WebOTX 運用編(運用管理の概要)

WebOTX 運用編

バージョン: 7.1

版数: 第 4 版

リリース: 2008 年 10 月

目次

1.はじめに

本書は WebOTX 実行環境を運用するための運用操作法について概要や具体的な設定項目や設定方法について記載しています。

対象読者

このマニュアルは WebOTX Application Server Web Edition、Standard-J Edition、Standard Edition、Enterprise Edition を使って運用環境を構築するシステムエンジニア、日々の運用を行うオペレータを対象としています。

表記について

パス名表記

本書ではパス名の表記については特に OS を限定しない限りセパレータはスラッシュ'/'で統一しています。Windows 環境においては'¥'に置き換えてください。

環境変数表記

インストールディレクトリやドメインルートディレクトリなど環境によって値の異なるものについては環境変数を用いて表します。

${*env*} または $(*env*)で表しています。

例)
${AS_INSTALL} : インストールディレクトリ
${INSTANCE_ROOT}: ドメインルートディレクトリ

コマンド操作について

本書中では運用操作に用いるコマンドの詳細についての説明は省略しています。

コマンドの詳細は「運用管理コマンド」、「運用管理コマンドリファレンス」を参照してください。

2.運用管理の概要

WebOTX を運用管理の概要について説明します。

2.1.ドメイン

WebOTX ではシステムをドメインという構成単位で管理を行います。ドメインについて説明します。

2.1.1.ドメインとは

WebOTX の基本的な構成単位をドメインと呼びます。このドメインは JMX 仕様で定義されているドメインに対応します。ドメインは WebOTX で管理するリソースやサービス群を論理的にグルーピングしています。ドメインは独立して運用され、コンフィグレーション情報など構成情報はドメイン単位で独立したディレクトリに管理されます。それぞれの業務システムの要件にあわせてドメインを構成することができます。例えば 1 つのホストで商用環境と評価環境を構築したい場合、そのホスト上に商用のドメインと評価用のドメインを構成させることができます。また稼動、待機構成のクラスタ構成を行いたい場合は、複数のホストで 1 つのドメインを定義できます。

2.1.2.ドメインの構成要素

ドメインを構成するモジュール群について説明します。ドメインは大きくドメイン情報を管理する運用管理エージェントとアプリケーションサーバとしての機能を提供するサービス群(以下の表で示します)で構成されます。

運用管理エージェントはそのドメイン内で動作するサービスやコンフィグレーションを管理します。またリモートから運用操作が行えるためのインタフェースを外部に提供します。

サービスには次のものを提供しており、そのドメインで必要なサービスをドメイン作成時に指定します。

WebOTX が提供するサービス群

サービス名称	Web Edition	Standard-J Edition	Standerd Edition	Enterprise Edition
HTTP サーバ	○	○	○	○
Web コンテナ	○	○	○(注 1)	○(注 1)
EJB コンテナ	×	○	○(注 2)	○(注 2)
JNDI サービス	○	○	○	○
JMS サービス	×	○	○	○
Object Broker サービス	○	○	○	○
Transaction サービス	○	○	○	○
TP モニタ	×	×	○	○
WatchServer	×	×	×(注 3)	○
キャッシュ名前サーバ	×	×	×(注 3)	○

(注 1) Web Edition および Standerd-J Edition の Web コンテナは WebOTX エージェントの Java VM 内で動作します。 Standerd Edition および Enterprise Edition の EJB コンテナは WebOTX エージェントとは独立した Java VM 上で動作します。ただし従来どおり WebOTX エージェントの Java VM 内で動作させることも可能です。

(注 2) Standerd-J Edition の EJB コンテナは WebOTX エージェントの Java VM 内で動作します。Standerd

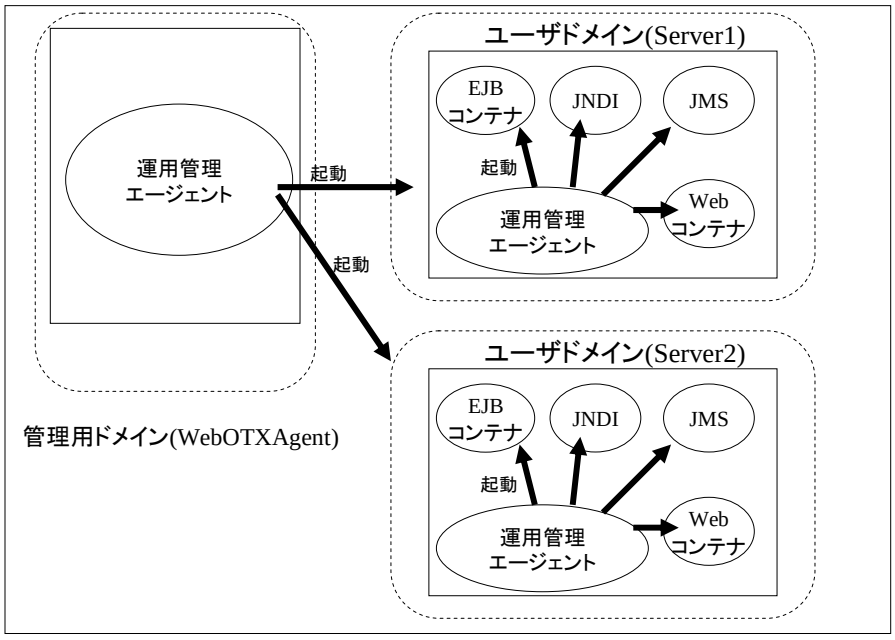
Edition および Enterprise Edition の EJB コンテは WebOTX エージェントとは独立した Java VM 上で動作します。

(注 3) Standard Edition は WebOTX クラスタ環境と組み合わせることにより、WatchServer やキャッシュ名前サーバを利用することが出来ます。

2.1.3.ドメインのタイプ

WebOTX では大きく 2 つのドメインにタイプ分けしています。

- 管理ドメイン
  WebOTX をインストールした時点で、ホスト単位で 1 つ作成されます。管理ドメインはそのホスト上で動作するユーザドメインの管理を行います。管理ドメインを通して以下の操作を行うことができます。
  ① ユーザドメインの作成、削除
  ② ユーザドメインの起動、停止
  ③ ユーザドメインの一覧取得
  ④ ユーザドメインの依存関係の設定
- ユーザドメイン
  システム構成によってユーザが任意に作成するドメイン。ドメイン作成時に、そのドメイン上で管理するリソースやサービスを任意に指定(WebOTX の Edition により指定できるリソースやサービスに制限されます)することができます。ユーザドメイン上で実際の業務システムを構成するアプリケーションやサービスが動作します。



2.1.4.シングルドメインモードとマルチドメインモード

ドメイン構成として V6.4 からシングルドメインモードとマルチドメインモードをインストール時に選択することが可能になりました。ドメインを1つしか利用しないケースでは管理ドメインを起動せずに済むためマシンリソースを節約することができます。

補足:マルチドメインモードはすべての WebOTX Edition でサポー

ドメインモードの選択

インストール時にドメインモードを選択してインストールを行います。インストール後にもドメインモードの変更は可能です。

- マルチドメインモード（デフォルト）

マルチドメインモードでは、複数のドメインを作成することができ、管理ドメインと複数のユーザドメインから構成されるシステム構成です。

- シングルドメインモード（V6.4 新機能）

シングルドメインモードでは、管理ドメインは起動せず、ユーザドメインとして domain1 だけが起動し、1つのドメインから構成されるシステムです。

ドメインモードの切り替え

ドメインモードを切り替える場合は以下の手順で変更します。

1.WebOTXを停止します。

2.ドメインモードの変更は、以下の設定ファイル中の AS_BOOT_DOMAIN の値を修正します。

Windows ： ${INSTALL_ROOT}/config/asenv.bat

UNIX ： ${INSTALL_ROOT}/config/asenv.conf

AS_BOOT_DOMAIN の値でサーバ起動時に最初に起動されるドメイン名を指定します。マルチドメインモードの場合は管理ドメイン名である WebOTXAdmin を、シングルドメインモードの場合は domain1 を指定します。

マルチドメインモード ： AS_BOOT_DOMAIN=WebOTXAdmin

シングルドメインモード ： AS_BOOT_DOMAIN=domain1

シングルドメインモード時の制限事項

- シングルドメインモードでは管理ドメインが起動されません。統合運用管理ツールからの接続時は、ドメイン名に domain1、ポート番号に 6212(デフォルト)を指定し接続する必要があります。
- シングルドメインモードでは新規にユーザドメインを作成することはできません。ドメインの生成、削除を行う場合には、マルチドメインモードに変更し管理ドメインを起動する必要があります。
- 統合運用管理ツールのサーバー覧機能は表示できません。
- 統合運用管理ツールからリモートによるユーザドメイン起動・停止はできません。

2.1.5.ドメインの構成

業務システムのユーザドメインの構成について説明します。ドメイン構成は大きく次の 6 つのグループに分割することができます(DB サーバについては省略します)。これらについてそれぞれ独立したドメインに割り当てることも、同一のドメインに割り当てることも可能ですが、基本的な構成について次に説明します。

- Web Editionドメイン
- Standard-J Editionドメイン
- Standard/Enterprise Editionドメイン
- Webサーバドメイン

トされています。

補足:Web Edition において AP サーバドメインを作成することはできません。

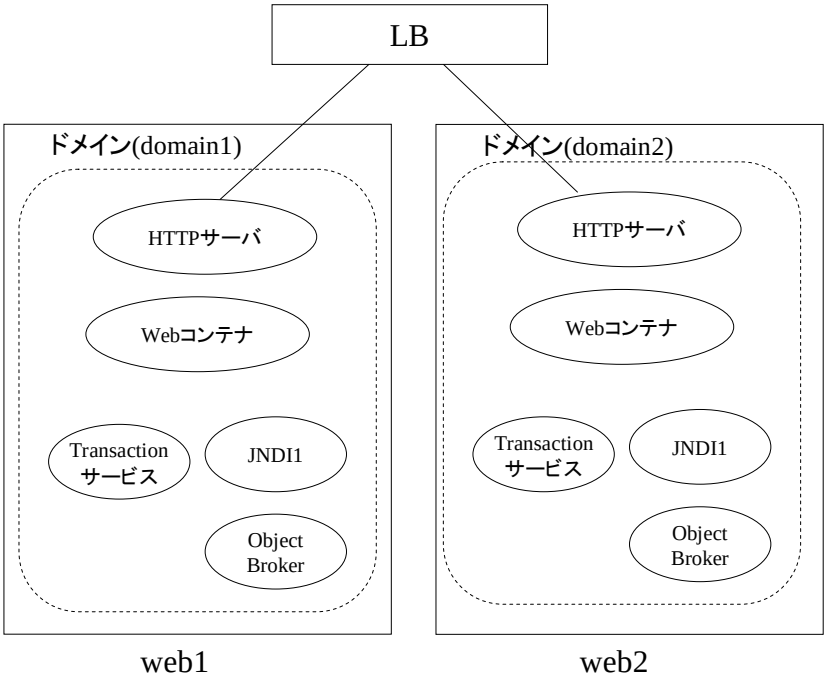
- APサーバドメイン
- 名前サーバドメイン

Web Edition ドメイン

Web Edition で動作するサービスを 1 つのドメインで動作させます。Web アプリケーションシステムを構築する上でもっともシンプルなドメイン構成です。1 台のマシンで全て動作させる場合は通常この構成となり以下のサービスが動作します。

