WebOTX Application Server V11 構築ガイド

2022 年 10 月 24 日

第 1 版

NEC

改訂履歴

版　数	更新日付	更新内容【概要】
1	2022/10/24	初版作成

目次

1. はじめに 1
1.1. はじめに 2
2. 詳細設計 4
2.1. 性能 5
2.2. 多重度 12
2.3. 通信/タイムアウト 29
2.4. 通信/セッション 40
2.5. セキュリティ 45
2.6. データベース連携 51
2.7. ログ 64
2.8. 監視 67
2.9. 配備 71
2.10. 分散 79
3. その他 92
3.1. 使用上の条件の確認 92
3.2. パラメーター覧 96
3.3. WebOTX で動作するプロセス 109

1.はじめに

「WebOTX Application Server V11 構築ガイド」は、WebOTX Application Server V11.1 以降の Standard または Standard Extended Option を導入される方を対象に、標準的な設計指針や構築方法についてわかりやすく記載しています。

WebOTX によく寄せられるお問合せの中から、性能、多重度、タイムアウト設計など、構築時にネックになりがちな項目に焦点を当て、システム構築におけるポイントや、パラメータ設計方法、構築手順などをご紹介し、WebOTX への理解をより深めていただけることを目的に作成されました。

多くのシステムに活用いただける内容を記載していますが、個々のシステム毎に環境や手順が異なる場合があります。本ガイドに記載できなかった内容については製品付属のマニュアルもあわせてご確認ください。

1.1.はじめに

本構築ガイドでは、以下の製品を対象としております。

- WebOTX Application Server Standard V11.1
- WebOTX Application Server Standard V11.1 Extended Option (※1)

(※1) Java および C++の CORBA アプリケーションの実行基盤を提供する WebOTX Application Server Standard のオプション製品。同一ライセンス方式の WebOTX Application Server Standard も同じ数必要。利用する場合、WebOTX Application Server Standard + Extended Option をインストール。

1.1.1.「WebOTX Application Server」とは

「WebOTX Application Server」とは 拡張性に優れた業務システム構築を実現するアプリケーションサーバです。

コストパフォーマンスに優れた小規模モデルからシステムの安定稼動を実現する高信頼な大規模モデルまで、長期に渡って安心してお使いいただける、企業システムの基盤インフラとして最適なミドルウェア製品です。

1.1.2.構築ガイドの目的

WebOTX Application Server を採用するシステム担当者が、導入・構築作業において迷うことがないように最低限設計すべき項目や設定を推奨する項目を標準化し、ガイドどおりに一連の作業を行うことで、一定の品質を保った WebOTX Application Server を利用したシステム構築や評価ができることを目指します。

