオンライン・エクスペリエンスを拡張可能な方法で提供し、常に顧客と適切な関係性を構築することでコンバージョン率を高め、クロスチャネル販売を促進できます。ビジネス・ユーザーは、検索、カテゴリ選択またはファセット絞込みに応答してコンテンツを提示する方法、場所、時期および提示するコンテンツのタイプを制御できます。

これらのコンポーネントは、SEO、ソーシャルおよびモバイル・チャネル・サポート用の追加モジュールとともに、顧客エクスペリエンス管理プラットフォームであるOracle Endeca Experience Managerの中核を構成しています。Oracle Endeca Experience Managerは、すべての顧客のタッチ・ポイント全体にわたって、それぞれの顧客の各ステップに対して最も適切で、ターゲティングされ、最適化されたエクスペリエンスを提供することに重点を置いています。

このガイドについて

このガイドでは、Oracle Endeca Commerceで構築されたアプリケーションを管理および保守する際に必要なタスクについて説明します。Endeca実装の初期デプロイ時の実施作業と、システム管理者がシステムを保守する際に取り組む必要がある問題とのギャップを解消します。

このガイドは、Endecaソフトウェアが開発サーバーにすでにインストールされていることを前提としています。本番環境にインストールされていてもかまいません。また、システム管理者またはOracleサービス担当者がDeployment Templateを使用して、開発サーバーにアプリケーションを構成済であることも前提としています。

このガイドの対象読者

このガイドは、Endeca実装を保守するシステム管理者を対象としています。

*『Oracle Endeca Commerceコンセプト・ガイド』*に記載されているEndecaの概念もよく理解している必要があります。

このガイドで使用する規則

このガイドでは次の表記規則を使用します。

コード例、コード要素に対するインライン参照、ファイル名およびユーザーが入力するテキストは、固定幅フォントで設定されています。コードの行が長い場合や、インラインの固定幅テキストが行末にある場合は、内容が次の行に続くことを示すために、記号「¬」が使用されます。

このようなコード例をコピーして貼り付ける場合は、この記号とそれに対応する改行が削除され、余分なスペースが残っていないことを確認してください。

Oracleサポートへの連絡

Oracleサポートでは、Oracle Endecaソフトウェア、実装に関する質問、製品とソリューションのヘルプ、および全般的なニュースや更新内容に関する重要な情報を登録ユーザーに対して提供しています。

Oracleサポートには、Oracleサポート・ポータルのMy Oracle Support (*https://support.oracle.com*)を通して連絡できます。

第 1 部

環境間での実装の管理

第 1 章

概要

この項では、Endeca実装の操作および保守を制御する段階について説明します。システムの基本的な管理コンポーネントであるEndeca Application Controller (EAC) およびDeployment Templateも紹介します。

Endeca実装の所有権の取得

システム管理者は、開発ライフサイクルの特定の段階でEndeca実装の所有権を取得します。このトピックでは、Endeca実装の安定した操作を維持するために管理タスクを実行する状況について説明します。

このガイドでは、システム管理者またはそのチームによって、この時点で次のことが実行されていることを前提としています。

- ステージング環境および本番環境に必要なハードウェアが計画され、プロビジョニングされていること。
- MDEX Engine、プラットフォーム・サービス、ツール、フレームワークなどのEndecaコンポーネントがインストールされていること。
- *『Oracle Endeca Commerceスタート・ガイド』*を通読していること。
- Endecaソリューションをステージング環境にデプロイし、本番環境へのデプロイが準備済または本番環境にすでにデプロイ済であること。

Oracle Endeca Application Controllerについて

Oracle Endeca Application Controller (EAC) は、Endeca実装内のコンポーネントを制御、管理および監視するのに使用できる管理システムです。

EACにより、Endecaプロジェクトの設計からデプロイメント、ランタイムまでをサポートするインフラストラクチャが提供されます。これは非推奨の管理インタプリタに替わるものですが、Endecaツール (Developer StudioおよびOracle Endeca Workbench) はほとんど変更されていません。

EACではWebサービス記述言語 (WSDL) などのオープン・スタンダードを使用しているため、Application Controllerはプラットフォームや言語に依存しません。その結果、Application Controller

は本番の多様なアプリケーションをサポートできます。これにより、部分更新、デルタ更新、段階的なMDEX Engine更新などの機能をサポートする複雑な操作環境に対応できます。

EACアーキテクチャ

EACは、Endecaソフトウェアを実行する各マシンにインストールされ、分散環境で実行するのが一般的です。

Endeca実装におけるEACの役割に応じて、EACの各インスタンスは次のいずれかの役割を実行します。

- EAC Centralサーバー
- EACエージェント

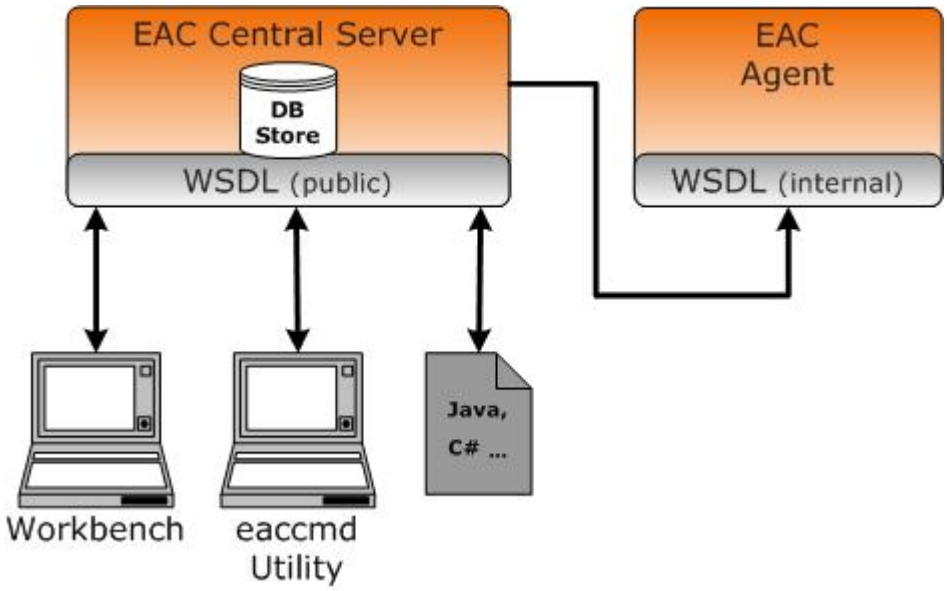
次の3つの方法のいずれかを使用して、EACと通信し、インスタンス構成およびリソース構成情報をEAC Centralサーバーに提供できます。

- Endeca Workbench。Endeca Workbenchは、WSDLインタフェースを介してEAC Centralサーバーと通信します。Endeca Workbenchを使用して、アプリケーションをプロビジョニング、実行および監視できます。詳細は、*Oracle Endeca Workbench Administrator's Guide*を参照してください。
- コマンドライン・ユーティリティ`eaccmd`。`eaccmd`を使用して、Perl、シェル、バッチなどの言語でEACのスクリプトを作成できます。
- WebサービスをサポートするEndeca WSDL対応のインタフェースおよび言語 (Javaなど) を介した、プログラムによる直接管理。

> 注: Endeca Deployment Templateでは、この方法を使用してEAC Centralサーバーと通信します。

これらの方法のいずれかを使用して、EACに対してEndeca実装内で様々な操作を実行するように指示できます。たとえば、コンポーネント (Forge、Dgraphなど) やユーティリティ (コピー、シェル環境など) の起動/停止などです。

次の図は、EACアーキテクチャおよびその通信方法を示します。後続の各項では、EAC CentralサーバーおよびEACエージェントの役割を説明します。



EAC Centralサーバー

EACのインスタンスの1つは、実装のEAC Centralサーバーとして機能します。このインスタンスにはWSDL対応のインタフェースが含まれ、このインタフェースを介してEACと通信します。通信は、標準のWebサービス・プロトコルであるSOAPを使用して実装されます。

EAC Centralサーバーには、プロビジョニング情報 (つまり、EACが管理するホスト、コンポーネント、アプリケーションおよびスクリプトに関するデータ) を格納するリポジトリも含まれます。

注: 特定のアプリケーションに対して構成できるEAC Centralサーバーは1つのみです。同じEACエージェントで複数の中央サーバーが同じアプリケーションに対してプロビジョニングされると、EACに問題が発生する可能性があります (たとえば、複数の中央サーバーからエージェントに送信された複数のクリーン・アップ指示が混同されて、スクリプトが中断する可能性があります)。

EACエージェント

EACのその他のインスタンスはすべてエージェントとして機能します。エージェントは、ホスト・マシンに対して、Endeca実装の実際の作業を実行するように指示します。たとえば、Forgeコンポーネントを使用してデータを処理したり、複数のMDEX Engineの作業を1つのMDEX Engineコンポーネントに集約するように調整します。

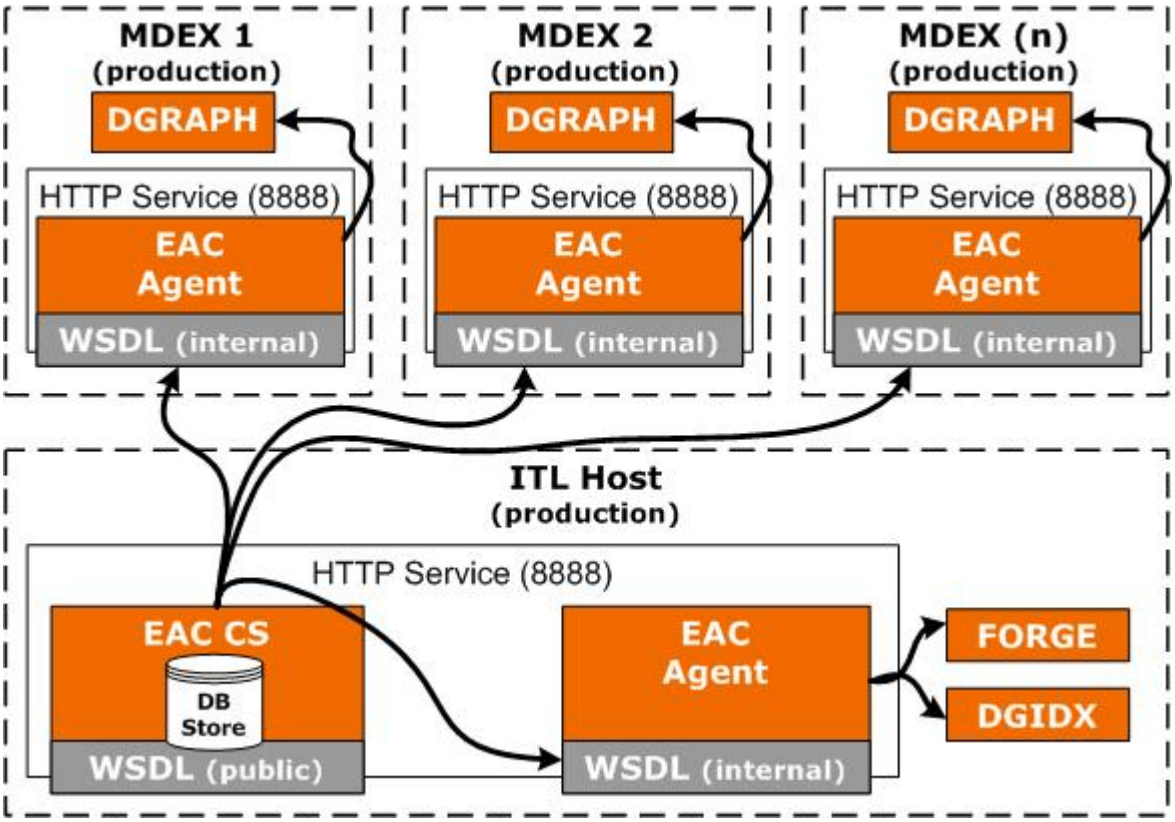
各エージェントには、そのエージェントのみで使用する小規模のリポジトリも含まれます。EAC Centralサーバーは、内部のWebサービス・インタフェースを介してエージェントと通信します。エージェントと直接通信することはできません。すべてのコマンド、管理および監視機能は、EAC Centralサーバーを介して送信されます。

EACアーキテクチャの例

一般的なEndeca実装は、複数のホスト・サーバーにわたり広がっています。これらの物理サーバーにはそれぞれ、サーバーにインストールされたコンポーネントを管理する1つのEACエージェントが含まれる必要があります。

次の図に、EACのアーキテクチャを示します。

EAC Centralサーバーは、実装全体 (または実装を構成するコンポーネント) をホストする各マシン上で稼働しているEACエージェントと通信します。EACサーバーは、インスタンス構成およびリソース構成に関する情報をエージェントと通信します。エージェントは、各マシンで必要なコンポーネントとそのプロセス (Forge、Dgidx、Dgraphなど) を実行します。



Deployment Templateについて

Deployment Templateによって、開発およびアプリケーションのデプロイメントの開始ポイントとして機能する一連の操作コンポーネントが提供されます。

テンプレートには、Endeca Application Controller (EAC) スクリプト、構成ファイル、共通スクリプト機能を備えたバッチ・ファイルまたはシェル・スクリプトなど、デプロイメントに必要な完全なディレクトリ構造が含まれています。

Deployment Templateを使用して、アプリケーションのデプロイメント環境を作成することをお薦めします。

AppConfig.xmlファイルについて

[appdir]/config/script/AppConfig.xmlファイルは、Deployment Templateの中心的な構成ファイルです。EACに認識されたEndeca実装を構成するホストおよびEndecaコンポーネントを定義します。定義したコンポーネントを実行することで、更新を編成するスクリプトも定義します。

ステージング環境または開発環境にすべてのコンポーネントが必要でない場合、または各コンポーネントが複数必要な場合は、AppConfig.xmlファイルを変更してコンポーネントを追加または削除できます。

注: このガイドでは、AppConfig.xmlで実行できる最も頻繁に使用されるタスクについて説明します。各コンポーネントをAppConfig.xmlに定義する方法の詳細は、Endecaの*Tools and Frameworks Deployment Template Usage Guide*を参照してください。

一般的なワークフロー

この項では、Endeca実装内で発生する基本的なワークフローを定義します。

管理フレームワーク

このガイドでは、Oracle Endeca Application Controller (EAC) とともに、主要管理フレームワークとしてDeployment Templateを使用することを前提にしています。

前述したように、一般的なEndecaの実装には、異なる物理サーバーで実行可能な様々なコンポーネント (CAS、Workbench、Forge、Dgidx、Dgraphなど) が含まれます。各サーバーは、EACエージェントやEAC Centralサーバーのホストにもなります。つまり、EACは、Endeca実装の各物理サーバー上でこれらのコンポーネントを管理し、コンポーネント間の通信を調整する管理システムです。

Deployment Templateにより、Oracle Endeca Application Controller (EAC) を使用してコンポーネントを起動および実行でき、ベースライン更新や部分更新も実行できます。

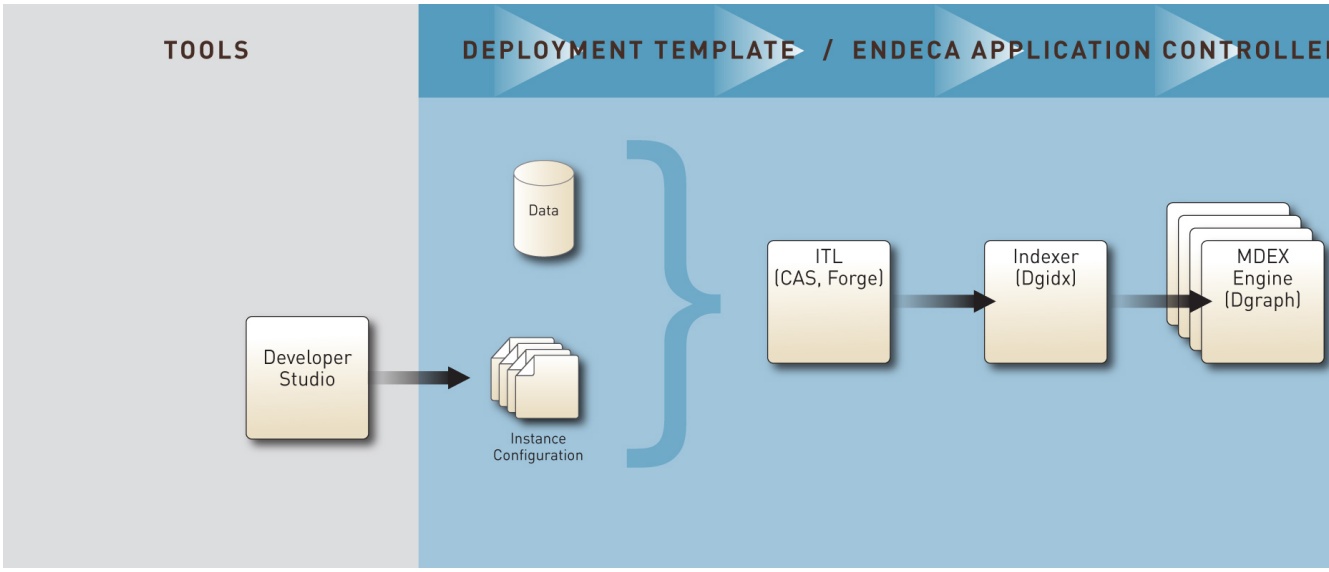
EACはEndeca実装の必須コンポーネントですが、Deployment Templateはオプションで、EACとの通信に使用されます。他の方法を使用してEACと通信することは可能ですが、Deployment Templateは便利なフレームワークとして機能し、タスクを簡略化する一連のスクリプトを提供できるため、Deployment Templateを使用して実装を管理することをお薦めします。

データおよび構成のワークフロー

このトピックにある図は、データおよび構成の基本ワークフローを示しています。このワークフローは、Deployment Templateの実行中に実行するタスクを表します。

基本ワークフロー

次の図は、Deployment Templateの実行中に発生する基本ワークフローを示しています。

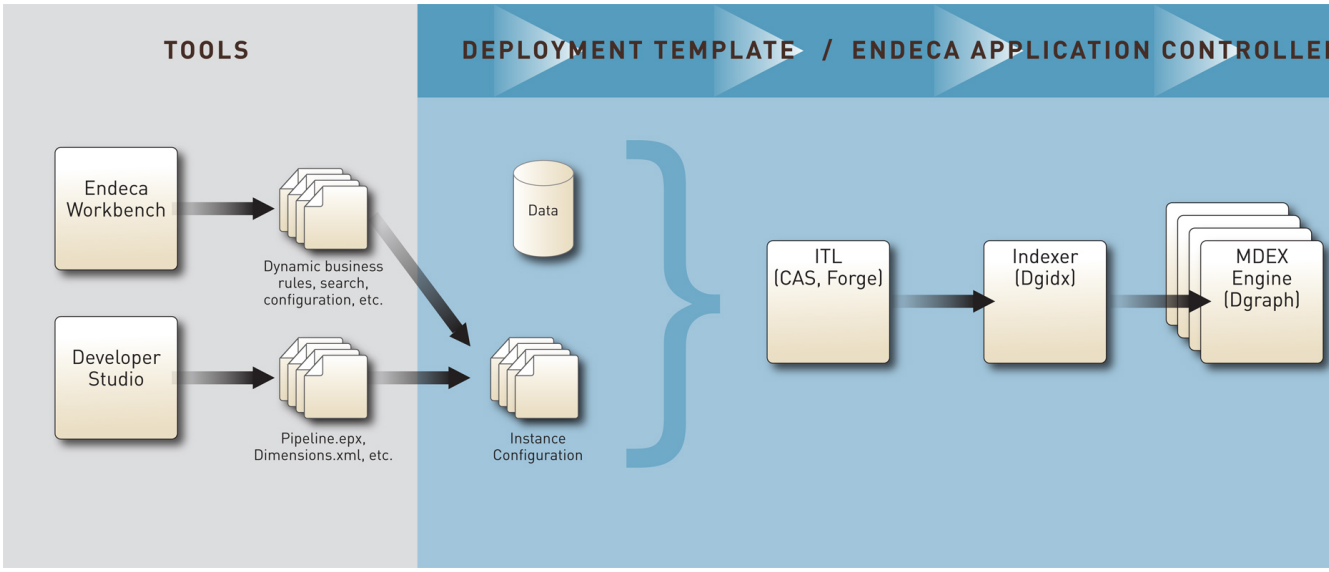


Deployment Templateによって、ディレクトリ構造が作成され、サンプルのwineアプリケーション・データが適切なディレクトリに追加されます。 (独自のデータを使用する場合は、Developer Studioで作成したパイプラインによってインスタンス構成ファイルが提供されます。)

必要に応じて、Content Acquisition System (CAS) を使用してソース・データを取得し、パイプラインでさらに処理できるようにします。次に、Deployment Templateによってデータがロードされ、アプリケーションがOracle Endeca Application Controller (EAC) に対してプロビジョニングされて、ベースライン更新スクリプトが実行されます。これにより、データがMDEX Engineによって索引付けされ、ユーザー問合せを処理できるようになります。

Oracle Endeca Workbenchを含む基本ワークフロー

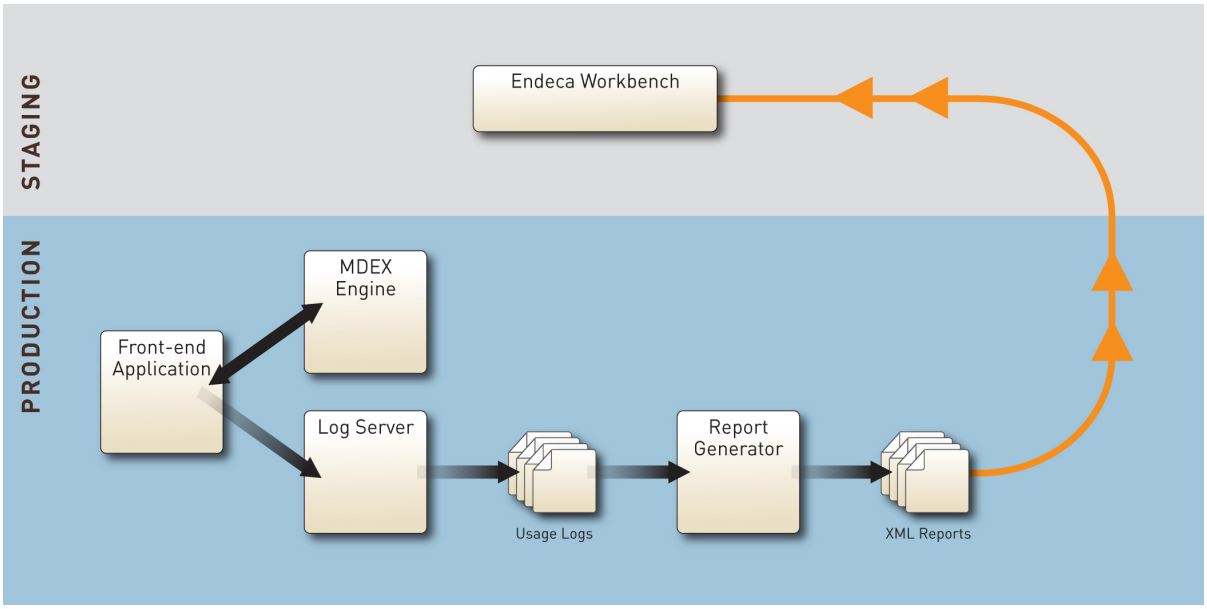
次の図は、デプロイメントにEndeca Workbenchが含まれる場合に、Deployment Templateの実行中に発生する基本ワークフローを示しています。



この図では、環境にOracle Endeca Workbenchが含まれるため、ビジネス・ユーザーはビジネス・ルール、検索構成、シソーラスおよびその他のオプションを変更できます。

ロギングおよびレポートのワークフロー

このトピックにある図は、ロギングおよびレポートのワークフローを示しています。



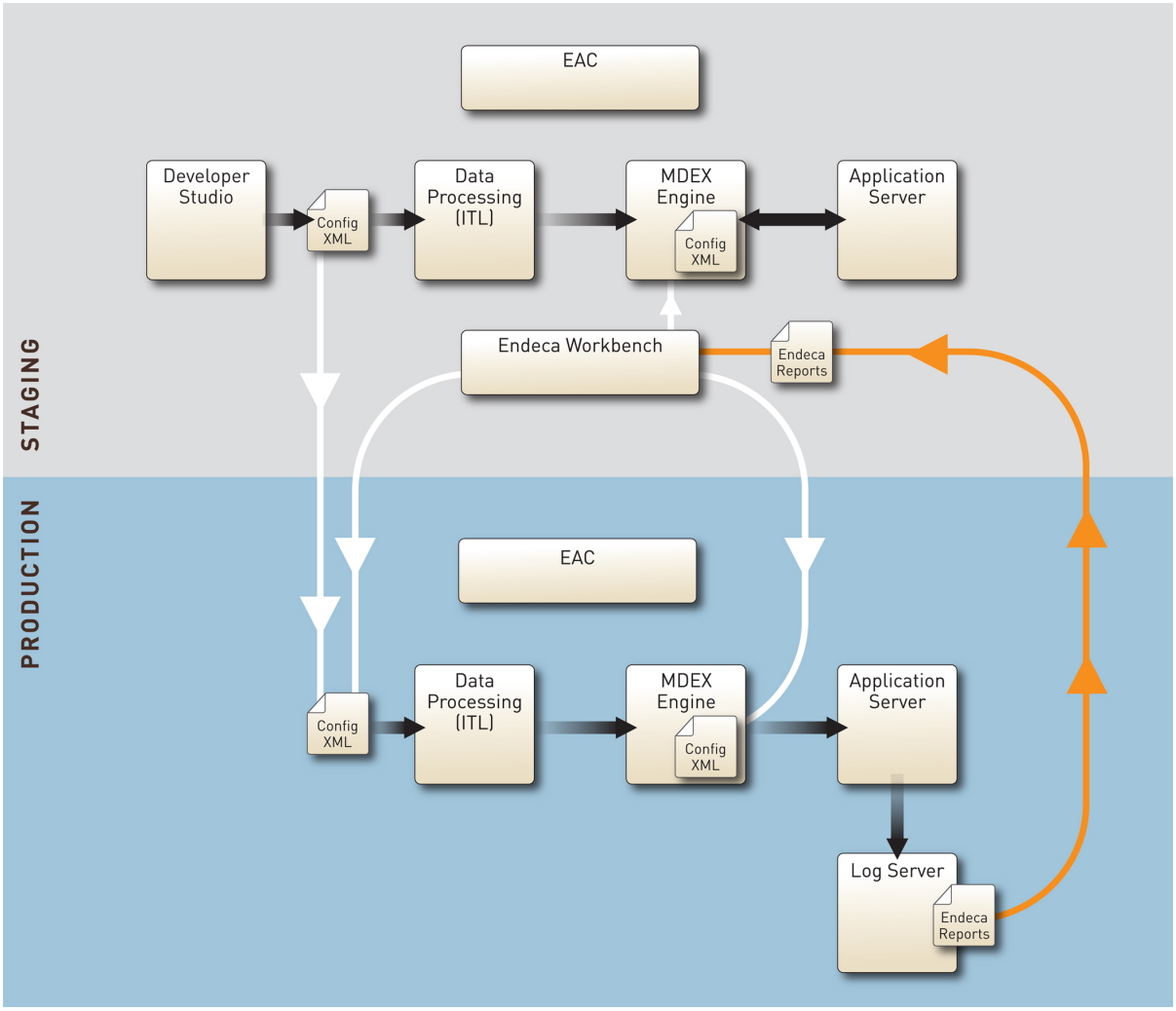
次に、この図の内容を説明します。

- フロントエンド・アプリケーションでは、Endeca Logging APIを使用してログ・サーバーと通信します。(フロントエンド・アプリケーションでは、Endeca Presentation API、Webサービスと XQuery、またはRAD Toolkit for ASP.NETを使用して、MDEX Engineと通信します。)
- レポート・ジェネレータ用の使用ログが生成され、レポート・ジェネレータでOracle Endeca Workbench用のXMLレポートが作成されます。ビジネス・ユーザーは、Oracle Endeca Workbench を使用してレポートを分析し、その分析に従ってビジネス・ルールの検索構成設定を調整できます。

詳細は、*Endeca Platform Services Log Server and Report Generator Guide*を参照してください。

本番レポートに基づくアプリケーションの調整

このトピックにある図は、ステージング環境と本番環境の間で発生する、ロギングおよびレポートのワークフローを示しています。このプロセスでは、本番環境からレポートを取得し、ステージング環境で構成を変更して本番環境に戻すことにより、Endecaアプリケーションを調整できます。



この図では、次のアクションが発生します。

1. プロジェクトが作成され、ステージング環境で実行されます。
2. データおよび構成情報がステージングから本番に送信されます。これは、ステージング環境から本番環境への矢印で表されています。
3. プロジェクトが作成され、本番環境で実行されます。
4. 問合せからのデータがロギング・サーバーに渡され、ロギング・サーバーでEndecaレポートが生成されます。
5. EndecaレポートはOracle Endeca Workbenchで使用されます。この情報に基づいて、ビジネス・ユーザーはOracle Endeca Workbenchを使用してプロジェクト構成を調整できます。

> 注: Oracle Endeca Workbenchで分析できるEndecaレポートに加えて、本番環境で実行中のアクティブなMDEX Engineからの問合せログを使用して、ステージング環境でのMDEX Engineのパフォーマンスを微調整し、MDEX Engineの本番サーバーにその変更を反映できます。このサイクルを繰り返して、パフォーマンスを最適化できます。

第 2 章

複数サーバー環境の作成

プロジェクト要件に基づいて、ステージングおよび本番環境の作成に進むことができます。ここでは、通常、専用サーバーの追加を行います。サーバーおよびMDEX Engineを追加するには、ステージング (または本番) 環境内のサーバーの正しいホスト名を指すように、Deployment Templateの`AppConfig.xml`ファイルを調整します。

複数サーバー環境について

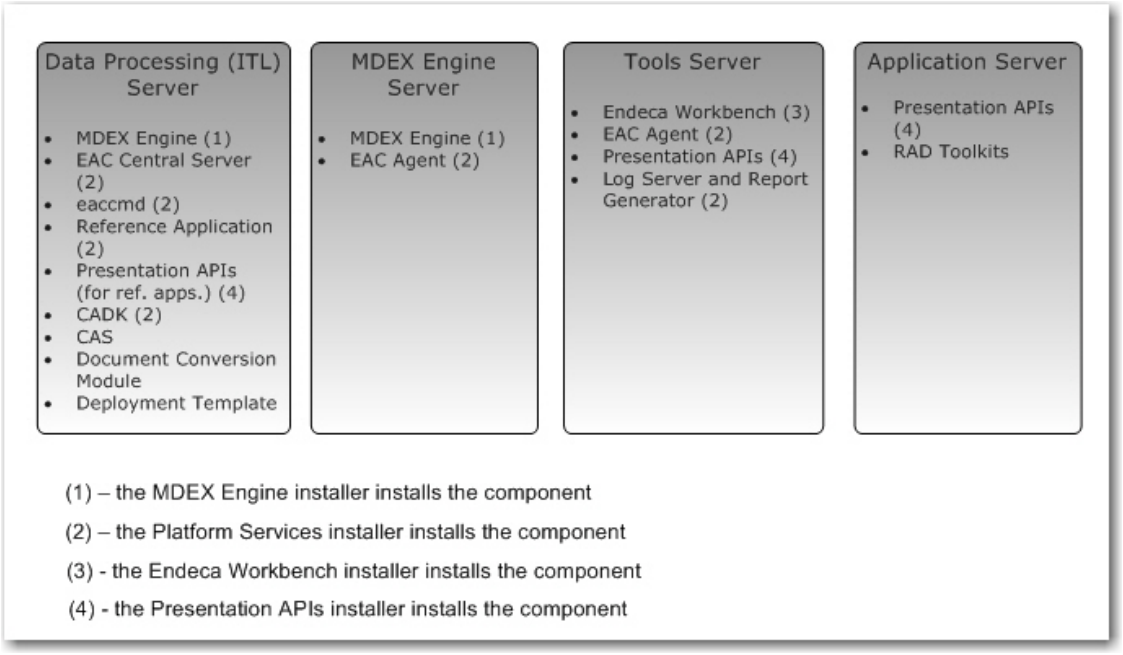
開発環境 (ステージング環境や本番環境も同様) は、複数のサーバーを使用して構築するのが一般的です。このトピックにある図は、複数サーバー環境にインストールする必要があるEndecaコンポーネントを示しています。