- HTTP サーバ
- Web コンテナ
- JNDI サービス
- Transaction サービス
- Object Broker サービス

マルチサーバ構成の場合、マシン単位でドメインを 1 つ作成し負荷分散装置(Load Balancer)により分散を行います。



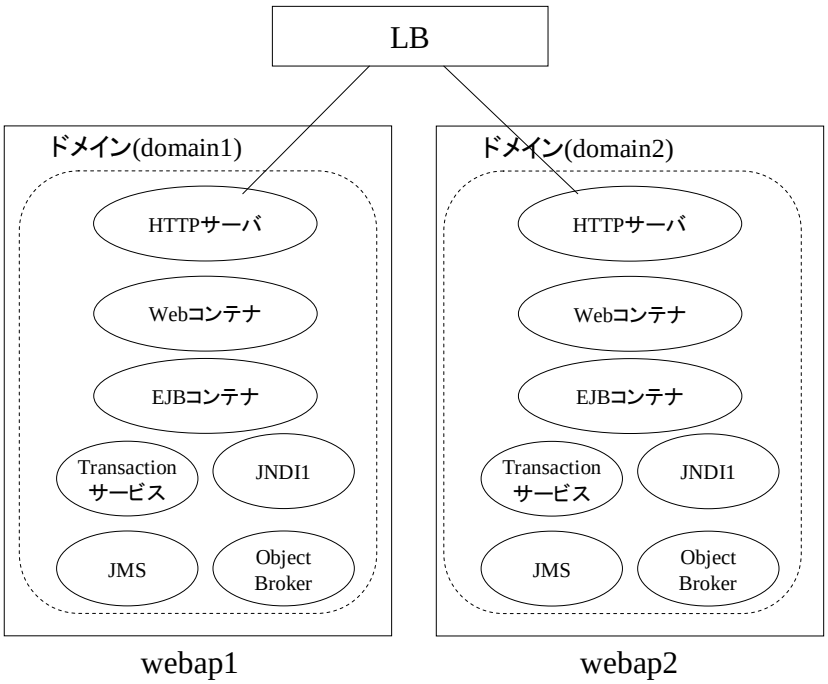
Standard-J Edition ドメイン

Standard-J Edition で動作するサービスを 1 つのドメインで動作させます。J2EE アプリケーションシステムを構築する上でもっともシンプルなドメイン構成です。1 台のマシンで全て動作させる場合や Web 層と AP 層を分割する必要のない場合は通常この構成となり以下のサービスが起動します。

- HTTP サーバ
- Web コンテナ
- EJB コンテナ
- JNDI サービス
- JMS サービス
- Transaction サービス

- Object Broker サービス

マルチサーバ構成の場合、マシン単位でドメインを 1 つ作成し負荷分散装置により分散を行います。

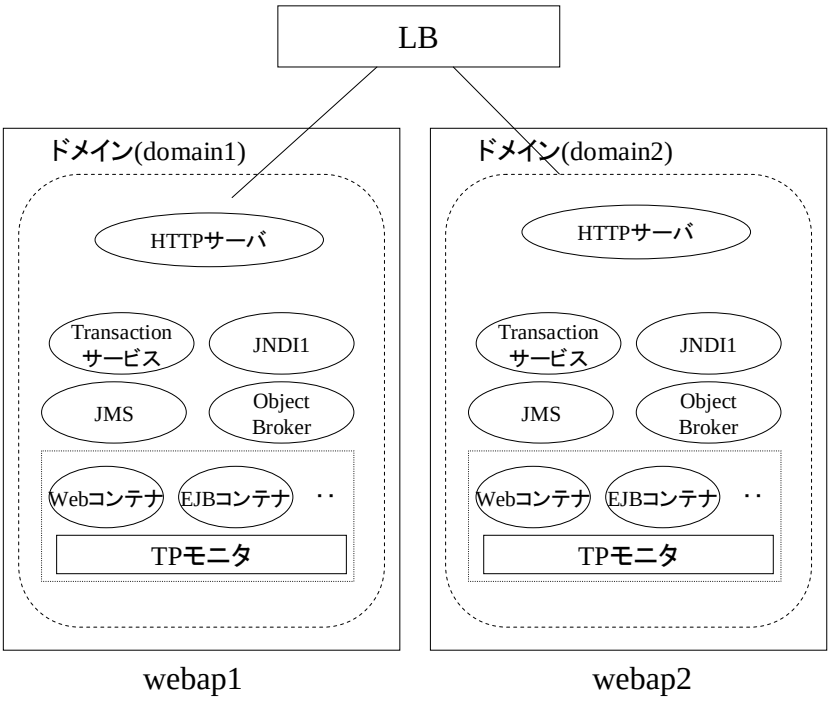


Standard/Enterprise Edition ドメイン

Standard/Enterprise Edition で動作するサービスを 1 つのドメインで動作させます。システムを構築する上でもっともシンプルなドメイン構成です。1 台のマシンで全て動作させる場合や Web 層と AP 層を分割する必要のない場合は通常この構成となり以下のサービスが起動します。

- HTTP サーバ
- Web コンテナ（独立 Java VM）
- EJB コンテナ（独立 Java VM）
- JNDI サービス
- JMS サービス
- Transaction サービス
- Object Broker サービス
- TP モニタ

マルチサーバ構成の場合、マシン単位でドメインを 1 つ作成し負荷分散装置により分散を行います。

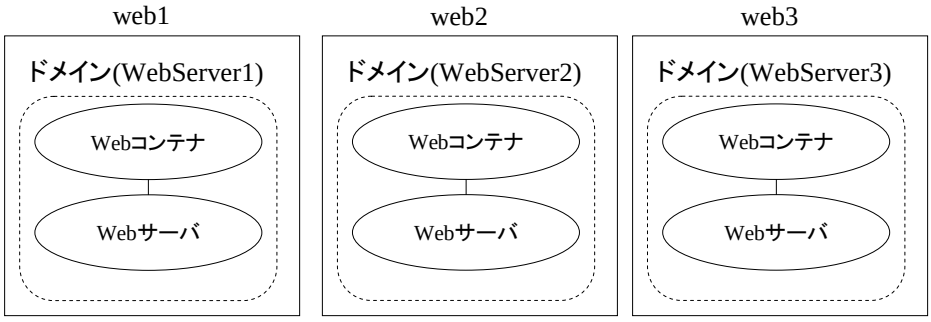


Web サーバドメイン

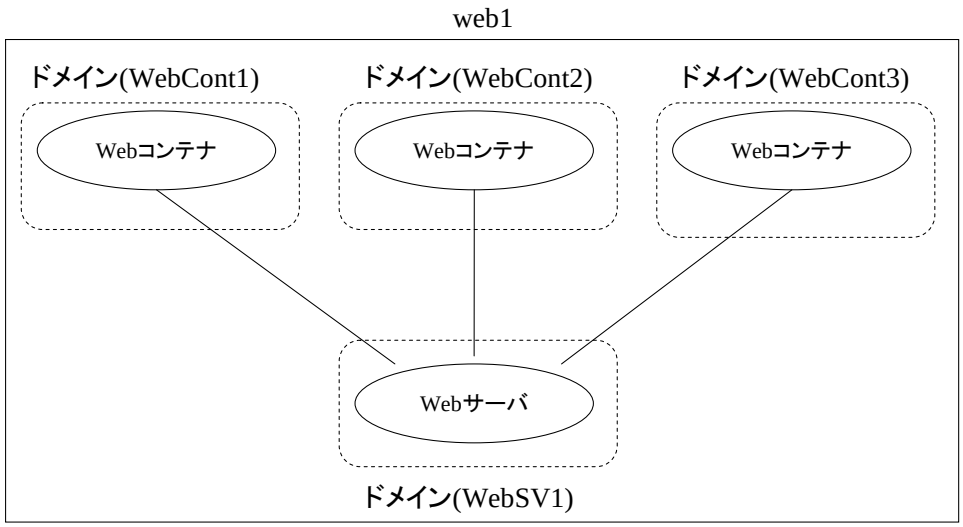
Web サーバ層を構成するドメインでは次のサービスが動作します。

- HTTP サーバ
- Web コンテナ

Web サーバはクラスタ化する場合、基本的にはマルチサーバ負荷分散構成となります。さらに 1 つのマシンでは 1 つの Web サーバ、Web コンテナが動作させる構成が基本です。



また、パターンによっては 1 マシン内で Web サーバと Web コンテナが 1 対 n になるような構成も可能です。この場合はそれぞれ(Web サーバ、Web コンテナ)が独立したドメイン構成となります。



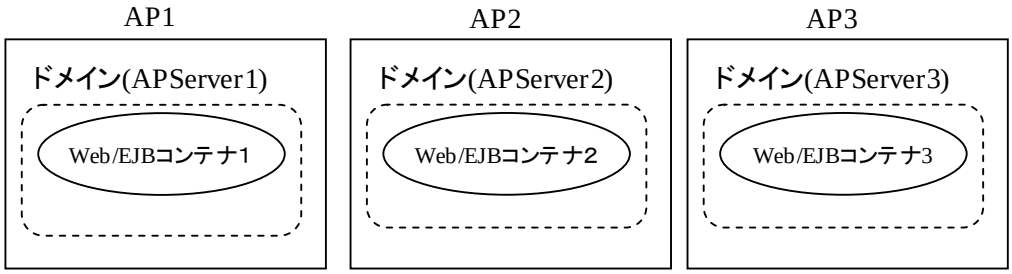
AP サーバドメイン

AP サーバ層を構成するドメインでは次のサービスが動作します。

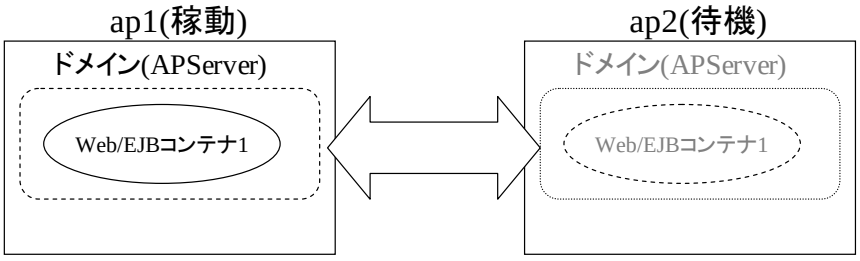
- Web/EJB コンテナ

AP サーバ層のクラスタとしては負荷分散がメインですが稼動待機構成(シングルスタンバイ、相互スタンバイ)も用いられ、また業務の特性に応じてマルチシステム構成となるため、柔軟な構成が望まれています。