1.1.3.構築ガイドを利用する対象者

想定する対象読者と主な利用方法は次のとおりです。

1) 営業(システム提案)、方式設計者、アプリケーション基盤担当者

   WebOTX Application Server の概要を理解し、サイジングやシステムの方式設計ができるようになります。

   WebOTX Application Server を利用したシステム基盤の基本設計・詳細設計にて利用することを想定しています。

2) システム基盤担当者、アプリケーション開発者、運用担当者

   WebOTX Application Server の構築、パラメータ設定、動作確認ができるようになります。

   WebOTX Application Server を利用したシステム環境の構築・評価にて利用することを想定しています。

1.1.4.本文の記述に関して

これ以降、本構築ガイドでは「WebOTX Application Server」を、「WebOTX」と記載します。

共通

共通タグ:プロセスグループ、エージェントプロセス共通項目です。

プロセスグループ

プロセスグループタグ:プロセスグループでの動作に関する記述がされております、エージェントプロセスで動作する場合は不要な項目です。

エージェントプロセス

エージェントプロセスタグ:エージェントプロセスでの動作に関する記述がされております、プロセスグループで動作する場合は不要な項目です。

なお、タグが付いていない章については、すべて共通項目となります。

2.詳細設計

詳細設計では、各設計項目に対しての説明と、設計指針、決定すべき項目を記述します。

1.　性能の設計

サーバの CPU、メモリなど限られたリソースの中で効率よく処理を行えるようにパラメータを設計します。

2.　多重度設計

プロセスの多重度、システム全体の多重度の構成方法を示します。

3.　通信/タイムアウト設計

通信する際のプロセスのタイムアウト、システム全体のタイムアウトの構成方法を示します。

4.　セッション設計

セッション情報の保持、タイムアウトなどセッション管理に関する考え方を示します。

5.　セキュリティ設計

通信時のセキュリティ設定に関する考え方を示します。

6.　DB連携設計

DBとの接続設定、コネクション管理、状態監視について設計の考え方を示します。

7.　ログ設計

ログの出力、ローテートに関する考え方を示します。

8.　監視設計

プロセス、メッセージ監視にて障害が検知された時の対処法について考え方を示します。

9.　配備設計

配備の仕組み、ライブラリの構造、読み込み順序が影響する点に関する考え方を示します。

10. 分散設計

負荷分散について説明します、特に AP サーバでの負荷分散方法で用いられる CNS の処理について説明します。

2.1.性能

システムの構築において、サーバの CPU、メモリ、OS のリソースを超えた運用を行うことはできません。そのため、限られたリソースの中で効率よく処理を行えるようにパラメータを設計します。性能設計では、主に以下の項目に対し、説明を行います。

1） プロセスグループに配備するか、エージェントプロセスに配備するかの選択

2） WebOTX が使用するメモリサイズの調整

3） 使用していないサービスを抑止する

性能設計においては、クライアントからの接続多重度やアプリケーションの実行多重度が大きく影響します。多重度設計に関しては「2.2 多重度」で詳しく説明します。

2.1.1.概要説明

1） アプリケーションの配備先の選択 共通

2） WebOTX が使用するメモリについて 共通

① 管理ドメインのプロセス

② ユーザドメインのプロセス

③ Java VM に対する設定

3） 使用していないサービスについて 共通

1） アプリケーションの配備先の選択 共通

Jakarta EE アプリケーションサーバにおいて、Web アプリケーションは Web コンテナ上で、EJB アプリケーションは EJB コンテナ上で動作します。WebOTX における Web 及び EJB コンテナは以下、何れかのパターンで動作させることができます。

➢ Web コンテナ・EJB コンテナ共にプロセスグループ上で動作する。

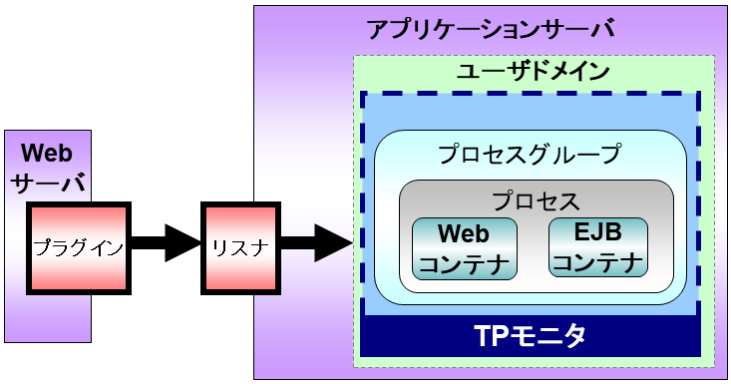


図 2.1.1. Web コンテナ、EJB コンテナの動作パターン①

Web アプリケーション、EJB アプリケーションをプロセスグループに配備した場合、Web コンテナ、EJBコンテナ上で動作するアプリケーションを、高信頼性基盤(TP モニタ)上で運用することができます

TP モニタ上で運用することにより、アプリケーションに対し以下の処理を実施することが可能となります。

- リクエストの流量制御
- アプリケーションのモニタリング
- 詳細な統計情報の取得

このため、信頼性、可用性が向上し、結果、ミッションクリティカル性が向上します。「WebOTX Application Server

Standard」では、プロセスグループによる運用を推奨しています。

➢ Web コンテナはエージェントプロセス上で、EJB コンテナはプロセスグループ上で動作する。

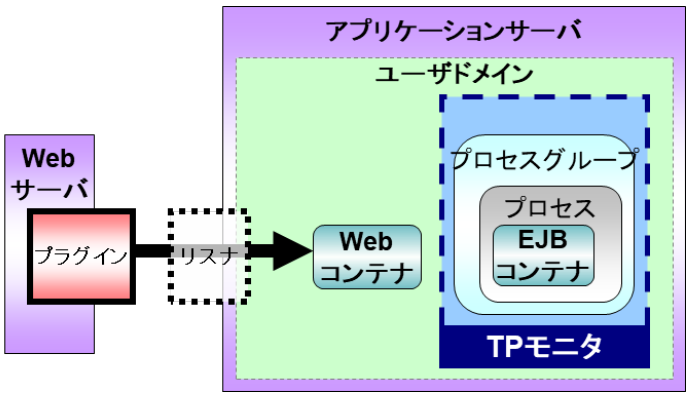


図 2.1.2. Web コンテナ、EJB コンテナの動作パターン②

Web アプリケーションをエージェントプロセスに、EJB アプリケーションをプロセスグループに配備した場合、EJB アプリケーションは TP モニタ上で運用することができますが、Web コンテナ上で動作する Web アプリケーションを TP モニタ上で運用することはできません。しかし、Web コンテナがプロセスグループ上に生成されないため、プロセスグループを複数作成した場合、リソースの節約ができます。