複数サーバー環境では、次をホストできます。

- 1台のサーバー上に、MDEX Engine、プラットフォーム・サービス・パッケージ (EAC Centralサーバーと EACエージェントを含む)、アプリケーションのデータ、およびDeployment Template。これは、*データ処理 (ITL) サーバー*です。
- 1台以上の追加サーバー上に、MDEX EngineおよびEACエージェント。これらは、*MDEX Engine*サーバーです。
- 別の1台のサーバー上に、Oracle Endeca WorkbenchおよびEACエージェント。これは、*ツール・サーバー*です。

また、専用のアプリケーション・サーバーを設定してフロントエンド・アプリケーションをホストでき、既存のサーバーの1台をこの目的で使用することもできます。開発環境でアプリケーション・サーバーを設定するかどうかは任意ですが、ステージング環境と本番環境では、1台以上の専用アプリケーション・サーバーを設定するのが一般的です。

次の図は、Endecaデプロイメント専用のサーバーにインストールする必要があるEndecaパッケージまたはその一部を示しています。



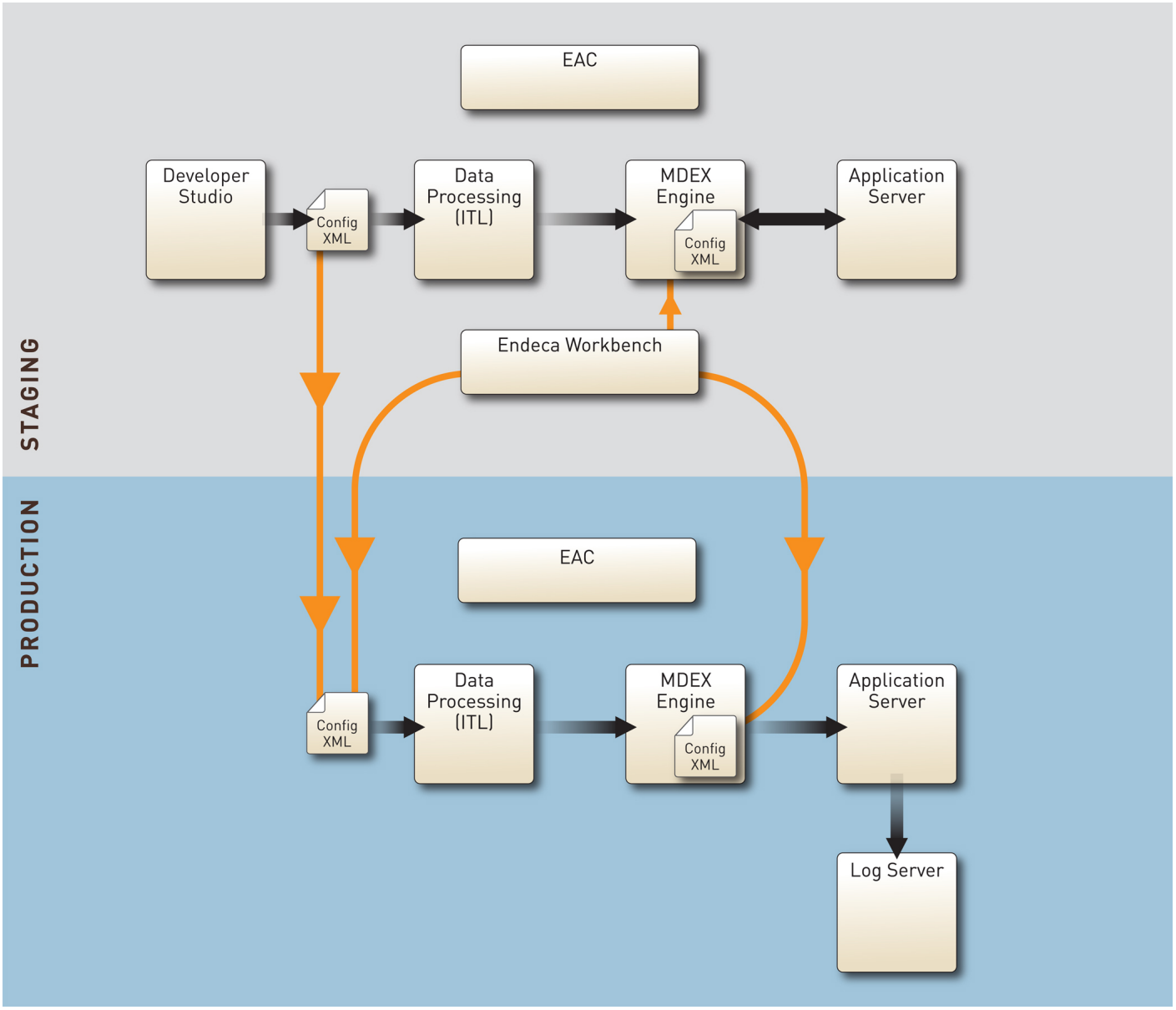
次に、この図の内容を説明します。

- 各サーバーはEndeca環境内で機能し、1台の物理マシンを表します。ただし、データ処理(ITL)、MDEX Engineおよびフロントエンド・アプリケーション用に複数のサーバーを構成できます。
- 一般的に、データ処理 (ITL) サーバーは、プラットフォーム・サービス・パッケージに含まれるEAC Centralサーバーをホストします。
- 一般的に、Windows上で稼働するデータ処理 (ITL) サーバーはDeveloper Studioもホストします。
- データ処理 (ITL) サーバーは、MDEX Engineインストール・パッケージに含まれるDgidxコンポーネントをホストする必要があります。
- MDEX Engineサーバーは、MDEX Engineインストール・パッケージに含まれるDgraphコンポーネントをホストします。
- MDEX Engineサーバーおよびツール・サーバーはそれぞれ、プラットフォーム・サービス・パッケージに含まれるEACエージェントをホストする必要があります。
- アプリケーション・サーバーでEACエージェントは必要ありません。

ステージング環境と本番環境の概要

ステージング環境と本番環境は、ハードウェア設定が同一または非常によく似ていますが、その機能は異なります。

この図では、ステージング環境と本番環境の相違点を示し、これらが相互にどのように関連しているかを説明します。



次に、この図の内容を説明します。

- 図の上部は、ステージング環境におけるデータおよび構成のワークフローを示しています。このワークフローでは、パイプラインを作成し、Deployment TemplateまたはOracle Endeca Workbenchのいずれかを使用してアプリケーションをEACに対してプロビジョニングおよび初期化し、ベースラインまたは部分更新スクリプトを実行します。次に、データが索引付けされ、MDEX Engineで処理されます。MDEX Engineでは、フロントエンド・アプリケーションから問合せを受け取り、結果をアプリケーション・サーバー上のフロントエンド・アプリケーションに返します。
- 図の下部は、本番環境におけるデータおよび構成のワークフローを示しています。このワークフローでは、本番サーバー上で更新を実行し、フロントエンド・アプリケーションがユーザー問合せをMDEX Engineサーバーに処理のために送信することもできます。この時点までに、構成ファイルおよび操作設定は本番環境で稼働するMDEX Engineにステージング環境から複製されているため、本番環境にDeveloper StudioとOracle Endeca Workbenchが含まれていないことに注意してください。

- 最も重要なのは、この図でステージング環境と本番環境をつなぐ矢印が、ステージング環境から本番環境へのデータの送信を表していることです。プロジェクトのデータおよび構成をステージング環境から本番環境に送信するには、通常、次のハイレベルの2ステップを実行します。

  1. パイプライン (Developer Studio)、および (使用している場合のみ) Oracle Endeca Workbenchから、本番環境のデータ処理サーバー上の(Forgeのために使用される)受信ディレクトリに構成ファイルをコピーします。このタスクを実行するには、Deployment Template内のスクリプトを実行します。
  2. 構成ファイル (通常はOracle Endeca Workbenchで作成される) を本番環境のMDEX Engineサーバーにプロモートします。これにより、MDEX Engineはプロジェクトの構成設定を使用できます。Deployment Templateの構成マネージャを使用すると、Oracle Endeca Workbenchで作成されたファイルの中で、本番環境のMDEX Engineサーバーにプロモートする必要があるファイルを指定できます。

サーバー・トポロジの計画

ステージング環境と本番環境の要件を評価し、サーバーの正しいホスト名を指すようにDeployment TemplateのAppConfig.xmlファイルを調整します。

このトピックでは、高レベルのステップについて説明します。Deployment Templateのカスタマイズの詳細は、*Tools and Frameworks Deployment Template Usage Guide*を参照してください。

サーバー・トポロジを構成する手順は、次のとおりです。

1. Endecaアプリケーションのパフォーマンスについての目標を分析します。データ・セットの予測サイズおよび他の特性を調査します。ハードウェア・ベンチマークおよびパフォーマンスの詳細は、*Endeca MDEX Engine Performance Tuning Guide*を参照してください。
2. 期待するパフォーマンスの目標に基づいて、ステージング環境と本番環境の要件を算出します。

   たとえば、ステージング環境と本番環境でプロビジョニングする必要があるサーバー数、および各MDEX Engineサーバーで実行する必要があるMDEX Engine数を見積もる必要があります。ステージング環境では、期待する本番環境の完全な複製をプロビジョニングすることが必要な場合や、本番サーバーのサブセットをプロビジョニングすることで十分な場合があります。

   👉 注: Workbenchサーバーでは、アセンブラ・クライアントごとに個別のスレッドが必要です。使用している1つ以上のアプリケーションに対して多数のアセンブラ・クライアント (50台以上) を構成する場合は、Workbenchサーバーに十分なメモリーがあることを確認する必要があります。この数では、推奨の4 GBを超えるメモリーが必要な可能性があります。

3. ステージング環境と本番環境の両方に対するサーバー・トポロジの構成に進みます。サーバーの正確なホスト名を指すように、Deployment TemplateのAppConfig.xmlファイルを調整します。

   通常は、各環境でDeployment Templateのdeployスクリプトを実行し、各環境を自己完結型プロジェクトとして構成します。

サーバーに対する調整を終了した後は、Deployment Templateを使用してアプリケーションを構成し、プロジェクトに対するベースライン更新スクリプトのカスタマイズを開始します。

関連リンク

サーバーへのMDEX Engineサーバーおよびの追加 25 ページ
Deployment Templateを使用して新しいアプリケーションを初めてデプロイすると、同じMDEX Engineサーバーに2つのDgraphが稼働する環境が設定されます。1つのDgraphはオーサリング・アプリケーション用で、もう1つのDgraphはライブ・アプリケーション用です。この構成は必要に応じて変更が可能で、オーサリング・アプリケーションまたはライブ・アプリケーションに1つ以上のMDEX Engineサーバーと1つ以上のDgraphを追加できます。

複数サーバーでのアプリケーションの構成

複数サーバーでアプリケーションを構成するには、 `[appDir]/config/script`に格納されているアプリケーション構成ファイルを変更して、使用している環境に各サーバーを指定します。

複数サーバー環境では通常、アプリケーション構成ファイルに次の内容を記述します。

- 1台以上のMDEX Engineサーバー
- データ処理 (ITL) サーバー
- ツール・サーバー

複数サーバーでアプリケーションを構成する手順は、次のとおりです。

1. アプリケーションはすでにデプロイされている必要があります。Oracle Endecaツールおよびフレームワークに付属しているDeployment Templateを使用します。手順については、*Tools and Frameworks Deployment Template Usage Guide*を参照してください。
2. 次のトピックに基づいて、 `[appDir]/config/script`にあるアプリケーション構成ファイルを変更します。

関連リンク

サーバーへのMDEX Engineサーバーおよびの追加 25 ページ
Deployment Templateを使用して新しいアプリケーションを初めてデプロイすると、同じMDEX Engineサーバーに2つのDgraphが稼働する環境が設定されます。1つのDgraphはオーサリング・アプリケーション用で、もう1つのDgraphはライブ・アプリケーション用です。この構成は必要に応じて変更が可能で、オーサリング・アプリケーションまたはライブ・アプリケーションに1つ以上のMDEX Engineサーバーと1つ以上のDgraphを追加できます。

追加のカスタマイズ・タスク 27 ページ
受信データの場所、および環境内のサーバーのトポロジを反映するようにDeployment Templateワークフローを調整した後は、プロジェクトのベースライン更新スクリプトの操作を開始できます。後で、Deployment Template内で更新スクリプトを実行できます。

サーバーへの**MDEX Engine**サーバーおよび**Dgraph**の追加

Deployment Templateを使用して新しいアプリケーションを初めてデプロイすると、同じMDEX Engineサーバーに2つのDgraphが稼働する環境が設定されます。1つのDgraphはオーサリング・アプリケーション用で、もう1つのDgraphはライブ・アプリケーション用です。この構成は必要に応じて変更が可能で、オーサリング・アプリケーションまたはライブ・アプリケーションに1つ以上のMDEX Engineサーバーと1つ以上のDgraphを追加できます。

このタスクでは、LiveDgraphCluster.xmlファイルで設定を変更して、ライブ・アプリケーションにMDEX Engineサーバーとそのサーバー上にDgraphを追加する方法について説明します。ただし、このタスクは、基本的にはオーサリング・アプリケーションにMDEX Engineサーバーを追加するタスクと同じです。オーサリング・アプリケーションでは、LiveDgraphCluster.xmlのかわりにAuthoringDgraphCluster.xmlを変更します。各構成ファイル内の設定の定義は、host、dgraphおよびdgraph-clusterの各オプションの定義と同じです。

サーバーにMDEX Engineサーバーおよびを追加する手順は、次のとおりです。

1. *[appDir]*/config/scriptに移動し、テキスト・エディタでLiveDgraphCluster.xmlを開きます。
2. このファイルのLive MDEX Hostsセクションを検索して、MDEX Engineサーバーを表す1つ以上のhost要素を追加します。新しいサーバーのホスト名とEACエージェント・ポート情報を指定します。
   この例では、LiveServerB.CompanyName.comというサーバーを追加しています。

```
  <!--
    ########################################################################
    # Live MDEX Hosts - The machines used to host all MDEX processes
    # for the 'live environment' MDEX cluster.
    #
  -->
  <host id="LiveMDEXHostA" hostName="LiveServerA.CompanyName.com"
port="8888" />
  <host id="LiveMDEXHostB" hostName="LiveServerB.CompanyName.com"
port="8888" />
```

3. Dgraph Clusterブロックに、新しいサーバー上で稼働するDgraphを追加します。
   この例では、新しいLiveServerB.CompanyName.comサーバーにDgraphB1とDgraphB2を追加しています。

```
  <!--
    ########################################################################
    # Live Dgraph Cluster - The 'live environment' MDEX cluster.
    #
  -->
  <dgraph-cluster id="LiveDgraphCluster" getDataInParallel="true" en¬
abled="true">
    <dgraph ref="DgraphA1" />
    <dgraph ref="DgraphA2" />
    <dgraph ref="DgraphB1" />
    <dgraph ref="DgraphB2" />
  </dgraph-cluster>
```

4. 新しいサーバー上に各DgraphのDgraphプロセス定義を追加します。
   この例では、新しいLiveServerB.CompanyName.comサーバーにDgraphB1とDgraphB2を追加しています。

```
...
  <dgraph id="DgraphB1" host-id="LiveMDEXHostB" port="15010"
          post-startup-script="LiveDgraphPostStartup">
    <properties>
      <property name="restartGroup" value="1" />
      <property name="DgraphContentGroup" value="Live" />
    </properties>
    <log-dir>./logs/dgraphs/DgraphB1</log-dir>
    <input-dir>./data/dgraphs/DgraphB1/dgraph_input</input-dir>
```

```
    <update-dir>./data/dgraphs/DgraphB1/dgraph_input/updates</update-dir>

  </dgraph>

  <dgraph id="DgraphB2" host-id="LiveMDEXHostB" port="15010"
          post-startup-script="LiveDgraphPostStartup">
    <properties>
      <property name="restartGroup" value="2" />
      <property name="DgraphContentGroup" value="Live" />
    </properties>
    <log-dir>./logs/dgraphs/DgraphB2</log-dir>
    <input-dir>./data/dgraphs/DgraphB2/dgraph_input</input-dir>
    <update-dir>./data/dgraphs/DgraphB2/dgraph_input/updates</update-dir>

  </dgraph>
...
```

5. 内容を保存して、`LiveDgraphCluster.xml`を閉じます。

追加のカスタマイズ・タスク

受信データの場所、および環境内のサーバーのトポロジを反映するようにDeployment Templateワークフローを調整した後は、プロジェクトのベースライン更新スクリプトの操作を開始できます。後で、Deployment Template内で更新スクリプトを実行できます。

ベースライン更新スクリプトをプロジェクトのデータに対して実行する方法は、サンプル・データに対して実行する方法と同じです。Deployment Templateのカスタマイズと機能の詳細は、*Tools and Frameworks Deployment Template Usage Guide*を参照してください。

第 3 章

環境間でのアプリケーション定義の複製

Endecaアプリケーションは通常、開発、テスト、使用および修正のライフ・サイクルを経験します。異なる環境間でアプリケーション定義を複製する機能により、このライフ・サイクルの管理が大幅に簡略化され、バックアップやリカバリも容易になります。

アプリケーション定義の複製について

環境間でアプリケーション定義を複製する機能によって、組織では体系化されたアプリケーション・ライフ・サイクルをサポートできます。このライフ・サイクルの中で、アプリケーションの開発、テスト、デプロイ、修正、再テストおよび再デプロイを最小限の作業で行うことができます。

また、アプリケーション定義を複製することによって、Endecaアプリケーションのバックアップやリカバリなどの管理プロセスが簡略化されます。

この項では、ターゲット環境の特性が大きく異なる場合でもアプリケーションを複製するための計画および手順について説明します。

Endecaの管理者や開発者は、次のようなハードウェア環境やソフトウェア環境で同じアプリケーション定義を使用する場合があります。

- 主要開発環境。アプリケーション構成およびファイルを作成して保守します。
- ステージング環境およびテスト環境。アプリケーションをテストして修正します。
- 本番環境。
- 二次開発環境。新規アプリケーションの基礎としてアプリケーション定義を再利用できます。
- バックアップまたは障害リカバリ環境。主要本番環境で問題が発生した場合に、アプリケーションを再デプロイして再起動できます。

これらの環境は、サーバーの数やタイプ、オペレーティング・システムおよびネットワーク・アーキテクチャの相違など、特性が大きく異なる場合があります。たとえば、開発環境では1台のUNIXワークステーションを使用し、テスト環境では1台のWindowsサーバー上で複数の仮想マシンを実行し、本番環境ではITLサーバー、MDEX Engine、複数のアプリケーション・サーバーおよびログ・サーバー用の様々なシステムに接続した専用サブネットを使用する場合があります。

多様な環境にわたって任意のタイプのアプリケーションを複製することは容易ではありません。しかし、計画を立て、いくつかのスクリプトを作成し、特定の手順に従うと、特性が異なる環境間でEndecaアプリケーション定義を複製するプロセスをほとんど自動化できます。

解決方法の1つは、Endeca Deployment Templateを使用して開発することです。前の*「複数サーバー環境の作成」*で説明したように、Deployment TemplateではEACを使用して、各環境内の複数のサーバー間でアプリケーション定義を管理します。各環境には複数のマシン (MDEX Engineサーバー、ITLサーバー、アプリケーション・サーバーなど) が含まれる場合がありますが、アプリケーション定義はその中の1つであるEAC Centralサーバーに格納されます。Deployment Templateでは、EAC Centralサーバーに格納されている定義を使用して他のサーバーを更新するアプリケーション管理スクリプトが生成されます。

ただし、環境間でアプリケーション定義を正常に複製するには、これ以外の手順も必要です。

次の図に、一般的な状況を示します。この例で、管理者は、開発用、アプリケーションのステージング、テストおよび調整用、および本番でのアプリケーション実行用の3つの環境を保守します。

管理者が3つの環境間でアプリケーションを同期化すると、開発環境での変更がステージング環境にすぐに移動し、ステージング環境での調整が本番環境に簡単にデプロイされて、その調整が開発環境にも反映されます (ここでは、開発中の新規バージョンのアプリケーションに反映できます)。

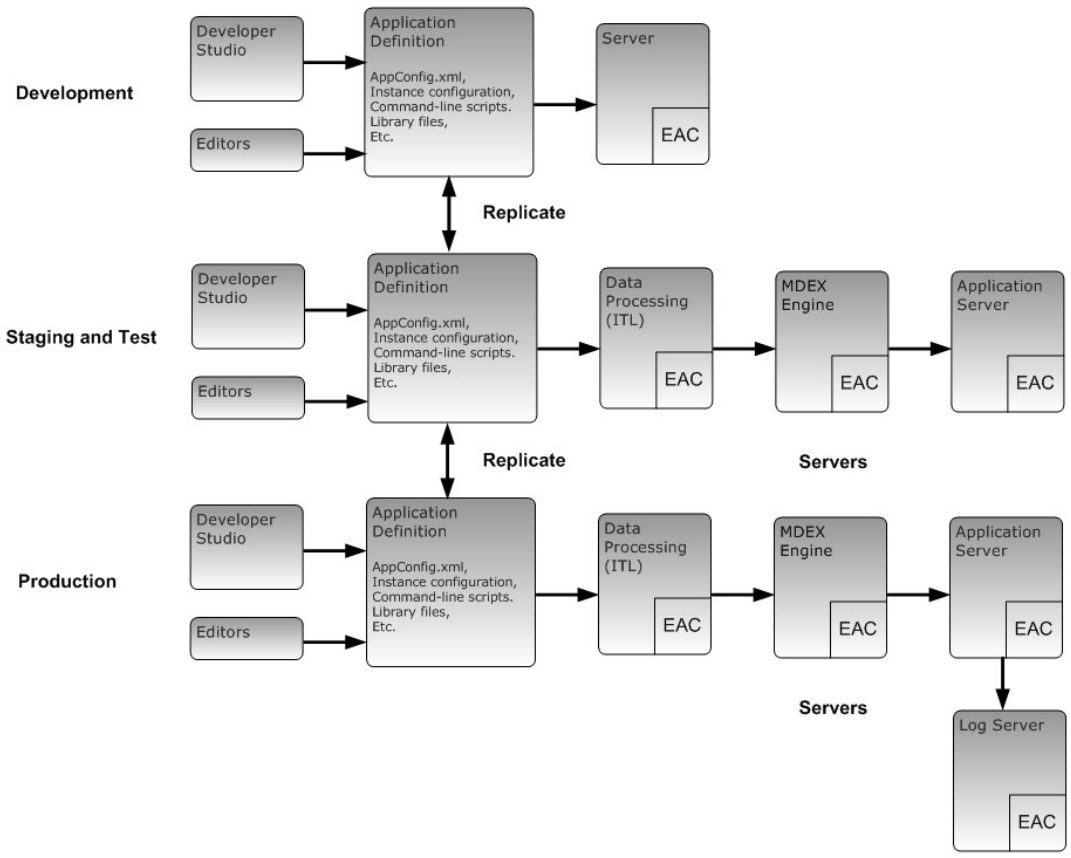
管理者がDeployment Templateを使用して開発している場合は、新規または修正済のアプリケーション定義が各環境のEAC Centralサーバーに移動し、EACによってアプリケーション定義の必要な部分がその環境内のサーバー間で伝播されます。

ただし、複製プロセスを可能なかぎり簡略化するために、管理者は次の作業も行う必要があります。

- すべての環境で修正なし (または、ほぼ修正なし) で使用できるアプリケーション定義の作成。
- アプリケーション定義要素の複数環境間における複製プロセスの自動化。

管理者が次の操作を行う場合も、同様の作業が必要です。

- アプリケーション定義を二次開発環境に複製し、新規アプリケーションの基礎として使用する場合。
- 安全な場所に格納できるアプリケーションのコピーを作成し、そのコピーをバックアップ環境または障害リストア・サイト (そのサイトの特性が元の環境と異なる場合でも) に迅速に複製する場合。



環境間でアプリケーションを複製するステップは次のとおりです。

- アプリケーション定義を構成するアーティファクトを認識します。
- 環境固有の設定用にカスタム・ファイルを作成します。
- 環境間の相互運用性を確保するためにパスを管理します。
- ファイル収集を自動化します。
- 環境間でアプリケーション定義を複製します。
- 同期の競合を回避する方法を選択します。

アプリケーションを構成するアーティファクトの識別

通常のEndecaアプリケーションを構成するアーティファクトには、次のものがあります。

AppConfig.xml構成ファイル

`AppConfig.xml`ファイルとその関連ファイルには、アプリケーションのプロビジョニング情報が記載されています。Deployment Templateでは、`AppConfig.xml`ファイルが`[appdir]/config/script`ディレクトリに作成されます。

アプリケーション構成はOracle Endeca Workbenchを使用して変更できますが、変更は`AppConfig.xml`ファイルでのみ行うことをお薦めします。これによって、アプリケーション構成

は、環境間で共有できるようにディスクに保存されます。Workbenchは、構成のレビュー、個々のコンポーネントの起動や停止など、構成の変更に関連しない他のタスクに使用できます。

`AppConfig.xml`ファイルの一部には、アプリケーションの名前、ファイル・システム・パス、環境内のホスト・アドレスなど、環境固有の設定が含まれています。これらの設定は、収集してカスタム・ファイルに保存する必要があります。このファイルの作成方法の詳細は、*環境固有の設定用のカスタム・ファイルの作成* 33 ページを参照してください。

インスタンス構成**XML**ファイル

インスタンス構成は、MDEX EngineにロードされたITLプロセスとデータを管理する一連のXMLファイルです。これらのファイルには、ディメンション定義、検索構成、Forgeパイプライン、Experience Managerランディング・ページなどの構成データが含まれています。Developer StudioおよびOracle Endeca Workbenchでは、ユーザーのツール使用時に、XMLファイルへの書込みによってインスタンス構成が変更されます。

Experience Managerテンプレート

Experience Managerテンプレートは、エクスペリエンス・マネージャで作成できる動的なランディング・ページを実行するために使用されます。テンプレートは、Workbenchによって保存されたXMLファイルで定義されます。

コマンドライン・スクリプト

通常、アプリケーション・デプロイメントには、アプリケーションに関する一般的な操作を実行するコマンドライン・スクリプトが含まれています。たとえば、スクリプトには、`initialize_services`、`baseline_update`、`partial_update`などがあります。Deployment Templateを使用して新しいアプリケーションをデプロイすると、複数のスクリプトが`[appdir]/control`ディレクトリに作成されます。

必要に応じて、追加のスクリプトを作成し、`[appdir]/control`ディレクトリに保存できます。これらのスクリプトは、その入力データおよび出力ディレクトリとともに、アプリケーション定義の一部分を構成します。たとえば、開発者は、Webページをクロールするためのスクリプトを、ベースライン更新の準備で作成できます。これらのスクリプトは、シードリスト・ファイルを入力として受け入れ、`[appdir]`下のカスタム・ディレクトリに出力ファイルを作成します。

これらのコマンドライン・スクリプトは、その入力データおよび出力ディレクトリとともに、開発、ステージングおよび本番の各環境間で共有される必要があります。

ライブラリ・ファイル

ライブラリ・ファイルは、アプリケーションの多くの部分で使用されます。たとえば、JavaまたはPerlマニピュレータを実行するForgeパイプラインでは通常、これらのマニピュレータを実装するために、ライブラリ・ファイルにアクセスする必要があります。BeanShellスクリプトでは、アプリケーション固有またはEndeca固有のJavaクラスを使用できます。ライブラリ・ファイルは、`[appdir]/config/lib`下に保持されます。

Forgeの状態ファイル

Forgeの状態ファイルは、`[appdir]/data/state`ディレクトリにあります。

多くの場合、これらのファイルは、アプリケーション定義の一部として含める*必要はありません*。ただし、自動生成ディメンションまたは外部ディメンションのディメンション値をアプリケーションで使用している場合、Forgeの状態ファイルは、複数の環境にわたって同期化されている*必要があります*。この状況での状態ファイルには、これらのディメンション値のIDが含まれており、Forgeの実行回数に関係なく、同じディメンション値が常に同じIDを取得します。これらのディメンション値は、動的ビジネス・ルール、ランディング・ページ、ディメンションの順序指定、優先順位ルールなど様々な方法で使用できます。

つまり、自動生成ディメンションまたは外部ディメンションがアプリケーションで使用され、これらのディメンションの値がアプリケーション構成のいずれかの場所 (たとえば、動的ビジネス・ルール、Experience Managerランディング・ページ、明示的なディメンションの順序指定、優先順位ルールなど) で参照されている場合、Forgeの状態ファイルはアプリケーション定義の一部として識別する必要があります。

環境固有の設定用のカスタム・ファイルの作成

ほとんどのアプリケーション構成設定はすべての環境間で共有できますが、環境固有の設定もあります。このような設定は`AppConfig.xml`ファイルから削除して、別のファイルに格納する必要があります。このため、各環境には、同期中に変更されない環境固有の設定用に独自のカスタム・ファイルがあります。

一般的に、環境固有の設定には、アプリケーションの名前、ファイル・システム・パス、環境内のホスト・アドレス、およびDgraphクラスタと呼ばれるMDEXプロセスの定義が含まれます。

`custom.xml`ファイルを作成する手順は、次のとおりです。

1. Deployment Templateで生成された`AppConfig.xml`ファイルを、次のような行を含むように編集します。
   ```
   <spr:import resource="custom.xml" />
   ```
2. 環境固有のすべての要素を`AppConfig.xml`から新しい`custom.xml`ファイルに移動します。対象の要素には、`<app>`、`<host>`、`<dgraph-cluster>`、`<dgraph>`および`<logserver>`要素が含まれます。
3. 環境ごとに`custom.xml`ファイルを作成し、その環境に該当する設定を含めます。

次に、独自のファイルを作成する際に一般的な参照として使用できるサンプルの`custom.xml`ファイルを示します。

```
- <!--
    ############################################################################

     # Global variables

     -->
- <app appName="wine" eacHost="ConfigMig1" eacPort="8888" dataPrefix="wine"
 sslEnabled="false" lockManager="LockManager">
  <working-dir>${ENDECA_PROJECT_DIR}</working-dir>
  <log-dir>./logs</log-dir>
  </app>
- <!--
    ############################################################################
```

```
     # Servers/hosts
     #
     # The "webstudio" host and its "webstudio-report-dir" directory use
    # predefined names to inform Workbench where it should look for reports

     # for this application.
     #
      -->
  <host id="ITLHost" hostName="ConfigMig1" port="8888" />
  <host id="MDEXHost" hostName="ConfigMig1" port="8888" />
- <host id="webstudio" hostName="ConfigMig1" port="8888">
- <directories>
  <directory name="webstudio-report-dir">./reports</directory>
  </directories>
  </host>
- <!--
    ###################################################################################

     # Dgraph Cluster
     #
      -->
- <dgraph-cluster id="DgraphCluster" getDataInParallel="true">
  <dgraph ref="Dgraph1" />
  </dgraph-cluster>
- <!--
    ###################################################################################

     # Dgraphs
     #
      -->
- <dgraph id="Dgraph1" host-id="MDEXHost" port="15000">
- <properties>
  <property name="restartGroup" value="A" />
  <property name="updateGroup" value="a" />
  </properties>
  <log-dir>./logs/dgraphs/Dgraph1</log-dir>
  <input-dir>./data/dgraphs/Dgraph1/dgraph_input</input-dir>
  <update-dir>./data/dgraphs/Dgraph1/dgraph_input/updates</update-dir>
  </dgraph>
- <!--

    ###################################################################################

     # LogServer
     #
      -->
- <logserver id="LogServer" host-id="ITLHost" port="15010">
- <properties>
  <property name="numLogBackups" value="10" />
  <property name="targetReportGenDir" value="./reports/input" />
  <property name="targetReportGenHostId" value="ITLHost" />
  </properties>
  <log-dir>./logs/logservers/LogServer</log-dir>
  <output-dir>./logs/logserver_output</output-dir>
  <startup-timeout>120</startup-timeout>
  <gzip>false</gzip>
```

```
  </logserver>
```