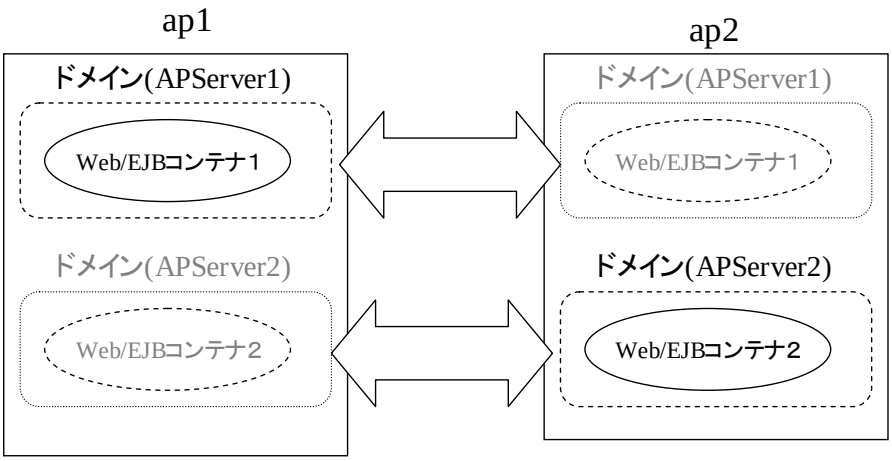
負荷分散構成:



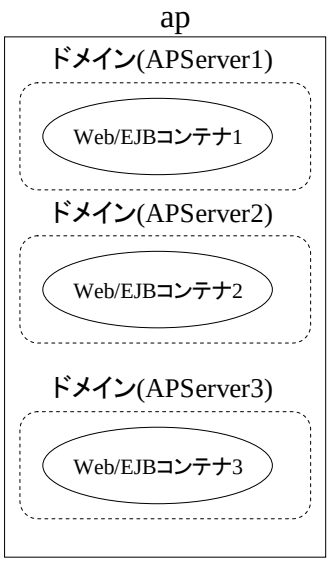
シングルスタンバイ構成:



相互スタンバイ構成:



マルチシステム構成:



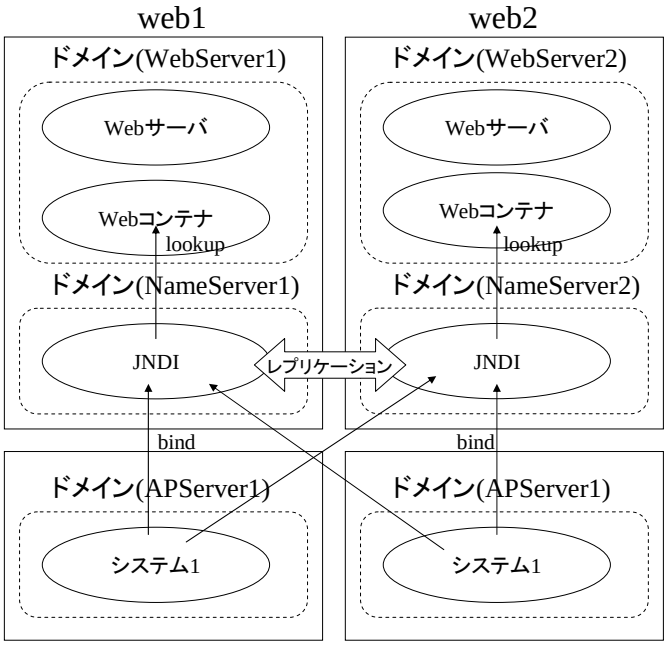
名前サーバドメイン

名前サーバ層を構成するドメインでは次のサービスが動作します。

- Object Broker 名前サーバ
- JNDI サーバ

名前サーバ自身は負荷分散クラスタなど、負荷を分散する構成にする必要はないが、基本的にはシステムで 1 つの存在であり、SPOF(Single Point of Failure)とならないための対策が必要となります。また名前サーバ専用にマシンを割り当てる構成は、コスト的に敬遠されるためさまざまな箇所に配置されることが想定されます。

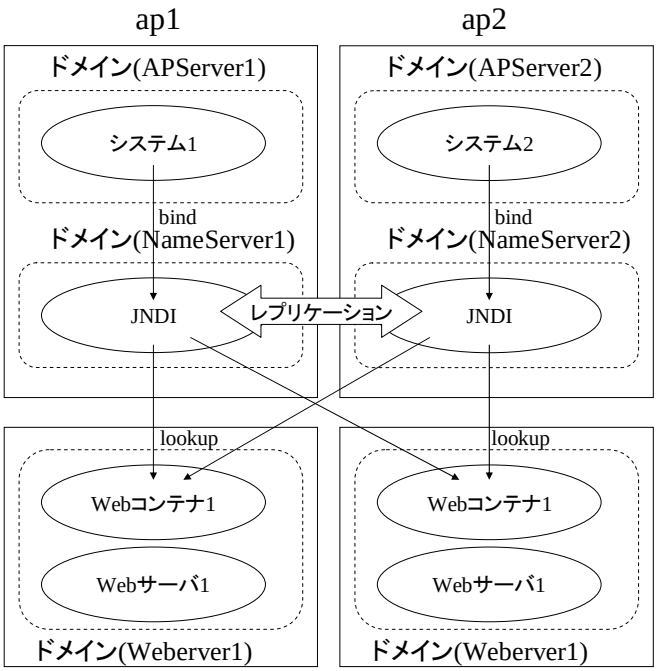
- Web サーバに配置
  Web サーバ上に配置する場合は、Web サーバと 1 対 1 で配置します。各名前サーバには全ての AP サーバの名前情報が格納されるように設定します。Web サーバ上の名前サーバは、AP サーバ負荷分散実装するためを全ての AP サーバ上の情報を持ち、どの Web サーバの名前サーバも同一の情報を保持しています。



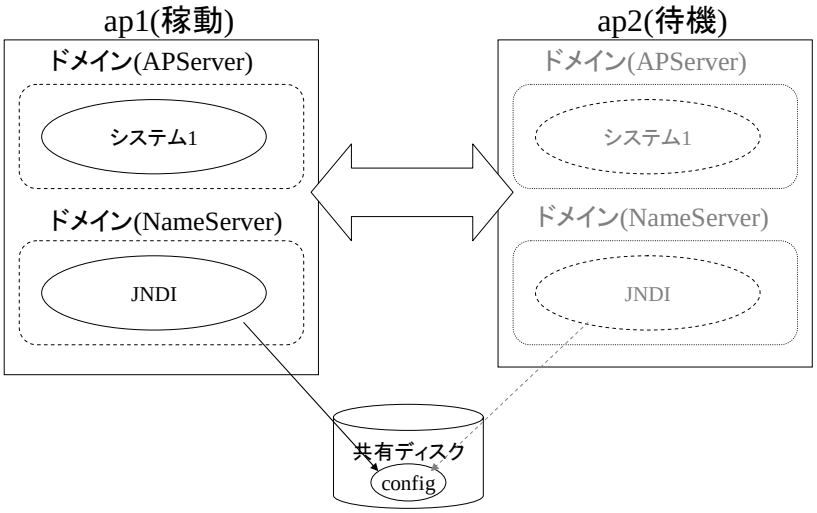
- AP サーバに配置

AP サーバ上に配置する場合は、AP サーバのクラスタ構成によって構成が変わります。

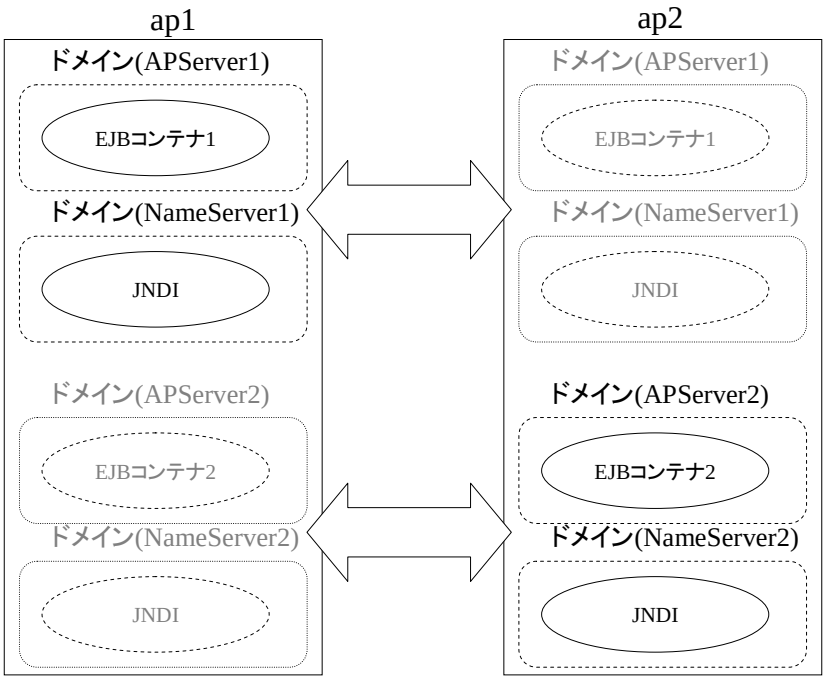
負荷分散構成:
負荷分散構成の場合、1 マシンに 1 名前サーバを配置します。AP サーバ負荷分散実装するため名前サーバ間でレプリケーションを行い、全ての AP サーバ上の情報を格納します、名前サーバは全て同一の情報を保持しています。



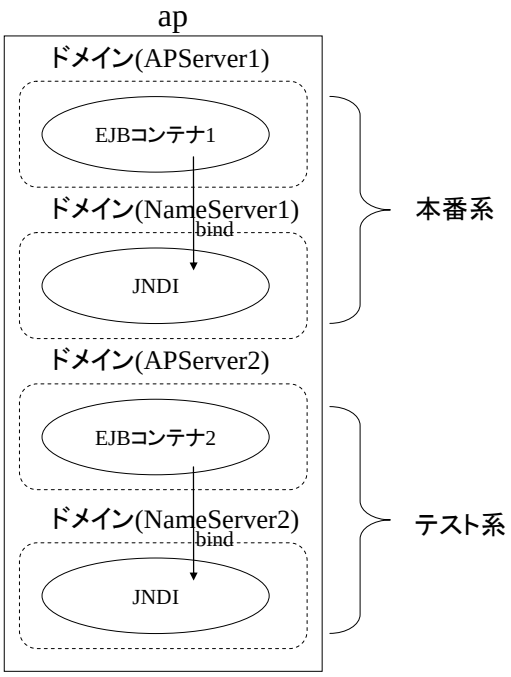
シングルスタンバイ構成:
シングルスタンバイ構成の場合、稼動系マシンに 1 名前サーバとなります。異常時には AP サーバと同様、名前サーバもフェイルオーバします。この場合、フェイルオーバしても登録情報を引き継げる必要があるため情報を永続化します。