また、Web アプリケーションをエージェントプロセスに配備した場合、Web サーバから Web コンテナに転送する処理量が少なくなるため、Web サーバとアプリケーションサーバ間での通信速度を向上させることができます。

➢ Web コンテナ・EJB コンテナ共にエージェントプロセス上で動作する。

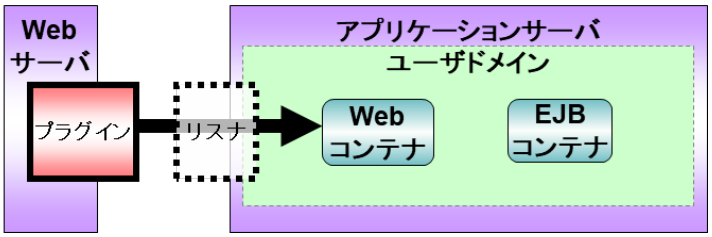


図 2.1.3. Web コンテナ、EJB コンテナの動作パターン③

Web アプリケーション、EJB アプリケーションをエージェントプロセスに配備した場合、Web アプリケーション、EJB アプリケーションの両方とも TP モニタ上で運用することができません。しかし、プロセスグループを使用しないため、マシンのリソースをより節約することができます。

また、Web アプリケーションをエージェントプロセスに配備した場合、Web サーバから Web コンテナに転送する処理量が少なくなるため、Web サーバとアプリケーションサーバ間での通信速度を向上させることができます。

プロセスグループ上での動作、エージェントプロセス上での動作のどちらかを選択し運用を行います。

2） WebOTX が使用するメモリについて 共通

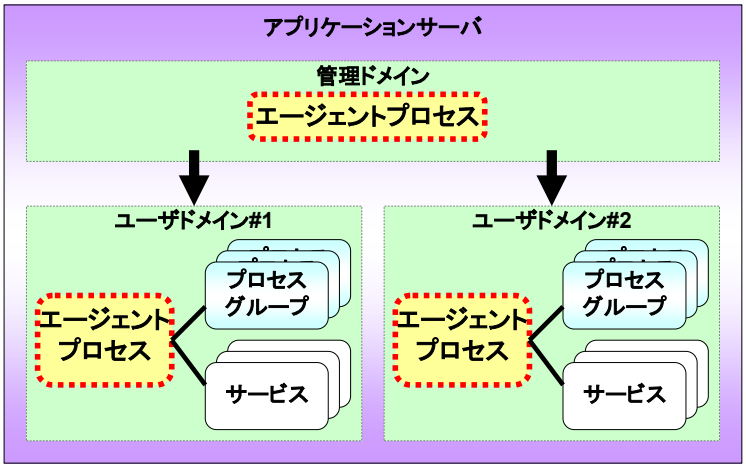


図 2.1.4. アプリケーションサーバ構成

WebOTX は管理ドメイン(admin)と、ユーザドメインから構成されており、これらのドメインも多くのプロセスにより構成されています。
このユーザドメインの中に、エージェントプロセス、サービスが提供するプロセス、プロセスグループのプロセスが含まれます。WebOTXが使用するメモリの調整が必要なプロセスとして、ユーザドメインのエージェントプロセスとプロセスグループのプロセスがあります。

- 管理ドメインのプロセス
  - 管理ドメイン(admin)のエージェントプロセス
    ユーザドメインの起動・停止を制御する管理用の Java プロセスです。サーバ単位に1つの管理ドメインが起動します。1つの管理ドメインと、複数のユーザドメインを起動することができます。この起動している管理ドメインを使用することで、ユーザドメインの操作を行うことができます。

    管理ドメインは Java プロセスとしておよそ、256MByte のメモリを使用します。

- ユーザドメインのプロセス
  - ユーザドメインのエージェントプロセス
    エージェントプロセスとは、ドメインにおいてコアとなる Java プロセスのことです。エージェントプロセスでは、サービスの管理、アプリケーションの配備処理や統計情報の収集を行います。配備するアプリケーションやセッション情報に保持するデータ量などによって必要なメモリサイズが変わってきます。