環境間の相互運用性を確保するためのパスの管理

開発、ステージングおよび本番の各環境に含まれるサーバーは、オペレーティング・システムやディレクトリ構造が異なる場合が多くあります。スラッシュ、相対パス名および変数を使用すると、環境間で伝播されたアプリケーション定義がターゲット環境で機能します。

スラッシュの使用

すべての場所でパスが機能するには、構成ファイルやスクリプトで可能なかぎりスラッシュ (/) を使用します。スラッシュを使用すると、WindowsおよびUNIX環境は正常に機能します。ただし、Windowsの.batファイルやUNIXのシェル・スクリプトなど、プラットフォーム固有のスクリプトは例外です。

パイプラインでの相対パスの使用

Forgeパイプラインで絶対パスを使用すると、アプリケーションは1つの環境内の特定の場所に固定されます。したがって、相対パスを使用することをお薦めします。

レコード・アダプタでは、パスは`[appdir]/data/processing`に対する相対パスです。ここには、パイプラインが実行される前の`[appdir]/data/incoming`のコンテンツが含まれます。したがって、`[appdir]/data/incoming`からのファイルは、ディレクトリを指定せずに名前で直接参照できます。

Javaマニピュレータ・パスは`[appdir]`に対する相対パスです。これは、マニピュレータのクラス・パス、およびJavaコードのアクセス対象になるすべてのファイルが該当します。これらのファイル名はパススルーとして指定される場合が多いため、このようなパススルー値は`[appdir]`に対して相対的にしてください。

スクリプトでの変数の使用

スクリプトで相対パスを指定できますが、スクリプトは常に同じディレクトリから起動されるわけではないことに注意してください。たとえば、`[appdir]/control`内に`crawl.sh`というスクリプトがあるとします。このスクリプトで`[appdir]/control`に対する相対パスを使用すると、スクリプトがこのディレクトリに格納されている間に起動された場合のみ、スクリプトは正常に機能します。スクリプトが`[appdir]`から`/control/crawl.sh`として起動されると、パスは不正になり、スクリプトは失敗する場合があります。

この問題を回避するには、特定の絶対パスを使用せずに、スクリプトで事前設定の変数を使用して既知のディレクトリを参照します。Windowsのバッチ・ファイルでは、変数`%~dp0`によってスクリプトを含むディレクトリが解決されます。UNIXのBashスクリプトでは、変数`$0`によってスクリプト名が解決され、`dirname`コマンドを実行するとスクリプトのディレクトリが示されます。`AppConfig.xml`のBeanShellスクリプトでは、変数`${ENDECA_PROJECT_DIR}`が`[appdir]`を指します。これらの変数を使用すると、場所に依存しないスクリプトに移植可能なパスを作成できます。

Deployment Templateで生成されるすべてのスクリプトはこのテクニックを使用し、サンプルとして使用できます。

UNIXの場合、通常は次の行でスクリプトが開始されます。

```
WORKING_DIR=`dirname ${0} 2>/dev/null`
. "${WORKING_DIR}/../config/script/set_environment.sh"
```

これにより、WORKING_DIR変数が実行対象のスクリプトを含むディレクトリに設定され、WORKING_DIRに対する相対パスが作成されて別のディレクトリ内にあるset_environment.shファイルを指します。スクリプトがどの場所で起動されても、set_environment.shへのパスは常に正しく解決されます。

Windowsの場合、通常は次の行でスクリプトが開始されます。

```
call %~dp0..\config\script\set_environment.bat
```

これにより、%~dp0に対する相対パスを使用してset_environment.batを指します。スクリプトがどの場所で起動されても、参照は正しくなります。

アプリケーションを構成するファイルの収集

アプリケーション構成を変更した後は、アプリケーション定義に関するファイルとアーティファクトをすべて収集し、他の環境に複製できるようにアプリケーションの[appdir]ディレクトリに格納します。このため、Deployment Templateの[appdir]ディレクトリ、Workbench、アプリケーションのスクリプトで使用する任意の場所など、様々な場所からファイルを収集する必要があります。

アプリケーションを構成するファイルを収集する手順は、次のとおりです。

1. 任意のエディタを使用して、アプリケーション定義のWorkbench部分をすべて指定します。
2. 同じファイルで、Deployment Templateの[appdir]ディレクトリに格納されている残りのアプリケーション定義に、Workbench部分を結合します。

スクリプトを実行すると、すべてのアーティファクトが収集され、[appdir]の下にあるサブディレクトリ (/config、/control、/test-data、data/state、data/incomingなど) に格納されます。

次に、サンプルの収集スクリプトcollect-app.batを示します。使用するEndeca環境にあわせて同様のスクリプトを記述できます。このサンプルは参照用として提供されています。

3

```
set app=wine
set wbench=localhost:8006
call %~dp0..\config\script\set_environment.bat
call %~dp0runcommand.bat ConfigManager updateWsConfig
xcopy /y %~dp0\..\data\complete_index_config %~dp0\..\config\pipeline
set templ=%~dp0\..\config\templates
mkdir %templ%
emgr_update --app_name %app% --host %wbench% --action get_templates --dir
%~dp0\..\config\templates
```

更新されたアプリケーション定義の関連部分を[appdir]ディレクトリからWorkbenchにアップロードするスクリプトを作成することもできます。次に、サンプル・スクリプトのupload.batを示します。

```
set app=new-wine
set wbench=localhost:8006
call %~dp0..\config\script\set_environment.bat
emgr_update --app_name %app% --host %wbench% --action update_mgr_settings
--dir %~dp0\..\config\pipeline --prefix %ENDECA_PROJECT_NAME%
emgr_update --app_name %app% --host %wbench% --action set_templates --dir
%~dp0\..\config\templates
```

注: collect_app.batおよびupload.batスクリプトを実行するときは注意が必要です。これらのスクリプトによって、Developer StudioおよびWorkbenchの設定が上書きされる可能性があります。

環境間でのアプリケーション定義の初回配布

アプリケーション定義に関するファイルとアーティファクトをすべて収集して中央の場所に格納した後、ターゲット環境のEAC Centralサーバーでそれらを[appdir]ディレクトリにコピーすることにより配布できます。アプリケーションを環境に初めてデプロイするときは、いくつかの追加ステップを実行する必要があります。たとえば、開発からステージングへ、またはステージングから本番へ、初めてアプリケーションを移動する場合です。

環境間でアプリケーション定義を初めて配布する手順は、次のとおりです。

注: これらのステップは、バージョン管理システム下で環境が稼働していることを前提にしています。

1. ソース環境のEAC Centralサーバーで、collect-app.batスクリプトを実行して、アプリケーション構成のすべてのアーティファクトを[appdir]ディレクトリに格納します。
2. [appdir]ディレクトリのコンテンツをバージョン管理システムにコミットします。
3. ターゲット環境で、ITLサーバーに[appdir]ディレクトリを作成します。
4. ソース環境の[appdir]ディレクトリのコンテンツを、ターゲット環境の[appdir]ディレクトリにコピーします。
5. 定義の環境固有部分を編集し、ターゲット環境に対して正しい値を設定します。
   詳細は、*環境間の相互運用性を確保するためのパスの管理* 35 ページを参照してください。
6. ターゲット環境のEAC Centralサーバーで、Deployment Templateのinitialize_servicesスクリプト ([appdir]/controlディレクトリ内) を実行します。
7. ターゲット環境のEAC Centralサーバーで、Workbenchのupload.batスクリプトを実行して、ターゲット環境のWorkbenchが更新されたことを確認します。

更新されたアプリケーション定義の他の環境への配布

更新されたアプリケーション定義を他の環境に配布する手順は、次のとおりです。

👉 注: これらのステップは、バージョン管理システム下で環境が稼働していることを前提にしています。

1. ソース環境のEAC Centralサーバーで、`collect-app.bat`スクリプトを実行して、アプリケーション構成のすべてのアーティファクトを`[appdir]`ディレクトリに格納します。
2. `[appdir]`の関連するサブディレクトリのコンテンツをバージョン管理システムにコミットします。
3. ターゲット環境のEAC Centralサーバーで、`[appdir]`ディレクトリを更新します。
4. ターゲット環境のEAC Centralサーバーで、`upload.bat`スクリプトを実行して、ターゲット環境のWorkbenchが更新されたことを確認します。

同期の競合を回避する方法

複数のユーザーがアプリケーション定義を同時に変更する場合があります。たとえば、複数のアプリケーション開発者がそれぞれの開発環境内で同じアプリケーションに対して作業を行ったり、ビジネス・ユーザーがステージング環境内でアプリケーション構成を変更する一方で、開発者が開発環境内で別の変更を行う場合があります。同期の競合を回避するには、次の方法を検討してください。

トークン・システム

トークン・システムを使用して、1つの環境内で1名のユーザーのみにアプリケーション構成の変更を許可することにより、同期の競合を防ぐことができます。「トークン」を保持するユーザーのみ変更を行うことができます。トークンは同期中に環境間で渡されます。アプリケーション定義が伝播されるときに、ターゲット環境はソース環境からトークンを取得します。トークンを取得する前に実行された変更は、警告なしで上書きされます。

トークン・システムは単純で、特定の時点で変更が許可されている環境を簡単に識別でき、トークンを保持している環境内でアプリケーションの任意の部分を変更できます。ただし、トークン・システムを使用すると、同時開発はできません。

Separation of concerns

Separation of concerns手法では、環境内で許可される変更の範囲が制限されます。環境ごとに、構成の異なる部分が対象となります。たとえば、ある環境ではExperience Managerのテンプレートが対象で、別の環境ではディメンション定義が対象になります。対象の領域への変更は、該当する環境でのみ行うことができます。したがって、ある環境で行われた変更によって、別の環境で行われた変更が上書きされることはありません。

Separation of concerns手法を使用すると同時開発が容易になりますが、Separation of concernsは事前に適切に定義して、一貫して適用する必要があります。

手動による解決

複数の環境で同時変更が制限されていないために発生した競合は、競合しているすべてのファイルを手動で比較し、各相違を個別に処理することによって解決できます。

ただし、手動による解決は時間がかかり、Endecaのコンポーネントや内部形式について幅広い知識が必要になるため、人為的なエラーが発生しやすく、広範な対応はできません。したがって、複数のEndeca環境を同期化する場合、手動による解決は避けてください。

第 2 部

システム操作の実行

第 4 章

EACを使用したシステム操作の実行

この項では、Endecaアプリケーションに関連する操作タスクについて説明します。Deployment Templateを使用してEACと相互作用するための推奨プラクティスに重点を置いていますが、一部のタスクの実行に関する代替方法にも触れています。また、EACログに対する指示など、EAC操作に関連する基本的な管理情報も記載されています。

アプリケーションをプロビジョニングするためのオプション

*プロビジョニング*とは、EACに対するEndecaアプリケーションを管理するEndecaリソース (Forge、Dgidx、1つまたは複数のDgraphなど) の場所と構成を定義するタスクです。

アプリケーションは3種類の方法でプロビジョニングできます。

方法	説明
Deployment Templateの`initial¬ize_services`スクリプトを実行する	これはアプリケーションをプロビジョニングする最も簡単な方法です。 詳細は、このガイドおよび*Oracle Endeca Deployment Template Usage Guide*を参照してください。
Oracle Endeca Workbenchを使用する	詳細は、*Oracle Endeca Workbench*ヘルプを参照してください。
EACに直接`eaccmd`ツールを使用する	詳細は、*Oracle Endeca Application Controller Guide*を参照してください。

関連リンク

アプリケーション・プロビジョニングの更新 44 ページ
`AppConfig.xml`ファイルに変更を加えた場合は、Deployment Template構成のアプリケーション・プロビジョニングも更新する必要があります。Deployment Templateの環境では、ベースライン更新を実行せずにアプリケーション・プロビジョニング定義を更新できます。

eaccmdを使用したEACアプリケーション・プロビジョニングのバックアップ 44 ページ
既存のアプリケーションのプロビジョニングをバックアップするために、(Deployment Templateではなく) `eaccmd`を使用する場合は、`eaccmd describe-app`コマンドを実行し

て、その出力をファイルにリダイレクトできます。不要な変更を加えた場合にアプリケーションのプロビジョニングを簡単に元に戻すことができるため、これはテストの際に役立つテクニックです。

アプリケーション・プロビジョニングの更新

AppConfig.xmlファイルに変更を加えた場合は、Deployment Template構成のアプリケーション・プロビジョニングも更新する必要があります。Deployment Templateの環境では、ベースライン更新を実行せずにアプリケーション・プロビジョニング定義を更新できます。

Deployment Template構成のAppConfig.xml定義を更新する手順は、次のとおりです。

次のように--update-definitionオプションを指定してruncommandスクリプトを実行します。

オプション	説明
Windowsの場合	`C:\apps\myApplication\control\runcommand.bat --update-definition`
UNIXの場合	`/apps/myApplication/control/runcommand.sh --update-definition`

--update-definitionオプションによって、アプリケーション・プロビジョニングが更新されます。

eaccmdを使用したEACアプリケーション・プロビジョニングのバックアップ

既存のアプリケーションのプロビジョニングをバックアップするために、(Deployment Templateではなく)eaccmdを使用する場合は、eaccmd describe-appコマンドを実行して、その出力をファイルにリダイレクトできます。不要な変更を加えた場合にアプリケーションのプロビジョニングを簡単に元に戻すことができるため、これはテストの際に役立つテクニックです。

この目的にはeaccmd describe-appコマンドを使用できますが、これは、このコマンドの出力がeaccmd define-appコマンドで使用されるのと同じXML形式のためです。

eaccmdコマンドの詳細は、*Oracle Endeca Application Controller Guide*の「*Eaccmdツールの使用*」の章にある*プロビジョニング・コマンド*に関する項を参照してください。

eaccmdを使用して、既存アプリケーションのプロビジョニングをバックアップする手順は、次のとおりです。

eaccmd describe-app --app application_name ?canonical>application.xmlコマンドを実行します。

このコマンドにより、define-appコマンドで後で簡単に使用できるXMLがファイルに書き込まれます。eaccmd describe-app ?canonicalコマンドには、記述するアプリケーションの名前を取得する?appパラメータが必要です。--canonicalフラグにより、アプリケーションの記述のすべてのエントリに絶対パスが使用され、.や..などの参照が解決されて簡単なパスが生成

されるように強制されます。これにより、アプリケーションの記述の読取りが非常に容易になります。

生成されたアプリケーション記述ファイルは、後で`eaccmd define-app`コマンドで定義ファイルとして使用できます。

システム操作を実行するためのオプション

二者択一のアプローチ (Deployment Templateを利用するスクリプト、またはOracle Endeca WorkbenchのEAC管理コンソールでプロビジョニングした独自のスクリプト) を使用して、システム操作を実行できます。

このガイドでは、Endeca実装で操作タスクを実行する主要な手段として、Deployment Templateを使用することを前提とし、EACに関する基本的な情報を提供します。

方法	説明
Deployment Template環境でのカスタマイズ済スクリプト	Deployment Templateに付属のサンプル・スクリプトは、EACを介してEndeca操作タスクを制御します。このスクリプトを使用したり、このスクリプトに基づいてカスタム・スクリプトを作成できます。 通常、スクリプトでは定型的なベースライン更新や部分更新などのプロセスを実行します。また、Dgraphの停止前または起動後に実行する特定のスクリプトを追加できます。
Oracle Endeca Workbench環境での独自のJavaスクリプト	ベースライン更新および部分更新を実行するための独自の操作スクリプトを作成し、Oracle Endeca Workbenchからそのスクリプトを実行するようにスクリプトをプロビジョニングできます。 スクリプトは、各サーバーのプロセスを制御するEAC Centralサーバーと通信する必要があります。 EACと直接通信する独自のカスタムJavaスクリプトを使用する際のスクリプトの記述方法は、*Oracle Endeca Application Controller Guide*を参照してください。

EACコンポーネントのステータスの確認

いくつかの方法を使用して、EACにプロビジョニングされるアプリケーションのコンポーネント (Dgidx、Forge、Dgraphなど) のステータスを確認できます。

EACコンポーネントのステータスを確認するには、次のいずれかの方法を使用します。

- **Oracle Endeca Workbench**。**EAC**管理コンソールを使用して、アプリケーションの特定のコンポーネントのステータスを監視します。各コンポーネントのステータスは、「**auto-refresh**」に設定できます。
- **eaccmd**ユーティリティ。コマンドラインで次のコマンドを実行します (UNIXおよびWindows)。

```
eaccmd status --app <app_id> --comp <comp_id
```

  👉 注: `--comp <comp_id>`引数を`--script <script_id>`引数に置き換えて、EACスクリプトのステータスを確認することもできます。

- **Deployment Template**。コマンドラインで次のコマンドを実行します。

```
Windows:
<PROJECT_DIR>\control\runcommand.bat --print-status
UNIX :
<PROJECT_DIR>/control/runcommand.sh --print-status
```

  このコマンドにより、現行アプリケーションの各コンポーネントのステータスが出力されます。

- 自分のアプリケーション。プログラムによるWebサービス・コールを使用してEACを問い合せるように、アプリケーションを作成できます。`getComponentStatus() API`メソッドの使用方法の詳細は、*Oracle Endeca Application Controller Guide*のEndeca Application Controller APIインタフェース・リファレンスに関する項を参照してください。

ゾンビ**EAC**プロセスの回避

UNIXシステムでは、psコマンドにより、EACで発生した多数のゾンビ・プロセスがレポートされる場合があります。これは、既知の予定されたEACの動作で、必ずしも問題を示すものではありません。

たとえば、psコマンドによって、次の出力を参照できます。

```
> ps -ef | grep endeca
endeca 1924 1875 0 - ? 2:00 <defunct>
[...]
```

さらに、この形式の警告メッセージが、影響を受けたサーバーの`$ENDECA_CONF/logs/process.0.log`ファイルに表示されます。

```
Apr 17, 2009 11:24:17 AM
com.endeca.esf.delegate.procctrl.ExecutableProcessHandle
tryCleanShutdown
WARNING: Process 1924 did not shutdown cleanly after 30 seconds.
Terminating forcefully.
```

これらの警告メッセージの原因は、次のとおりです。EACがDgraphなどの子プロセスをシャットダウンするときは、最初にそのプロセスに対して正しいexitコマンド (Dgraphの場合は`admin?op=exit`) を送信し、プロセスの終了を30秒間待機します。ただし、Dgraphが長時間実行の問合せを処理している場合やそのリクエスト・キューが長い場合は、30秒間でシャットダウンできない場合があります。

30秒間を経過してもプロセスが終了しない場合、EACでは、前述の警告メッセージがログに記録され、そのプロセスはオペレーティング・システムの`kill`コマンドを使用して停止されます。これが発生したとき、影響を受けたプロセスは`ps`によって、状態が`<defunct>`としてレポートされます。この状態では、メモリー、ディスク領域、ポートは使用されないため、システムに関する問題はありません。

あるいは、この状態は、`shutdown.sh`スクリプトを使用しないで、EACプロセスを直接停止した場合に発生する可能性があります。この場合、EACプロセスは即時に停止し、子プロセスは`<defunct>`状態のままとなります。

ゾンビEAC処理を回避するには、次の推奨事項を考慮してください。

- リクエスト・ログは、キューや長い処理時間のために、Dgraphによる`admin?op=exit`コマンドへの時間内対応が阻止されているかどうかを示しています。阻止されている場合は、トラフィックをより多くのMDEX Engineミラーに分散するか (キューの場合)、または問合せの複雑度を下げる (長い処理時間の場合) ことで、Dgraphではexitコマンドに対してより迅速に対応できるようになります。
- 他の選択肢として、サーバーの`$ENDECA_CONF/conf/eac.properties`ファイルにある`com.endeca.eac.process.shutdownTimeoutSecs`設定の値を変更して、EACシャットダウンに対するデフォルトの30秒タイムアウト時間を上書きできます。

  この値は、サーバー上のEACエージェントがプロセスの停止を待機する時間 (秒) を示します。この設定に高い値を指定すると、`<defunct>` EAC子プロセスの作成が抑制されますが、EACエージェントはすべてのプロセスの停止をより長く待機するため、EACの更新も遅くなります。

  👉 注: この設定に対する変更は、Endeca HTTPサービス (EACが含まれている) がサーバーで再起動されるまでは無効です。

EACメモリー使用

このトピックでは、EACメモリー使用の最適化に関する推奨事項の概要を説明します。たとえば、サーバーで複数のMDEX Engineと複数のEACエージェントが実行されている場合は、複数のEACエージェント・プロセスの大きなメモリー・フットプリント(EACエージェント・プロセスによって使用されるRAMの量)に起因して、パフォーマンスの問題が発生する可能性があります。

EACメモリー使用を最適化するには、次の推奨事項を使用します。

- EACエージェント・プロセスの仮想メモリー使用量と常駐セット・サイズ (RSS) を計測するには、サードパーティのユーティリティ (topなど) を使用します。
- 仮想メモリー使用量が大きい場合でも、プロセスのワーキング・セット・サイズがRAMに適合しているかぎり、この状態が問題になることはないことに注意してください。

  MDEX Engineでのメモリー使用量と、プロセスで使用されるRSS、ワーキング・セット・サイズおよび仮想メモリー の相互関係に関する詳細情報は、*Oracle Oracle Endeca MDEX Engine*パフォーマンス・ガイドを参照してください。

- MDEX EngineとEACエージェント・プロセスのためにメモリーを開放するには、サーバーで実行しているクリティカルではないEndeca以外のプロセスを削除します。また、スワップ領域の量の増加も考慮します。

- 現在の実装にメモリー制約があり、プロセス実行でメモリーが不足する場合は、その実装のトポロジの再構成を検討してください。たとえば、10台のMDEX Engineサーバー (それぞれに2つのMDEX Engineをホストする8つのコアがある) が以前あった場合は、トポロジを再構成して、サーバー当り8つのスレッドで実行する、2つでなく1つのMDEX Engineの設定を検討します。このような構成によって、メモリーはさらに効率的になります。

  注: 双方に十分なRAMがある場合、1つの8スレッドDgraphでは、2つの4スレッドDgraphほどのスループットは発生しませんが、MDEX Engineを実行しているサーバーに物理的なメモリー制約がある場合は、1つの8スレッドDgraphによってスループットを増加させることができます。これは、メモリーの問題に及ぶことによるMDEX Engineの再起動が回避されるためです。

Deployment TemplateとOracle Endeca Workbenchの相互作用

このトピックでは、Deployment TemplateとOracle Endeca Workbenchの相互作用に関する概要を示します。

目的の環境でDeployment Templateに加えてOracle Endeca Workbenchを使用する場合は、これらの相互作用の概要を示す次の各ポイントに注意してください。

注: このトピックでは、これら2つのコンポーネント間の相互作用に関する概要を示し、管理操作に対する推薦事項をリストします。ステージング環境と本番環境におけるOracle Endeca WorkbenchとDeployment Template両方の実装方法の詳細は、*Tools and Frameworks Deployment Template Usage Guide*を参照してください。

- **Deployment Template**を介した、操作環境に対する変更の管理

  Forge、DgidxおよびDgraphに対するコマンドライン引数など、EACコンポーネント構成に対する変更は、WorkbenchのEAC管理コンソールではなく、Deployment Template (操作タスクに使用するように選択した場合) を介して管理することをお薦めします。

- インスタンス構成に対する変更の**Oracle Endeca Workbench**への手動アップロード

  最初にアプリケーションをデプロイした後で、インスタンス構成ファイルに変更を加え (新しいディメンションの追加や動的なビジネス・ルールに対する新しいゾーンの定義など)、次にベースライン更新や部分更新のためにDeployment Templateスクリプトを実行した場合は、変更した構成ファイルと設定が、Deployment TemplateによってWorkbenchに自動的にアップロードされません。

  必要な場合、構成の更新は、Deployment Templateスクリプトを介してWorkbenchに手動でアップロードできます。これは、動的なビジネス・ルール・ゾーンへの変更やスタイルの更新、ディメンションの追加または削除時に必要な場合があり、Workbenchは、更新した構成に基づいてビジネス・ユーザーがルールを保守できるように更新されている必要があります。

  通常、Workbenchと変更の同期化は、本番環境でのEndeca実装のデプロイ前またはEndeca実装の更新時に実行すると役立ちます。これらのパイプライン変更をWorkbenchを介してデプロイするには、`update_web_studio_config.[sh|bat]`スクリプトを使用します。

`update_web_studio_config.[sh|bat]`のコールには、Workbenchで使用可能なすべてのロックが必要です。これは、すべてのユーザーがシステムからログアウトし、リソースに関するロックを保持していない必要があることを意味しています。`update_web_studio_con¬fig.[sh|bat]`を実行するときは、Workbenchインタフェースを介してロックに関する問題を解決する必要があります。

- **Oracle Endeca Workbench**でのいくつかの設定の管理

  通常は、Deployment Templateフレームワークを使用すると、そのスクリプトが、EACコンポーネントとインスタンス構成ファイルに対するすべての変更を管理する指定環境になります。

  ただし、ルール、キーワード・リダイレクト、検索、ディメンション順序およびレポートの管理にもWorkbenchを使用する場合は、Deployment Templateで構成マネージャを有効にすることができます。構成マネージャは、Deployment Templateに対して、Developer Studioではなく、Oracle Endeca Workbenchを使用する必要がある変更を通知します。

  注: Deployment Templateでは、実際に、構成ファイルや操作設定の作成、変更または管理は実行されません。これらの機能は、Developer StudioとWorkbenchによって実行されます。ただし、スクリプトの実行時に、Deployment Templateでは、使用する設定とファイルについて、Developer Studio (これがデフォルトの想定です) またはWorkbenchからの情報が必要です。Developer StudioまたはWorkbenchで発生した変更を使用するかどうかを決定するために、Deployment Templateでは、その構成マネージャが使用されます。ただし、構成マネージャにEndeca Workbenchで保守されているファイルを指定した場合は、それらのファイルのDeveloper Studioバージョンが Deployment Templateフレームワーク内で実行するスクリプトに使用されていないことを確認します。

  要約すると、いくつかの変更をOracle Endeca Workbenchを介して保守する場合は、構成マネージャ・コンポーネントを有効にして、Workbenchで保守されている構成ファイルが、Deployment Templateのスクリプト実行時に使用されることを確認します。このコンポーネントの使用に関する詳細は、*Oracle Endeca Deployment Template Usage Guide*の「アプリケーション構成」の章を参照してください。

関連リンク

EACのDeployment Templateによって設定されたロックの解放 50 ページ
場合によっては、Deployment Templateスクリプトによって設定されているEAC内の特定のコンポーネントのロックを手動で解放することが必要な場合があります。

Dgraphログ・ファイルのアーカイブ

Deployment Templateを使用してベースライン更新スクリプトを実行すると、Dgraphファイルが自動的にアーカイブされます。また、Dgraphファイルを詳細な基準でアーカイブする場合は、MDEX Engine処理を停止してDgraphログ・ファイルをアーカイブし、Dgraphを再起動するカスタムのDeployment Templateスクリプトを作成できます。

ベースライン更新スクリプトの`applyIndex()`メソッドを使用すると、Dgraphが停止してログ・ファイルがアーカイブされ、Dgraphが再起動します。このメソッドは、Deployment Templateのデフォルトのベースライン更新スクリプトの`DistributeIndexAndApply`ステップにあります。このメソッドを使用して、カスタム・スクリプトを作成できます。

Dgraphを停止してそのログをアーカイブし、Dgraphを再起動する手順は、次のとおりです。

1. 次の例に示すように、`applyIndex()`メソッドをDeployment Templateのプロジェクト内の新しいスクリプトにコピーし、必要に応じて変更します。

```
<!--########################################
 # CUSTOM: Restart all dgraphs, archiving log files
 #
 -->
<script id="DgraphRestartWithArchive">
<log-dir>./logs</log-dir>
 <bean-shell-script>
   <![CDATA[
    for ( DgraphComponent dgraph : DgraphCluster.getDgraphs() )
    {
       if ( dgraph.isActive() )
       {
         dgraph.stop();
       }
    dgraph.archiveLogDir();
    dgraph.start();
    }
  ]]>
  </bean-shell-script>
</script>
```

2. Dgraphログをアーカイブするために、アプリケーションの`/control`ディレクトリに移動し、`runcommand DgraphRestartWithArchive`スクリプトを実行します。

EACのDeployment Templateによって設定されたロックの解放

場合によっては、Deployment Templateスクリプトによって設定されているEAC内の特定のコンポーネントのロックを手動で解放することが必要な場合があります。

Endeca実装では、様々なタイプのロックが表示される可能性があります。最初のタイプのロックは、Windowsファイル・システムのロックです。たとえば、Windows Explorerを`data\forge_output`の場所で開いていて、ベースライン更新でそのフォルダのクリーン・アップが試行された場合、`explore.exe`によってディレクトリのロックが保持されているため失敗します。ファイル・システムのロックを解放するには、プロセスおよびユーザーが関連するフォルダ(または、それらのフォルダ内のファイル)を開いていないことを確認してください。(UNIXでは、ファイルを排他的にシステム・ロックしません。)

発生する可能性がある他のタイプのロックとしては、Deployment Templateの`baseline_update`スクリプト、`partial_update`スクリプトまたは他のスクリプトによるEACのロック (またはEACフラグ) があります。デフォルトのロックは、`update_lock`と呼ばれます。これは、Deployment Templateスクリプトの次の行で作成されます。

```
LockManager.acquireLock("update_lock")
```

実行中のDeployment Templateスクリプトが未処理例外により実行の途中で中断された場合、またはスクリプトの実行中にユーザーが[Ctrl]-[C]キーを押して手動で中断した場合、EAC内のロックは設定されたままです。

たとえば、Deployment Templateログに次の例外エラーが表示される場合があります。

```
[10.17.09 06:52:09] SEVERE: Caught an exception while
 invoking method 'run' on object 'BaselineUpdate'.
Releasing locks.
Caused by java.lang.reflect.InvocationTargetException
...
[10.17.09 06:52:09] INFO: Released lock 'update_lock'.
```

この例外は他の原因による場合もありますが、通常はDgraphでのロック解放の失敗が原因です。

EAC内のコンポーネントのロックを解放するには、次のコマンドを実行します。

1. eaccmdツールを使用して、次のコマンドを実行し、アプリケーション内のすべての未処理フラグのリストを取得します。remove-all-flagsコマンドを実行する前に、これらのフラグを確認することが必要な場合があります。
   `list-flags --app application_name`
2. コマンドラインまたはeaccmdツールを使用して、次のコマンドを実行します。

オプション	説明
コマンドラインを使用する場合:	`.<AppDir>/control/`ディレクトリに移動します。 次のDeployment Templateコマンドを実行します。 Windowsの場合: `.\runcommand.bat LockManager releaseLock update_lock` UNIXの場合: `./runcommand.sh LockManager releaseLock update_lock`
eaccmdツールを使用する場合:	Windowsの場合: `eaccmd.bat remove-all-flags -app <your application>` UNIXの場合: `eaccmd.sh remove-all-flags -app <your application>` LockManagerは内部的にEACフラグ設定機能を使用するため、Deployment Templateのロックは`eaccmd`の`remove-all-flags`機能によって事実上解除されます。