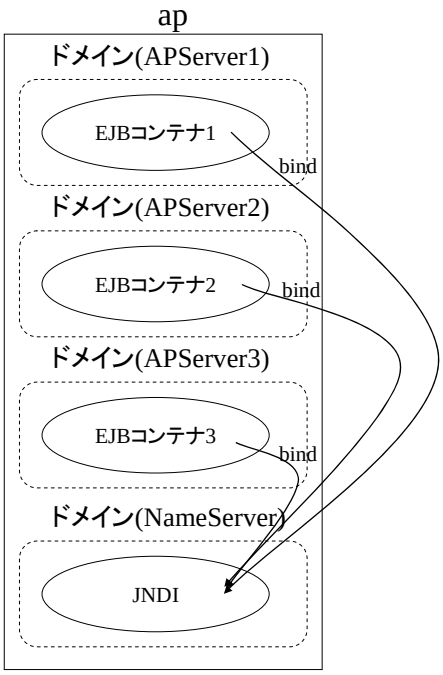


相互スタンバイ構成:
相互スタンバイ構成の場合、AP サーバドメイン 1 つにつき 1 名前サーバとなります。この場合、縮退時には 1 マシン上に複数名前サーバが稼動します。



マルチシステム構成:
マルチシステム構成の場合、システムごとに名前サーバを配置するか、1 つにするかは要件に依存します。通常は 1 つで名前空間をわけることで問題はありませんが、本番、テスト系のような構成の場合、運用上名前サーバも本番、テスト系で別運用となるため独立運用できることが必要です。





2.2.実装している仕様について

WebOTX の運用管理基盤で、Java で標準仕様とされている以下の仕様に対応しています。WebOTX が提供するシステム管理ツールを利用してシステムを運用管理行う上ではこの仕様をほとんど意識する必要はありませんが、システム構成を検討したり、システム構築したり、運用管理 API を使って独自の運用管理アプリケーションを作成する 上で参考になります。それぞれの仕様の詳細についてはそれぞれの JSR より公開されている仕様を参照してください。

- Java Management Extensions (JMX) – JSR-3
- Java Management Extensions (JMX) Remote API – JSR-160
- J2EE Management – JSR-77

2.2.1.JMX

WebOTX における JMX の実装について説明します。

JMX 仕様は Java 言語によるネットワークマネージメントアーキテクチャで、マルチサーバで構成されたシステムをリモートの運用管理アプリケーションから一元的に運用管理するための API やサービスを提供します。WebOTX では JMX で規定された運用管理基盤を提供しています。

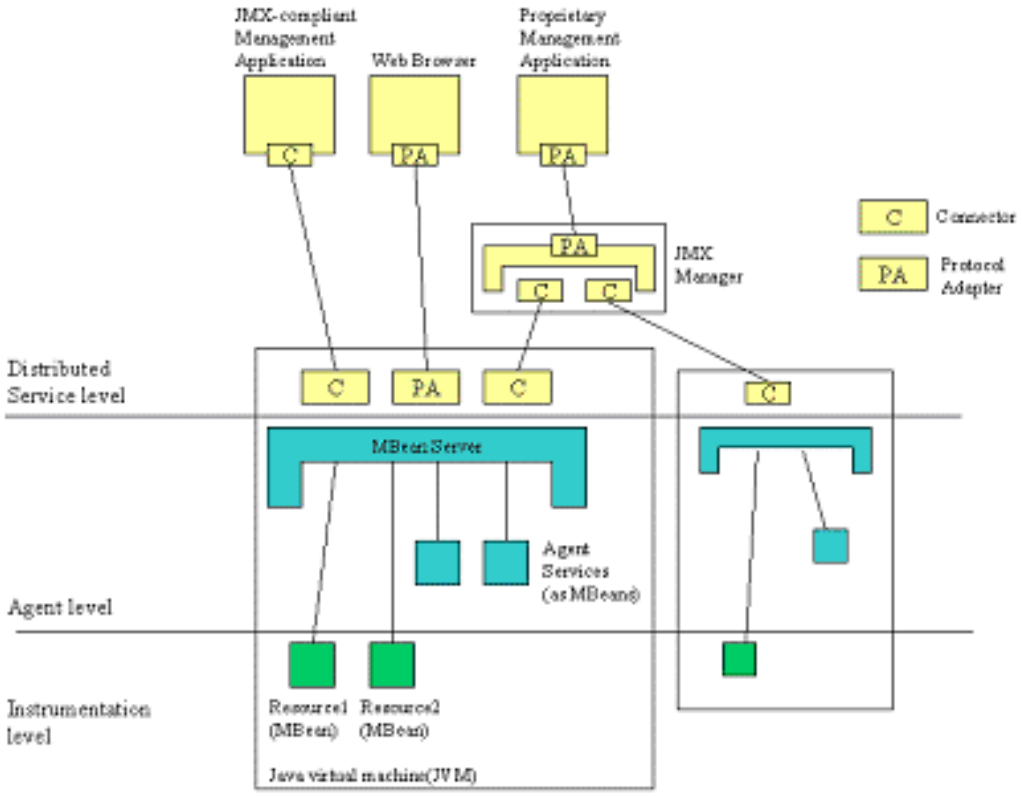
JMX のアーキテクチャ

JMX で規定されているアーキテクチャについて説明します。

JMX アーキテクチャの構成

JMX アーキテクチャは次の 3 レベルで構成されており、エージェントを通して運用管理クライアントから管理リソースを制御する仕組みとなっています。

- Instrumentation level
- Agent level
- Distributed services level



Instrumentation level

Instrumentation level は JMX 管理リソースの実装を提供します。JMX 管理リソースは具体的にはサービス、デバイス、ユーザ、アプリケーションを指します。この管理リソースを JMX 準拠の Java Bean である Managed Bean(MBean)によって管理します。MBean は JMX エージェントにより管理されます。

Instrumentation level ではさらに Notification メカニズムを提供します。Notification 機能により管理リソースで発生したさまざまなイベント(状態の変更、コンフィグレーションの変更、重大障害の発生など)をリモートの運用

管理クライアントに通知することができます。

MBean

MBean は対象管理リソースのマネージメントインタフェースを提供する Java オブジェクトです。管理リソースごとに MBean は提供されます。

MBean の要素

MBean のマネージメントインタフェースは次の要素を提供します。

- 属性（Attribute）
  管理リソースのコンフィグレーション情報を属性として定義します。MBean の属性を取得、設定することによりコンフィグレーション情報の参照、更新が行えます。MBean は属性を取得、設定するためのメソッド（getter メソッド、setter メソッド）を提供します。
- オペレーション（Operation）
  起動、停止など管理リソースに対する操作をオペレーションとして定義します。
- 通知（Notification）
  管理リソースから通知するイベントを notification として定義します。運用管理クライアントは notification を受信することにより管理リソースで発生したイベントをリアルタイムで知ることができます。
- コンストラクタ（Constructor）
  コンストラクタのインタフェースを提供することにより、MBean を生成しエージェントに登録を行うことができます。

MBean の属性およびオペレーションは Java の public メソッドとして提供されます。運用管理クライアントは管理対象リソースの運用管理を行うため、JMX エージェントを通じて MBean の属性を取得、設定したり、オペレーションを実行したり、Notification イベントを受信したりします。

MBean のタイプ

JMX は 4 つのタイプの MBean を規定しています。WebOTX では全ての MBean を model MBean として提供しています。よって本書ではこれ以降 MBean と表現している箇所は全て model MBean として扱います。

- standard MBean
  もっともシンプルデザインの MBean、実装の MBean でメソッド名がそのままマネージメントインタフェースとなります。
- dynamic MBean
  特定のインタフェースをインプリメントし、実行時に動的にマネージメントインタフェースを提供することができます。
- open MBean
  dynamic MBean の一種で、属性やオペレーションやコンストラクタの型が basic data tpye（プリミティブな型）で構成されており、ユーザフレンドリな管理環境を提供します。
- model MBean
  dynamic MBean の一種で、起動時に自動生成され、リソースを動的に効率よく管理するためのさまざまなポリシーが実装されています。

Notification

Notification は運用管理クライアントにリソースの状態やコンフィグレーションが変更されたことを通知するために利用します。

運用管理クライアントはリスナーを該当リソースの MBean に登録します。MBean は登録しているリスナーに対して 1 度のみイベントの通知を行ないます。JMX の仕様では notification モデルは MBean と同じ JVM にある Agent 間の通知について規定しており、リモートプロセスへの通知についてはサポートしていませんが、JMX Remote API でリモートへの通知をサポートしており WebOTX でもリモートプロセスへの通知を実装しています。

以下に JMX で規定されている Notification インタフェースについて説明します。

- Notification クラス
  発生したイベントの内容を定義します。JMX では次の内容が規定されています。

① Notification Type
発生した Notification がどのようなイベントかを特定するための識別子です。Notification ごとに一意に識別できる'.' セパレートの文字列です。なお JMX で規定されている Notification は"jmx."というプレフィックスをつけています。WebOTX で規定されている Notification は"webotx."というプレフィックスをつけています。

② Sequence Number
エージェントごとに発生した Notification を識別するための番号です。イベントが発生するたびにインクリメントされます。

③ Time Stamp
イベントが発生した時間

④ Message
イベントの詳細を説明する文字列

⑤ User Data
リスナーにイベントについて補足するための任意の情報
WebOTX では Notification クラスを拡張した WebOTXEventNotification クラスを提供し次の独自の情報を追加しています。

⑥ Subsystem ID
イベントの発生元のサブシステムを識別するための ID

⑦ Catalog ID
イベントの発生元のモジュールを識別するための ID

⑧ Message ID
発生したイベントの種別を識別するための ID

- MonitorNotification クラス
Notification からの派生クラスで、モニタリングイベントを受信するための実装です。MBean の属性のモニタリングを行なうためにモニタリング設定を行い、閾値オーバなどのイベントが発生した場合に MonitorNotification で定義している内容が通知されます。

- NotificationListener インタフェース
MBean から送られるイベントを受信するための実装です。イベントを受信したいクライアントはエージェントにリスナーの登録を行います。具体的には NotificationListener インタフェースを MBeanServer の AddNotificationListener メソッドによりエージェントに登録します。

- NotificationFilter インタフェース
MBean から送られる通知をフィルタリングするための実装です。該当 MBean が提供する Notification について受信したい Notification を特定するためにフィルタを設定します。例えば Notification Type や Subsystem ID などでフィルタを設定することができます。なおフィルタはリスナー登録時(MBeanServer の AddNotificationListener メソッド)に設定します。

- NotificationEmitter インタフェース
MBean が通知を行うための実装です。Notification をサポートする MBean は必ず実装しなければなりません。WebOTX で提供する MBean は全て NotificationEmitter インタフェースを実装しています。

- MBeanNotificationInfo クラス
JMX では MBean がどのような Notification をサポートしているかを事前に取得することが可能です。それが MBeanNotificationInfo クラスで Notification についてのさまざまな情報を得ることができます。

model MBean のポリシー

model MBeanはリソース管理を容易に行えるようにするためさまざまなポリシーが規定されています。なお MBeanのポリシーの詳細については「2.5.1ポリシー」を参照ください。