  - 各サービスプロセス
    各ドメインには、Jakarta EE サービスを提供するための各種プロセスが起動されます。Web サーバ、JMS(Java Messaging Service)、ObjectBroker サービス、TP モニタ関連プロセスなどが含まれます。 通常、各サービスプロセスのメモリサイズの調整は不要です。

  - プロセスグループのプロセス
    業務アプリケーションはエージェントプロセスから分離された個別のプロセス上で動作します。このプロセスは多重化することができ、多重化したプロセス群のことを「プロセスグループ」と呼んでいます。Webアプリケーション、EJBアプリケーションなどの業務アプリケーションは、プロセスグループ上で動作します。業務アプリケーションが使用するメモリに応じてプロセスグループのメモリサイズを調整します。

- Java VM のヒープメモリについて
  - Java VM のメモリサイズ

    Java VM はヒープ領域と非ヒープ領域(Metaspace 領域、C ヒープ領域)から構成されます。これらの領域の内、Java VM のメモリサイズを調整する場合、対象としては下記の設定を考慮する必要があります。

    - 最大ヒープサイズ
    - Metaspace 領域の最大値(-XX:MaxMetaspaceSize)

- Java VM のガーベッジコレクション(GC)
  Java VM はヒープの空きメモリ領域が少なくなるとガーベッジコレクション(GC)を実行し使用済みのオブジェクトを解放して空き領域を確保します。Full GC 実行中は、全ての処理が停止(数秒～数十秒)するため性能に大きな影響を及ぼします。そのために、最大ヒープサイズに適正値を設定する必要があります。
  一般的に最大ヒープサイズは使用ヒープサイズの 1.5～2 倍が良いとされています。

3) 使用していないサービスについて 共通
使用していないサービスを停止することで、サーバのリソースを節約することができます。

2.1.2.設計指針

性能設計をする上で必要となる指針を記載します。

1) プロセスグループに配備するか、エージェントプロセスに配備するかの選択 共通

2) WebOTX が使用するメモリサイズの設計 共通 プロセスグループ

- ヒープサイズの設計
- Metaspace 領域の設計
- GCログの出力設定

3) 使用していないサービスを抑止する 共通

- JMS サービスの無効化

1) プロセスグループに配備するか、エージェントプロセスに配備するかの選択 共通

概要説明を参照の後、システムの要件にしたがって選択してください。

2) WebOTX が使用するメモリサイズの設計 共通 プロセスグループ

共通

WebOTX のプロセスは Java 仮想マシン(Java VM)で動作しています。Java VM のメモリ領域をチューニングすることで、使用メモリ、ガーベッジコレクション(GC)の頻度などを考慮した設計を行うことができます。

設計対象となるのは、ユーザドメインのエージェントプロセス、プロセスグループに対するメモリサイズです。それらに対し別々にヒープサイズ、Metaspace、GC の観点から設計を行う必要があります。

- ヒープサイズの設計

  Java のヒープ領域は新規オブジェクトの配置場所である New 領域と長期間必要なオブジェクト Old 領域に分かれます。

<table>
<tr><th colspan="3">Heap</th></tr>
<tr><th>New</th><th colspan="2">Old</th></tr>
<tr><td>アプリケーション実行時に一時的に格納される部分</td><td>New領域に設定したサイズ部分</td><td>常駐する可能性があるオブジェクト</td></tr>
</table>

図 2.1.5. ヒープ領域の構成

- New 領域

  アプリケーション実行時に新規オブジェクトが格納される領域です。
  アプリケーションの実装に依存するため、期待通りのスループットが得られるか確認しながら値を調整する必要があります。

- Old 領域

  アプリケーション実行時に長期間必要であると判断されたオブジェクトが格納される領域です。New 領域のオブジェクトが登録される可能性があるため、New 領域に設定したサイズと、常駐する可能性があるオブジェクト

のサイズを合わせた値を指定する必要があります。

Old 領域の常駐する可能性があるオブジェクトのサイズを見積もるためには、実測値を計測する必要があります。

計測に際しては、アプリケーションを配備した上で、GC ログの出力設定を行い FullGC 直後のメモリサイズを確認してください。ここで計測した現在使用しているヒープサイズが最終的に常駐する可能性が高いオブジェクトとなります。