これによって、EACのロックが解放されます。

構成からのコンポーネントの削除

非アクティブなコンポーネントを削除するために、複数のオプション (Deployment Templateでのそれらのコンポーネントの停止および削除、`initialize_services`の実行、`eaccmd`の使用、またはOracle Endeca WorkbenchのEAC管理コンソールの使用) を使用できます。

コンポーネントを停止して、そのコンポーネントをEAC定義から削除する手順は、次のとおりです。

次のオプションのいずれかを使用します。

オプション	説明
Deployment Templateの**runcommand**の使用	コンポーネントが実行中でその定義を削除する場合は、Deployment Templateの`runcommand`を使用してそのコンポーネントを停止します。その後に、`AppConfig.xml`から目的のコンポーネントを削除できます。 1. コンポーネントが実行中の場合は、`[appDir]/control/runcommand`*`component_name`* `stop`で停止します。 2. コンポーネントの定義を`[appDir]/control/runcommand`*`component_name`*`removeDefinition`で削除します。 3. `AppConfig.xml`ファイルから非アクティブなコンポーネントを削除します。
Deployment Templateの**initialize_services**スクリプトの実行	このスクリプトにより、`AppConfig.xml`に指定されているコンポーネントにプロビジョニングがリセットされます。また、実行中のコンポーネント (Dgraphやログ・サーバーなど) が停止されて、EACからアプリケーションが削除され、アプリケーション全体がEACに再度プロビジョニングされます。 👉 重要: `initialize_services`スクリプトの再実行により、Oracle Endeca Workbenchに存在する構成 (ルールの構成など) が消去されます (Workbenchで構成を作成した場合)。
Endeca Workbenchの**EAC**管理コンソールの使用	Oracle Endeca Workbenchでは、EACと通信してコンポーネントを停止および削除します。
eaccmdツールを使用したコンポーネントの停止および削除	eaccmdツールでは、非アクティブなコンポーネントを停止して、EACから削除します。 1. コンポーネントの停止: - Windowsの場合: `eaccmd.bat host:port stop --app <appname> --comp <component id>` - UNIXの場合: `eaccmd.sh host:port stop --app <appname> --comp <component id>` 2. コンポーネントの削除: - Windowsの場合: `eaccmd.bat host:port remove-component --app <appname> --comp <component id>` - UNIXの場合: `eaccmd.sh host:port remove-component --app <appname> --comp <component id>`

関連リンク

Deployment Template環境でのコンポーネントの削除 53 ページ
このトピックでは、AppConfig.xmlファイルから削除されたコンポーネントをDeployment Templateがどのように取り扱うかについて説明します。Deployment Templateの増分プロビジョニングでは、既存のコンポーネントは更新されますが、AppConfig.xmlに定義されていないコンポーネントは削除されません。

Deployment Template環境でのコンポーネントの削除

このトピックでは、AppConfig.xmlファイルから削除されたコンポーネントをDeployment Templateがどのように取り扱うかについて説明します。Deployment Templateの増分プロビジョニングでは、既存のコンポーネントは更新されますが、AppConfig.xmlに定義されていないコンポーネントは削除されません。

通常、initialize_servicesの実行により、EACおよびOracle Endeca Workbenchの両方のストアからアプリケーションが削除されて、アプリケーションが再プロビジョニングされるため、その時点でEACには、Deployment Templateにプロビジョニングされたコンポーネントのみが確保されます。

Deployment Templateがそのプロビジョニング・チェックを実行する際は、AppConfig.xmlに現在リストされているすべてのコンポーネントがプロビジョニングされ、コンポーネントのEACコピーで最新になっていることが検証されます。新規または変更されたコンポーネントは、この時点でプロビジョニングされるか、EACでプロビジョニングが更新されます。

ただし、Deployment TemplateはAppConfig.xmlに宣言されていないコンポーネント (ある時点では宣言されていて、その後削除されたコンポーネントも含む) は考慮しません。これは、EACにプロビジョニングされているコンポーネント (以前にDeployment Templateでプロビジョニングされたコンポーネントも含む) が残存し、EACと相互作用する唯一のツールではない環境をDeployment Templateがサポートすることを意味します。

シナリオの例

プロビジョニングの手順の後、Deployment Templateは、ベースライン更新を実行するたびにApp¬Config.xmlファイルの定義がEAC Centralサーバーに保存されている定義と比較して変更されたかどうかをチェックします。新しいフラグや追加のコンポーネントなどの変更を取得します。

一方、AppConfig.xmlファイルのDgraphを削除してベースライン更新スクリプトを変更すると、baseline_updateスクリプトは変更内容を順守しますが、EAC Centralサーバーに保存されたプロビジョニング構成には反映されません。スクリプトを実行すると、不整合に関するメッセージが発行されます。削除したDgraphは、AppConfig.xmlファイルにすでに記載されていなくても引き続き実行されます。

サービスURLを使用したEACの状態の判別

次のサービスURLを使用して、EAC CentralサーバーまたはEACエージェントが稼働中かどうかを判別できます。

EACの状態を判別する手順は、次のとおりです。

- EAC Centralサーバーの場合は、`http://machine_name:8888/eac/ProvisioningSer¬vice?wsdl`にアクセスします。
- EACエージェントの場合は、`http://machine_name:8888/eac-agent/IDelegateServ¬er?wsdl`にアクセスします。

EAC Centralサーバーのログ

特定のEACコンポーネント、ユーティリティおよびスクリプトに関連付けられたログ・ファイル以外は、EAC Centralサーバーとそのサービスが、そのワークスペース・ディレクトリにログ・ファイルを生成します。

EACのログは、`%ENDECA_CONF%\logs` (Windowsの場合) または`$ENDECA_CONF/logs` (UNIXの場合) に配置されます。

具体的には、`/logs`ディレクトリには、Endeca HTTPサービス、およびEAC CentralサーバーやEACエージェントなど、その内部で動作するアプリケーションによって生成された多数のファイルが格納されます。

EACのログには1GBのデフォルトのサイズ制限があります。ログは`main.rotation number.log`という名前で、最大サイズに達すると自動的に移行する2つのログの一部です。2番目のログ・ファイルが最大サイズに達すると、最初のログが上書きされます。つまり、`main.0.log`が1GBの制限に達すると、`main.1.log`への書込みが開始されます。`main.1.log`が1GBのサイズ制限に達すると、`main.0.log`が上書きされます。

EACの開発およびデバッグには、通常、次のログ・ファイルが必要であり役立ちます。

EACログのタイプ	説明
メイン・ログ。具体例: `main.0.log`	EACのほとんどのロギングは、このログ・ファイルに格納されます。たとえば、シェルまたはコンポーネントの起動時にスローされた例外がこのログに記録されます。 たとえば、存在しないホストでユーティリティを起動しようとすると、`"The host "my_host" in application "my_app" does not exist"`のような例外がこのファイルに記録されます。
プロセス・ログ。具体例: `pro¬cess.0.log`	EACプロセス制御モジュールで生成されたログがこのファイルに格納されます。このログには、プロセスの制御およびリカバリに関連付けられたメッセージが保存されます。 これらのメッセージには、スクリプト、コンポーネントおよびユーティリティの起動と停止、失敗したプロセスのリカバリ、およびアクティブなプロセスへの再バインドに関する情報が含まれます。
起動ログ。具体例: `invoke.0.log`	このファイルには、EAC Webサービスの起動に関連付けられたログが格納されます。たとえば、このファイルには、Web

EACログのタイプ	説明
	サービスのリクエストとレスポンスの正確なXMLコンテンツが記録されます。
Tomcat/Catalinaログ。具体例: - `catalina.out` - `catalina.[date].log` - `tomcat_[std¬ out|stderr].log`	これらのログは、EACのロード時にエラーが発生した場合に役立ちます。 たとえば、EACのコンテキスト構成でEAC WARファイルに間違ったパスが指定された場合、Endeca HTTPサービスの起動時にエラーが発生し、これらのログ・ファイルにログが記録されます。 あるいは、Endeca HTTPサービスのクリーン・スタートアップでは、例外が発生せずに正常に起動したことを示す"`Server startup in [n] ms`"というメッセージが出力されます。

EAC Central ServerマシンのIPアドレスの変更

このトピックでは、リモートMDEXHostマシンが機能する環境内で、EAC Central ServerマシンのIPアドレスを変更する方法について説明します。ここで、`AppConfig.xml`ファイルは、マシンのIPアドレスではなく、ホスト名を使用します。

EAC Central ServerマシンのIPアドレスを変更する手順は、次のとおりです。

1. EAC Centralサーバーとリモート MDEXHostの両方で、Endeca HTTPサービスを停止します。
2. 両方のマシンで、`java.security`ファイルの`networkaddress.cache.ttl`を0に変更します。このファイルは、デフォルトで、Windows上の`%ENDECA_ROOT%\j2sdk\jre\lib\security`、UNIX上の`$ENDECA_ROOT/j2sdk/jre/lib/security`にあります。
3. EAC CentralサーバーのIPアドレスを変更し、MDEXHostオペレーティング・システムでEAC Centralサーバーを新しいIPアドレスで解決できることを確認します。
4. 両方のマシンで、Endeca HTTPサービスを再起動します。
5. (オプション) 今後のDNSなりすまし攻撃を防ぐために、前述のステップ2で説明した`java.security`ファイルに戻り、`networkaddress.cache.ttl`を-1に戻して、両方のマシンでEndeca HTTPサービスを再起動できます。

第 5 章

Oracle Endeca Workbenchを使用したシステム操作の実行

この項では、Workbenchの管理セクションで可能なシステム管理および保守タスクについて説明します。

Oracle Endeca WorkbenchのEAC管理コンソールについて

管理者は、EAC管理コンソール・ページを使用して、システム・プロビジョニングの確立と変更、システム・コンポーネントの起動と停止、およびEACスクリプトの実行ができます。

Oracle Endeca Workbenchの管理コンソールは、次の3つのセクションに分割されています。

- Hosts - プロビジョニングしたホストで編成されたアプリケーションのビューが表示されます。このビューには、ホスト名、ホスト別名、ポートおよび構成オプションが表示されます。ホスト構成オプションの変更、ホスト上でのコンポーネントの起動または停止、ホスト上でのコンポーネントのステータスの表示ができます。
- Components - アプリケーションに対してプロビジョニングしたEndecaコンポーネントで編成されたアプリケーションのビューが表示されます。このタブでコンポーネントを作成できますが、ホストは作成できません。
- Scripts - アプリケーションで使用可能なEACスクリプトが表示され、EACスクリプトを追加、削除、実行および監視できます。ベースライン更新など、EACスクリプトによって実行されたシステム操作を停止および起動できます。

Oracle Endeca Workbenchを使用したアプリケーションのプロビジョニングについて

プロビジョニングとは、Endecaアプリケーションを管理するEndecaリソース (Forge、MDEX Engineなど) の場所と構成を定義するタスクです。

EAC管理コンソール・ページを使用して、Endecaアプリケーションをプロビジョニングできます。

EAC管理コンソールのプロビジョニング機能は、開発環境およびステージング環境で便利な機能です。本番環境では、Deployment Templateを使用してアプリケーションをプロビジョニングすることをお薦めします。Deployment Templateの詳細は、*Tools and Frameworks Deployment Template Usage Guide*を参照してください。

注: Workbenchでは、`AppConfig.xml`ファイルに対して、Deployment Templateのデフォルト要件とは異なるいくつかの変更が必要であることに注意してください。これらの相違点については、「アプリケーション構成ファイルの変更について」で説明します。

関連リンク

アプリケーション構成ファイルの変更について 59 ページ

Discover Electronics参照アプリケーションには更新された`AppConfig.xml`ファイルが含まれており、カスタム・アプリケーションを作成する際の基礎としてこのファイルを使用する必要があります。

Workbenchを使用したアプリケーションのプロビジョニング

EAC管理コンソール・ページを使用して、Endecaアプリケーションをプロビジョニングできます。

注: EAC管理コンソールのプロビジョニング機能は、開発環境およびステージング環境で便利な機能です。本番環境では、Deployment Templateを使用してアプリケーションをプロビジョニングすることをお薦めします。Deployment Templateの詳細は、*Deployment Template Usage Guide*を参照してください。

Workbenchを使用してアプリケーションをプロビジョニングする手順は、次のとおりです。

1. Webブラウザを使用して、Workbenchにログインします。
2. 1つ以上のアプリケーションを追加します。
3. 1つ以上のホストを追加します。
4. ホストにEndecaコンポーネントを追加します。1つのForge、少なくとも1つのインデクサ(Dgidx)、および1つのMDEX Engine (Dgraph) を設定する必要があります。
5. EACスクリプトを追加します。

これらの手順は、*Oracle Endeca Workbench Administrator's Guide*を参照してください。

EAC管理コンソールでのプロビジョニングの無効化

管理ロールがプロビジョニング情報を変更できないようにすることができます。

デフォルトでは、Workbench管理者は、EAC管理コンソール・ページのプロビジョニング情報を変更できます。必要に応じて、管理ロールがプロビジョニング情報を変更できないようにすることができます。

注: 本番環境では、Deployment Templateを使用してプロビジョニングを管理することをお薦めします。

プロビジョニングを無効にした場合も、EAC管理コンソールを使用して、Endecaコンポーネントおよびスクリプトを起動および停止したり、コンポーネントまたはスクリプトのステータスを監視できます。

管理ロールがプロビジョニング情報を変更できないようにする手順は、次のとおりです。

1. Endeca Tools Serviceを停止します。
2. %ENDECA_TOOLS_CONF%\conf (Windowsの場合) または$ENDECA_TOOLS_CONF/conf (UNIXの場合) に移動します。
3. テキスト・エディタで、webstudio.propertiesファイルを開きます。
4. 次のように、com.endeca.webstudio.allow.eac.プロビジョニング・プロパティをtrueからfalseに変更します。
   ```
   com.endeca.webstudio.allow.eac.provisioning=false
   ```
5. 内容を保存してファイルを閉じます。
6. Endeca Tools Serviceを起動します。

アプリケーション構成ファイルの変更について

Discover Electronics参照アプリケーションには更新されたAppConfig.xmlファイルが含まれており、カスタム・アプリケーションを作成する際の基礎としてこのファイルを使用する必要があります。

変更されたファイルには、次の変更が加えられています。

- 新しいSiteHandler Beanの追加

```
<spr:bean id="SiteHandler" class="com.endeca.dt.ifcr.SiteHandler">
    <spr:property name="appName" value="@@PROJECT_NAME@@"/>
    <spr:property name="repoUrl" value="http://@@HOST@@:@@WORK¬
BENCH_PORT@@/ifcr"/>
    <spr:property name="repoUserName" value="admin"/>
    <spr:property name="repoPassword" value="admin"/>
  </spr:bean>
```

- InitialSetupブロック内の新しいBeanへのコールの追加

```
<script id="InitialSetup">
    <bean-shell-script>
      <![CDATA[
    if (ConfigManager.isWebStudioEnabled()) {
      log.info("Updating Workbench configuration...");
      ConfigManager.updateWsConfig();
      log.info("Finished updating Workbench.");
    }
    SiteHandler.provisionSite();
      ]]>
    </bean-shell-script>
  </script>
```

これらの追加によって、Endeca構成リポジトリ内にデフォルト・サイト構造が設定されます。これらの変更を組み込むには、独自のアプリケーション構成ファイルを作成する際の基礎として、%ENDECA_TOOLS_ROOT%\server\bin\reference\discover-data\script\AppConfig.xmlファイル (UNIXでは

$ENDECA_TOOLS_ROOT/server/bin/reference/discover-data/script/AppConfig.xml)を使用する必要があります。

さらに、リポジトリでデフォルトの「admin」ログインとパスワードを使用しない場合は、SiteHandler Beanの関連する値を更新できます。

システム操作の実行について

システム操作には、更新の実行、Endecaコンポーネントの起動と停止、プロジェクトのバックアップなどが含まれます。

EAC管理コンソールの「Scripts」タブでは、Endecaコンポーネント (Forge、Dgidx、MDEX Engine、レポート・ジェネレータ、ログ・サーバーなど)を管理するベースライン更新を実行します。「Hosts」および「Components」タブでは、各Endecaコンポーネントを実行します。

その他のシステム操作、ステージング環境から本番環境へのインスタンス構成の転送、およびemgr_updateユーティリティの使用については、*『Endeca Commerce管理者ガイド』*の「環境間でのEndeca実装の転送」を参照してください。

Oracle Endeca Workbenchからのベースライン更新の実行について

ベースライン更新によって、ソース・データに対するForgeの実行、Endecaのレコードと索引を作成するためのDgidxの実行、および新しい索引を使用した1つ以上のMDEX Engineの起動を含めて、Endecaアプリケーションが完全に再構築されます。

さらに、ベースライン更新スクリプトを実行すると、Endeca構成リポジトリからMDEX Engineにコンテンツが発行されます。

EACスクリプトを使用してベースライン更新を実行するプロセスについては、*Oracle Endeca Workbench Administrator's Guide*を参照してください。

Endeca構成リポジトリからMDEX Engineへのコンテンツの発行について

Endeca構成リポジトリ内の情報は、Experience Managerまたはシソーラスで変更された後にMDEX Engineに発行されます。発行プロセス中にエラーが発生した場合は、使用可能なログ・ファイルを確認する必要があります。

ログ・ファイルは、Windowsでは%ENDECA_TOOLS_CONF%\logs\sling.<yyyy-mm-dd>.log、UNIXでは$ENDECA_TOOLS_CONF/logs/sling.<yyyy-mm-dd>.logにあります。

元の発行エラーの原因を特定して修正した後は、アプリケーションのcontrolディレクトリに移動し、次のコマンドを実行して再発行できます。

```
runcommand.bat IFCR push<Authoring|Live>ContentToDgraphById <Dgraph ID>
```

たとえば、Discover Electronics参照アプリケーションのオーサリングのコンテンツを送信するには、次のコマンドを使用します。

```
runcommand.bat IFCR pushAuthoringContentToDgraphById AuthoringDgraph
```

選択したDgraphは、AuthoringまたはLiveコンテンツ・グループに属している必要があります。Discover Electronics参照アプリケーションでは、これらの設定は<app

dir>\config\script\AuthoringDgraphCluster.xmlおよびLiveDgraphCluster.xmlファイルで構成されます。

注: このコマンドは各Dgraphに適用され、Dgraphクラスタまたはグループには適用されません。

MDEX Engineの起動と停止について

MDEX Engineのステータスは、EAC管理コンソールの「Components」タブで表示および変更できます。

MDEX Engineのステータスが「Running」の場合は、「Start」リンク (MDEX Engineラベルの横) が無効になり、「Stop」リンクのみ使用可能です。ステータスが「Stopped」の場合は、「Start」リンクのみ使用可能です。MDEX Engineを起動すると、コンポーネント構成の「Arguments」フィールドで指定したオプションを使用して起動されます。

MDEX Engineを起動および停止するプロセスについては、*Oracle Endeca Workbench Administrator's Guide*を参照してください。

ログ・サーバーの起動と停止について

ログ・サーバーは、EAC管理コンソールの「Components」タブから起動および停止できます。

ログ・サーバーのステータスが「Running」の場合は、「Start」リンク (ログ・サーバー・ラベルの横) が無効になり、「Stop」リンクのみ使用可能です。ステータスが「Pending」または「Stopped」の場合は、「Start」リンクのみ使用可能です。

ログ・サーバーを起動および停止するプロセスについては、*Oracle Endeca Workbench Administrator's Guide*を参照してください。

ログ・サーバーのログのロール

ログ・サーバーで作成されたログは、EAC管理コンソールからロールできません。ただし、URLコマンドを使用すると、ログをロールできます。

ログをロールするには、次のURLコマンドを使用します。

```
http://logserverhost:logserverport/roll
```

たとえば、このコマンドは次のように指定します。

```
http://web002:8002/roll
```

web002という名前のホストの8002ポートで稼働しているログ・サーバーがロールされます。

システム・ステータスの監視について

Endecaアプリケーションでプロビジョニングする各ホストおよびコンポーネントは、Oracle Endeca WorkbenchのEAC管理コンソール・ページにそのシステム・ステータスが表示されます。

Oracle Endeca Workbenchでは、「Hosts」タブおよび「Components」タブの縮小ビューに、コンポーネントのステータスのサマリーが表示されます。各コンポーネントの横にあるステータス・リンクを使用して、各コンポーネントの詳細にアクセスできます (ログ・サーバーは独自のアクティビティを記録しないため除く)。ステータス・リンクをクリックすると、開始時間、経過時間 (コンポーネントが実行された時間)、およびOracle Endeca Workbenchでコンポーネントのステータスが最後にチェックされた時間が表示されます。「Auto-Refresh」を選択すると、Oracle Endeca Workbenchではステータスが頻繁に自動リフレッシュされます。

コンポーネント・ログの表示

Endecaコンポーネント (独自のアクションをログに記録しないログ・サーバーを除く) の最新ログを表示するには、管理コンソール・ページの「Components」タブに示されているコンポーネントのログ・ファイルの値を確認します。次に、ログ・ファイル・ディレクトリを参照して、コンポーネント・ログを開きます。

ステータス情報のリフレッシュ

ステータス表示を手動でリフレッシュするには、「**Refresh Status**」ボタンをクリックします。

「**Auto Refresh**」チェック・ボックスをクリックして、ページを (あらかじめ設定されている間隔で) 自動的にリフレッシュするように設定することもできます。このオプションは、ベースライン更新が進行中でシステム状態の変更頻度が高い場合に便利です。デフォルトでは、このオプションは、システム全体の状態の変更頻度が低いためにオフになっています。

第 3 部

Oracle Endeca Commerceの各コンポーネントの管理

第 6 章

Dgidxの管理

この項では、Dgidxプロセスについて説明し、適切に操作するために役立つ管理タスクを概説します。Dgidxパフォーマンスを向上させ、問題をトラブルシューティングするためのヒントも記載されています。

Dgidxの処理とメモリー使用

DgidxはMDEX Engineのコンポーネントで、取得したレコードを構造に編成し、Dgraphによって使用されるいくつかの結果を部分的に事前計算し、Endeca索引を作成します。

DgidxはMDEX Engineインストール・パッケージの一部であるため、ITLサーバーとMDEX Engineサーバーの両方にインストールされます。Dgidxはベースライン更新時に実行するオフライン処理の一部であるため、ITLサーバーで実行することがベスト・プラクティスです (Deployment Templateはそのスクリプトでこのプラクティスを実施します)。

きわめて高いレベルでは、Dgidxプロセスは次のステップで構成されます。

1. ディメンションおよびレコードをそのプロセスのメモリーに読み取ります。レコードは、必要に応じて徐々にロード、処理およびリリースされます。
2. レコードを1つずつ索引付けし、関連する情報をすべての索引に追加します。これらの索引はディスクに保存されます。
3. 索引の生成をマージします。

処理の一環として、Dgidxは取得したレコードをソートし、ナビゲーション索引、テキスト索引およびワイルドカード索引を作成します。

Dgidxのメモリー使用

Dgidxは、ディスクの索引に対するオペレーティング・システム・キャッシングに依存し、メモリー・マッピングされたI/Oを使用して索引を取得します。これは、Dgidx作業プロセスに割り当てられた仮想メモリーのサイズに影響し、メモリー・サイズが周期的に増加します。

メモリーに関する考慮事項およびMDEX Engineによるメモリーの使用方法の詳細は、*Endeca MDEX Engine Performance Tuning Guide*を参照してください。

関連リンク

Dgidx索引付け時間の変動 72 ページ
Dgidxログを分析したときに、索引付け時間が予想した時間より周期的に長くなることに気づく場合があります。

Deployment Templateを使用したDgidxプロセスの実行

Dgidxプロセスは通常、Deployment Templateの`runcommand`ユーティリティを使用して起動します。

`runcommand`ユーティリティを使用すると、リモートのEndecaデータ処理サーバーでDgidx索引付けプロセスを起動できます。

Dgidxプロセスを実行する手順は、次のとおりです。

Deployment Templateから次のコマンドを実行します。

オプション	説明
Windowsの場合	`runcommand.bat MyDgidx run`
UNIXの場合	`./runcommand.sh MyDgidx run`

ここで、`MyDgidx`は、`AppConfig.xml`ファイルに指定されている`<dgidx id="MyDgidx" host-id="ITLHost">`などの`dgidx`要素の`id`の値です。

注: Deployment Templateの`runcommand`を使用したDgidxの実行に加えて、コマンドラインからDgidxのバイナリ実行ファイルを実行することもできます。これは、トラブルシューティングを目的としている場合に役立ちます。

関連リンク

コマンド・プロンプトでのDgidxバイナリの実行 66 ページ
まれな例ですが、EACおよびDeployment Template構成ではなく、コマンド・プロンプトからDgidxバイナリを実行することが必要な場合があります。これは、Dgidxプロセスが失敗し、問題の考えられる原因がDeployment Templateスクリプト、EACまたはDgidx自体にあるかどうかを識別する必要がある場合に役立ちます。

コマンド・プロンプトでのDgidxバイナリの実行

まれな例ですが、EACおよびDeployment Template構成ではなく、コマンド・プロンプトからDgidxバイナリを実行することが必要な場合があります。これは、Dgidxプロセスが失敗し、問題の考えられる原因がDeployment Templateスクリプト、EACまたはDgidx自体にあるかどうかを識別する必要がある場合に役立ちます。

注: コマンド・プロンプトでDgidxバイナリを直接実行するのは、個別のテスト環境でDeployment Templateのジョブを複製する必要があるまれな場合のみです。通常の設定で特定のプロセスを再実行する必要がある場合は、Deployment Templateの`runcommand`を使用することをお薦めします。

コマンド・プロンプトでDgidxバイナリを実行する前に、次の前提条件タスクを実行してください。

- 必要なファイルを、Dgidxプロセスによって検出可能な場所にコピーします。
- `dgidx_output`ディレクトリを作成します。これは、Dgidxを実行する前に存在する必要があり、コマンド・プロンプトからのDgidxの実行の過程で自動的に作成されることはありません。

コマンド・プロンプトでDgidxバイナリを実行する手順は、次のとおりです。

1. `%ENDECA_MDEX_ROOT%\bin`ディレクトリ (Windowsの場合) または`$ENDECA_MDEX_ROOT/bin`ディレクトリ (UNIXの場合) に移動します。
2. `dgidx` (Windowsの場合) や`dgidx` (UNIXの場合) などのDgidxコマンドを入力します。

   Dgidxバイナリの使用方法の情報が表示されます。

   Dgidxのパラメータは、個別の実装に依存します。Dgidxコマンドの使用方法を確認して、次のステップで実行するコマンドを構成します。
3. 次の例のようにコマンドを実行します。

オプション	説明
Windowsの場合	`%ENDECA_MDEX_ROOT%\bin\dgidx --out \localdisk2\endeca\`*`version`*`\dgidx.log --dtddir %ENDECA_ROOT%\conf\dtd \localdisk2\endeca\`*`version`*`\endeca \localdisk2\endeca\`*`version`*`\dgidx_output\endeca`
UNIXの場合	`$ENDECA_MDEX_ROOT/bin/dgidx --out /localdisk2/endeca/`*`version`*`/dgidx.log --dtddir $ENDECA_ROOT/conf/dtd /localdisk2/endeca/`*`version`*`/endeca /localdisk2/endeca/`*`version`*`/dgidx_output/endeca`

   このコマンドにより、Dgidxログ、DgidxのDTDディレクトリおよびDgidx出力ファイルが指定されます。

索引付け時間を高速化するためのヒント

パフォーマンスが最高になるようにDgidxを最適化するとともに、構成およびフロントエンド・アプリケーションを調整して索引付け時間を高速化できます。

索引付けを高速化するために、次のいくつかの使用について考慮してください。

- レコードまたはフィールド。
- テキスト検索可能なフィールド。
- ワイルドカード検索可能なフィールド。

👉 注: 特定の機能のパフォーマンスに関する詳細は、*Performance Tuning Guide*を参照してください。

Dgidxエラーのトラブルシューティング

このトピックでは、原因の特定と修正または発生の防止に役立つように、Dgidxクラッシュの考えられる主な原因を示します。

Dgidxを使用したDeployment Templateエラーのトラブルシューティング

Dgidxは、Deployment Templateスクリプト内で実行する必要がある最初のコンポーネントの1つであるため、多くの場合、最初に失敗するコンポーネントです。

たとえば、次のDgidxエラーが表示されることがあります。

```
SEVERE: Batch component 'Dgidx' failed. Refer to component
 logs in /usr/local/endeca/[version]/endeca/project/sample
/control/.././logs/dgidxs/Dgidx on host ITLHost.
Occurred while executing line 32 of valid BeanShell script:
[[

29| Forge.archiveLogDir();
30| Forge.run();
31| Dgidx.archiveLogDir();
32| Dgidx.run();
33|
34| // distributed index, update Dgraphs
35| DistributeIndexAndApply.run();

]]
```

次の手順を使用して、Deployment TemplateのDgidxエラーを調査します。

- Dgidxエラー・ログを見つけて検査します。
- DgidxはMDEX Engineインストールの一部であるため、データ処理 (ITL) サーバーにもMDEX Engineがインストールされていることを確認します。
- プラットフォーム・サービスのworkspace/confフォルダにあるeac.propertiesファイル (たとえば、/endeca/PlatformServices/workspace/conf/eac.properties) をチェックします。com.endeca.mdexRootプロパティが適切なMDEX_ROOTの場所とバージョン (たとえば、/usr/local/endeca/MDEX/*[version]*) に設定されていることを確認し、HTTPサービスを再起動します。
- コマンドラインからDgidxを実行します。この方法は、Deployment TemplateとEAC構成のレイヤーを使用せずにDgidxに直接アクセスします。

  デバッグのためにDgidxを直接実行することは有用です。たとえば、Deployment Templateのruncommandを使用してDgidxを実行するとDgidxが失敗となり、コマンドラインでは正常に実行できる場合、これは、データの問題とは対照的に、EACまたはDeployment Templateのいずれかに問題があることを意味します。

Dgidxでのメモリー割当エラーのトラブルシューティング

正常に実行するために必要な仮想メモリーまたはスワップ領域の不足により、まれにDgidxプロセスが失敗する場合があります。

たとえば、次のようなメモリー割当エラーが表示されることがあります。

```
FATAL DATE 13:50:40.753 UTC DGIDX {dgidx,baseline}:
memory allocation failure
```

```
-------------------------------------------------------------
Endeca fatal error detected.
-------------------------------------------------------------
```

次のヒントを使用して、メモリー割当クラッシュをトラブルシューティングします。

- Dgidxエラー・ログを見つけて検査します。
- 別のベースライン更新を実行し、`vmstat` (UNIXの場合) を使用してDgidxメモリー使用状況およびITLサーバーで使用可能なメモリー量およびスワップ領域を綿密に監視します。`vmstat`の出力を保存してから調査し、空きおよびスワップ・メモリーの量が少ないか、十分残っているかを確認できます。
- UNIXの場合は、`top`および`prtconf`のコマンドを使用して出力を調査します。
- このサーバーで実行しているいくつかのプロセスを一時的にシャットダウンし、Dgidxプロセスが一貫して失敗し続けるかどうかを調査します。
- プロセスが失敗しない場合は、プロセス実行中のピークの仮想メモリー使用状況を監視します。

関連リンク

Dgidxログ 69 ページ
アプリケーションのDgidxログの位置を特定するには、Deployment Templateの`AppConfig.xml`ファイルでDgidx定義を確認してください。

Dgidxログ

アプリケーションのDgidxログの位置を特定するには、Deployment Templateの`AppConfig.xml`ファイルでDgidx定義を確認してください。

アプリケーション名が`MyApp`、Dgidxプロセス名が`Dgidx1`の場合、デフォルトでは、Dgidxログは`MyApp/logs/dgidxs/Dgidx1`に格納されます。