- Notification ポリシー
- Notification ロギングポリシー
- Persistence ポリシー
- Cache ポリシー

- Protocol Map ポリシー
- Export ポリシー
- Visibility ポリシー
- Presentation ポリシー

MBean インタフェース

WebOTX の提供する MBean は ModelMBean インタフェースを実装しています。ModelMBean インタフェースは管理リソースをコントロールするためのインタフェースについて規定しています。以下にその概要を示します。

- MBean 情報(MBeanInfo)の提供
  MBean がどのような属性、オペレーション、コンストラクタ、Notification を提供しているかを取得するインタフェースです。また MBean、属性、オペレーション、コンストラクタ、Notification が提供しているディスクリプタについても MBeanInfo を通して取得することができます。
- 属性の取得/設定
  MBean の属性の取得/設定するためのインタフェースです。
- オペレーションの実行
  MBean のオペレーションを実行するためのインタフェースです。
- MBean の永続化
  MBean が持つ永続化ポリシーの設定がある属性の値を外部ストレージに格納したり、外部ストレージからその属性値を復元したりするためのインタフェースです。WebOTX では永続化情報は xml ファイルとして格納します。
- Notification リスナーの登録/削除
  MBean が提供する Notification を受信するためのインタフェースです。
- Notification の通知
  MBean に Notification 通知を行わせるためのインタフェースです。

Agent level

Agent level はエージェントの実装を提供します。エージェントは直接管理リソースをコントロールし、リモートのマネージメントアプリケーション(運用管理クライアント)からリソースの利用を可能にします。

JMX Agent は MBean server と MBean を扱うためのサービス群で構成されています。Distributed Service level でエージェントにアクセスするためのアダプタやコネクタを利用します。

MBeanServer

MBeanServer はエージェント内で 1 つ作成され、エージェントに登録された全ての MBean を管理します。外部モジュールに対して MBean にアクセスするためのインタフェースを提供します。ただし MBeanServer インタフェースはエージェント内の同一 Java VM にあるモジュールで利用することができますが、リモートプロセスに対してそのインタフェースを公開していません。リモートプロセスに対してのインタフェースについては JMX Remote API で規定されています。

MBeanServer インタフェース

以下に MBeanServer が提供するインタフェースの概要について説明します。

- Notification リスナーの登録/削除
- MBean の作成/登録/削除
- 登録されている MBean の数の取得
- MBean の検索
- 指定した MBean の属性取得/設定
- 指定した MBean のオペレーション実行
- ドメイン名の取得

MBean の識別

MBean はエージェントにより一元管理されます。よってエージェントやクライアントが MBean を一意に識別するための識別子が必要となります。JMX では ObjectName を、各 MBean を識別するための識別子として規定しています。以下に ObjectName のフォーマットについて説明します。クライアントは MBean を特定するのに ObjectName を使用することになります。

[フォーマット]

[domainName]:property=*value*[,property=*value*,…]

[説明]

domainName	大文字・小文字を区別する文字列で、MBean の名前空間を指定します。 MBeanServer 自身の DomainName はデフォルトドメイン名として扱われ、省略することができます。省略した場合は MBeanServer により自身の DomainName を補完して MBean の検索を行います。これにより運用管理クライアントはドメイン名を知らなくても省略してプロパティ値のみで MBean の指定を行うことができます。 DomainName は'.'セパレートされたネットワークドメイン名とは逆の記述で表現します。 例) jp.co.nec.mydomain
property=value	個々の MBean を一意に決定するためのプロパティとその値を示す文字列です。 少なくとも必ず 1 つは指定しなければなりません。

補足:
ObjectNameのフォーマットについてはJ2EE Management仕様でさらに詳細に規定されています。WebOTXのMBeanのObjectNameについてはJ2EE ManagementのObjectNameフォーマットに準拠しています。詳細は「2.2.3J2EE Management」を参照ください。

MBean の検索

クライアントが管理リソースに対して何かアクション(属性の取得、設定、オペレーションの実行、Notification リスナーの登録)を行う場合、まずそのリソースに対応する MBean の ObjectName を取得する必要があります。MBeanServer インタフェースで MBean にアクセスするためのメソッドの引数には必ず ObjectName を指定します。そこで MBeanServer のインタフェースでは MBean の検索するためのメソッド(queryNames,queryMBeans)を提供しています。JMX における MBean の検索方法について説明します。

queryNames,queryMBeans メソッドも引数は同じです。返却値が引数で指定した条件にマッチする ObjectName クラスの set か MBean クラスの set かの違いがあるだけです。

queryNames,queryMBeans メソッドの第一引数は ObjectName です。この ObjectName では以下のワイルドカードを使用することが可能です。指定した ObjectName にマッチする MBean を検索することになります。

'*':空の 1 つを含むどんな文字シーケンスとも一致します。

'?':1 つのどんな単一文字とも一致します。

以下に例を示します。

次の ObjectName で MBeans が MBeanServer に登録されると仮定します。

デフォルトドメイン名は"DefaultDomain"とします。

```
MyDomain:description=Printer,type=laser
MyDomain:description=Disk,capacity=2
DefaultDomain:description=Disk,capacity=1
DefaultDomain:description=Printer,type=ink
DefaultDomain:description=Printer,type=laser,date=1993
Socrates:description=Printer,type=laser,date=1993
```

- "*:*" は MBean サーバの全てのオブジェクトと一致します。 パターンとして使用される無効のストリング・オブジェクトあるいは空のストリング("")名は、"*:*"と等価です。
- ":*" はデフォルトのドメイン(DefaultDomain)の全てのオブジェクトと一致します。

- “MyDomain:*”は MyDomain の中の全てのオブジェクトが一致します。
- “??Domain:*”は MyDomain の中でオブジェクトとすべて一致します。
- “*Dom*:*”は MyDomain と DefaultDomain の中の全てのオブジェクトと一致します。
- “*:description=Printer,type=laser,*” は、次のオブジェクトと一致します。
  MyDomain:description=Printer,type=laser
  DefaultDomain:description=Printer,type=laser,date=1993
  Socrates:description=Printer,type=laser,date=1993
- “*Domain:description=Printer,*” は、次のオブジェクトと一致します。
  MyDomain:description=Printer,type=laser
  DefaultDomain:description=Printer,type=ink
  DefaultDomain:description=Printer,type=laser,date=1993

queryNames,queryMBeans メソッドの第二引数は QueryExp です。QueryExp はユーザ定義の属性値を基に MBean のフィルタリング条件を指定します。なお QueryExp を null に指定するとフィルタリングは行いません。

QueryExp は属性値の constraints(数値については“equals”や”less-than”、文字列については”matches”のようなもの)から構成されています。この constraints は”and”や”or”のようなオペレーションも含みます。

例えば次のような Expression が定義できます。

“Retrieve the MBeans for which the attribute age is at least 20 and the attribute name starts with G and ends with ling”

(属性”age”の値が少なくとも 20 であり、属性”name”の値が G から始まり ling で終わる MBean を検索する)

AgentService

AgentService は MBeanServer にデフォルトで登録されている MBean のオペレーションを実行するためのオブジェクトです。JMX では次に示す AgentService を規定しています。WebOTX のエージェントでもサポートしています。

- Dynamic class loading
  management applet(m-let)のを利用して、任意のネットワークロケーションから動的に MBean のクラスやライブラリをロードするサービスです。
- Monitors
  MBean の属性値を監視し、更新があった場合、他のいくつかのオブジェクトに通知を行うサービスです。
- Timers
  one-time alarm clock での通知もしくは周期的に繰り返す通知ベースのスケジューリングメカニズムを提供するサービスです。
- Relation service
  relation type を元に MBean 間の関連を管理するサービスです。

Distributed service level

Distribute Services level は JMX Manager の実装を提供します。Agent を操作するまたは Agent を階層化するためのインタフェースやコンポーネントについて定義します。

これらのコンポーネントの機能について以下に説明します。ただし JMX 1.2 では Distribute Services level について具体的な仕様を規定していません。

- コネクタを通してエージェントにアクセスするためのインタフェースを提供します。
- リソースを運用管理するためのさまざまな形態のクライアントを提供します。
- 複数の JMX エージェントの運用管理を提供します。
- セキュリティ(ユーザ認証、暗号化)を提供します。

コネクタ・アダプタ

リモートの運用管理クライアントが、管理リソースをコントロールするには MBean に対してアクションするためのインタフェースと通信プロトコルおよびリモート通信におけるセキュリティが必要です。このインタフェースと通信プロトコルおよびセキュリティを提供するのがコネクタ・アダプタです。JMX で規定されたクライアントとエージ

ェントとの間で利用するのがコネクタで、既存プロトコル(HTTP を利用した Web ブラウザや SMTP を利用した運用管理アプリケーション)を使用するアプリケーションからエージェントにアクセスを行うために利用するのがアダプタです。アダプタは既存プロトコルから JMX のプロトコルにプロトコル変換を行う機能を有しています。

コネクタについては JMX Remote API で仕様が規定されています。

運用管理クライアント

運用管理クライアントは大きく 2 種類に分かれます。なお以下のツールの詳細については「2.3システム管理ツール」を参照ください。

- JMX 準拠の運用管理クライアント
  JMX(厳密には JMX Remote API)で規定している API を用いてエージェントにアクセスするクライアントです。
- 既存プロトコルを利用している運用管理クライアント
  HTTP や SMTP など既存で定義されているプロトコルを用いてエージェントにアクセスするクライアントです。

WebOTX では運用管理を行うため次のクライアントを提供しています。これ以外にも運用管理 API を利用して独自にカスタマイズした運用管理クライアントを作成することが可能です。

- 統合運用管理ツール
  JMX 準拠のインタフェース、複数サーバに分散している全てのエージェントにアクセス可能な GUI クライアントです。
- 運用管理コンソール
  HTTP アダプタを通して HTTP プロトコルでエージェントにアクセスする Web ベースの GUI クライアントです。
- 運用管理コマンド
  JMX 準拠のインタフェースでエージェントにアクセスするコマンドベースのクライアントです。シェルスクリプトやバッチファイルから利用できます。

2.2.2.JMX Remote

WebOTX における JMX Remote API の実装について説明します。

JMX 仕様ではリモートの運用管理クライアントからどのようにエージェントにアクセスするのかはアダプタ・コネクタを用いてアクセスするというガイドラインは示していましたが、詳細な仕様を規定していません。JMX Remote API では JMX 準拠の運用管理クライアントが利用するコネクタの仕様について規定しています。WebOTX では JMX Remote API で規定されたコネクタとそのコネクタを利用した JMX 準拠の運用管理クライアントを提供しています。

コネクタ

リモートの運用管理クライアントから MBeanServer にアクセスするために JMX Remote API ではいくつかのプロトコルを利用したコネクタインタフェース(MBeanServerConnection インタフェース)について規定しています。コネクタインタフェースについてはプロトコルに依存せず、MBeanServer のインタフェースと同様なものを提供しています。