なお、全体として空きヒープ領域が少なくなると Full GC が頻発し性能が劣化します。

また、プロセスグループ、及びエージェントプロセスに対して上記の観点以外に考慮すべき留意点は下記の通りです。

- プロセスグループに対する留意点

  プロセスグループの場合、ヒープ領域が不足すると、プロセスグループのログファイル(<WebOTX インストールディレクトリ>/domains/<ドメイン名>/logs/tpsystem/<アプリケーショングループ名>/<プロセスグループ名>/<プロセスグループ名>.<プロセス ID>.log)に“OutOfMemoryError”のメッセージが出力されプロセスグループが異常終了します。プロセスグループでは異常終了した場合、自動でプロセスの再起動が行われます。

  なお、セッション情報を登録するアプリケーションをプロセスグループに配備する場合、セッション情報はプロセスグループ上に登録されるため、セッションオブジェクトのためのメモリが必要となります。そのため、セッション情報を登録する場合は、登録されるセッションオブジェクトの数、サイズ、使用しなくなったセッションオブジェクトを破棄するまでのタイムアウト時間などを考慮して、サイズを決定してください。

- エージェントプロセスに対する留意点

  エージェントプロセスの場合、一般的な設計に加え、登録するアプリケーション数やセッション情報のデータ量に応じてエージェントプロセスの最大ヒープサイズを拡張する必要があります。

  エージェントプロセスでは登録したアプリケーションを管理するため、アプリケーションのインタフェース数、オペレーション数が多くなると、より多くのヒープメモリを必要とします。

  ここでのインタフェースとは EJB や CORBA のインタフェースを、オペレーションとは EJB のメソッドや CORBA のオペレーションを指します。Web アプリケーションをプロセスグループに配備する場合、これに加えて、Web アプリケーションごとに POST/GET の 2 つがオペレーションとして計上されます。

  運用管理コンソール上では

  <EJB>

  ◆アプリケーション

  +-WebOTX Jakarta EE アプリケーション

  + -アプリケーション名

  + -EJB モジュール名

  + -各 Bean 名

  + -インタフェース名

  + -メソッド名

  <CORBA>

  ◆アプリケーション

  +-CORBA アプリケーション

  + -CORBA アプリケーション名

  + -CORBA モジュール名

  + -インタフェース名

+ -オペレーション名

の箇所で表されます。

およそ、インタフェース数が100増える毎に約 10MByte、オペレーション数が 100 増える毎に約 15MByte のヒープ領域が必要になります。

セッション情報を登録するアプリケーションをエージェントプロセスに配備する場合、セッション情報はエージェントプロセス上に登録されるため、セッションオブジェクトのためのメモリが必要となります。

また、エージェントプロセス、プロセスグループに関わらず、JNDI サーバにセッション情報を登録する場合は、JNDI サーバがエージェントプロセス上で動作するため、セッションオブジェクトのためのメモリが必要となります。そのため、セッション情報を登録する場合は、登録されるセッションオブジェクトの数、サイズ、使用しなくなったセッションオブジェクトを破棄するまでのタイムアウト時間などを考慮して、サイズを決定してください。

- Metaspace 領域の設計

Metaspace 領域はクラスの情報が格納される領域となります。 設計に際して、アプリケーションの実装により依存するため、実際にアプリケーションを配備した上で動作確認を行い、GC ログから使用状況を見定めて設定する必要があります。

- GC ログの出力

Java VM の使用メモリサイズ、Full GC の発生状況を確認するには Java VM のGCログを確認します。システムが安定するまではGCログ出力設定を追加し定期的にメモリ使用状況を確認するようにします。

- エージェントプロセスに対する留意点

Java VM の使用メモリサイズ、Full GC の発生状況を確認するには Java VM のGCログを確認します。システムが安定するまではGCログ出力設定を追加し定期的にメモリ使用状況を確認するようにします。
以下は、otxadmin コマンドを使用した例となります。このとき、verbose-gc-enabled の値を true に変更することで GC ログ出力が可能となります。

1） WebOTX をサービスから起動します。

2） 対象のドメインにログインします
otxadmin > login --password <パスワード> --user <ユーザ名> --host <ホスト名> --port <ポート番号>

3） otxadmin コマンドにて verbose-gc-enabled の値を変更します。
otxadmin > set server.java-config.verbose-gc-enabled=true

4） WebOTX を再起動します

GC ログは出力先オプションを指定しない場合、

<ドメインディレクトリ>/domains/<ドメイン名> /logs/server.log に出力されます。

プロセスグループ

- プロセスグループに対する留意点

Java VM の使用メモリサイズ、Full GC の発生状況を確認するには Java VM のGCログを確認します。システムが安定するまではGCログ出力設定を追加するようにします。プロセスグループの Java VM オプションに -verbose:gc を追加してください。