たとえば、次の`AppConfig.xml`のDgidx定義には、Dgidxログの位置が記載されています。

```
# Dgidx
   #
 -->
 <dgidx id="Dgidx1" host-id="ITLHost">
   <properties>
    ...
   </properties>
   <directories>
     <directory name="incomingDataDir">./data/forge_output</directory>
     <directory name="configDir">./data/forge_output</directory>
   </directories>
   <args>
     <arg>-v</arg>
   </args>
   <log-dir>./logs/dgidxs/Dgidx1</log-dir>
   <input-dir>./data/dgidxs/Dgidx1/dgidx_input</input-dir>
   <output-dir>./data/dgidxs/Dgidx1/dgidx_output</output-dir>
   <data-prefix>Test-part0</data-prefix>
   <temp-dir>./data/dgidxs/Dgidx1/temp</temp-dir>
   <run-aspell>true</run-aspell>
 </dgidx>
```

次の例で、Dgidxログ・ファイルの一般的ないくつかの項目とその説明を示します。

注: また、Dgidxによってレコードごとに`Endeca.DataSize`、`Endeca.NumAssigns`および`Endeca.NumWords`という`admin`タイプの3つのプロパティが作成されることに注意してください。これらのプロパティは、Dgidxログおよび Dgraphの主要プロパティに表示されます。これらのプロパティは将来のリリースでサポートされなくなる可能性があるため、ログ内のこれらのプロパティは無視し、これに基づいてフロントエンド・アプリケーション・ロジックを作成しないことをお薦めします。

例**1**

```
=== DGIDX: Finished phase
"Read raw dimensions,
 properties, and records"
=== Phase Time: 19 minutes, 44.11 seconds
```

このログ・エントリは、Dgidxがすべてのデータを読み取り、すべての索引を作成していることを示します。

例**2**

```
$->tail Dgidx.log
Sorting... 22.16 seconds
Writing cycle 255 to temporary file
Parsing text fields...
...
179,600,000 text fields,
5,726,985,428 elements
179,700,000 text fields,
5,730,167,391 elements
...
```

このログ・エントリは、次のことを示します。

- `text fields`。Text fieldsは、ディメンション検索またはレコード検索 (あるいはその両方) に対してDgidxが索引に追加するレコードの個々のエントリです。Dgidx出力ログには、レコードの各ディメンションまたはプロパティのテキスト・フィールド合計数がリストされ、テキスト検索の索引付けで処理されたテキスト・フィールド数が定期的に出力されます。

  テキスト検索の索引付けでは、すべてのディメンションまたはプロパティのテキスト・フィールドを操作するため、定期的に出力される合計はそれぞれの合計を超える可能性があります。

  テキスト・フィールドには1つまたは複数の用語が格納されます。テキスト・フィールド数と要素数が大きく異なるのは、1つのプロパティまたはディメンション (あるいはその両方) に対して多数の用語がレコードに格納されることに起因します。

- `elements`。Elementsは、索引に送信された個々の用語または用語関連オブジェクトの数を表します。

  要素は、テキスト検索 (適用可能な場合はディメンション検索も含む) 用にソートおよび格納されます。

たとえば、`Name: John Lee Age: 24 Hired: 2008-08-14 Description: Permanent`という従業員レコードがあるとします。`Name`はディメンション検索に使用可能なディメンション、`Description`はテキスト検索に使用可能なプロパティである場合、ログには2つのテキスト・フィールドと3つの要素を備えたレコードとして表されます。

注: これらの数は概算で、索引の項目の規模を反映します。これらの数をデータ・コーパスで一意の正確な用語数とは解釈しないでください。他にも考慮事項はありますが、単一の入力語によって複数の索引要素が生成され、その異なるタイプの索引ごとに、Dgidxでは異なるタイプの一意の要素が使用されます。

関連リンク

テキスト検索索引付けに対するDgidxログの詳細 71 ページ
Dgidxログのテキスト検索索引付け部分を検査し、このトピックの情報を使用して、索引付け時間に影響を与えているログ内の項目を識別できます。

欠落または重複レコード指定値があるレコードのDgidx処理 72 ページ
Dgidxで、欠落または重複レコードの指定子(指定)値があるレコードが処理されると、そのDgidxは正常に完了しますが、非常に大きなログ・ファイルが生成されます。

Dgidx索引付け時間の変動 72 ページ
Dgidxログを分析したときに、索引付け時間が予想した時間より周期的に長くなることに気づく場合があります。

テキスト検索索引付けに対するDgidxログの詳細

Dgidxログのテキスト検索索引付け部分を検査し、このトピックの情報を使用して、索引付け時間に影響を与えているログ内の項目を識別できます。

ステミングおよびスペルは、Dgidxログのテキスト検索索引付け部分のログ数には影響を与えません。ただし、ワイルドカード検索では、索引に対して作成されるエントリ数が増加します。

テキスト検索索引付けに関連する次の項目がDgidxログに表示されます。

Dgidxログ項目	説明
`Records`	`recsearch_indexes.xml`ファイルにリストされている実際のレコード数およびディメンション数に対応します。これは、テキスト検索用にDgidxによって索引付けされる必要があるレコード数およびディメンション数を表します。
`Text fields`	テキスト検索に使用できる合計ペア数に対応します。 (ペアとは、ディメンションまたはプロパティとそれらに対応する値との関連です。)
`Entries`	索引に対して作成された合計エントリ数に対応します。 注: ワイルドカード検索が有効な場合は、`entries`の数が増加します。
`Rec`	標準索引を反映します。
`RecWC`	標準索引に加えて作成されるワイルドカード索引を反映します。

欠落または重複レコード指定値があるレコードの**Dgidx**処理

Dgidxで、欠落または重複レコードの指定子 (指定) 値があるレコードが処理されると、そのDgidxは正常に完了しますが、非常に大きなログ・ファイルが生成されます。

ログには、レコード全体を出力する警告レベルのメッセージが含まれます。これらの警告メッセージは、レコードが、プロジェクトの構成済レコード指定プロパティから不適切に割り当てられたプロパティ値であるため、Dgidxログに表示されます。

- アプリケーションでレコード指定プロパティが定義されている場合、各レコードには、該当するプロパティの単一の固有値が含まれている必要があります。レコードにレコード指定プロパティ値が含まれていない場合、Dgidxでは、次の例のように、"record... has no value assigned to it from any record specifier property"という警告が出力されます。

```
WARN 08/16/09 15:49:23.897 UTC DGIDX {dgidx,baseline}: The record
with the following properties has no value assigned to it from any
record specifier property. This record cannot be modified with
rapid updates:
[Record Id=4]
Dimension[6200,"Wine Type"]: Value[8013] "White"
Dimension[8,"Region"]: Value[4294967254] "Mendocino Lake"
[...]
Property["P_Body"]: Value[0xce1bd0] "Ripe"
Property["P_DateReviewed"]: Value[0xce1470] "02/28/95"
[...]
```

- レコードに、他のレコードですでに使用しているレコード指定プロパティ値 (一意でない値) が含まれている場合、Dgidxでは、次の例のように"Two records cannot share the value... for specifier property"という警告が出力されます。

```
WARN 08/16/09 15:49:23.897 UTC DGIDX {dgidx,baseline}: Two records
cannot share the value "34699" for specifier property "P_WineID";
removing this record:
[Record Id=2]
Dimension[6200,"Wine Type"]: Value[8013] "White"
Dimension[8,"Region"]: Value[4294967282] "Sonoma"
[...]
Property["P_Body"]: Value[0xce1870] "Crisp"
Property["P_WineID"]: Value[0xcd6e40] "34699"
[...]
```

どちらの場合も、Dgidxでは、レコードのプロパティ値とディメンション値がログに出力されるため、後日の更新時には識別して訂正できます。

前述の場合、この完全なレコードの表示を抑制する方法はありません。かわりに、ログに記録されたレコード指定に関する問題を、プロジェクトのForgeパイプラインまたはそのレコード指定プロパティ選択を変更して訂正する必要があります。レコード指定プロパティ値がないレコードに値を割り当て、重複が発生しないように、各レコードに一意のレコード指定プロパティ値が割り当てられていることを確認します。Dgidxログに出力されたレコードの詳細を使用すると、レコードに一意のレコード指定プロパティ値がない場合でも、影響を受けたレコードを識別できます。

Dgidx索引付け時間の変動

Dgidxログを分析したときに、索引付け時間が予想した時間より周期的に長くなることに気づく場合があります。

索引付け操作は、索引付け時間の変動の原因となる既存レコードの特性の多少の変化ではなく、ユーザーによる通常のレコードの追加のように見える場合があります。

Dgidxでは、多数の索引付け生成ファイルのマージ・プロセスが定期的に行われます。特に、生成ファイルの数が多くなると、Dgidxではそれらのファイルをマージして、オープンしているファイル数を減らします。生成ファイルの数が200を超えると、この特別なマージ・ステップが行われます。

第 7 章

Dgraphの管理

この項では、Dgraphの基本的な管理タスクについて説明します。Dgraphのトラブルシューティングのヒントを紹介し、Dgraphのログについても説明します。

pingコマンドを使用したDgraphの確認

Dgraphの可用性を速やかに確認する方法は、このトピックで説明するURLにアクセスすることです。

Dgraphが実行中かどうかを確認する手順は、次のとおりです。

Dgraphの場合は、次にアクセスします。

```
http://DgraphServerNameOrIP:DgraphPort/admin?op=ping
```

Dgraphによって、次の内容を含む軽量なHTMLレスポンス・ページがすぐに返されます。

```
dgraph host:port responding at date/time
```

👉 注: 「MDEX Engine Statistics」ページを表示して、MDEX Engineが実行され、問合せを受け入れているかどうかを確認することもできます。

Deployment TemplateでのDgraphに対する引数の指定

Deployment Templateを使用している場合は、`AppConfig.xml`ファイルの`<dgraph-defaults>`要素の下にDgraph引数を指定します。

Dgraphに引数を指定する手順は、次のとおりです。

次の例のように、`AppConfig.xml`ファイルにある`<dgraph-defaults>`の`<args>`要素に引数を追加します。

```
<dgraph-defaults>
<properties>
  ...
</properties>
<directories>
```

```
  ...
</directories>
<args>
  <arg>--threads</arg>
  <arg>2</arg>
  <arg>--spl</arg>
  <arg>--dym</arg>
</args>
<startup-timeout>120</startup-timeout>
</dgraph-defaults>
```

注: 値を取得する引数 (`--threads`など) は、連続する2つの`<arg>`要素に分ける必要があります。

デバッグ情報の収集

MDEX Engineに関する問題のデバッグを試みる前に、次の情報を収集します。

- ハードウェアの仕様および構成。
- Endecaトポロジの説明 (サーバー、Dgraphの数)。
- 「MDEX Engine Statistics」ページのデータ。
- `AppConfig.xml`ファイル。
- パイプライン・ディレクトリの内容。
- Dgraph入力。
- 部分更新ファイル。
- 通常の部分更新の説明。
- 影響を受けるDgraphの説明。

関連リンク

Dgraphによって作成されるログ 76 ページ
Dgraphでは最大5つのログが作成されますが、これらのログの一部は実装および使用するEndecaコンポーネントによって決まります。このトピックでは、これらのログの概要を示します。

Dgraphによって作成されるログ

Dgraphでは最大5つのログが作成されますが、これらのログの一部は実装および使用するEndecaコンポーネントによって決まります。このトピックでは、これらのログの概要を示します。

これらのDgraphのログを使用して、MDEX Engineの問合せをトラブルシューティングしたり、特定の問合せや更新のパフォーマンスを追跡します。

Dgraphリクエスト・ログ

Dgraphリクエスト・ログは常に作成されます。これを使用して、問合せと更新の両方の処理をデバッグします。処理された各問合せに対して1つのエントリが格納されます。

次のいずれかの方法を使用して、このログへのパスを設定できます。

- Deployment TemplateのAppConfig.xmlファイルに、Dgraphコンポーネントの<log-dir>要素で設定したパスを指定します。これに基づいて、Deployment Templateは$component.reqlogの形式でファイルを作成します。たとえば、Dgraph1.reqlogは、Dgraph1という名前のDgraphコンポーネントのDgraphログへのパスです。
- コマンドラインからDgraphを使用している場合は、リクエスト・ログへのパスをdgraph.reqlogというファイル名でDgraph作業ディレクトリに作成します。

Dgraphリクエスト・ログの詳細は、*Performance Tuning Guide*を参照してください。

Dgraphエラー・ログ

Dgraphエラー・ログは、コマンドラインまたはdgraph --outフラグを使用して、stderrをファイルにリダイレクトする場合にのみ作成されます。それ以外の場合、エラー・メッセージはstderrに表示されます。

Dgraphエラー・ログには、警告およびエラーのメッセージに加え、起動メッセージが格納されます。Dgraphのフラグ(-vなど)を使用して構成できます。また、このドキュメントの付録に記載されているconfig?op=log-enable操作により、特定の機能について詳細を記録できます。

AppConfig.xmlファイルに、Dgraphコンポーネントの<log-dir>要素で$component.log形式を使用してパスを指定できます。たとえば、Dgraph1.logは、Dgraph1という名前のDgraphコンポーネントへのパスです。

更新ログ

Dgraph更新ログは、Dgraphを--updatelogフラグ付きで実行した場合、またはDeployment Templateを介して実行した場合にのみ作成されます。

このログへのパスは、Deployment TemplateのDgraphコンポーネントの<log-dir>要素に$component.updatelogの形式で設定できます。たとえば、Dgraph1.updatelogは、Dgraph1という名前のDgraphコンポーネントのDgraph更新ログへのパスです。

プロセス起動ログ

Dgraphプロセス起動ログは、Dgraphプロセスの起動時に発生するメッセージ用にDeployment TemplateおよびEAC環境で作成されます。通常、このログは空です。Deployment TemplateまたはEACの問題をデバッグする際に役立ちます。

このログへのパスは、Dgraphコンポーネントの<log-dir>要素で$component.start.logの形式を使用して設定できます。たとえば、Dgraph1.start.logは、Dgraph1という名前のDgraphコンポーネントに対するプロセス起動ログを示します。

問合せ当たりの**EQL**統計ログ

問合せ当たりのEQL統計ログは、Endeca問合せ言語 (Endeca Query Language: EQL) を使用する場合に役立ちます。これは、--log_stats *path* フラグを使用してDgraphを実行した場合にのみ作成されます。このログは、Deployment Template主導のデフォルト構成では作成されません。

Endeca問合せ言語の詳細は、*Advanced Development Guide*を参照してください。

ベースライン更新エラーのトラブルシューティング

ベースライン更新エラーをデバッグするには、最初にDgraphリクエスト・ログとベースライン更新ログを検査し、次にEACプロセス・ログを検査します。

次の推奨事項を使用します。

- **Dgraph**リクエスト・ログの確認。ベースライン更新に失敗した時間帯のログを確認し、Dgraphの問題を除外します。

  Dgraphへのヘルス・チェックの送信時間、Dgraphの再起動時間、部分更新の発行時間および最終問合せの発行時間を確認します。

  たとえば、このDgraphリクエスト・ログの変更概要には一定期間のアクティビティが表示されます。

  ```
  12096521815/1/09 14:29 last search query
  12096522265/1/09 14:30 health check
  12096526095/1/09 14:36 last health check for x time
  12096571605/1/09 15:52 health checks resume
  12096574435/1/09 15:57 last empty health check
  12096601195/1/09 16:41 Dgraph startup
  12096601435/1/09 16:42 first query
  ```

  Dgraphは14:29から15:57までの間、ヘルス・チェック以外のリクエストを受信していません。ログにエラー・メッセージはありません。この間にDgraphは再起動していません。これは、この例のベースライン更新の失敗を引き起こした問題がDgraphの外部で発生したことを示します。

- ベースライン更新出力ログの確認。たとえば、次の場合は、Dgraphコンポーネントを停止している最中にエラーが発生しました。

  ```
  [05.01.09 10:07:54] INFO: Stopping component 'Dgraph1'.
  [05.01.09 10:17:54] SEVERE: Error communicating with EAC agent while
  stopping component.
  Occurred while executing line 5 of valid BeanShell script:
  ```

  EACがDgraphコンポーネントを停止できなかった理由を詳細に調査するには、EACプロセスについてログを確認し、その冗長性を増やします。

- **EAC**プロセス・ログの冗長性の増加。

  [ENDECA_CONF]`/conf/logging.properties`ファイルで、EACロギング構成を指定します。`com.endeca.eac.invoke`、`com.endeca.eac.process`および`com.endeca.eac.main`のログ・レベルを`FINE`に設定します。これにより、ベースライン更新プロセスに再度失敗した場合に追加のデバッグ情報が提供されるようになります。

  👉 注: *ENDECA_CONF*`/logs`ディレクトリのサイズを監視し、ディスクが満杯にならないようにします。

  さらに例では、たとえば、EACプロセス・ログはWebサーバーとDgraphサーバー間のハードウェア・コンポーネントの停止期間を示します。これらのログは、ベースライン更新に再度失敗した場合に役立つ可能性があります。

部分更新のトラブルシューティング

このトピックには、部分更新のトラブルシューティングに役立つ様々な指針が含まれています。

失敗した更新ファイルへのアクセス

失敗した更新ファイルを格納するためにMDEX Engineで使用されるデフォルト・ディレクトリは、`<updatedir>/failed_updates/`です。

Dgraphで`--failedupdatedir <dir>`コマンドを使用し、これらのファイルに対して別のディレクトリを指定できます。

索引ディレクトリにアクセスするための権限

部分更新に関連するDgraphエラーが発生した場合は、索引ディレクトリにアクセスするための権限がDgraphにあることを確認してください。

Dgraphは、部分更新を適用する前に、索引ディレクトリに対する権限をチェックします。必要な読取り/書込み権限が見つからない場合、Dgraphは標準エラー・ログにエラーを発行し、更新の適用は失敗となります。

Dgraphが冗長モードで実行されている場合は、Dgraphに読取り/書込み権限がない索引ディレクトリへのパスもログに記録されます。

Dgraphは、 `[AppDir]/dgidx_output/myApp_indexes`の次のディレクトリに対する権限をチェックします。

- `/committed`
- `/generations`

(これらのファイルパスは、Deployment Templateスクリプトを使用してアプリケーションが設定されたと想定しています。)

Dgraphがこれらのディレクトリにアクセスするためには、この両方のディレクトリに読取りおよび書込みの権限が必要です。ただし、Endeca実装のトポロジと一体化したファイル・システムの問題やハードウェア保守の問題のために、これらの権限はリセットされる可能性があります。その場合、Dgraphはこれらのディレクトリにアクセスできなくなります。

接続エラーの識別

Dgraph標準出力ログに`connection broken`メッセージがある場合は、Dgraphに問題が発生しているように見えますが、問題の実際の原因は通常、フロントエンド・アプリケーションをホストするサーバーとDgraphをホストするサーバー間の接続が機能していないことにあります。

接続エラーの場合は、Endeca実装の様々な部分から次のエラーおよび警告メッセージが発行されます。

- .NET APIでは、次の例外が出力されます。

```
Endeca.Navigation.ENEConnectionException:
Error reading from the connection. The operation has timed out
   at Endeca.Navigation.OptiBackendRequest.GetContent()
  at Endeca.Navigation.OptiBackend.GetNavigation(OptiBackendRequest req)
```

```
   at Endeca.Navigation.HttpENEConnection.Query(ENEQuery neq)
```

Java APIでも、同様の例外が出力されます。

- Dgraph標準出力ログには、次のような警告が含まれます。

```
WARN [DATE TIME] UTC (1239830549803)
DGRAPH {dgraph}: Aborting request: connection broken: client 10.10.21.21
```

- 最後に、Dgraphリクエスト・ログには、次のような`status 0`異常メッセージが含まれます。

```
1239830549803 10.6.35.35 - 349 0 19.35 0.00 0 - 0 0 - -
```

`connection broken`メッセージは通常、クライアントとDgraph間の接続に予期せぬエラーが発生したことを意味します。このタイプのエラーは、ネットワークなどのDgraph以外で発生したり、クライアント・アプリケーション・セッションのタイムアウトが原因で発生する場合があります。

クライアントとDgraph間の接続を調査してください。たとえば、クライアント・アプリケーション・セッションのタイムアウトを防止するために、フロントエンド・アプリケーションによる再試行を実装することを決定できます。

Dgraphでのソケットおよびポート・エラーのトラブルシューティング

Dgraphのプロセスがソケットにバインドできず、そのポートが初期化されない場合、Dgraphは起動できません。このエラーは、MDEX Engineをアップグレードし、サーバー上の別のプロセスがすでに使用しているポートを使用しようとしたときに発生する傾向があります。

Deployment Templateからのベースライン更新スクリプトが完了しても、MDEX Engineが起動しません。Dgraphログに次のエラーが表示されます。

```
ERROR (date and time)
DGRAPH {dgraph,baseline}: Unable to bind
to socket [err=`Result too large',errno=34]
FATAL (date and time)
DGRAPH {dgraph,baseline}: Unable to initialize the
main server port: 8000
```

通常、`"Unable to bind to socket"`エラーは問題のポートが別のプロセスにすでに使用されていることを示します。

Windowsのコマンドライン・ユーティリティ`netstat -ano`で、使用中のすべてのポートおよびそのポートを使用しているプロセスのプロセスIDのリストを表示します。このユーティリティを使用してポート8000を使用しているプロセスIDを識別し、そのプロセスをWindowsのタスク・マネージャで検索して別のプロセスが使用していることを確認します。このプロセスにより、Dgraphの起動が阻止されます。

Windowsシステムで使用中のポートを識別する手順は、次のとおりです。

1. `netstat -ano`を実行します。

   このコマンドにより、ポートおよび実行中のすべてのプロセスのプロセスIDのリストが表示されます。

2. Dgraphの使用予定ポートを使用しているプロセスを確認します。この例では、ポートは8000です。
3. Dgraphを別のポートで実行するか、すでに使用されているポートをMDEX Engineで使用するように解放できることを確認します。

Dgraphコア・ダンプ・ファイルの管理

Dgraphクラッシュでは、まれにDgraphがディスクにコア・ダンプ・ファイルを書き込むことがあります。

大規模なデータ・セットに対してDgraphを実行している場合は、索引サイズのそのメモリー内表現が物理RAMのサイズを超える可能性があります。このようなDgraphプロセスが失敗すると、Dgraphには潜在的に非常に大きいコア・ダンプ・ファイルをディスクに書き込む必要が生じます。

Dgraphをトラブルシューティングするために、多くの場合は、そのようなエラーの結果として書き込まれた一連のコア・ファイル全体を保持することが効果的です。十分なディスク領域がない場合は、このプロセスが停止するまでにファイルの一部のみがディスクに書き込まれます。最も役立つトラブルシューティング情報はコア・ファイルの最後の部分に格納されるため、このファイルをトラブルシューティングに有効利用できるように、ファイルを完全に取得できる十分なディスク領域をプロビジョニングすることが重要です。

目的に応じて2つの状況が考えられます。

- Dgraphクラッシュをトラブルシューティングする場合は、一連のコア・ファイル全体を取得できる十分なディスク領域をプロビジョニングします。この場合、ファイルは保存されますが、潜在的にディスクが満杯になる可能性があります。
- ディスクが満杯になることを回避する場合は、これらのファイル・サイズをオペレーティング・システム・レベルで制限できます。この場合、大規模なDgraphアプリケーションでは、コア・ファイルの一部のみがディスクに保存されます。これによりデバッグでの有用性が制限される可能性があります。

関連リンク

WindowsでのDgraphクラッシュ・ダンプ・ファイルの管理 81 ページ
Windowsでは、デフォルトの場合、すべてのDgraphクラッシュ・ダンプ・ファイルはディスクに保存されます。 (MDEX Engineは、Microsoft `DbgHelp`ライブラリから`MiniDump`機能を使用します。)

LinuxおよびSolarisでのDgraphコア・ダンプ・ファイルの管理 82 ページ
Dgraphコア・ダンプ・ファイルには、`ulimit -c unlimited`設定を使用することをお薦めします。無制限のコア・ファイルには、メモリー内のすべてのDgraphデータ (DgraphのRSS) が格納されます。

WindowsでのDgraphクラッシュ・ダンプ・ファイルの管理

Windowsでは、デフォルトの場合、すべてのDgraphクラッシュ・ダンプ・ファイルはディスクに保存されます。 (MDEX Engineは、Microsoft `DbgHelp`ライブラリから`MiniDump`機能を使用します。)

この推定に基づいてコア・ファイルを格納する十分なディスク領域をプロビジョニングします。

- これらのファイル・サイズの予測上限は、最大でMDEX Engineが使用する物理メモリーに索引サイズを加えたサイズと同じです。通常はそれより少ない領域がファイルに使用されます。

LinuxおよびSolarisでのDgraphコア・ダンプ・ファイルの管理

Dgraphコア・ダンプ・ファイルには、`ulimit -c unlimited`設定を使用することをお薦めします。無制限のコア・ファイルには、メモリー内のすべてのDgraphデータ (DgraphのRSS) が格納されます。

大規模なMDEXアプリケーションによりすべての使用可能なRAMが使用される可能性があるため、コア・ダンプ・ファイルも大きくなり、ディスク上の物理的なRAMサイズに索引サイズを加えた領域が使用されます。

👉 注: RSSの詳細は、*MDEX Engine Performance Tuning Guide*を参照してください。

この推定に基づいてコア・ファイルを格納する十分なディスク領域をプロビジョニングします。

- これらのファイル・サイズの予測上限は、最大で物理RAMサイズと等しくしてください。通常はそれより少ない領域がファイルに使用されます。

👉 注: `ulimit -c unlimited`を設定していない場合、一部のLinuxインストールでは`ulimit -c`のデフォルトが0に設定されるため、MDEX Engineのクラッシュが発生してコア・ファイルがディスクに書き込まれない可能性があります。

あるいは、コア・ファイルのサイズを制限するために、`ulimit -c <size>`コマンドを使用できます (お薦めの方法ではありません)。このようにして制限サイズを設定した場合、コア・ファイルはデバッグに使用できなくなりますが、コア・ファイルの存在によってDgraphがクラッシュしたことを確認します。クラッシュをトラブルシューティングできるようにするには、この設定を`ulimit -c unlimited`に変更し、コア・ファイル全体を取得している間にクラッシュを再現させます。同様に、クラッシュをトラブルシューティングできるようにするには、完全なコア・ファイルを取得している間にクラッシュを再現する必要があります。

第 4 部

実装のバックアップとリストア

第 8 章

Oracle Endecaアプリケーションのバックアップ

この項では、Endecaアプリケーション内でバックアップの対象になるディレクトリとファイルのリストを示します。アプリケーションのディレクトリ構造は、Deployment Templateを使用して最初に構築された環境に基づきます。使用する実装がここで説明する構造と異なる場合、ディレクトリとファイルのリストは、バックアップやリカバリの計画を作成する開始ポイントとして使用できます。

バックアップに必要なファイル

完全にバックアップする場合は、デプロイメント・テンプレート・アプリケーション・フォルダ全体のコピーを作成してアプリケーションをバックアップできます。選択的にバックアップする場合は、次の表内のディレクトリをバックアップします。

次のディレクトリ構造は、Deployment Templateで管理する環境に基づきます。パスにある変数の意味は次のとおりです。

- `[appdir]`は、`deploy`スクリプトの実行時にアプリケーションがデプロイされたパスを示します。たとえば、アプリケーション名が`MyApp`で、デプロイメント・ディレクトリを`C:\Endeca\apps`に指定した場合、アプリケーションのテンプレートは`C:\Endeca\apps\MyApp`にインストールされます。
- `ENDECA_CONF`は、Endeca HTTPサービスのワークスペース・ディレクトリのパスを示します。デフォルトのインストールの場合、この値は`C:\Endeca\PlatformServices\workspace` (Windowsの場合) または`endeca/PlatformServices/workspace` (UNIXの場合) です。
- `ENDECA_TOOLS_ROOT`は、Endeca Tools Serviceのワークスペース・ディレクトリのパスを示します。デフォルトのインストールの場合、この値は`C:\Endeca\Workbench\workspace` (Windowsの場合) または`endeca/Workbench/workspace` (UNIXの場合) です。

製品	ディレクトリ	内容	注釈
Endecaアプリケーション	`[appdir]\config`	構成ファイルおよびパイプライン・ファイル	
	`[appdir]\control`	スクリプト	
	`[appdir]\logs`	ログ・ファイル	最初にバックアップをするときに、この

			ディレクトリをコピーする必要があります。その後、このディレクトリをバックアップするかどうかはオプションです。
	[appdir]\data\dgidx_output	初期索引	
	[appdir]\data\partials\cumulative_partials	部分更新ファイル	これらのファイルを初期索引に適用する必要があります。
	[appdir]\data\state	状態ファイル	これらのファイルには、アプリケーションのディメンション値IDが含まれます。アプリケーションによっては、更新後に、次の更新までIDを保持することが重要な場合があります。
	[appdir]\test_data	テスト・データ・ファイル	これらのファイルはオプションです。
プラットフォーム・サービス	ENDECA_CONF\conf		このディレクトリには、サーバーのコンテキスト・ファイルおよび構成ファイルが格納されます。
	ENDECA_CONF\etc		
	ENDECA_CONF\reports		カスタム・レポート・ディレクトリがAppConfig.xmlファイルのwebstudio-report-dirパラメータに指定されている場合は、このディレクトリをバックアップする必要があります。
Workbench	ENDECA_TOOLS_ROOT\conf	Workbenchの拡張機能およびWorkbenchのユーザー定義が含まれます。	

	ENDECA_TOOLS_ROOT\state\emanager		

注: 索引のバックアップの詳細は、*Oracle Endeca MDEX Engine Partial Updates Guide*および*Oracle Endeca MDEX Engine Migration Guide*を参照してください。

CAS構成のバックアップ

CAS構成のバックアップは、ファイルのコピーおよびアーカイブを実行する場合と方法が異なります。

CAS構成を抽出するには、*CAS Developer's Guide*の*コンテンツ取得の管理*に関する項の説明に従って、特別なコマンドを実行する必要があります。CAS環境でバックアップが必要なコンポーネントは次のとおりです。

- クロール構成
- クロール・データ
- レコード・ストア構成
- レコード・ストア・データ
- カスタムWebクローラ

バックアップしないファイル

次の表に、リカバリ用にバックアップする必要のないファイルを示します。

ファイル・タイプ	ファイル	注釈
アーカイブ	タイムスタンプ付きのアーカイブ・ファイル (`forge_output`、`dgidx_output`、`logs`など) はオプションです。	

処理ファイル	`[appdir]/data/complete_data_config`
	`[appdir]/data/forge_output`
	`[appdir]/data/incoming`
	`[appdir]/data/partials`
	`[appdir]/data/processing`
	`[appdir]/data/temp`
	`[appdir]/data/web_studio`
		次のファイルはバックアップする必要があります。
		`data/partials/cumulative_partials`。

Dgraphインスタンス	`[appdir]/data/dgraphs`	

主要環境とリカバリ環境が異なる場合

特定の状況では、リカバリ環境のサーバー数、オペレーティング・システムおよびネットワーク・アーキテクチャが主要本番環境とは異なる場合があります。

Deployment Templateおよび次の特定の手順を使用することで、ある環境で実行中のEndecaアプリケーションを、特性が異なる他の環境に円滑に複製できます。

異種環境間でEndecaアプリケーションを複製するためのテクニックについては、このガイドの第3章で説明しています。

カスタマイズしたファイルのバックアップ

コマンドライン・スクリプトやライブラリ・ファイルなど、カスタマイズしたアプリケーション用に作成したファイルは、管理者が[appdir]に格納することをお薦めします。一方、このプラクティスに従わないことを選択した場合は、これらのファイルの格納場所を追跡し、ファイルの場所をバックアップ・プロセスに含める必要があります。

オプションの1つは、これらのファイルを収集してそのコピーを[appdir]に格納するスクリプトを作成することです。このスクリプトをバックアップの直前に実行することで、カスタマイズしたファイルのコピーがアプリケーション・ファイルとともに確実にバックアップされます。

このガイドの第3章に記載したサンプル・スクリプトのcollect-app.batを利用できます。使用するEndeca環境にあわせて同様のスクリプトを記述できます。