コネクタの種類

JMX Remote API としては、クライアント、サーバ間の通信プロトコルにより、標準で次の 2 つのコネクタについて規定しています。

- RMI コネクタ
  リモート呼び出しを行う場合、Java で一般的に用いられている Remote Method Invocation (RMI)で通信を行います。
- Generic コネクタ
  クライアント、サーバ間で JMX Remote API で独自に規定したメッセージをやりとりすることにより通信を行うコネクタです。実際に通信に利用するプロトコルについては任意ですが、JMX Remote API では TCP ベースの JMX Messaging Protocol (JMXMP)について実装を提供しています。

WebOTX では以下のコネクタを提供しています。

- JMXMP コネクタ

エージェントへの接続

コネクタを利用してエージェントに接続する方式について説明します。JMX Remote API では次の 2 通りの方式について規定しています。

- JMXServiceURL でサーバを指定
  クライアントがエージェントのアドレス、ポート番号、コネクタのサポートプロトコルがわかっている場合、JMXServiceURL を用いてエージェントに直接アクセスすることができます。Web ブラウザが直接 URL を指定して目的のホームページにアクセスするのと同じイメージです。
  JMXServiceURL は次の様な形式です。

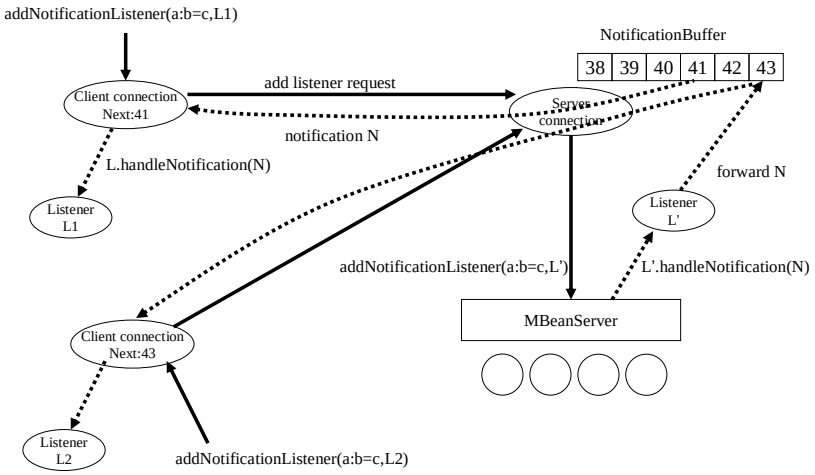
  service:jmx:*protocol*://*host*[:*port*][*url-path*]
  例) service:jmx:jmxmp//agent1:6212
  　　ホスト名"agent1"の JMXMP コネクタ(ポート 6212)にアクセスする場合

Notification

JMX Remote API ではリモートの運用管理クライアントに対してエージェントで発生したイベントを通知する機能(Notification)について規定しています。JMX Remote API の Notification は非同期で実装されており、バッファ上にイベントが存在する限り、そのイベントを受け取ることができます。WebOTX でもこれに基づいたリモートへの Notification を提供します。

コネクタは全ての MBean で発生する Notification イベントをハンドルして運用管理クライアントに通知する機能を実装しています。そのためにコネクタは MBean の作成、削除イベントを監視し、MBean が作成されると、リスナーのプロキシ(下図の L')を MBeanServer に登録します。こうして、コネクタは運用管理クライアントの接続の有無にかかわらず、エージェントで発生する全てのイベントをハンドルします。非同期通知を実装するため、コネクタはハンドルしている全ての Notification イベントを格納する NotificationBuffer を保持しています。
NotificationBuffer はシーケンス番号を元にイベントを管理しており、新しいイベントは前回のシーケンス番号をインクリメントして割り振られます。また、バッファのサイズは有限であり、もしバッファがいっぱいになった場合は、最も古いイベントが削除されます。

クライアント側で addNotificationListener メソッドによりエージェントに対してリスナーの登録を行うと、クライアントコネクションは次に検索するシーケンス番号を保持します。コネクタ側で、クライアントが持つシーケンス番号からイベントを検索しフィルタ条件に一致するイベントがあれば、そのイベントを運用管理クライアントに通知します。



2.2.3.J2EE Management

WebOTX における J2EE Management の実装について説明します。

J2EE Management 仕様は J2EE において、Java エンタープライズ・マネージメントやモニタリング・サービスを実現する共通フレームワークとして、JMX を基盤機能として規定されています。J2EE Management 仕様の目的は、J2EE が提供する情報にアクセスするため、より良い定義モデルを供給する J2EE アーキテクチャの、基本的な管理可能の概観を抽象化することにあります。加えてこの仕様では、プラットフォームリソースのモニタリングやコントロールに参加する J2EE コンポーネントに操作が共通の標準 API を定義しています。J2EE Management 仕様で規定している基本機能は次のとおりです。

- ディスカバリ – 管理システムの管理オブジェクトを発見したり操作したりする機能
- イベント – 管理オブジェクトで発生した重要なイベントの通知を受け取る機能
- 状態管理 – 管理オブジェクトの実行状態を監視しコントロールする機能
- パフォーマンス – 管理オブジェクトの基本性能統計をモニタリングする機能

なお J2EE Management 仕様は上記以外に既存マネージメントプロトコル(SNMP/CIM)とのマッピングについても規定していますが、これについては WebOTX V6 では未サポートであるため説明は省略します。

管理オブジェクト

J2EE Management 仕様では、管理リソース単位で管理オブジェクト(Managed Object 以下 MO と略します)を定義しています。管理オブジェクトはリソースのコンフィグレーションの取得や設定、状態管理、パフォーマンス情報の取得をするためのメソッドを提供します。管理オブジェクトは JMX 仕様で規定されている MBean で実装されています。J2EE Management 仕様では J2EE アプリケーションサーバを構成する全ての管理リソースについて具体的に管理オブジェクトのインタフェースを規定しています。これに従い WebOTX も全ての管理リソースを管理オブジェクトとして JMX の model MBean の形式で提供しています。

また J2EE Management 仕様では、次に示す管理オブジェクトとしての振る舞いを規定しています。MO は MBean であるので属性、オペレーション、コンストラクタ、notification をインタフェースとして提供します。

WebOTX では基本的には J2EE Management 仕様に準拠した MO として実装していますが、J2EE RI など流用元で提供されている一部 MBean(統計情報を管理する StatsHolderMBean など)については以下に示す ObjectName フォーマットに従っていないものもあります。

ObjectName

JMX 仕様で個々の MBean を識別するための情報として ObjectName が用いられていますが、J2EE Management でも MO の識別には ObjectName を使用しています。J2EE の ObjectName は JMX フォーマットで定められた形式であるがいくつか必須のプロパティを規定しています。

[フォーマット]

[domainName]:J2EEType=*j2eeType*,name=*name*, <parent-j2eeType>=<*parent ManagedObject name*>
[,property=*value*,…]

[説明]

domainName	JMX 仕様	大文字・小文字を区別する文字列で、MBean の名前空間を指定します。MBeanServer 自身の DomainName はデフォルトドメイン名として扱われ、省略することができます。省略した場合は MBeanServer により自身の DomainName を補完して MBean の検索を行います。これにより運用管理クライアントはドメイン名を知らなくても省略してプロパティ値のみで MBean の指定を行うこと

		ができます。DomainName は'.'セパレートされたネットワークドメイン名とは逆の記述で表現します。 例）jp.co.nec.mydomain
J2EEType=*j2eeType*	J2EE 仕様	MO の種別を表す。J2EE においていくつかの J2EEType が規定されています。WebOTX 独自の J2EEType もいくつか規定しており、WebOTX のプレフィックスをつけています。必須プロパティです。
name=*name*	J2EE 仕様	個々の MO につけられたオブジェクト名称。必須プロパティです。
<parent-j2eeType>= *<parentManagedObject name>*	J2EE 仕様	MO 間の親子関係を識別するため、子の MO については親の J2EEType とその name プロパティの値を指定します。子の MO については必須プロパティです。
property=*value*	JMX 仕様	上記のプロパティ以外で、個々の MO を一意に決定するためのプロパティとその値を示す文字列です。

補足:J2EE 仕様では MO の種別を表すプロパティとして j2eeType を用いますが、WebOTX の提供する一部は tpye プロパティを用いています。

[例]

- ドメイン名が FirstEverBank の、J2EEDomain 管理オブジェクトの objectName

  FirstEverBank:j2eeType=J2EEDomain,name=FirstEverBank

- このドメインの J2EEServer

  FirstEverBank:j2eeType=J2EEServer,name=BankServer1

- この J2EEServer に配備した J2EEApplication

  FirstEverBank:j2eeType=J2EEApplication,name=AccountsController,J2EEServer=BankServer1

- その J2EEApplication の EJBModule

  FirstEverBank:j2eeType=EJBModule,name=BankAccount,J2EEApplication=AccountsController,J2EEServer=BankServer1

- EJBModule のエンティティ Bean

  FirstEverBank:j2eeType=EntityBean,name=Account,EJBModule=BankAccount,J2EEApplication=AccountsController,J2EEServer=BankServer1

必須属性

MO として必ず持たなければならない属性が J2EE Management 仕様により規定されています。MO がどのような特性をもつのかを表します。これらの属性は J2EEManagedObject インタフェースで規定されており、MO は必ずこのインタフェースを実装しなければなりません。

- boolean stateManageable

  true であれば MO が状態管理を行っていることを示します。

- boolean statisticsProvider

  true であれば MO がパフォーマンス情報を提供していることを示します。

- boolean eventProvider

true であれば MO がイベント通知を行うことを示しています。イベントは JMX の Notification 機能を利用して通知されます。

状態管理

J2EEMangement では状態のあるリソースの MO には状態管理を行い、運用管理クライアントに対してその状態を取得できるインタフェースを提供するように規定されています。さらに状態に変更があった場合、その変更をイベントとして通知することも規定されています。状態管理を行うためのインタフェースとして StateManageable インタフェースが用意されており、MO はそのインタフェースを実装するようになっています。WebOTX で提供する MBean も状態管理を提供するものは全てこのインタフェースを実装しています。

StateManageable インタフェースで規定されているインタフェースの概要を以下に示します。

- 状態を表す属性
- 開始した時刻を示す属性
- 開始メソッド
- 停止メソッド

パフォーマンスモニタリング

J2EE Managementでは管理リソースのパフォーマンス情報を提供するため、リソース毎に提供すべきパフォーマンス情報について規定しています。それぞれのMOに対して提供すべきパフォーマンス情報を定義するインタフェースを規定しています。例えばEJBのMOについてはEJBStatsというインタフェースを規定し、そのインタフェースを実装したMOを提供しています。パフォーマンス情報を提供するために以下のインタフェースを規定しています。なおパフォーマンスモニタリングの詳細については「運用編(モニタリング)」を参照してください。

Statistic インタフェース

Statistic インタフェースおよびそのサブインタフェースは、パフォーマンスデータを供給するのに必要なアクセサを規定します。以下のインタフェースが規定されています。