出力先は
<WebOTX インストールディレクトリ>/domains/<ドメイン名>/logs/tpsystem/<アプリケーショングループ名>/<プロセスグループ名>/<プロセスグループ名>.<プロセス ID>.log となります。

3） 使用していないサービスを抑止する 共通

ドメイン作成時の既定値では、全てのサービスが起動するように設定されます。運用上必要のないサービスについては、起動させない設定にすることにより WebOTX が使用するメモリを削減することができます。

- JMS サービスの無効化
  JMSサービスを利用していなければ、ライフサイクルモジュール設定により JMS サービスの起動設定を無効にします。

2.1.3.設定項目

性能設計で決定すべきパラメーター一覧です。

1） WebOTX が使用するメモリサイズの設定項目

メモリサイズの設計では、以下の内容を決定します。

共通

エージェントプロセス

設定パラメータ(属性名)	説明	既定値	推奨値
最大ヒープサイズ max-heap-size	ドメインのエージェントプロセスの Java VM オプションとして最大ヒープサイズを指定します。	768m	要件に合わせて設定してください。
JVM オプション jvm-option	JVM オプションを指定します	なし	-XX:MaxMetaspaceSize=<size> size は要件に合わせて設定してください。
GC ログ verbose-gc-enabled	ドメインのエージェントの Java VM オプションとして GC ログオプションを指定します。	false	true

プロセスグループ

プロセスグループ

設定パラメータ(属性名)	説明	既定値	推奨値
最大ヒープサイズ maxHeapSize	プロセスグループの Java VM オプションとして指定します。	-1 (JavaVM の既定値 ※)	要件に合わせて設定してください。
その他の引数 otherArguments	プロセスグループの Java VM オプションのその他の引数として指定します。	なし	-XX:MaxMetaspaceSize=<size> -verbose:gc size は要件に合わせて設定してください。 複数のオプションを指定する場合はスペース区切りで指定します。

※ 最大ヒープサイズの既定値は物理メモリの 1/4 と 1GB で少ない方の値となります。ただし物理メモリが 1GB より小さい場合は、物理メモリの 1/2 の値となります。

2） サービスの抑止の設定項目

サービス抑止の設計では、以下の内容を決定します。

共通

設定パラメータ(属性名)	説明	既定値	推奨値
JMS サービスの起動の有効化 enabled	サーバ起動時における JMS サービスの起動可否を指定します。	true	false

2.2.多重度

システムを構築する上で、以下の多重度において考慮する必要があります。

1） WebOTX Webサーバ (※本章では、以降 Web サーバと記述します。)の接続多重度
クライアントからの全てのリクエストを、サーバが受け付けた場合、システムに多大な負荷がかかり処理が実行できなくなる可能性があります。クライアントに対するレスポンスを保証するために、一度に受け付け可能なクライアント接続数とリクエスト数について設定する必要があります。

2） アプリケーションサーバの実行多重度
レスポンスを保証するために、アプリケーションの同時実行数を設定する必要があります。

3） データベースコネクションプールの多重度
データベースコネクションの多重度を設定する必要があります。
データベースコネクション設定は 2.4 DB連携設定 で詳しく説明しています。

2.2.1.概要説明

多重度の設計を以下の 3 つの部分に分けて説明します。

1） Webサーバの多重度設定 共通
Webサーバの受付リクエスト数をもとに、WebOTX Web サーバに関するパラメータの設定を行います。
また、Webサーバとアプリケーションサーバ間(プラグインとリスナ)の通信に関するパラメータの設定を行います。

2） アプリケーションサーバの実行多重度設定 プロセスグループ エージェントプロセス
アプリケーションサーバの実行多重度を制御するために、必要なパラメータの設定を行います。