第 9 章

Endecaアプリケーションのバックアップとリストア

この項では、様々な種類のEndecaアプリケーション・データをバックアップおよびリストアする方法について説明します。

Endecaアプリケーションのバックアップとリストアについて

バックアップ・プロセスによって、Experience Managerのコンテンツと検索構成、ユーザー情報、権限データなど、アプリケーションのスナップショットを作成できます。

このプロセスに、アプリケーションのプロビジョニング情報は含まれません。

バックアップを行うために、Endecaアプリケーションは次の3つの部分で構成されています。

- アプリケーション構成 - 特定のアプリケーションのEndeca構成リポジトリに格納された構成 (Experience Managerのコンテンツを含む)。
- アプリケーションの状態 - `webstudiostore`および`emanager`ディレクトリには、アプリケーションの状態に関する情報 (ユーザーと権限の設定、プレビュー・アプリケーションの設定、コンテンツXML、リソース・メタデータなど) が含まれています。
- Workbench構成ファイル -Oracle Endeca Workbenchインストールの動作をカスタマイズするXMLおよびプロパティ・ファイル。

アプリケーションのバックアップは、アプリケーションの状態および構成が対象になります。Workbench構成をバックアップすることによって、Workbenchユーザーは、アプリケーションを移行またはリストアしたときに一貫性のある処理を実行できます。

Endeca構成リポジトリからのアプリケーションのバックアップ

Deployment Templateに付属の`export_site`スクリプトを使用して、Endeca構成リポジトリのアプリケーション構成をバックアップします。

このスクリプトにより、アプリケーション構成がEndeca構成リポジトリに再インポートできる形式でエクスポートされます。また、`AppConfig.xml`の構成に基づいて、現在のアプリケーションのEndeca構成リポジトリ・インスタンスに接続されます。

👉 重要: データの一貫性を確保するために、バックアップ・プロセスの間にベースライン更新または部分更新が実行されていないことを確認してください。

Endeca構成リポジトリのアプリケーション構成をバックアップする手順は、次のとおりです。

1. デプロイ済アプリケーションの`control`ディレクトリに移動します (例: `C:\Endeca\apps\discover\control`)。
2. 次の例に示すように、`export_site`スクリプトを実行して、エクスポート・ファイルに対するオプションの名前を渡します。

   Windowsの場合:

   ```
   export_site.bat ..\ECR-backups\20121221.xml
   ```

   UNIXの場合:

   ```
   ./export_site.sh ../ECR-backups/20111221.xml
   ```

   ファイル名を指定しない場合は、`<site-name>-DD-MM-YYYY.xml`というパターンに基づいた名前のファイルが作業ディレクトリ内にデフォルト設定されます。

アプリケーションの状態のバックアップ

`webstudiostore`および`emanager`ディレクトリには、アプリケーションの状態に関する情報 (ユーザーと権限の設定、プレビュー・アプリケーションの設定、コンテンツXML、リソース・メタデータなど) が含まれています。

アプリケーションの状態のディレクトリをバックアップする手順は、次のとおりです。

1. Endeca Tools Serviceを停止します。
2. `webstudiostore`ディレクトリとそのサブディレクトリを、`%ENDECA_TOOLS_CONF%\state\` (Windowsの場合) または`$ENDECA_TOOLS_CONF/state/` (UNIXの場合) から別の場所にコピーします。

   このディレクトリには、ユーザーと権限、およびプレビュー・アプリケーションの設定などの情報が含まれています。
3. `emanager`ディレクトリとそのサブディレクトリを、`%ENDECA_TOOLS_CONF%\state\` (Windowsの場合) または`$ENDECA_TOOLS_CONF/state/` (UNIXの場合) から別の場所にコピーします。

   このディレクトリには、リソース・メタデータ、状態の情報およびコンテンツXMLが含まれています。
4. Endeca Tools Serviceを起動します。

Workbench構成ファイルのバックアップ

Workbenchでは、`%ENDECA_TOOLS_CONF%\conf` (Windowsの場合) または`$ENDECA_TOOLS_CONF/conf` (UNIXの場合) にある複数の構成ファイルを使用して、Workbenchの様々な側面の動作をカスタマイズします。

これらのファイルには、Workbench構成、ユーザー認証構成、およびWorkbenchのメニューと拡張機能の定義が格納されています。次のファイルのいずれかをデフォルトの状態から手動で変更した場合は、それらのファイルをバックアップの場所にコピーする必要があります。

ファイル名	説明
`Login.conf`	LDAPを使用したユーザー認証用の構成
`webstudio.properties`	Workbenchのその他の構成パラメータ
`webstudio.log4j.properties`	Workbenchのシステム・ログおよび監査ログ用の構成
`ws-extensions.xml`	Workbench拡張機能の定義
`ws-mainMenu.xml`	Workbenchのナビゲーション・メニューおよび起動ページの定義

Endeca構成リポジトリへのサイトのバックアップのリストア

Deployment Templateに付属の`import_site`スクリプトを使用して、アプリケーション構成のバックアップをEndeca構成リポジトリのインスタンスにリストアできます。

このスクリプトにより、前のエクスポートで作成されたファイルが取得されて、その内容がEndeca構成リポジトリにインポートされます。また、`AppConfig.xml`の構成に基づいて、現在のアプリケーションのEndeca構成リポジトリ・インスタンスに接続されます。

👉 重要: データの一貫性を確保するために、バックアップ・プロセスの間にベースライン更新または部分更新が実行されていないことを確認してください。

Endeca構成リポジトリのバックアップのリストアは、同じリリース・バージョンのインスタンス間でのみサポートされています。

Endeca構成リポジトリにアプリケーション構成のバックアップをリストアする手順は、次のとおりです。

1. デプロイ済アプリケーションの`control`ディレクトリに移動します (例: `C:\Endeca\apps\discover\control`)。
2. 次の例に示すように、`import_site`スクリプトを実行して、バックアップのファイル名を渡します。

   Windowsの場合:

   ```
   import_site.bat discover-21-12-2012.xml
   ```

   UNIXの場合:

   ```
   ./import_site.sh discover-21-12-2012.xml
   ```

3. load_baseline_test_dataスクリプトを実行します。
4. baseline_updateスクリプトを実行して、更新された情報をMDEX Engineに発行します。
5. set_templatesスクリプトを実行して、更新されたテンプレートをリポジトリに送信します。
6. promote_contentスクリプトを実行します。

アプリケーションの状態のバックアップのリストア

アプリケーションの状態ディレクトリのwebstudiostoreとemanagerのバックアップを、同じバージョン以降のインストールにリストアできます。

アプリケーションの状態ディレクトリのバックアップをリストアする手順は、次のとおりです。

1. Endeca Tools Serviceを停止します。
2. %ENDECA_TOOLS_CONF%\state\ (Windowsの場合) または$ENDECA_TOOLS_CONF/state/ (UNIXの場合) から、webstudiostoreディレクトリを削除します。
3. webstudiostoreディレクトリ (すべてのサブディレクトリを含む) のバックアップを、%ENDECA_TOOLS_CONF%\state (Windowsの場合) または$ENDECA_TOOLS_CONF/state/ (UNIXの場合) から別の場所にコピーします。
4. %ENDECA_TOOLS_CONF%\state\ (Windowsの場合) または$ENDECA_TOOLS_CONF/state/ (UNIXの場合) から、emanagerディレクトリを削除します。
5. emanagerディレクトリ (すべてのサブディレクトリを含む) のバックアップを、%ENDECA_TOOLS_CONF%\state\ (Windowsの場合) または$ENDECA_TOOLS_CONF/state/ (UNIXの場合) から別の場所にコピーします。
6. Endeca Tools Serviceを起動します。

Oracle Endeca Workbench構成ファイルのバックアップのリストア

構成ファイルのバックアップをリストアできるのは通常、正確に同じバージョンのWorkbenchに対してのみです (例: 3.1.0バージョンから他の3.1.xバージョンではなく、3.1.0バージョンから3.1.0バージョン)。

インストールをアップグレードすると、構成の変更が発生し、構成ファイルを手動でマージすることが必要になる場合があります。Oracle Endeca Workbench構成ファイルに対する変更の詳細は、*MDEX Engine Migration Guide*を参照してください。

Workbench構成ファイルのバックアップをリストアする手順は、次のとおりです。

1. Endeca Tools Serviceを停止します。
2. バックアップ・ファイルを%ENDECA_TOOLS_CONF%\conf (Windowsの場合) または$ENDECA_TOOLS_CONF/conf (UNIXの場合) にコピーします。
3. Endeca Tools Serviceを起動します。

付録 A

管理および構成の操作

MDEX Engineは、単純なURLを介してアクセスできる多数の管理および構成の操作をサポートしています。これらの操作およびそのロギング変数を使用して、MDEX Engineの動作をシステム内で手際よく制御できます。

管理および構成の操作について

管理および構成の操作を使用すると、実行中のDgraphを停止せずにDgraph統計をチェックしたり、診断フラグを有効または無効にできます。Dgraphの停止および再起動も操作できます。この項では、Dgraphで公開されているURLのリストを示し、各URLの機能を説明し、それらのURLの構文を定義します。

管理および構成操作の構文

次のリストで、<host>はMDEX Engineのホスト名またはIPアドレス、<port>はMDEX Engineがリスニングするポートを示します。これらのURLに対する問合せは、他のリクエストと同様にMDEX Engineのリクエスト・キューで処理されるため、先入れ先出し方式で処理されます。また、他のリクエストと同様にMDEX Engineリクエスト・ログにレポートされます。

管理操作の構文は次のとおりです。

```
http://<host>:<port>/admin?op=<supported-operation>
```

構成操作の構文は次のとおりです。

```
http://<host>:<port>/config?op=<supported-operation>
```

👉 注: HTTPSモードを使用している場合は、URLにhttpsを使用してください。

管理操作のリスト

このトピックのリストに記載されている管理(admin)操作を使用して、システム内からMDEX Engineの動作を制御できます。

MDEX Engineでは、次のadmin操作を認識します。

admin操作	説明
`/admin?op=help`	すべてのadmin操作の使用方法ページを返します。
`/admin?op=ping`	MDEX Engineの稼働をチェックし、簡単なメッセージを返します。
`/admin?op=flush`	MDEX Engineが動的キャッシュをフラッシュする時期を指定します。
`/admin?op=exit`	MDEX Engineの実行を停止します。
`/admin?op=restart`	MDEX Engineを再起動します。
`/admin?op=audit`	「MDEX Engine Auditing」ページを返します。
`/admin?op=auditreset`	「MDEX Engine Auditing」ページをリセットします。
`/admin?op=stats`	「MDEX Engine Statistics」ページを返します。
`/admin?op=statsreset`	「MDEX Engine Statistics」ページをリセットします。
`/admin?op=logroll`	問合せログ・ロールを強制的に適用し、`stdout`が再マッピングされる二次的作用が生じます。
`/admin?op=update`	部分更新ファイルをMDEX Engineに適用します。
`/admin?op=updateaspell`	MDEX Engineを停止および再起動せずに、データ・コーパスからスペル修正用のaspellディクショナリを再作成します。
`/admin?op=updatehistory`	MDEX Engineが最近処理した更新ファイルのリストを表示します。
`/admin?op=reload-services`	アプリケーションのメインおよびライブラリ・モジュールを再ロードするWebサービス操作です。

関連リンク

help 95 ページ
`/admin?op=help`は、すべての管理操作の使用方法ページを返します。

ping 95 ページ
`/admin?op=ping`は、MDEX Engineの稼働をチェックし、簡単なメッセージを返します。

flush 95 ページ
`/admin?op=flush`は、Dgraphキャッシュをフラッシュします。

exit 96 ページ
`/admin?op=exit`は、MDEX Engineの実行を停止します。

restart 96 ページ
`/admin?op=restart`は、Dgraphを再起動します。

audit 96 ページ
`/admin?op=audit`は、「MDEX Engine Auditing」ページを返します。

auditreset 97 ページ
`/admin?op=auditreset`は、「MDEX Engine Auditing」ページをリセットします。

help

`/admin?op=help`は、すべての管理操作の使用方法ページを返します。

ping

`/admin?op=ping`は、MDEX Engineの稼働をチェックし、簡単なメッセージを返します。

「MDEX Engine Statistics」ページを表示して、MDEX Engineが実行され、問合せを受け入れているかどうかを確認できます(ただし、多少のオーバーヘッドがあります)。`ping`コマンドは、Dgraphの可用性を速やかに確認できる手段です。

`ping`コマンドは、次のようなMDEX Engineおよび現在の日時のリストを示す軽量なページを返します。

```
dgraph example.endeca.com:8000 responding at Wed Oct 25 15:35:27 2009
```

この操作を使用して、MDEX Engineの可用性をロード・バランサの可用性チェックとして監視できます。

flush

`/admin?op=flush`は、Dgraphキャッシュをフラッシュします。

`flush`操作は、Dgraphキャッシュからすべてのエントリを消去します。次のメッセージが返されます。

```
flushing cache...
```

exit

`/admin?op=exit`は、MDEX Engineの実行を停止します。

`exit`操作は、未処理のトランザクション (問合せ) すべてが完了するまでDgraphを保留します。すべてのトランザクションが完了すると、Dgraphはシャットダウンします。プロセスを手動で停止する方法とは対照的に、すべてのトランザクションが正常に完了して円滑に終了するため、この方法でDgraphをシャットダウンすることをお薦めします。

次のような出力が表示されます。

```
Dgraph admin, OK
Dgraph shutting down at Wed May 20 11:23:08 2009
```

Oracle Endeca Application Controller (EAC) と非推奨の管理インタプリタの両方とも、MDEX Engineを停止する際は`exit`操作をバックウラウンドで使用します。ただし、問題のMDEX EngineがEACまたは管理インタプリタによって当初起動された場合、この操作を手動で発行すると、EACまたは管理インタプリタは結果として生じるMDEX Engineのシャットダウンを失敗と解釈し、MDEX Engineを即時に再起動するため、MDEX Engineのシャットダウンは永久に失敗します。EACまたは管理インタプリタによって起動されたMDEX Engineは、当初起動した管理フレームワークを介してシャットダウンする必要があります。

restart

`/admin?op=restart`は、Dgraphを再起動します。

`restart`操作は、シャットダウンの後にDgraphが再起動すること以外は、`exit`操作と同様に機能します。プロセスを手動で停止および起動する方法とは対照的に、再起動の前にすべてのトランザクションが正常に完了して円滑に終了するため、この方法でDgraphを再起動することをお薦めします。

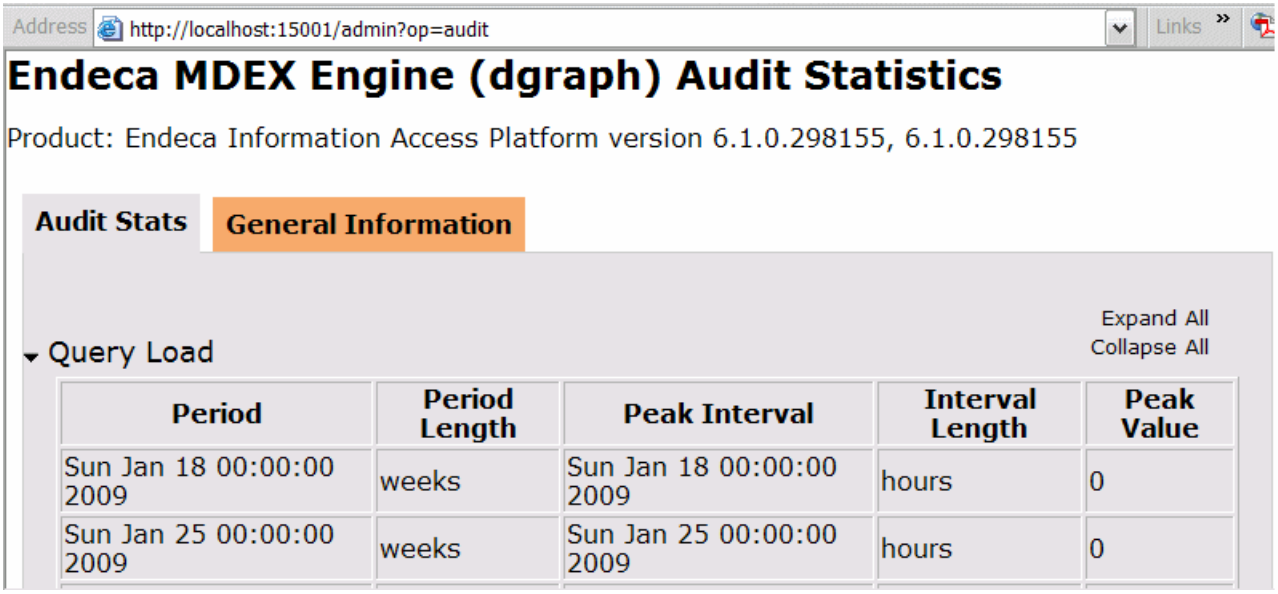
`restart`操作では、次のような出力が返されます。

```
Dgraph admin, OK
Dgraph restarting at Wed May 20 11:25:19 2009
```

audit

`/admin?op=audit`は、「MDEX Engine Auditing」ページを返します。

「MDEX Engine Auditing」ページには、Dgraphメトリックの集計が時間とともに表示されます。進行中の使用状況統計を追跡するXMLレポートの出力が用意されています。これらの統計は、プロセスを再起動しても維持されます。このデータは、ライセンス条件に準拠しているかどうかを検証する際に使用可能で、製品の使用状況を追跡する場合も役に立ちます。



注: 「MDEX Engine Auditing」ページの詳細は、*Performance Tuning Guide*を参照してください。

auditreset

`/admin?op=auditreset`は、「MDEX Engine Auditing」ページをリセットします。

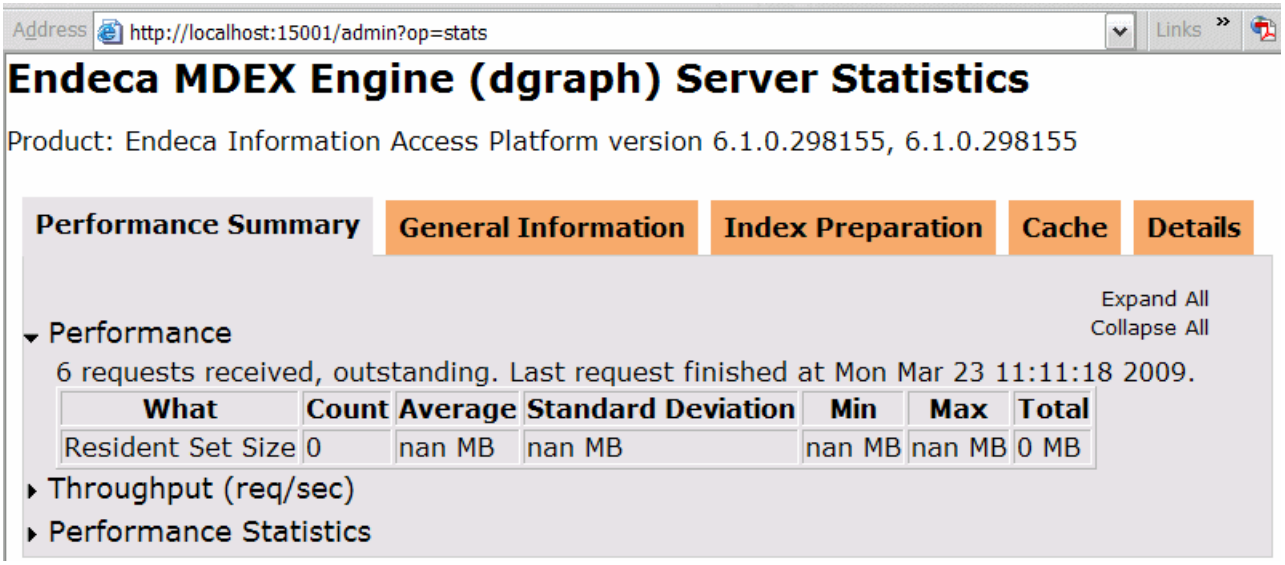
次のメッセージが返されます。

```
resetting auditing stats...
```

stats

`/admin?op=stats`は、「MDEX Engine Statistics」ページを返します。

「MDEX Engine Statistics」ページは、Dgraphの動作について詳細な分析を提供し、Endeca実装の構成およびパフォーマンスに関する有用なソースです。起動時間、最終索引付け時間、索引データ・パスなどの情報があります。このページで、チューニングおよびロード・バランシングの対応に集中できます。このページを検査することで、Dgraphが時間を費やしている場所を確認できます。「Details」タブの「Hotspots」セクションで合計が最も高い機能を識別することにより、チューニングを開始します。



注: 「MDEX Engine Statistics」ページの詳細は、*Performance Tuning Guide*を参照してください。

statsreset

`/admin?op=statsreset`は、「MDEX Engine Statistics」ページをリセットします。

`statsreset`操作では、次のメッセージが返されます。

```
resetting server stats...
```

logroll

`/admin?op=logroll`は、問合せログ・ロールを強制的に適用し、`stdout`が再マッピングされる二次的作用が生じます。

`logroll`コマンドでは、次のようなメッセージが返されます。

```
rolling log... Successfully remapped stdout/stderr to specified
path "C:\Endeca\apps\JanWine\logs\dgraphs\Dgraph2\Dgraph2.log".
Successfully rolled log file.
```

update

`/admin?op=update`は、部分更新ファイルをDgraphに適用します。

この操作は、Dgraphに対してその更新ディレクトリの部分更新ファイルを処理するように指示します。部分更新の詳細は、*Partial Updates Guide*を参照してください。

URL更新コマンドを受信すると、Dgraphはデフォルトで次の一連の操作を実行します。

1. 更新の処理と同時に問合せの処理を続行します。
2. 更新ディレクトリをチェックし、アップロードされていない部分更新をすべてアップロードします。
3. 更新ファイルを処理し、そのファイルを削除します。

単一ファイルからの更新の処理

場合によっては、Dgraphに単一のファイルを示すことで部分更新を実行する必要があります。この場合は、`admin?op=update&updatefile=`*`filename`* オプション (`filename`は、更新ディレクトリにある更新ファイルの名前) を実行します。

複数のファイルがある場合は、このコマンドを再実行します。MDEX Engineによって部分更新の結果が正常に適用されると、更新ファイルは削除されます。

updateaspell

`admin?op=updateaspell`管理操作を使用して、MDEX Engineを停止および再起動せずに、データ・コーパスからスペル修正用のaspellディクショナリを再作成できます。

`admin?op=updateaspell`操作は、次のアクションを実行します。

- すべての用語のテキスト検索索引をクロールします。
- `aspell`ワード・リストのテキスト・バージョンをコンパイルします。
- このワード・リストを`aspell`に必要なバイナリ形式に変換します。
- Dgraphは、先行する既存の問合せをすべて処理し、受信した問合せの処理を一時的に停止します。
- 以前のバイナリ形式のワード・リストを、更新したバイナリ形式のワード・リストで置換します。
- `aspell`スペル・ディクショナリを再ロードします。
- Dgraphが、キューで待機中の問合せの処理を再開します。

更新された設定を適用するためにDgraphを再起動する必要はありません。

一度に処理できる`admin?op=updateaspell`操作は1つのみです。

`admin?op=updateaspell`操作では、次のような出力がDgraphエラー・ログに返されます。

```
...
aspell update ran successfully.
...
```

👉 注: `-v`フラグを使用してDgraphを起動した場合は、次のような行も出力に表示されます。

```
Time taken for updateaspell, including wait time on any
previous updateaspell, was 290.378174 ms.
```

updatehistory

`/admin?op=updatehistory`は、Dgraphが起動してから処理した更新ファイルのリストを表示します。

`updatehistory`操作では、次のような出力が返されます。

```
Endeca Dgraph Server update directory history contents

Checking for update directory for directory "..\data\partition0\dgraph_in¬
put\updates\"

Files in update directory history
```

```
"..\data\partition0\dgraph_input\updates\\wine-
sgmt0.records.xml_2009.05.19.10.31.25"
"..\data\partition0\dgraph_input\updates\\wine-
sgmt0.records.xml_2009.05.19.10.30.08"
"..\data\partition0\dgraph_input\updates\\wine-
sgmt0.records.xml_2009.05.19.10.32.13"
```

reload-services

`/admin?op=reload-services`は、アプリケーションのメインおよびライブラリ・モジュールを再ロードするWebサービス操作です。

`admin?op=reload-services`操作により、Dgraphは先行する既存の問合せをすべて処理し、他の問合せの処理を一時的に停止して、`admin?op=reload-services`の処理を開始します。この操作の処理が終了すると、Dgraphは、一時的に後回しにしたリクエストをキューに戻して問合せの処理を再開します。

注: `admin?op=reload-services`は、作成してコンパイルが必要なXQueryモジュールの数によっては、時間がかかる操作になる可能性があります。

構成操作のリスト

このトピックのリストに記載されている構成(config)操作を使用して、システム内からMDEX Engineの構成およびロギングに関する情報を変更できます。

Dgraphでは、次のconfig操作を認識します。

config操作	説明
`/config?op=help`	すべてのconfig操作の使用方法ページを返します。
`/config?op=update`	更新されたMDEX Engine構成ファイルをロードして適用します。
`/config?op=log-enable`	指定された1つ以上の変数に対して冗長ロギングを有効にします。
`/config?op=log-disable`	指定された1つ以上の変数に対して冗長ロギングを無効にします。
`/config?op=log-status`	冗長ロギング・ステータスを返します。

関連リンク

help 101 ページ
`/config?op=help`は、すべてのconfig操作の使用方法ページを返します。

update 101 ページ
`config?op=update`操作は、更新されたDgraph構成ファイル(シソーラス・エントリや動的ビジネス・ルールなどの項目に関する情報を含む)をロードします。更新された設定を適用するためにDgraphを再起動する必要はありません。

help

`/config?op=help`は、すべてのconfig操作の使用方法ページを返します。

update

`config?op=update`操作は、更新されたDgraph構成ファイル(シソーラス・エントリや動的ビジネス・ルールなどの項目に関する情報を含む)をロードします。更新された設定を適用するためにDgraphを再起動する必要はありません。

`config?op=update`操作により、Dgraphは先行する既存の問合せをすべて排出し、その他の問合せ処理を一時的に停止して、`config?op=update`の処理を開始します。更新されたすべての構成ファイルがロードされ、適用されます。これらの操作の処理が終了すると、Dgraphは、一時的に後回しにしたリクエストをキューに戻して問合せの処理を再開します。

一度に処理できる`config?op=update`操作は1つのみです。開発およびデバッグ時に、MDEX Engineを再起動する割込みを回避して構成の変更をすばやくロードする必要がある場合は、このコマンドが便利です。

注: Dgraphで更新を処理する必要がある構成ファイルの数によっては、`config?op=update`操作は時間がかかる可能性があります。

更新操作では、次のような出力が返されます。

```
Processing configuration file
"<full path to XML configuration file>"
Successfully processed configuration file
"<full path to XML configuration file>"
Finished processing config updates.
```

MDEX Engineのロギング変数について

config操作ではロギング変数を使用できます。これにより、Dgraphの停止と再起動または構成の更新を実行しなくても、予期しないアプリケーションの動作またはパフォーマンスの問題を診断するのに役立つDgraph処理に関する詳細情報を取得できます。

Dgraphコマンドラインで`--v`フラグを指定して通常の冗長ロギングを指定することもできますが、有効にするにはDgraphを再起動する必要があります。

関連リンク

ロギング変数の操作構文 102 ページ
MDEX Engineロギング変数は、`/config?op=log-enable&name=<variable-name>`および`/config?op=log-disable&name=<variable-name>` の操作を使用して切り替えます。

サポートされているロギング変数のリスト 102 ページ
次の表では、指定した機能に対するロギング冗長性を切り替えるために関連config操作で使用できる、サポートされているロギング変数について説明します。

ロギング変数の操作構文

MDEX Engineロギング変数は、`/config?op=log-enable&name=<variable-name>` および`/config?op=log-disable&name=<variable-name>` の操作を使用して切り替えます。

1つのリクエストに複数のロギング変数を指定できます。認識できないロギング変数には警告が生成されます。

たとえば、次の操作を考えてみます。

```
/config?op=log-enable&name=merchverbose
```

この操作と同時に、動的ビジネス・ルール機能の冗長ロギングがオンになります。

```
config?op=log-enable&name=textsearchrelrankverbose&name=textsearchspellver¬
bose
```

テキスト検索関連ランキング機能とスペル機能の両方の冗長ロギングがオンになります。

一方、次の操作を考えてみます。

```
config?op=log-enable&name=allmylogs
```

`unsupported logging setting`メッセージが返されます。

さらに、次の操作がサポートされます。

- `/config?op=log-status`は、すべてのロギング変数とその値 (trueまたはfalse) のリストを返します。
- `all`という特別な名前を`/config?op=log-enable`または`/config?op=log-disable`で使用して、すべてのロギング変数を設定できます。

サポートされているロギング変数のリスト

次の表では、指定した機能に対するロギング冗長性を切り替えるために関連config操作で使用できる、サポートされているロギング変数について説明します。

ロギング変数名は大/小文字が区別されません。

変数	説明
`verbose`	冗長モードを有効にします。
`requestverbose`	各リクエストに関する情報をstdoutに出力します。
`updateverbose`	更新処理時に冗長メッセージを表示します。
`recordfilterperfverbose`	レコード・フィルタ・パフォーマンスに関する冗長情報を有効にします。
`merchverbose`	マーチャンダイズ・ルール処理時に冗長デバッグ・メッセージを有効にします。
`textsearchrelrankverbose`	検索問合せ処理時に関連ランキングに関する冗長情報を有効にします。
`textsearchspellverbose`	スペル修正機能に対する冗長出力を有効にします。

変数	説明
dgraphperfverbose	コアDgraphナビゲーション計算時の冗長パフォーマンス・デバッグ・メッセージを有効にします。
dgraphrefinementgroupverbose	絞込みの冗長/デバッグ・メッセージを有効にします。

関連リンク

log-enable 103 ページ
: log-enable操作は、冗長ロギングをオンに切り替えます。

log-disable 103 ページ
: log-disable操作は、冗長ロギングをオフに切り替えます。

log-status 103 ページ
: log-status操作は、すべてのロギング変数とその値 (trueまたはfalse) のリストを返します。

log-enable

log-enable操作は、冗長ロギングをオンに切り替えます。

1つのリクエストに複数のロギング変数を指定できます。認識できないロギング変数には警告が生成されます。

たとえば、次の操作を考えてみます。

```
/config?op=log-enable&name=merchverbose
```

この操作と同時に、動的ビジネス・ルール機能の冗長ロギングがオンになります。

```
config?op=log-enable&name=textsearchrelrankverbose&name=textsearchspellver¬
bose
```

テキスト検索関連ランキング機能とスペル機能の両方の冗長ロギングがオンになります。

一方、次の操作を考えてみます。

```
config?op=log-enable&name=allmylogs
```

“unsupported logging setting”メッセージが返されます。

log-disable

log-disable操作は、冗長ロギングをオフに切り替えます。

引数のない/config?op=log-disableは、log-statusと同じ出力を返します。

log-status

log-status操作は、すべてのロギング変数とその値 (trueまたはfalse) のリストを返します。

たとえば、任意の機能に対して冗長ロギングを有効にしていない場合は、次のようなメッセージが表示されます。

```
Logging settings:

verbose - FALSE
requestverbose - FALSE
```

```
updateverbose - FALSE
recordfilterperfverbose - FALSE
merchverbose - FALSE
textsearchrelrankverbose - FALSE
textsearchspellverbose - FALSE
dgraphperfverbose - FALSE
dgraphrefinementgroupverbose - FALSE
```

付録 B

Endecaフラグ・リファレンス

この付録では、DgidxおよびDgraphプログラムで使用されるフラグ (オプション) について説明します。Forgeフラグについては、*Platform Services Forge Guide*を参照してください。