- Statistic

  パフォーマンスデータとして基本的な属性を提供します。

属性名	概要
name	パフォーマンスデータの名称
unit	パフォーマンスデータの単位
description	パフォーマンスデータの概要
startTime	パフォーマンスデータを採取し始めた時間
lastUpdateTime	パフォーマンスデータを最後に取得した時間

- CountStatistic

  カウンタで表されるパフォーマンスデータのためのインタフェース。例えばオペレーションの呼び出し回数などが該当します。

属性名	概要
count	パフォーマンスデータの値(回数)

- TimeStatistic

  時間で表されるパフォーマンスデータのためのインタフェース。例えばオペレーションのレスポンスタイム

が該当します。

属性名	概要
count	パフォーマンスデータを採取した回数。
maxTime	採取した中での最大値。
minTime	採取した中での最小値。
totalTime	採取したパフォーマンスデータの合計値。count で割ると平均値となる。

- RangeStatistic

  値が増減するパフォーマンスデータのためのインタフェース。例えば単位時間あたりのオペレーション呼び出し回数が該当します。

属性名	概要
highWaterMark	採取した中での最大値。
lowWaterMark	採取した中での最小値。
current	カレント値

- BoundaryStatistic

  上限値、下限値をもつパフォーマンスデータのためのインタフェース

属性名	概要
upperBound	上限値
lowerBound	下限値

- BoundedRangeStatistic

  RangeStatistic と BoundedRangeStatistic を派生したインタフェース、値が増減し上限値、下限値をもつパフォーマンスデータのためのインタフェース

属性名	概要
highWaterMark	採取した中での最大値。
lowWaterMark	採取した中での最小値。
current	カレント値
upperBound	上限値
lowerBound	下限値

Stats インタフェース

Statsインタフェースおよびそのサブインタフェースは、おのおの指定されたMOのためのパフォーマンスデータのアクセサを規定します。各Statsインタフェースは提供するパフォーマンスデータを適切なStatisticインタフェースとして提供します。J2EE Managementとして以下のインタフェースが規定されています。なおWebOTXでは以下に加え独自の統計情報を取得するためのStasインタフェースを拡張しています。拡張したインタフェースについては「運用編（モニタリング）」を参照してください。

- EJBStats

  EJB に関するパフォーマンスデータ

- EntityBeanStats

  Entity Bean に関するパフォーマンスデータ

- JavaMailStats

  JavaMail に関するパフォーマンスデータ

- JCAStats

  JCA リソースアダプタに関するパフォーマンスデータ

- JDBCStats

  JDBC に関するパフォーマンスデータ

- JMSStats

  JMS に関するパフォーマンスデータ

- JTAStats

  JTA に関するパフォーマンスデータ

- JVMStats

  JVM に関するパフォーマンスデータ

- MessageDrivenBeanStats

  メッセージドリブン Bean に関するパフォーマンスデータ

- ServletStats

  Servlet に関するパフォーマンスデータ

- SessionBeanStats

  Session Bean に関するパフォーマンスデータ

- StatefulSessionBeanStats

  ステートフル Session Bean に関するパフォーマンスデータ

- StatelessSessionBeanStats

  ステートレス Session Bean に関するパフォーマンスデータ

- URLStats

  URL に関するパフォーマンスデータ

イベント

J2EEMangement ではイベント通知の実装については JMX の Notification モデルで実装することを規定しています。WebOTX でも JMX および JMX Remote API の Notification モデルをサポートします。以下に WebOTX が標準的に提供するイベントの概要について示します。

- MO(MBean)が登録された、削除された
- MO(MBean)の属性値の値が更新された

2.3.システム管理ツール

WebOTX ではアプリケーション実行環境を運用操作するための次のツールを提供しています。これらのツール群を利用することでリモートの運用コンソールより複数の実行環境(Web サーバ,AP サーバ)の統合的な運用ができます。

システム管理ツールの詳細については「運用編(統合運用管理ツール)」を参照してください。

2.3.1.統合運用管理ツール

GUI により複数のサーバに分散した運用操作が行えます。

2.3.2.Web 版統合運用管理コンソール

Flash を利用したブラウザベースの GUI により、分散した複数のサーバの運用操作が一箇所で行なえます。

（セキュリティ上の配慮から、統合運用管理ツールに比べていくつか機能の制限があります）

2.3.3.運用管理コンソール

ブラウザベースで GUI により運用操作が行えます。運用管理コンソールでは主に Web コンテナの操作を行うことができます。

2.3.4.運用管理コマンド

コマンドベースでコマンドライン上から運用操作が行えます。運用管理コマンドでは全ての運用操作を行うことができます。

以下に運用管理コマンドで操作可能な項目について示します。

2.3.5.運用管理 API

プログラミングにより、独自の統合運用管理ツールを作成することができます。標準の JMX および JMX Remote API を利用してプログラミングすることが可能です。

2.4.ドメイン構成情報ファイル

ドメインの設定情報を格納する各構成情報ファイルについて説明します。

2.4.1.domain.xml

各ドメインの構成情報については xml ファイル(domain.xml)にその設定内容を保持しています。domain.xml の構成情報については config MBean にて全て定義されているため MBean の属性の取得もしくは変更により構成情報を参照、設定することができます。通常 domain.xml を直接参照、変更する必要はありません。

注意事項

- domain.xml を編集する場合は該当するドメインは停止している状態で行ってください。
- 変更を行った結果、XML の解析が行えない場合はドメインの起動ができなくなります。変更する場合は必ずバックアップを作成してください。
- 他のドメインの domain.xml をコピーして使用することはできません。

2.4.2.domains-config.xml

この設定ファイルは、管理ドメイン(WebOTXAdmin)にのみ定義されるファイルです。

管理ドメインが管理するユーザドメインの情報を保持しています。 管理ドメインはこの設定ファイルに従いユーザドメインの自動起動の設定、起動順番などを制御します。domains-config.xml の構成情報については config MBean にて全て定義されているため MBean の属性の取得もしくは変更により構成情報を参照、設定することができます。通常 domains-config.xml を直接参照、変更する必要はありません。

注意事項

- domains-config.xml を編集する場合は管理ドメインが停止している状態で行ってください。
- この設定ファイルで直接ドメインの追加、削除はできません。削除した場合、ドメインのサービスが不正に残る可能性があります。
- 変更を行った結果、XML の解析が行えない場合は管理ドメインの起動ができなくなります。変更する場合は必ずバックアップを作成してください。

2.4.3.log4otx.xml

ドメイン内で動作するWebOTXの各システムサービスのログ出力に関する設定を保持しています。log4j の設定ファイルのフォーマットに従っています。WebOTXのログレベルについては、統合運用管理ツール、および、運用管理コマンドから動的な設定変更が可能です。ログのファイルサイズや世代数など、ログレベル以外の設定を行う場合には直接 log4otx.xml を編集します。ドメインの再起動後に有効となります。

2.4.4.log4j.xml

ドメイン内で動作する log4j パッケージを利用しているアプリケーションの為にログ出力に関するデフォルト設定を保持しています。log4j パッケージや commons-logging を利用して作成したアプリケーションを利用する場合は、log4j の設定ファイルのフォーマットに従い、直接 log4j.xml を編集します。

また、システムプロパティ log4j.configurationを変更し独自の設定ファイルを定義することも可能です。詳細は、「運用編(ロギング)」編の「2.6 アプリケーションがlog4jを利用する場合の注意点」を参照してください。

2.4.5.logging.properties

ドメイン内で動作する Java LoggingAPI を利用しているアプリケーションの為にログ出力に関するデフォルト設定を保持しています。Java LoggingAPI を利用して作成したアプリケーションを利用する場合は、Java Logging の設定ファイルのフォーマットに従い、直接 logging.properties を編集します。

2.4.6.server.policy

ドメインのセキュリティポリシに関する設定を保持しています。エージェントのセキュリティポリシを変更するためにはこのファイルを編集します。

2.4.7.otx.conf

各ドメイン共通のシステム環境変数を保持しています。
UNIX では /etc/WebOTX/otx.conf に格納されています。各ドメインに共通のシステム環境変数を与えたい場合はこのファイルを編集します。

なお Windows ではこのファイルを編集するのではなく、OS の[システムのプロパティ]で設定してください。

Standard/Enterprise Edition の場合、otx.conf で行った設定は各プロセスグループの設定に反映されます。

例) Oracle の設定を加える場合
ORACLE_HOME=/opt/oracle/u01/app/oracle/product/10.1.0
LD_LIBRARY_PATH=/opt/oracle/u01/app/oracle/product/10.1.0/lib
export ORACLE_HOME LD_LIBRARY_PATH
※ 環境変数を有効にするため、必ず export してください。

また、例えば LD_LIBRARY_PATH が既に設定されている場合は、次のように記述すると設定を引き継ぐことができます。
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/oracle/u01/app/oracle/product/10.1.0/lib
export LD_LIBRARY_PATH

Oracle ライブラリの読み込みを行う場合、上記のように ORACLE_HOME を設定するだけではなく、oracle のインストールディレクトリやファイルに対して、WebOTX 運用ユーザの参照権と実行権を付加する必要があります。

2.5.WebOTX Model MBean

model MBean は JMX 仕様で規定されている MBean であり、WebOTX ではこの ModelMBean を独自に機能強化しています。WebOTX Model MBean の機能と特長について説明します。

2.5.1.ポリシー

model MBean はリソース管理を容易に行えるようにするためさまざまなポリシーが規定されています。以下に model MBean で規定されているポリシーについて説明します。

なお以下に説明するポリシーのフィールドについては WebOTX 側で適切な値を設定しています。利用者がその値を変更することはできません。

Notification ポリシー

model MBean は提供する属性に対して更新（SetAttribute/SetAttributes メソッドが実行された）があった場合、属性変更通知（jmx.attribute.change）を通知する実装がなされています。

Notification ロギングポリシー

MBean で発生した全ての Notification 通知をログファイルに記録する仕組みを実装しています。ログ採取の有無やログファイル名を設定します。

なお WebOTX で提供する MBean については Notification ロギングポリシーフィールドを設定しません。

フィールド名	型	概要
Log	boolean	ロギングを行うかどうか
logFile	String	ログファイル名

Persistence ポリシー

MBean のコンフィグレーション情報などを永続的に扱うために、model MBean ではファイルなど外部リソースに属性の値を格納するインタフェースをサポートしています。永続化をサポートしない場合、MBean の属性値は一時的なものになり、システムの再起動などを行った場合、設定された値はリセットされます。WebOTX では永続化が必要なものについては MBean 毎に xml ファイルとして永続化する実装を行っています。

また、永続化するタイミングについて以下に示すポリシーを PersistPolicy フィールドに設定します。これらは MBean の属性の特性に応じて設定します。

- Never:

  永続化をサポートしない。属性は一時的である。

- OnTimer:

  persistPeriod で指定した間隔で、その間に属性値の更新が合った場合、外部リソースに保存します。

- OnUpdate：

  属性値の更新を行ったタイミングで外部リソースに保存します。

- NoMoreOftenThan：

  属性値の更新を行ったタイミングで、なおかつ前回更新時より persistPeriod 経過している場合、外部リソースに保存します。これは頻繁に更新されるデータに対して更新回数を抑えられる特長を持ちます。