3） データベースコネクションプールの多重度設定 共通
データベースコネクションの多重度設定を行います。

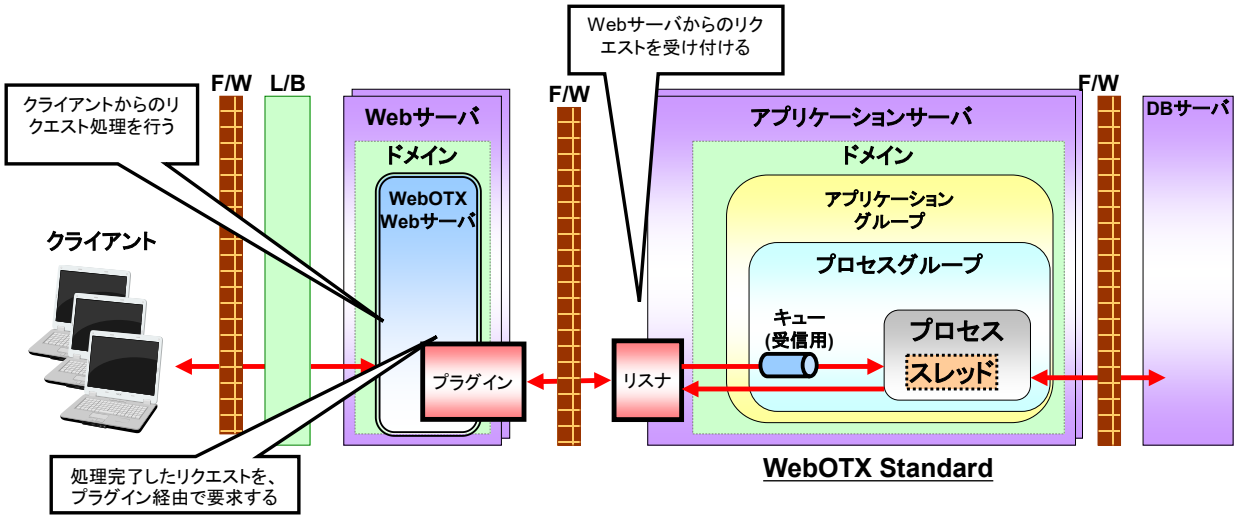


図 2.2.1.WebOTX での処理の流れ

上の図のような構成では、以下のようにリクエスト処理が行われます。
クライアントから要求されたリクエストは Web サーバで受け付けられます。Web サーバで受け付けたリクエストは、プラグインを経由してアプリケーションサーバに伝達されます。プラグインからアプリケーションサーバに送信されたリクエストは、アプリケーションサーバ内にてプロセスグループ上のスレッドで処理が行われます。