Dgidxフラグ

Dgidxプログラムでは、Forgeによって準備されたタグ付け済Endecaレコードが索引付けされ、Endeca MDEX Engine専用の索引が作成されます。

Dgidxの使用方法は次のとおりです。

```
dgidx [-qv] [--flags] <data export file> <output db_prefix>
```

<db_prefix>には、ディレクトリへのパスであり、Endecaアプリケーションのファイルに使用される接頭辞を指定します。

Dgidxでは、次のフラグがサポートされています。

フラグ	説明
-q	抑制モード。
-v	冗長モード。
--compoundDimSearch	アプリケーションに対する複合ディメンション検索を可能にします。このオプションを使用すると、索引付け時間が増加します。ただし、このオプションが索引時に有効でない場合、複合ディメンションの結果 (複数ディメンション値の結果) はMDEX Engineによって返されません。
--cov	ディメンションおよびプロパティに対するカバレッジ統計を計算し、レポートします。
--diacritic-folding	テキストの索引付け時に文字アクセントを無視します。発音区別符号付きの文字がASCIIの相当文字にどのようにマップされるかの詳細は、*MDEX Engine Advanced Development Guide*を参照してください。
--help	ヘルプ・メッセージを出力して終了します。

フラグ	説明
`--lang <lang-id>-u-<collation>`	`<lang-id>`で指定された言語でドキュメントを索引付けし、引数の`<collation>`部分でオプションの照合順序も指定します。未指定の場合、`<lang-id>`のデフォルトはen (米国英語) です。国際言語の使用方法の詳細は、*MDEX Engine Advanced Development Guide*を参照してください。
`--nostrictattrs`	厳密な属性チェックを無効にします。XML構成ファイルおよびDeveloper Studioのプロパティ・ビューでプロパティ (またはPROP_REF要素) が定義されていないプロパティについて、レコードでプロパティ値を保持できます。
`--numbins <num>`	Dgidxで読み取るレコードの数を制限します。
`--out <stdout/stderr file>`	stdout/stderrの再マップ先のファイル・パスを指定します (デフォルトでは、処理用のデフォルトのstdout/stderrが使用されます)。

`--sort <spec>`	データ・セットのデフォルトのソート指定を指定します。*<spec>*の書式は次のとおりです (引用符を含みます)。
	`"key|dir"`
	*key*はソート対象のプロパティまたはディメンションの名前、*dir*は`asc` (昇順) または`desc` (降順) のいずれかです (指定しない場合、順序は昇順です)。
	*key*には、次の例のようにジオコード・プロパティも指定できます。
	`"Location(43,73)|desc"`
	次の書式で複数のソート・キーを指定できます。
	`"key_1[|dir_1]||key_2[|dir_2]||...||key_n[|dir_n]"`
	複数のソート・キーを指定すると、レコードは最初のソート・キーでソートされ、キーが同じレコードは2番目のソート・キーで解決され、さらにキーが同じレコードは3番目のソート・キーで解決され、以下同様に続きます。
	Oracle Endeca Application Controller (EAC) を使用して環境を制御している場合は、`--sort`フラグから引用符を省略する必要があることに注意してください。かわりに次の構文を使用します。
	`--sort key_1|dir_1||key_2|dir_2||...||key_n|dir_n`

`--spellmode <mode>`	アプリケーションのスペル修正モードを指定します。サポートされているモードは次のとおりです。 - default - aspell - espell - aspell_OR_espell - aspell_AND_espell

フラグ	説明
`--spellnum`	`espell`モジュールを有効にするスペル・モードでは、espellディクショナリにワード以外の語 (数字、記号など) が含まれます。デフォルトでは、このような語は含まれません。
`--stemming-updates <file>`	デフォルトのステミング・ディクショナリに適用するステミング更新のオプションのXMLファイルを指定します。XMLの例およびファイル名の要件は、*Oracle Endeca MDEX Engineアドバンスト開発ガイド*を参照してください。
`--threads <num>`	索引付け処理のマルチスレッド部分に使用するソート・スレッド数を指定します。デフォルトは1です。このフラグを指定しないか、または1を指定した場合は、1つのソート・スレッドが使用されます。指定した値が1より大きい場合は、指定した数のスレッドがデータのソートに使用されます。 デフォルトで、Dgidxはマルチスレッド・モードで実行されることに注意してください。`--threads`フラグで制御できるソート・スレッド数以外に、デフォルトでバックグラウンドで実行され、データのソートには使用されない追加の保守スレッドが使用される場合があります。 索引付けのパフォーマンスを向上させるには、ソート・スレッド数を増やすことをお薦めします。Endecaアプリケーションの索引付けに専用サーバーが使用されるデプロイメントでは、Dgidxソート処理に対して、サーバーで許可される数のスレッドを割り当ててください。 パフォーマンスを最大にするには、指定するソート・スレッド数は、サーバーのコア数と相関させる必要があります。ソートは索引付け処理の一部にすぎないため、*N*個のソート・スレッドを使用してもDgidxが *N*倍速くなるわけではありません。
`--version`	バージョン情報を出力して終了します。

Dgraphフラグ

Dgraphプログラムは、MDEX Engineを起動します。

MDEX Engineは、Dgidxによって準備された一連の索引を示す対象となる、Dgraphというプログラムを実行して起動します。Dgraphには、MDEX Engineを調整できる多数のオプションがあります。たとえば、スペル、キャッシングなどを微調整できます。

Dgraphの使用方法は次のとおりです。

```
dgraph [-?] [--flags] <db_prefix>
```

`<db_prefix>`には、ディレクトリへのパスであり、Endecaアプリケーションのファイルに使用される接頭辞を指定します。

フラグ	説明
?	ヘルプ・メッセージを出力して終了します。
-d	*MDEX Engine 6.4.0では非推奨*です。デバッグ・モードで起動します。
-v	冗長モード。各リクエストに関する情報をstdoutに出力します。
--ancestor_counts	ルート・ディメンション値に対する件数を計算します。フラグが省略されている場合は、ルート・ディメンション値ではなく、選択した使用可能な絞込みに対する絞込み件数のみが計算されます。
--back_compat <api-version>	下位互換性を有効にして、Dgraphが以前のバージョンのPresentation APIと通信できるようにします。現在サポートされているバージョンのPresentation APIに加え、以前のバージョンである6.3.x、6.2.x、6.1.xおよび6.0.xがサポートされています。したがって、*<api-version>*の値は、次のいずれかにする必要があります。 - 630 (6.3.0バージョンのPresentation APIの場合)。 - 620 (6.2.2および6.2.1バージョンのPresentation APIの場合)。 - 614 (6.1.5および6.1.4バージョンのPresentation APIの場合)。 - 601 (6.1.3から6.0.1のバージョンのPresentation APIの場合)。
--backlog-timeout <seconds>	処理のために読み取られ、キューに入れられた問合せの待機制限(秒) を指定します。これは、問合せが処理キュー内で待機可能な最大秒数であり、この秒数を超えると、Dgraphによってタイムアウト・メッセージが返されます。デフォルト値は60秒です。
--cmem <MB>	MDEX Engineのメイン・キャッシュの最大メモリー使用量をMBで指定します。--cmemを指定しない場合、デフォルト値は、64ビット・プラットフォームのDgraphインストールの場合で1024MB (1GB) です。
--config <path>	起動時に読み取る構成ファイルを指定します。構成ファイルには、コマンドラインで使用される同じ書式の引数が含まれている必要があります (つまり、改行を含めて空白が無視されます)。
--disable_fast_aspell	aspellスペル・モジュールに対する高速モードを無効にします。高速モードを無効にした場合、スペル修正のパフォーマンスは低下しますが、さらに問合せを修正できる場合があります。 高速モードが有効な場合は、aspellモジュールでスペル修正機能を使用しているアプリケーションの速度が大幅に向上します。デフォルトで高速モードが使用されます。
--disable_web_services	*MDEX Engine 6.3.0では非推奨*です。起動時のXQueryモジュールの自動ロードを抑制します。
--dym	全文検索問合せに対するDYM (Did You Mean?) の明示的な問合せスペル提案を有効にします。

フラグ	説明
`--dym_hthresh <thresh>`	DYM (もしかして) 提案を生成しない下限値を指定します。デフォルトは`20`です。
`--dym_nsug <count>`	問合せに対して戻すDYM (Did You Mean?) 問合せ提案の最大数を指定します。デフォルトは`1`です。
`--dym_sthresh <thresh>`	DYM (Did You Mean?) エンジンで使用されるワードのしきいスペル修正スコアを指定します。デフォルトは`175`です。
`--dynrank_consider_collapsed`	このフラグを使用して、中間の縮小可能ディメンション値を動的ランキングに対する候補として考慮するようにMDEX Engineに強制します。 このフラグは、縮小可能ディメンション値があるディメンションを動的にランク付けする際のMDEX Engineのデフォルト動作を変更します。デフォルト (このフラグを指定しない場合) で、MDEX Engineでは、動的ランキングに対してリーフ・ディメンション値のみが考慮され、中間のディメンション階層はすべて考慮されません。 デフォルトの動作では、階層ディメンションの中間レベル値 (ルート値とリーフ値を除くすべて) がDeveloper Studioで縮小可能として構成されていて、ディメンションが動的絞込みランキングを使用するようにも設定されている場合、ディメンションは縮小され、すべてのナビゲーション問合せに対してリーフ値のみが表示されます。中間レベルのディメンション値は、ナビゲーション状態で示されているリーフ値の数に関係なく表示されることはありません。 MDEX Engineで、動的ランキングに (縮小可能として構成されている) 中間ディメンション値が考慮されるようにする場合は、このフラグを使用します。
`--esampmin <num>`	絞込み計算時にサンプリングする最小レコード数を指定します。デフォルトは`0`です。調整の推奨: - ほとんどのアプリケーションの場合、値が大きくなると、動的絞込みランキングの品質が向上せずにパフォーマンスが低下します。 - 非常に大きい構造化されていないディメンジョンを含む一部のアプリケーションの場合、値が大きくなると、低コストで動的絞り込みランキングの品質を大幅に向上させます。
`--failedupdatedir <dir>`	失敗した更新ファイルをMDEX Engineが保存するディレクトリを指定します。 失敗した更新ファイルを格納するためにMDEX Engineで使用されるデフォルト・ディレクトリは、`<`*`updatedir`*`>/failed_updates/`です。

フラグ	説明
`--help`	ヘルプ・メッセージを出力して終了します。
`--lang <lang-id>-u-<collation>`	`<lang-id>`で指定された言語で部分更新を索引付けして問合せを処理し、引数の`<collation>`部分で、検索結果に対するオプションの照合順序も指定します。 未指定の場合、デフォルト値はDgidxの`--lang`フラグから継承され、DgidxとDgraphのいずれに対してもフラグを指定していない場合、`<lang-id>`のデフォルトは`en` (米国英語) です。国際言語の使用方法の詳細は、*MDEX Engine Advanced Development Guide*を参照してください。
`--log <path>`	Dgraphリクエスト・ログ・ファイルのパスを指定します。デフォルトのログ・ファイルの名前は`dgraph.reqlog`です。
`--log_stats <path>`	EQL (Endeca問合せ言語) 統計ログのパスとファイル名を指定します。デフォルトで、このログはオフに設定され、このフラグを指定すると、EQLリクエストに対する統計のロギングがアクティブ化されます。
`--log_stats_thresh <value>`	Endeca問合せ言語リクエストに対する統計情報のログが記録されるようになる上限のしきい値を設定します。値はミリ秒 (1000ミリ秒=1秒) で指定されます。数字の後に`s`を付けることによって、1秒を`1s`のように、値を秒で指定することもできます。デフォルトは`60000`ミリ秒 (1分) です。このフラグは、`--log_stats`フラグの使用に依存することに注意してください。
`--mergepolicy <policy>`	部分更新に対するMDEX Engineのデフォルト・マージ・ポリシーを設定します。*<policy>*の値は`balanced`または`aggressive`のいずれかにする必要があります。このフラグを使用しない場合は、`balanced`がデフォルト・マージ・ポリシーです。マージ・ポリシーの詳細は、*Partial Updates Guide*を参照してください。
`--net-timeout`	Dgraphが、クライアントによるネットワークを介した問合せからのデータのダウンロードを待機する最大秒数を指定します。デフォルトのネットワーク・タイムアウト値は`30`秒です。
`--nomrf`	動的ビジネス・ルールのフィルタ処理を無効にします。
`--out <stdout/stderr file>`	stdout/stderrの再マップ先のファイル・パスを指定します (デフォルトでは、処理用のデフォルトのstdout/stderrが使用されます)。 Oracle Endeca Application Controller環境におけるDgraphの実行では、`dgraph-S0-R0.out`という名前のデフォルト・ファイルが作成されます。
`--persistdir`	選択したディレクトリにDgraph監査永続性ファイルを送信します。デフォルトでは、ファイルは、アプリケーションの作業ディレクトリにある`persist`というディレクトリに書き込まれます。

フラグ	説明
	監査永続性ファイルの詳細は、*Endeca Performance Tuning Guide*を参照してください。 重要: `--persistdir`フラグは、Dgraphの初回起動時にのみ使用してください。このディレクトリは、作成後に移動または名前変更しないでください。
`--phrase_max <num>`	テキスト検索に対する各フレーズの最大ワード数を指定します。デフォルト数は10です。フレーズ中の最大ワード数を超えた場合、そのフレーズは最大ワード数に切り捨てられ、警告が記録されます。
`--pidfile <pidfile-path>`	処理ID (pid) の書込み先のファイルを指定します。未指定の場合、pidファイルのデフォルト名はDgraphの起動方法によって異なります。 EAC環境またはコマンドラインからのDgraphの実行では、`dgraph.pid`という名前のファイルが作成されます。
`--port <num>`	サーバー (非対話) モードで使用するポートを指定します。デフォルトは`5555`です。
`--search_max <num>`	テキスト検索に対する最大用語数を指定します。デフォルトは`10`です。
`--snip_cutoff <num>`	MDEX Engineでスニペットを識別するために評価されるプロパティ内のワード数を制限します。*<num>*ワード内に一致が検出されない場合、その後でプロパティ値内で一致が検出された場合でも、MDEX Engineによってスニペットが返されることはありません。 フラグを指定しない場合、または*<num>*を指定しない場合、デフォルトは500です。
`--snip_disable`	スニペットをグローバルに無効にします。
`--spellpath <path>`	スペル・データ・ファイルの場所を指定します。パラメータは、スペル修正機能用の必要なaspellサポート・ファイルが格納されているディレクトリへのフル・パスである必要があります (`--dym`および`--spl`オプションを参照)。このパスには絶対パスを指定する必要があることに注意してください (相対パスはサポートされていません)。また、これは、少なくとも汎用pspell/aspellサポート・ファイルが格納されているディレクトリへのパスです。索引付けされたデータ・セットに対する.spelldatファイルの場所と同じである必要はありません。Dgraphでは通常、正しい.pwliファイルまたは書込み可能な.pwliファイルがこのディレクトリですでに使用可能でないかぎり、このディレクトリへの書込み権限が必要です。

フラグ	説明
`--spell_bdgt <num>`	スペルおよびDYM (Did You Mean?) 修正に対して考慮されるバリアントの最大数を設定します (デフォルトは32)。
`--spell_nobrk`	提案エンジンにおけるワード分割分析を無効にします。通常、提案エンジンでは、スペル修正を考慮する以外に、DYM (Did You Mean?) および自動修正に対する提案を生成するために、問合せに対する代替のワード分割ポイントが考慮されます。
`--spl`	レコード (全文) およびディメンション検索に対する自動提案スペル修正を有効にします。
`--spl_hthresh <thresh>`	自動修正提案を生成しない下限値を指定します。デフォルトは`1`であり、これは、ユーザーの検索に対して1つ以上のヒットがある場合、自動修正でスペル提案は提供されないことを意味します。言い換えると、デフォルトの1を使用し、ユーザーの検索に対するヒットがゼロ (0) の場合、スペル自動修正によって、代替のキーワード・スペルに対する提案が採用され、提供されます。
`--spl_nsug <count>`	自動修正提案を戻す最大数を指定します。デフォルトは`1`です。
`--spl_sthresh <thresh>`	自動修正提案として使用されるワードに対するしきいスペル修正スコアを指定します。デフォルトは`125`です。
`--sslcertfile <certfile-path>`	SSL通信用にクライアントに提供するためにDgraphで使用される、`eneCert.pem`証明書ファイルのパスを指定します。このフラグを使用すると、MDEX EngineがSSL用にクライアントに提示する証明書が提供されます。また、このオプションでは、HTTPではなくHTTPS接続が強制されます。提供しない場合、Dgraph通信に対してSSLは有効ではありません。
`--sslcafile <CA-certfile-path>`	他のEndecaコンポーネントとのSSL通信を認証するためにDgraphで使用される、`eneCA.pem`認証局ファイルのパスを指定します。このフラグでは、相互認証の目的で、クライアント接続の検証のためにMDEX Engineで使用される認証局ファイルが定義されます。提供しない場合、SSL相互認証は実行されません。 注: セキュアな接続を確立する必要があるが、認証済の接続を確立する必要がない場合は、`--sslcafile`フラグなしで`--sslcertfile`フラグを使用します。
`--sslcipher <cipher-list>`	SSLネゴシエーション時にDgraphで使用される、最小暗号化アルゴリズムを指定する暗号名 (RC4-SHAなど) を1つ以上設定します。複数の暗号を指定する場合は、名前をコロンで区切る必要があります。
`--stat-all`	使用可能なすべての動的ディメンション値属性を有効にします。このオプションはパフォーマンスに影響し、本番使用を目的としていないことに注意してください。

フラグ	説明
`--stat-abins`	集計済レコードに対する絞込み件数を有効にします。絞込み件数は、ディメンション値で絞り込んだ場合に、結果セットに含まれるレコードの数です。集計済レコードは、表示目的で単一のレコードに積み上げられる複数のレコードを表すレコードです。 このフラグを使用すると、絞込み件数には、ディメンション値で絞り込んだ場合にMDEX Engineによって結果セットに返される集計済レコード数が反映されます。 通常、MDEX Engineでは、絞込み件数は次のように計算されます。 - 通常 (非集計) レコードの結果を返す場合、MDEX Engineでは、絞込みごとの絞込み件数が計算されます。 (Developer Studioで絞込み件数を有効にします。) 通常レコードの絞込み件数は、MDEX Engineによって`Dgraph.Bins`プロパティとして返されます。 - 集計済レコードの結果を返す場合、`--stat-abins`フラグを使用すると、MDEX Engineによって集計済レコードに対する絞込み件数が返されます。これらの件数は、絞込みごとの集計済レコード数を正確に反映します。(このフラグを使用することによって集計済レコードに対する絞込み件数を有効にします。) 集計済レコードの絞込み件数は、MDEX Engineによって`Dgraph.AggrBins`プロパティとして返されます。 集計済レコードの動的統計は、MDEX Engineに対して負荷がかかる計算であることに注意してください。このフラグは、フロントエンド・アプリケーションで集計済レコードに対する絞込み件数を表示する目的でのみ使用してください。
`--syslog`	すべての出力をsyslogに送信します。
`--thesaurus_cutoff <limit>`	シソーラス置換の対象となるユーザーの検索問合せのワード数に制限を設定します。 *<limit>*のデフォルト値は3です。これは、ユーザーの検索問合せ中の3ワードまでがシソーラス・エントリで置換可能であることを意味します。シソーラス・エントリと一致する問合せ内の用語数がこれより多い場合、いずれのワードもシソーラス拡張されません。 このオプションは、非常に負荷がかかるシソーラス問合せに対するパフォーマンス保護として使用されます。値が小さいと、シソーラス・エンジンのパフォーマンスが向上します。
`--thesaurus_multiword_nostem`	複数ワード・シソーラス形式内のワードをフレーズと同様に処理する必要があり、ステミングしないことを指定します(これによって一部の問合せロードのパフォーマンスが向上します)。単一ワー

フラグ	説明
	ドの用語は、このフラグを指定しているかどうかに関係なくステミングの対象となります。 このフラグを使用すると、Dgraphでは、複数ワード・シソーラス形式のステミングによる拡張が行われなくなります。シソーラス・エントリは問合せ内のステミングされた形式と引き続き照合されますが、複数ワード拡張には、明示的にリストされた形式のみが含まれます。複数ワードのステミングされたシソーラス拡張を取得するには、様々な形式をシソーラスに明示的にリストする必要があります。
`--threads <num>`	MDEX Engineスレッド・プールのスレッド数を指定します。<num>の値は正数 (つまり1以上) にする必要があります。 <num>のデフォルトは2です。 MDEX Engineに対するスレッド数は通常、MDEX Engineサーバーのコア数と同じにすることをお薦めします。 CPUの負担が少なく、問合せ処理や更新には影響を与えない内部保守タスクの実行のためにも追加のスレッドが起動されます (これらの数は制御できません)。
`--tmpdir <dir>`	一時ファイルの格納に使用する一時ディレクトリへのパスを指定します (デフォルトは*db_prefix*のベース・ディレクトリです)。
`--unctrct`	Dgraphに対して、ナビゲーション結果に絞込みを表示する際に、暗黙的なディメンションは計算せず、明示的に指定したディメンションのみを計算および提示することを指定します。このフラグの指定では、表示される結果のレコード・セットのサイズは縮小しませんが、MDEX Engineのランタイム・パフォーマンスは向上します。 このフラグを使用する場合は、意味のあるナビゲーション絞込みを受け取るために、すべてのアウトバウンド問合せに対して最上位の優先順位ルールを機能させるようにする必要があることに注意してください。
`--updatedir <dir>`	完了した部分更新ファイルが格納されるディレクトリを指定します。また、部分更新ファイルがこのディレクトリから読み取られます。
`--updatelog`	更新関連のログ・メッセージに対するファイルを指定します。未指定の場合、更新ファイルのデフォルト名はDgraphの起動方法によって異なります。管理システム環境 (非推奨) またはコマンドラインからのDgraphの実行では、`dgraph.updatelog`という名前のデフォルトが作成されます。Endecaアプリケーション・マネージャ環境におけるDgraphの実行では、

フラグ	説明
	dgraph-S0-R0-update.logという名前のデフォルトが作成されます。
--updateverbose	更新処理時に冗長メッセージを表示します。
--version	バージョン情報を出力して終了します。
--warmupseconds <seconds>	更新後のキャッシュ・ウォーミングの期間 (秒) を指定します。
--wb_maxbrks	ワード分割分析で、問合せごとに挿入または削除する最大分割数を指定します。デフォルトは1です。
--wb_minbrklen	ワード分割分析で、新しいワード分割用語の最小長を指定します。デフォルトは2です。
--wb_noibrk	ワード分割で、ワード分割挿入分析を無効にします。
--wb_norbrk	ワード分割で、ワード分割削除分析を無効にします。
--wildcard_max <count>	句読点を含むワイルドカード問合せ (ab*c.def*など) でワイルドカードの用語を照合できる最大用語数を指定します。デフォルトは100です。
--whymatch	*MDEX Engine 6.2.0では非推奨です。* 全文検索問合せの結果として返される「Why Did It Match (照合理由) 」動的レコード属性の計算を有効にします。これらの動的属性には、一致の要因となったプロパティ/ディメンションのキーと値のコピーが、問合せ解釈ノート (スペル、シソーラスなど) とともに含まれます。
--whymatchConcise	*MDEX Engine 6.2.0では非推奨です。* --whymatchと同様ですが、プロパティ/ディメンションのキーおよび問合せ解釈ノートのみを含む、より簡潔な動的属性値を生成します。これは、ドキュメントの内容など、プロパティ値に大量のテキストが含まれる可能性がある場合に有効です。
--wordinterp	ワード解釈の動的補足(または参照提示)オブジェクトの計算を有効にし、全文(レコード)検索リクエストの処理時にテキスト検索エンジンで考慮されるユーザー問合せ用語の代替形式についてレポートします。
--xquery_fndoc <mode>	*MDEX Engine 6.3.0では非推奨です。* XQuery内のfn:doc()ファンクションの処理を指定します。次の3つの値がサポートされています。 - noneでは、fn:doc()のすべてのコールが失敗します。 - sandboxではfn:doc()は許可されますが、その引数はXQueryサービス・ディレクトリのXMLサブディレクトリ内の相対パスとして解釈されます。 - openでは、fn:doc()が許可され、その引数はURLとして解釈されます。

フラグ	説明
	指定しない場合、デフォルトはnoneです。デプロイ済アプリケーションで使用する場合、openはサポートされていないことに注意してください。
`--xquery_path <path>`	*MDEX Engine 6.3.0では非推奨です。*XQuery Webサービスのリソースが配置されるディレクトリを指定します。XQueryのメイン・モジュールおよびWSDLファイルは、このディレクトリからロードされます。ライブラリ・モジュールは`lib`サブディレクトリからロードされます。指定しない場合、ユーザーXQueryパスは使用されません。

付録 C

XML構成ファイル

この項では、Endecaアプリケーションで使用されるXML構成ファイルについて説明します。通常、システム管理者はこれらのファイルを作成しません。ただし、ファイルの変更や移動が必要になる場合に備えて、ファイルについて理解し、ファイルを特定できる必要があります。

XML構成ファイルについて

XML構成ファイルには、Forge、DgidxおよびDgraphプロセスの動作の側面を制御するために使用する設定が含まれています。

XML構成ファイル (詳細は*Endeca XML Reference*に記載) には、受信レコードのディメンション、ディメンション検索、ディメンション検索索引、優先順位ルール、プロパティ、絞込み、およびその他の様々な側面を定義します。XML構成ファイルは、Dgidxが読み取ってその一部を使用し、その他の部分がDgidx出力ディレクトリ、さらにDgraph入力ディレクトリに渡されます。具体的には、構成の更新時にDgraphがDgraph入力ディレクトリにあるXML形式で表された構成ファイルを再度読み取ります。これらのファイルには、ビジネス・ルール定義、キーワード・リダイレクト定義、ランディング・ページ定義、シソーラス・エントリ、検索インタフェース定義、導出されたプロパティ定義、レンダリング設定などのファイルがあります。

XML構成ファイルは、Forgeによってその出力ディレクトリにコピーされ、Dgidxインデクサ・プロセスが使用します。Forge出力ディレクトリとDgidx入力ディレクトリが異なる場合は、Deployment Templateスクリプトにより、Forge出力ディレクトリからDgidx入力ディレクトリに構成ファイルがコピーされます。

`Project.xml`ファイルは、Forgeで生成されたファイル (レコード、ディメンションおよびプロパティ/ディメンション構成) およびForgeを介してコピーされた構成ファイルの検索場所をDgidxに伝えます。通常、このファイルを編集する必要はありません。

👉 **注:** 場合によっては、XML構成ファイルの問題を修正する必要があります。Workbenchを使用している場合に、Workbench外部から構成ファイルを取得 (または既存のファイルをWorkbenchに送信) する方法は、このガイドの付録を参照してください。

XML構成ファイルの作成

XML構成ファイルは、次のいずれかの方法でアプリケーションの一部として作成されます。

- Deployment Templateを使用して新しいアプリケーションをデプロイすると、デフォルトのXML構成ファイル・セットがアプリケーションの`config/pipeline`ディレクトリに作成されます。これらのファイルは、最初にDeployment Templateで作成され、`MyApp.Dimensions.xml`など、アプリケーション名を含む接頭辞が付けられます。チームのメンバーは、Developer StudioまたはOracle Endeca Workbenchでこれらのファイルを編集して、デフォルト設定を変更できます。
- アプリケーション開発者がDeveloper Studioを使用してEndecaプロジェクトを作成すると、Developer StudioでXML構成ファイルが作成されます。アプリケーション開発者がプロジェクトを変更すると、Developer StudioおよびWorkbenchによって変更内容がXMLファイルに書き込まれます。

`config/pipeline`ディレクトリには、Developer Studioのパイプライン・ファイル`pipeline.epx`も格納されます。

`WorkbenchManager`の設定に応じて、ベースライン更新中に、Workbenchから構成ファイルを`<app name>/config/pipeline`ファイル・セットにマージできます。詳細は、*Tools and Frameworks Deployment Template Usage Guide*を参照してください。

Deployment Templateの出力接頭辞の変更

Deployment Templateを使用する場合は、Developer Studioのインデクサ・アダプタ・パイプライン・コンポーネントの`output-prefix`値を変更してこの設定値を保存すると、変更された接頭辞でXMLファイルの新しいセットが保存されることに注意してください。

ただし、Deployment Templateのプロジェクトでは、コンポーネントの名前のデフォルト接頭辞が保持され、スクリプトでForgeを実行する場合などに古いファイルを出力ディレクトリにコピーできます。これにより、最新の構成ファイルが上書きされる場合があります。したがって、XML構成ファイルのデフォルト接頭辞は変更しないことをお薦めします。

XML構成ファイルの作成と変更

XML構成ファイルを作成および更新する方法はいくつかあります。

プロジェクトはDeveloper StudioまたはOracle Endeca Workbenchを使用して構成し、通常と異なる状況で必要な場合のみ、XMLファイルを直接変更することをお薦めします。XML構成ファイルを直接変更する場合は、バイト順マーク (BOM) をファイルに書き込まないでください。

付録 D

Endecaの環境変数およびポートの使用

この項では、Endecaソフトウェアで使用される環境変数およびポートのリストを示します。インストールしたコンポーネントに応じて、使用する実装に該当しない場合があります。

Endecaの環境変数

Endecaインストール・プログラムでは、いくつかの環境変数が作成されます。

各変数について最初にリストされている値は、Windowsでデフォルト・インストール・パス(`C:\Endeca\<product name>`)を使用して、マシン単位のインストールを行う場合のパスです。ユーザー単位のインストールの場合、デフォルト・パスのルートは`%USERPROFILE%`ディレクトリです。

2番目の値は、UNIXでのインストール・ディレクトリ内のパスです。たとえば、Endecaを`/usr/local/`にインストールした場合、環境内での`ENDECA_ROOT`のフル・パスは`/usr/local/endeca/PlatformServices/version`です。

後述する変数が作成されるのに加えて、インストールによってEndecaのディレクトリが`PATH`変数に追加される場合があります。

👉 注: MDEX Engineインストールの場合は、インストールによって提供される`mdex_setup`スクリプトを実行すると、環境および`PATH`変数が設定されます。詳細は、*Oracle Endeca MDEX Engine Installation Guide*を参照してください。

MDEX Engineの変数

MDEX Engineで使用される変数は次のとおりです。

変数	説明	デフォルト値
`ENDECA_MDEX_ROOT`	MDEX Engineのルート・ディレクトリのパスを指定します。	- `C:\Endeca\MDEX\version` - `endeca/MDEX/version`

Endeca Tools Service

Endeca Tools Serviceでは、次の環境変数が使用されます。

環境変数	値	設定
JAVA_HOME	<ENDECA_TOOLS_ROOT>\server\j2sdk	JAVA_HOMEを、含まれているJava 2 SDKに設定します。
ENDE¬ CA_TOOLS_ROOT	<Tools and Frameworks directory>	値をツールおよびフレームワークのインストール・ディレクトリに設定します。
ENDE¬ CA_TOOLS_CONF	<ENDECA_TOOLS_ROOT>\server\workspace	値をツールおよびフレームワークのserver\workspaceディレクトリに設定します。
CATALINA_BASE	ENDECA_TOOLS_CONF	Tomcatインスタンスの場所をEndecaのworkspaceディレクトリに設定します。