フィールド名	型	概要
persistPorlicy	String	PersistPolicy の値以下うちいずれか Never,OnTimer,OnUpdate,NoMoreOftenThan
persistLocation	String	永続情報を格納するディレクトリ名
persistName	String	永続情報を格納するファイル名
persistPeriod	int	persistPolicy フィールドの値が OnTimer または NoMoreOftenThan である場合のみ、有効です。OnTimer については、その属性はその値が最初に設定する時に始まる各 persistPeriod の最初の部分で格納されます。NoMoreOftenThan については、前回の格納以来 persistPeriod が経過していない場合を除いては、それが更新されるごとに、属性は格納されます。

Cache ポリシー

MBean では属性もしくはオペレーションの戻り値に対して、キャッシュポリシーとともにデータのキャッシュ値および既定値を保持しています。運用管理クライアントから、属性値の取得要求があった場合、キャッシュポリシーに従い保持しているキャッシュ値が有効であると判断した場合、リソースに対してデータ取得要求を行わず、キャッシュ値を返却する仕組みを実装しています。リソースとの直接のやりとりが行われないため、実行時のアプリケーションリソースと性能の管理活動のインパクトを最小限にすることができます。

MBean の属性単位で、秒単位で表現される currencyTimeLimit フィールド(キャッシュ値の有効期限)と lastUpdatedTimeStamp フィールド（前回更新時刻）が設定でき、運用管理クライアントから属性取得要求があった時刻が lastUpdateTimeStamp+currencyTimeLimit を過ぎている場合、属性値は無効と判断され、過ぎていない場合有効と判断されます。有効と判断された場合は即座にキャッシュ値を返却し、無効と判断された場合は getMethod フィールドで設定された getter メソッドを呼び出してリソースに対して属性値取得要求を行います。getMethod フィールドで getter メソッドが定義されていない場合、常に default フィールドで設定された既定値が返されます。

フィールド名	型	概要
currencyTimeLimit	int	value がセットされるときから最新であり古くない秒単位の期間。value が current の場合、保存された値は返され、getMethod(もし定義されれば)は起動されません。currencyTimeLimit が 0 である場合、値はすべてのリクエストのために検索されなければなりません。currencyTimeLimit が -1 である場合、値は古くなりません。
default	String	value がセットされず、getMethod が定義されない場合、返されることになっている値です。

getMethod	String	属性の値を検索するために使用されるオペレーション・ディスクリプタからのオペレーション名。返されたオブジェクトは、value フィールドで保存されます。
lastUpdatedTimeStamp	int	value フィールドが最後に更新された時のタイムスタンプ。
setMethod	String	管理されたリソースの中で属性の値をセットするために使用されるオペレーション・ディスクリプタからのオペレーション名。新しい値も value フィールドで保存されます。
value	String	もしセットされていれば、このフィールドの値は属性の現在値を表わすオブジェクトです。これは currencyTimeLimit が古くなっていない場合、返される属性のキャッシュされた値です。

Protocol Map ポリシー

MBean のデフォルト動作と API はほとんどのアプリケーションの管理ニーズを満たすことになっています。しかしながら model MBean は複雑なリソース管理のシナリオも提供します。model MBean API はアプリケーションの MBean の属性と MIB や CIM オブジェクトのような既存マネージメントデータモデルとのマッピングを ProtocolMap フィールドの定義に従って行うことができます。

例えば MIB generator は JMX エージェントと相互に作用し、SNMP 管理システムによってロードされた MIB ファイルを作成します。作成した MIB ファイルは JMX エージェントによって知られている resource 情報を表現します。

なお WebOTX で提供する MBean については ProtocolMap フィールドを設定しません。

フィールド名	型	概要
protocolMap	String	このフィールドの値はディスクリプタ・オブジェクトでなければなりません。プロトコル名とマップされたプロトコル値のペアのセットを含んでいます。これは、特別のプロトコル用に属性が特別の識別子(CIM スキーマ、SNMP MIB Oid など)に関係していることを可能にします。このディスクリプタは管理されたリソースによってセットされ、管理アプリケーションにこの属性を表わすためにヒントとしてアダプタによって使用されるべきです。

Export ポリシー

JMX エージェントがマルチ環境で動作する場合、各 JMX エージェントは適切なディレクトリかルックアップサービスで存在および有効性を公示することが運用管理を容易にする。JMX では MBean が現在どの JMX エージェントに登録されているかを、他の JMX エージェントからも位置を確定できるようするしくみを提供している。このようにディレクトリおよびルックアップサービスにより MBean の位置を公示する場合は、Export フィールドを定義するようになっています。

なお WebOTX で提供する MBean については Export フィールドを設定ません。

フィールド名	型	概要
export	String	その値はシリアライズ可能で MBean の位置を確定できるようにするのに必要な情報を含んでいるすべてのオブジェクトであ

		りえます。null は、MBean が他の JMX agent に公開されてはならないことを示します。定義された値は、MBean が他の JMX agent に公開されるべきでありさらに、JMX agent アドレスが未知の場合、検出可能であるべきであることを示します。MBeans と MBean サーバの export がサポートされない場合、このフィールドが無視されます。

Visibility ポリシー

運用管理を行うにあたって、運用管理クライアントを GUI で提供することは必要であるが、多種多様な MBean の属性やオペレーションをどのように見せるかについてポリシーを規定しています。

例えば、エンタープライズマネージャは全ての情報を見て、より高いレベルの管理オブジェクトと対話したいと思うかもしれません。ドメインマネージャは、一般にアプリケーションの内容だけを見たいと思うでしょう。JMX ではこのようにユーザの関心度に対応したクライアントビューを見せる仕組みを提供しています。

Visibility フィールドは MBean、属性あるいはオペレーション単位に 1～4 の整数で設定します。最大は 1 で、ユーザの関心度が最も高いことを示しています。最小は 4 であまり重要性のない、特別の場合だけ必要な情報であることを意味しています。なお JMX 仕様はこれら 1-4 のレベルを厳密に定義していません。MBean 作成者に定義を委ねています。WebOTX の統合運用管理ツールでは通常ユーザでは Visibility が 1 のものしか表示しません。

フィールド名	型	概要
visibility	int(1-4)	MBean のために細分性のレベルを示す 1～4 の整数です。1 は、大きく分け、最もしばしば見られる MBeans のためにあります。4 は最も小さく、恐らく見られる度合いが最も少ない MBeans です。この値はアダプタあるいは管理アプリケーションによって使用されます。

Presentation ポリシー

PresentationString フィールドは任意のディスクリプタに定義することができる文字列です。この文字列はコンソールに関するヒントを提供するための、XML にフォーマットされた文字列であると規定されています。しかしながが JMX 1.2 ではプレゼンテーションフィールドの標準のセットはまだ定義されておらず、WebOTX でも PresentationString については何も設定しません。

フィールド名	型	概要
presentationString	String	どのように属性を示さなければならないか説明する XML にコード化された文字列です。

2.5.2.MBean のタイプ

MBean は以下に示すタイプで分類され、その MBean の持つ ObjectName の category プロパティでその識別を行います。以下に WebOTX で提供する MBean のタイプについて説明します。

コンフィグ MBean(category=config)

ドメインに関する構成情報を格納します。設定値は全てドメイン構成情報ファイル(domain.xml)に格納されます。

ランタイム MBean(category=runtime)

各サービスやリソースに対応した MBean で、J2EE Management 仕様で規定された MO として実装しています。サービスやリソースに対する設定やオペレーションが行えます。

「2.6管理対象リソース・サービス」に記述しているMBeanが該当します。

モニターMBean(category=monitor)

パフォーマンス情報を管理するStatsHolderMBeanやモニタリングを行なうMonitorMBeanが該当します。管理対象リソースのパフォーマンス情報を取得するgetterメソッドおよび子モニターMBeanの一覧を取得するオペレーションを提供します。「運用編(モニタリング)」に記述しているMBeanが該当します。

2.5.3.dottedname(CLIName)

ObjectName は MO を一意に識別するための識別子として用いますが、MO 間の親子関係を簡潔に示すために CLIName という識別子も定義されています。ObjectName と CLIName は 1 対 1 でマッピングされているます。WebOTX ではこの CLIName を統合運用管理ツールのツリー表示や運用管理コマンドで MO を指定する識別名として利用しています。

[フォーマット]

CLIName はルートから区切り文字'.'で親 MO の Name プロパティの値で連結したものです。例えば上記れいにもある ObjectName "domain1:j2eeType=J2EEApplication,name=AccountsController,J2EEServer=server"の CLIName は

"server.applications.j2ee-application.AccountsController"

と表されます。

なお先頭に現れるルート識別子について以下に説明します。

domain	ドメインの MO が該当します。
server 名 (通常"server"となっている)	J2EE サーバで管理しているコンフィグ MBean を示しています。配備されたアプリケーション、登録されているリソース、動作するサービスなどの MO が該当します。
tpsystem	アプリケーショングループやプロセスグループなど Standard/Enterprise Edition における TP モニタ関連の MO が該当します。
applications	Standard/Enterprise Edition でプロセスグループに配備したアプリケーションおよび共有コンポーネントの MO が該当します。

なおdottednameについては運用管理コマンド(otxadmin)のlistコマンドで一覧を確認でき、getコマンドやsetコマンドで属性値の取得/変更が可能です。なお具体的な運用管理コマンドについての説明は「運用編(統合運用管理ツール)」を参照してください。

2.6.管理対象リソース・サービス

WebOTX では全てのリソースやサービスを J2EE Managemant で規定される Managed Object(JMX 仕様では MBean)として提供しています。これにより、運用管理操作は全て JMX インタフェースで行なうことが出来ます。なお提供する MBean については「運用編 MO 定義リファレンス」を参照ください。

2.7.JMXMP プロトコルについて

WebOTX では、JMX Remote API の通信プロトコルとして JMXMP (JMX Messaging Protocol)をサポートしています。以下に JMXMP プロトコルについて説明します。なお JMX Remote API を利用する場合には、プロファイル情報をサーバ／クライアントで一致させる必要があります。

2.7.1.使用するプロファイル情報

WebOTX で利用している JMX Remote API のプロファイル情報は以下の通りです。

プロパティ	説明
jmx.remote.profiles	"TLS SASL/PLAIN"
jmx.remote.tls.socket.factory	(証明書の情報から作成した SSLScocketFactory オブジェクト)
jmx.remote.tls.enabled.protocols	"TLSv1"
jmx.remote.tls.enabled.cipher.suites	"SSL_RSA_WITH_NULL_MD5"
jmx.remote.sasl.callback.handler	(クライアント) new com.nec.webotx.enterprise.admin.jmx.remote. client.UserPasswordCallbackHandler(*user*, *password*)) (サーバ) new com.nec.webotx.enterprise.admin.jmx.remote. server.PropertiesFileCallbackHandler("*${INSTANCE_ROOT}/config/keyfile*")

Callback 関数は、wosv-rt.jar で提供されています。