注: WindowsでEndeca Tools Serviceからツールおよびフレームワークを実行しない場合は、前述の環境変数を手動で設定する必要があります。

プラットフォーム・サービスの変数

プラットフォーム・サービスで使用される変数は次のとおりです。

変数	説明	デフォルト値
ENDECA_ROOT	プラットフォーム・サービスのルート・ディレクトリのパスを指定します。	- C:\Endeca\PlatformSer¬ vices*version* - endeca/PlatformServices/*ver¬ sion*
ENDECA_REFERENCE_DIR	Endeca参照実装 (サンプルのwineプロジェクト、JSPおよび.NET UI参照など) が格納されるディレクトリのパスを指定します。	- C:\Endeca\PlatformSer¬ vices\reference - endeca/PlatformServices/ref¬ erence
ENDECA_CONF	Endeca HTTPサービス用のworkspaceディレクトリのパスを指定します。このディレクトリには、構成ファイル、ログ、および一時記憶域	- C:\Endeca\PlatformSer¬ vices\workspace - endeca/PlatformSer¬ vices/workspace

変数	説明	デフォルト値
	のディレクトリが格納されます。	
PERLLIB	perlルート・ディレクトリ、およびそのライブラリのディレクトリのパスを指定します。	- %ENDECA_ROOT%\perlおよび%ENDECA_ROOT%\perl\5.8.3\lib - $ENDECA_ROOT/lib/perl:$ENDECA_ROOT/lib/perl/Control:$ENDECA_ROOT/perl/lib:$ENDECA_ROOT/perl/lib/site_perl
PERL5LIB	PERLLIB変数と同じです。	PERLLIB変数と同じです。
UnixUtils	utilitiesディレクトリのパスを指定します。このディレクトリには、いくつかのUNIX共通ユーティリティのWindowsバージョンが格納されます。	- %ENDECA_ROOT%\utilities - UNIXでは使用できません

Endecaツールおよびフレームワークの変数

Endeca Workbenchで使用される変数は次のとおりです。

変数	説明	デフォルト値
ENDECA_TOOLS_ROOT	Endecaツールおよびフレームワークのルート・ディレクトリのパスを指定します。	- C:\Endeca\ToolsAndFrameworks*version* - endeca/ToolsAndFrameworks/*version*
ENDECA_TOOLS_CONF	Endeca Tools Service用のworkspaceディレクトリのパスを指定します。このディレクトリには、構成ファイル、ログ、および一時記憶域のディレクトリが格納されます。	- C:\Endeca\ToolsAndFrameworks\server\workspace - endeca/ToolsAndFrameworks/server/workspace

その他の変数

Endecaで使用されるその他の変数は次のとおりです。

変数	説明	デフォルト値
ENDECA_PROJECT_DIR	デプロイ済アプリケーションのパスを指定します。この変数は、Endeca Deployment Templateによって設定および使用されます。	値は、インストール時にユーザーが入力します。
ENDECA_PROJECT_NAME	プロジェクト名 (たとえば、プロジェクトのジョブ制御デーモンで定義したジョブのJCDジョブ接頭辞として使用する名前) を指定します。この変数は、Endeca Deployment Templateによって設定および使用されます。	値は、インストール時にユーザーが入力します。

Endecaのポート

このトピックでは、Endecaパッケージで使用されるポート、およびそのデフォルト・ポート番号について説明します。

使用するマシン上の既存のポートと混同しないかぎり、デフォルト・ポート番号は独自の番号に置換できます。ポート番号は32767以下にしてください。

サービスのポート

ポート	デフォルト
Endeca Tools Serviceのポート	8006
Endeca Tools Serviceのプロモーション・ポート	8007
Endeca Tools ServiceのSSLポート	8446
Endeca Tools Serviceのシャットダウン・ポート	8084
CASサービスのポート	8500
CASサービスのシャットダウン・ポート	8506
Endeca HTTPサービスのポート	8888
Endeca HTTPサービスのSSLポート	8443
Endeca HTTPサービスのシャットダウン・ポート	8090

Deployment Templateのポート

Deployment Templateインストールで推奨されるポート番号は次のとおりです。ただし、アプリケーションをデプロイするときに、他のポートを指定できます。

ポート	デフォルト
Dgraph1ユーザー問合せポート	15000
Dgraph2ユーザー問合せポート	15001
Endecaロギングおよびレポート・サーバーのポート 注: ロギング・サーバーのポート番号は32767以下にしてください。	15010

参照実装のポート

参照実装 (sample_wine_data) に付属する構成ファイルで使用されるポート番号は次のとおりです。

ポート	デフォルト
Endeca MDEX Engineのユーザー問合せポート	8000
Endecaロギングおよびレポート・サーバーのポート 注: JSP参照実装では、ロギング・サーバーのデフォルト・ポート番号は、対応するDgraphポート番号より2だけ大きい番号です。たとえば、Dgraphポートが15000の場合、参照実装のロギング・サーバーのデフォルト・ポートは15002です。Dgraphポートが15001の場合、参照実装のロギング・サーバーのデフォルト・ポートは15003です (これは、ロギング・サーバーがMDEX Engineと同じホスト上で稼働していることを前提にしています)。	8002

付録 E

環境間での**Endeca**実装の転送

この項は、プロジェクトでOracle Endeca Workbenchのみを使用し、Deployment Templateを使用しない場合に参照してください。この項では、Workbenchを使用するステージング環境からWorkbenchを使用する本番環境に、Endeca実装を転送する方法について説明します。

Deployment Templateを使用した実装の場合

この項では、Oracle Endeca Workbenchのみを使用する実装に限定していますが、実装の設定と保守に対する優先アプローチは、Deployment Templateの使用です。

Deployment Templateを使用した実装の作成と複製の詳細は、このガイドの*「複数サーバー環境の作成」*および*「アプリケーション定義の複製」*の項を参照してください。

実装の転送について

この項では、インスタンス構成およびMDEX Engineの環境間での転送について説明します。

ここでは、Endecaツールを使用して実装を手動で転送する方法、および`emgr_update`ユーティリティを使用して転送のスクリプト作成と自動化を行う方法を説明します。この項では、説明をわかりやすくするために、Endeca実装をステージング環境から本番環境に転送するとします。ただし、このケースに限定されるわけではありません。ここで説明する手順に従って、選択した任意の環境間で転送を行うことができます。

使用する環境および要件に応じて、環境間の移動を完了するにはフロントエンドWebアプリケーションの転送も必要になる場合があります。Endecaにおいてフロントエンドwebアプリケーションを転送するのに必要な作業は、`ENEConnection`オブジェクトで使用するMDEX Engineのホスト名とポートが新しい環境に対して正しいことを確認することのみです。

👉 注: `ENEConnection`インタフェースの使用の詳細は、*Endeca MDEX Engine Basic Development Guide*を参照してください。

Developer StudioでのOracle Endeca Workbenchインスタンス構成の取得

Endecaツールを使用すると、Oracle Endeca Workbenchを使用するステージング環境からOracle Endeca Workbenchを使用する本番環境に、手動による転送を実行できます。

Developer Studioを使用してOracle Endeca Workbenchインスタンス構成を取得する手順は、次のとおりです。

1. ステージング環境で、Developer Studioを起動し、新しいプロジェクトを作成します。
2. 「**Tools**」メニューから、「**Workbench Settings**」を選択します。
   「Workbench Settings」ダイアログ・ボックスが表示されます。
3. 目的のアプリケーション用のOracle Endeca Workbenchのホスト名とポートを指定します。指定したホスト名とポートが、情報を取得するOracle Endeca Workbenchと一致していることを確認してください。
4. 同じダイアログ・ボックスで、ドロップダウン・リストから目的のアプリケーションを選択するか、指定したアプリケーション名が、Oracle Endeca Workbenchから構成を取得するアプリケーションの名前と一致していることを確認してください。Oracle Endeca Workbenchには、作成された複数のアプリケーションのインスタンス構成が複数存在できるため注意してください。
5. Oracle Endeca Workbenchツールバーで、「**Get Instance Configuration**」:  」をクリックします。
6. 「**File**」メニューから「**Save**」を選択して、最新のインスタンス構成のプロジェクトを保存します。
7. (オプション) インスタンス構成から非アクティブな動的ビジネス・ルールを削除します。
8. 保存したプロジェクトのインスタンス構成ファイルを、Endeca Application Controller がそのファイルを認識できる本番環境の場所にコピーします。
9. Developer Studioで、自動生成ディメンションまたは外部ディメンションをロードするロード機能を使用している場合は、これらのForgeの状態ファイルを手動で同期化する必要があります。
10. このApplication Controllerを使用して、本番システムでベースライン更新を実行します。


注: ロード機能を使用して自動生成ディメンション値または外部ディメンション値をDeveloper Studioにロードした場合、ID割当はインスタンス構成に格納されません。これらのファイルは実装環境間で手動で同期化する必要があります。

関連リンク

自動生成ディメンション値および外部ディメンション値のID割当の転送 135 ページ
ロード機能を使用して自動生成ディメンション値または外部ディメンション値をDeveloper Studioにロードした場合、ID割当はインスタンス構成に格納されません。したがって、これらのファイルは実装環境間で手動で同期化する必要があります。

emgr_updateについて

emgr_updateユーティリティを使用すると、ステージング環境でEndecaツールを使用して行った変更に基づき、本番システムのインスタンス構成を更新できます。

適切な--action操作を使用することによって、emgr_updateを使用して次のタスクを実行できます。

- 選択した特定のアプリケーションのインスタンス構成ファイルを、ステージング環境から本番環境に転送します。転送後に、独自のEACスクリプトを使用してベースライン更新を実行します。すべてのインスタンス構成ファイルを転送するか、または、Oracle Endeca Workbenchで変更したインスタンス構成ファイルのみを転送するかを選択できます。
- 特定のアプリケーションのインスタンス構成をOracle Endeca Workbench環境間で転送します。
- 指定したアプリケーションのインスタンス構成情報をOracle Endeca Workbench構成から削除します。
- Forgeのディメンションを Oracle Endeca Workbenchに送信します。

emgr_update構文リファレンス

この項では、emgr_updateで使用可能なすべてのコマンドライン・パラメータのリストを示します。

emgr_updateユーティリティを使用すると、ステージング環境でEndecaツールを使用して行った変更に基づき、本番システムのインスタンス構成を更新できます。

emgr_updateはコマンドラインから実行します。コマンド・プロンプトまたはUNIXシェルを開き、プログラムを実行します。emgr_updateを実行する構文は、`emgr_update <parameters>`です。

次の表に、emgr_updateで使用可能なすべてのコマンドライン・パラメータを示します。ユーティリティの1回の呼出しで指定できるのは、1つの`--action`操作のみです。

次に、使用例を示します。

```
emgr_update --host localhost:8006 --action get_ws_settings
--prefix wine --dir /apps/endeca/data/forge_input --app_name wine
```

emgr_updateパラメータ	説明
`--host name:port`	Oracle Endeca Workbenchが稼働するマシンのホスト名、およびそのマシンのEndeca Tools Serviceのポートを指定します。 設定を取得する場合 (get操作を使用)、これは転送元の環境のホスト名です。設定を更新する場合 (set操作を使用)、これは転送先の環境のホスト名です。
`--action <op>`	アクションを1つ指定します。<op>は次にリストする操作のいずれかです。
`--action get_all_settings`	本番環境で使用するために、ステージング環境内でOracle Endeca Workbenchで実行されたプロジェクトのすべてのインスタンス構成設定を取得します。 必須パラメータ: `--dir`、`--prefix`

emgr_updateパラメータ	説明
	オプション・パラメータ: `--filter`
`--action get_ws_settings`	(すべての設定ではなく) Oracle Endeca Workbenchで変更可能なインスタンス構成設定のみを取得します。 これらの構成設定には、動的ビジネス・ルール、キーワード・リダイレクト、シソーラス・エントリ、自動フレーズ、ストップ・ワードおよびディメンション順序指定のOracle Endeca Workbench機能が含まれます。
`--action get_mdex_settings`	Oracle Endeca Workbenchで変更され、MDEX Engineを更新する際にベースライン更新が不要なインスタンス構成設定を取得します。 これらの構成設定には、動的ビジネス・ルール、キーワード・リダイレクト、シソーラス・エントリおよび自動フレーズのOracle Endeca Workbench機能が含まれます。 必須パラメータ: `--dir`、`--prefix` オプション・パラメータ: `--filter`
`--action set_post_forge_dims`	Forge後のディメンションを使用して、Oracle Endeca Workbench構成を更新します。
`--action get_post_forge_dims`	Forge後のディメンションについて、Oracle Endeca Workbench設定のコピーを取得します。通常、この操作はデバッグ目的で使用できます。
`--action update_mgr_settings`	ステージング環境内のOracle Endeca Workbench構成から抽出されたインスタンス構成設定を使用して、Oracle Endeca Workbenchの本番環境を更新します。
`--action remove_all_settings`	`--app_name`パラメータで指定するアプリケーションのOracle Endeca Workbenchから、すべてのインスタンス構成ファイルを削除します。 インスタンス構成を削除しても、アプリケーションの関連するプロビジョニング情報は削除されません。
`--app_name <string>`	EAC Centralサーバーに対してプロビジョニングされたアプリケーションの名前を指定します。

追加のアクション・パラメータ	説明
`--dir <pathname>`	インスタンス構成ファイルに対する書込み/読取りが行われるディレクトリの`pathname`を指定します。`set_post_forge_dims`および`remove_all_settings`を除き、すべての`--action`操作で必須です。
`--prefix <pathname>`	インスタンス構成ファイルで使用する接頭辞を指定します。このオプションは、`get_post_forge_dims`および`set_post_forge_dims`を除き、すべての`--action`操作で必須です。
`--filter`	状態が非アクティブな動的ビジネス・ルールをフィルタ処理します。(ルールによって、`endeca.internal.workflow.state`プロパティが`INACTIVE`に設定されます。) このオプションは、インスタンス構成を取得するときに、`get_all_settings`または`get_ws_settings`とともに使用できます。 非アクティブなルールの削除は必須ではありませんが、削除することをお薦めします。デフォルトのルール・フィルタを設定すると、MDEX Engineでは、状態が非アクティブなルールは起動されません。つまり、アクティブなルールと非アクティブなルールの両方を含めてインスタンス構成を転送できますが、MDEX Engineでは、ユーザー問合せに応答してアクティブなルールのみが起動されます。
`--post_forge_file <pathname>`	Forge後のディメンションが含まれるファイルへのパス名を指定します。このオプションは、`set_post_forge_dims`操作で必須です。

オプションのグローバル・パラメータ	説明
`--stop_on_warnings`	get操作の前にターゲット・ディレクトリが空でない場合、または、update操作の前にファイルの過不足が検出された場合は、確認なしでユーティリティを停止します。
`--ignore_warnings`	get操作の前にターゲット・ディレクトリが空でない場合、または、update操作の前にファイルの過不足がある場合でも、ユーティリティの実行を続行します。
`--help`	ユーティリティの使用パラメータが表示されます。
`--version`	ユーティリティのバージョン番号が表示されます。

関連リンク

emgr_updateユーティリティを使用した実装の転送について 130 ページ
Endecaツールと同様に、emgr_updateユーティリティを使用して、Oracle Endeca Workbenchを使用するステージング環境からOracle Endeca Workbenchを使用する本番環境にEndeca実装を転送できます。

emgr_updateユーティリティを使用した実装の転送について

Endecaツールと同様に、emgr_updateユーティリティを使用して、Oracle Endeca Workbenchを使用するステージング環境からOracle Endeca Workbenchを使用する本番環境にEndeca実装を転送できます。

emgr_updateユーティリティの主な利点は、環境間での転送のスクリプト作成および自動化が可能になることです。ステージング環境から本番環境への実装の転送は2ステップのプロセスで、ステージング環境からインスタンス構成を取得し、次に本番環境にその構成を設定 (更新) します。

この項では、Oracle Endeca Workbenchを使用するステージング環境からOracle Endeca Workbenchを使用する本番環境に、インスタンス構成ファイルを転送する方法について説明します。次の2つのシナリオを説明します。

- Endecaプロジェクトのすべてのインスタンス構成ファイルの転送
- Oracle Endeca Workbenchで変更可能なインスタンス構成ファイルのみの転送

注: ロード機能を使用して自動生成ディメンション値または外部ディメンション値をDeveloper Studioにロードした場合、ID割当はインスタンス構成に格納されません。これらのファイルは実装環境間で手動で同期化する必要があります。

関連リンク

emgr_update構文リファレンス 127 ページ
この項では、emgr_updateで使用可能なすべてのコマンドライン・パラメータのリストを示します。

自動生成ディメンション値および外部ディメンション値のID割当の転送 135 ページ
ロード機能を使用して自動生成ディメンション値または外部ディメンション値をDeveloper Studioにロードした場合、ID割当はインスタンス構成に格納されません。したがって、これらのファイルは実装環境間で手動で同期化する必要があります。

すべてのインスタンス構成ファイルの転送

すべてのインスタンス構成ファイルを転送するには、ステージング環境のファイルを本番環境のForge入力ディレクトリに移動します。

次に、Oracle Endeca Application Controllerを使用して、ベースライン更新を実行します。

すべてのインスタンス構成ファイルを本番システムに転送する手順は、次のとおりです。

1. ステージング環境で、Developer StudioまたはEndeca Workbench (あるいはその両方) を使用して、プロジェクトに変更を加えます。
2. `--action`の`get_all_settings`を指定して、`emgr_update`を実行します。
   a) `--host`パラメータには、ステージングのOracle Endeca Workbench環境のマシン名とポートを指定します。

b) `--dir`パラメータには、本番環境のForge入力ディレクトリを指定します。
c) `--app_name`パラメータには、転送するインスタンス構成のアプリケーション名を指定します。
d) `--filter`パラメータを使用して、非アクティブなビジネス・ルールを削除します。

次に、UNIXの例を示します。

```
emgr_update --host localhost:8006 --app_name My_application --action
get_all_settings --prefix wine --filter --dir /apps/endeca/data/forge_input
```

3. 転送先ディレクトリが空でない場合は、続行するかどうかを尋ねるプロンプトが表示されます。yと回答します。

   ユーティリティが終了すると、すべてのプロジェクト構成ファイル (プロジェクトおよびパイプライン・ファイルを含む) が`--dir`パラメータで指定された本番ディレクトリにコピーされます。

4. Oracle Endeca Application Controllerを使用して、本番システムでベースライン更新を実行します。

このユーティリティでは、Developer Studioの出力プロジェクト・ファイルの名前として`prefix.esp` (`prefix`は`--prefix`パラメータで指定した名前) が使用されます。本番ディレクトリに別の名前の既存プロジェクト・ファイルが存在する場合は、そのファイルを`prefix.esp`に変更することをお薦めします。

注: ロード機能を使用して自動生成ディメンション値または外部ディメンション値をDeveloper Studioにロードした場合、ID割当はインスタンス構成に格納されません。これらのファイルは実装環境間で手動で同期化する必要があります。

Endeca Workbenchで変更したインスタンス構成ファイルのみの転送

Oracle Endeca Workbenchで変更可能なインスタンス構成ファイルのみを、ステージング環境から本番環境に転送できます。

このタスクでは、ステージング環境のファイルを本番環境のForge入力ディレクトリに転送します。ただし、これらのファイルは、Oracle Endeca Workbenchで変更可能なインスタンス構成ファイルです。インスタンス構成ファイルの完全なセットではありません。Oracle Endeca Workbenchでは、次の機能のインスタンス構成ファイルを変更できます。

- 動的ビジネス・ルール
- シソーラス・エントリ
- 自動フレーズ
- ストップ・ワード
- ディメンション順序指定

後続のベースライン更新では、更新されたファイルがこれらの機能に対して使用されます。

Oracle Endeca Workbenchで変更したインスタンス構成ファイルを本番システムに転送する手順は、次のとおりです。

1. ステージング環境で、Oracle Endeca Workbenchを使用して、必要なインスタンス構成の変更を加えます。
2. `--action`の`get_ws_settings`を指定して、emgr_updateを実行します。

a) `--host`パラメータには、ステージングのWorkbench環境のマシン名とポートを指定します。
b) `--dir`パラメータには、本番環境のForge入力ディレクトリを指定します。
c) `--app_name`パラメータには、転送するインスタンス構成のアプリケーション名を指定します。
d) `--filter`パラメータを使用して、非アクティブなビジネス・ルールを削除します。

次に、Windowsの例を示します。

```
emgr_update.bat --host localhost:8888 --app_name My_application --action
 get_ws_settings--prefix wine --filter --dir c:\endecaproduction\da¬
ta\forge_input
```

3. 転送先ディレクトリが空でない場合は、続行するかどうかを尋ねるプロンプトが表示されます。yと回答します。
ユーティリティが終了すると、Endeca Workbenchで変更したプロジェクト・ファイルが`--dir`パラメータで指定された本番ディレクトリにコピーされます。
4. Oracle Endeca Application Controller (EAC) を使用して、本番システムでベースライン更新を実行します。

Workbench環境間での転送

`emgr_update`ユーティリティを使用して、インスタンス構成ファイルをOracle Endeca Workbench環境間で転送およびデプロイする処理は、`emgr_update`ユーティリティの `--action`操作の1つを使用して実行します。処理全体を記述することができます。

すべてのインスタンス構成ファイルを本番システムに転送およびデプロイする手順は、次のとおりです。

1. ステージング環境で、Developer Studio、Endeca Workbenchまたは両方の組合せを使用して、プロジェクトに変更を加えます。
2. `--action`の`get_all_settings`を指定して、`emgr_update`を実行します。
   a) `--host`パラメータには、ステージングのOracle Endeca Workbench環境のマシン名とポートを指定します。
   b) `--dir`パラメータには、本番環境のForge入力ディレクトリを指定します。
   c) `--app_name`パラメータには、転送するインスタンス構成のアプリケーション名を指定します。
   d) `--filter`パラメータを使用して、非アクティブなビジネス・ルールを削除します。

   次に、Windowsの例を示します。

```
emgr_update.bat --host localhost:8006 --app_name My_app --action
get_all_settings --prefix wine --filter --dir c:\endecaproduction\da¬
ta\forge_input
```

3. 転送先ディレクトリが空でない場合は、続行するかどうかを尋ねるプロンプトが表示されます。yと回答します。

   ユーティリティが終了すると、すべてのプロジェクト構成ファイルが`--dir`パラメータで指定された本番ディレクトリにコピーされます。

4. `--action`の`update_mgr_settings`を指定して、`emgr_update`を実行します。

a) `--host`パラメータには、Oracle Endeca Workbenchの本番環境用のマシン名とポートを指定します。
b) `--dir`パラメータには、Oracle Endeca Workbenchの本番環境を更新するために使用する、プロジェクト構成ファイルが含まれているディレクトリ (通常、ステップ2で使用したディレクトリと同じディレクトリ) を指定します。
c) `--app_name`パラメータには、転送するインスタンス構成のアプリケーション名を指定します。

次に、Windowsの例を示します。

```
emgr_update.bat --host localhost:8006 --app_name My_app --action up¬
date_mgr_settings --prefix wine --dir c:\endecaproduction\data\forge_input
```

5. Oracle Endeca Application Controller (EAC) を使用して、本番システムでベースライン更新を実行します。

Oracle Endeca Workbenchからのインスタンス構成ファイルの削除

Endeca WorkbenchのEAC管理コンソールでアプリケーションを削除しても、そのインスタンス構成ファイルは削除されません。アプリケーションのインスタンス構成ファイルを削除する場合は、`em¬gr_update`を使用します。

インスタンス構成ファイルを削除する手順は、次のとおりです。

- `--action`の`remove_all_settings`を指定して、`emgr_update`を実行します。
  a) `--host`パラメータには、Oracle Endeca Workbenchのステージング環境用のマシン名とポートを指定します。
  b) `--app_name`パラメータには、削除するインスタンス構成のアプリケーション名を指定します。

次に、Windowsの例を示します。

```
emgr_update.bat --host localhost:8006 --app_name My_app --action re¬
move_all_settings --prefix My_prefix
```

Forgeで生成されたディメンション・ファイルのEndeca Workbenchへの送信

ベースライン更新の実行に独自のスクリプトを使用している場合は、Forgeを実行した後に、Forgeで生成されたディメンション・ファイルを、使用しているアプリケーションのOracle Endeca Workbenchインスタンス構成に送信する必要があります。

注: この項を参照する必要があるのは、ベースライン更新の実行にDeployment Templateのデフォルトのスクリプトを使用せずに、この目的に独自のスクリプトを使用している場合のみです。

Forgeで生成されたディメンション・ファイルをOracle Endeca Workbenchに送信するには、次のように`emgr_update`を実行します。

- Forgeの出力ファイルにアクセスできるマシン (通常は、Forgeを実行したマシン) で、actionの`set_post_forge_dims`を指定して`emgr_update`を実行します。
  a) `--host`パラメータには、Oracle Endeca Workbenchの目的の環境用のマシン名とポートを指定します。
  b) `--app_name`パラメータには、この情報で更新するインスタンス構成のアプリケーション名を指定します。
  c) `--post_forge_file`パラメータには、Forgeがそのディメンションを格納した出力ファイルに対するフル・パス名を指定します。

次に、Windowsの例を示します。

```
emgr_update.bat ?-host localhost:8006 --action set_post_forge_dims ?-app_name
 wine ?-post_forge_file C:\sample_wine_data\data\partition0\forge_out¬
put\wine.dimensions.xml
```

インスタンス構成からの非アクティブなルールの削除

ステージングから本番にEndeca実装を転送する前に、非アクティブな状態の動的ビジネス・ルールを削除する必要があります。

👉 注: `emgr_update`の`--filter`オプションを使用すると、非アクティブなルールが削除されます。このトピックでは、手動で非アクティブなルールを削除する別のオプションについて説明します。

Workbenchの「**Rule Manager**」ページにあるルール・リストの「**State**」列は、ルールがアクティブか非アクティブかを示します。また、ルールに関する`merch_rule_group.xml`ファイルを検査して、ルールの`endeca.internal.workflow.state`プロパティの設定が`ACTIVE`か、または`IN¬ACTIVE`かを確認できます。

非アクティブなルールの削除は必須ではありませんが、削除することをお薦めします。デフォルトのルール・フィルタを設定すると、MDEX Engineでは、状態が非アクティブなルールは起動されません。つまり、アクティブなルールと非アクティブなルールの両方を含めてインスタンス構成を転送できますが、MDEX Engineでは、ユーザー問合せに応答してアクティブなルールのみが起動されます。

非アクティブな動的ビジネス・ルールを削除する手順は、次のとおりです。

1. 各ルール・グループのXMLファイルから非アクティブなルールを取り除くXSLTファイルを作成します。次の例を使用して、XSLTをコピーし、テキスト・ファイルに貼り付けます。

```
<?xml version="1.0" encoding="utf-8"?>
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

  <!-- explicitly output doctype -->
  <xsl:output method="xml" doctype-system="merch_rule_group.dtd" />

  <!-- copy all elements and their attributes -->
  <xsl:template match="* | @*">
```

```
  <xsl:copy><xsl:copy-of select="@*"/><xsl:apply-templates/></xsl:copy>

  </xsl:template>

  <!-- strip out the MERCH_RULE elements who have a PROP specifying
       the workflow state as INACTIVE -->
  <xsl:template
match="MERCH_RULE[PROP[@NAME='endeca.internal.workflow.state'][PVAL='IN¬
ACTIVE']]"
></xsl:template>

  </xsl:stylesheet>
```

2. `.xslt`接尾辞付きのファイルを選択した場所に保存します。
3. XSLTプロセッサを取得してインストールします。XSLTプロセッサをまだ取得してない場合は、一般的なXSLTプロセッサにXalanがあります。Xalanは、*http://xml.apache.org/xalan-j/*からダウンロードできます。
4. ルール・グループのDTDファイルをXalanの作業ディレクトリにコピーします。デフォルトのインストールでは、DTDは`C:\Endeca\MDEX\`*`version`*`\conf\dtd\merch_rule_group.dtd`にあります。
5. XSLTプロセッサをルール・グループごとに1回実行し、ルール・グループのXMLファイルの名前を渡します。たとえば、プロジェクトに5つのルール・グループがある場合は、Xalanを5回実行します。

   プロセッサによって、XSLT指示が各ルール・グループのXMLファイルに適用され、アクティブなルールのみが出力されます。XSLTプロセッサに必要な複数のパラメータには、次のものがあります。

   - 各ルール・グループのXMLファイルに対する名前とパス。ファイルは、`merch_rule_group_`*`name`*`.xml`という名前で、*name*の値には、リテラルのルール・グループ名を使用します。
   - ステップ1で作成したXSLTファイルに対する名前とパス。
6. 使用している環境の指示に従って、変更したインスタンス構成の転送を続行します。

自動生成ディメンション値および外部ディメンション値のID割当の転送

ロード機能を使用して自動生成ディメンション値または外部ディメンション値をDeveloper Studioにロードした場合、ID割当はインスタンス構成に格納されません。したがって、これらのファイルは実装環境間で手動で同期化する必要があります。

それ以外の場合は、(各環境のデータが同一でないかぎり)Forgeによって割り当てられたIDが、Developer Studioで割り当てられたIDと一致しない可能性があります。一致しないID割当は、Developer Studioに格納されているIDに依存するプロジェクト設定 (ディメンション値の順序指定、優先順位ルール、ビジネス・ルールなど) に影響を与える可能性があります。

自動生成ディメンション値および外部ディメンション値のID割当を転送する手順は、次のとおりです。

1. ディメンション値の順序指定、優先順位ルールおよびビジネス・ルールの変更に使用するEndecaアプリケーション・インスタンスのForgeの状態ディレクトリに移動します。
   次に例を示します。
   ```
   C:\Apps\staging\myapp\data\state
   ```
2. 自動生成ディメンション値および外部ディメンション値のID割当を含む、Forgeの状態ファイルを検索してバックアップします。
   次に例を示します。
   ```
   C:\Apps\staging\myapp\data\state\autogen_dimensions.xml.external.gz
   C:\Apps\staging\myapp\data\state\autogen_dimensions.xml.gz
   ```
3. Forgeの状態ファイルをコピーします。
4. これらのファイルを、実装の転送先アプリケーション・インスタンスのForgeの状態ディレクトリに貼り付けます。
   次に例を示します。
   ```
   C:\Apps\production\myapp\data\state
   ```
5. 追加の実装環境に対してこの処理を繰り返します。

これで、自動生成ディメンション値および外部ディメンション値のIDが両方の環境で同じになります。この処理は、ディメンション値の順序指定、優先順位ルールまたはビジネス・ルールを変更するたびに繰り返す必要があります。

Oracle Endecaアプリケーションの削除

Oracle Endecaアプリケーションを完全に削除するには、EAC管理コンソールでプロビジョニング情報を削除して、`emgr_update`でインスタンス構成ファイルを削除する必要があります。

EAC管理コンソールでアプリケーションを削除すると、アプリケーションのプロビジョニング情報が削除されますが、インスタンス構成ファイルは削除されません。`emgr_update`で`remove_all_settings`を実行すると、インスタンス構成ファイルは削除されますが、プロビジョニング情報は削除されません。アプリケーションに関するすべての情報を完全に削除するには、EAC管理コンソールでプロビジョニング情報を削除して、次に`emgr_update`でインスタンス構成ファイルを削除します。両方のステップを実行しない場合は、アプリケーションの不要な、または重複したファイルのセットが格納された状態になる場合があります。

Oracle Endecaアプリケーションを完全に削除する手順は、次のとおりです。

1. Oracle Endeca Workbenchで、削除するアプリケーションにログインします。
2. EAC管理コンソール・ページで、「**Delete**」をクリックします。あるいは、EAC Webサービス・クライアントを使用して、ステップ1と2を実行することもできます。これによって、プロビジョニング情報が削除されます。
3. `--action`の`remove_all_settings`を指定して`emgr_update`を実行し、インスタンス構成ファイルを削除します。

関連リンク

Oracle Endeca Workbenchからのインスタンス構成ファイルの削除 133 ページ

Endeca WorkbenchのEAC管理コンソールでアプリケーションを削除しても、そのインスタンス構成ファイルは削除されません。アプリケーションのインスタンス構成ファイルを削除する場合は、`emgr_update`を使用します。

索引

A

C

D

E

F

H

L

M

O

P

R

S

U

W

X

あ

い

こ

し

す

つ

て

と

は

ふ

ほ

ゆ

ら

ろ

わ