1） Webサーバの多重度設定 共通

Web サーバの多重度を考えるために、以下の図のような構成を考えます。

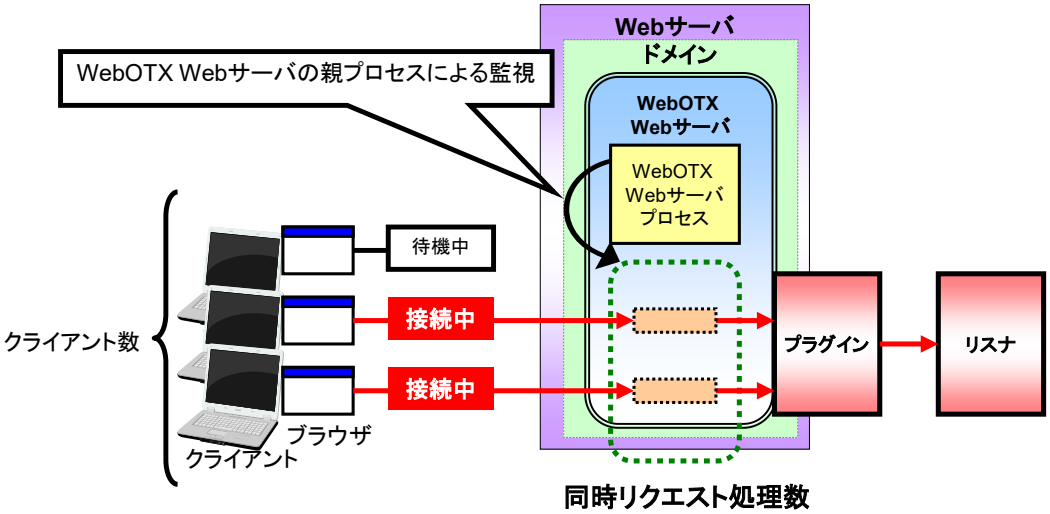


図 2.2.2.Webサーバ構成

- クライアント数
  Web サーバにアクセスすることのできるブラウザの総接続数です。

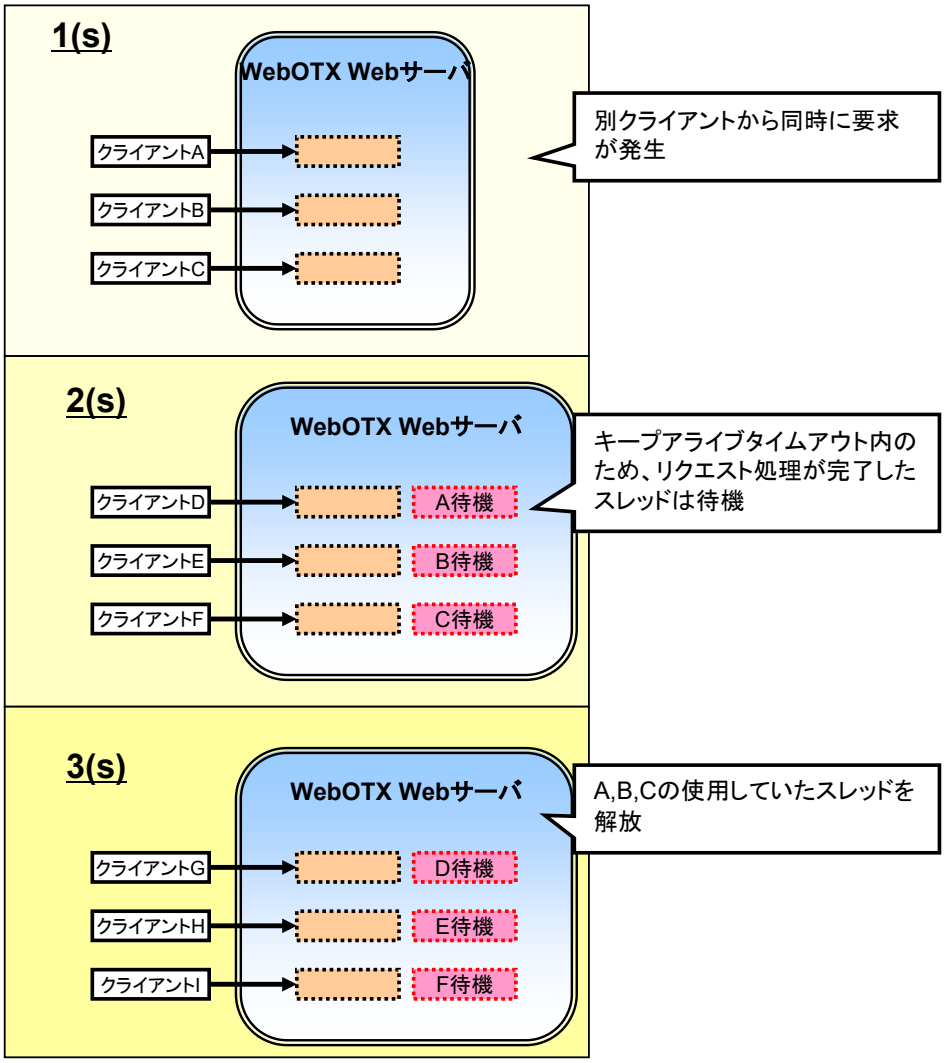
- 同時接続クライアント数
  クライアント数の内、Webサーバに同時に接続する可能性があるクライアント数です。

- 同時リクエスト処理数(MaxClients,ThreadsPerChild,ServerLimit)
  Web サーバが同時に処理するリクエストの数です。同時接続クライアント数に、1クライアントからの同時リクエスト数を掛けることで、算出することができます。

  同時リクエスト処理数 ＝ 同時接続クライアント数 × 1クライアントあたりの同時リクエスト数

  1 クライアントあたりの同時リクエスト数は、ブラウザの種類や、バージョンによって変わります。具体的な値については各 Web ブラウザのドキュメントで確認してください。

  また、キープアライブタイムアウトが設定されていた場合、WebOTX Web サーバの子プロセス内のスレッドは、リクエスト処理後に同じクライアントからのリクエストに備えて待機します。待機中のスレッドは、他のクライアントからのリクエストは受け付けることができません。



同時リクエスト処理数=3, KeepAliveTimeout=2(s), 1 件当たりのリクエスト処理時間≒0
1クライアントからの同時リクエスト数=1 の場合

図 2.2.3.KeepAliveTimeout 処理例

キープアライブタイムアウトを考慮すると、1クライアントからのリクエスト処理が 1 回で完了する場合、最大同時リクエスト数は以下のように算出されます。

最大同時リクエスト数 = 同時リクエスト処理数 × (1 件当たりのリクエスト処理時間 + キープアライブタイムアウト)

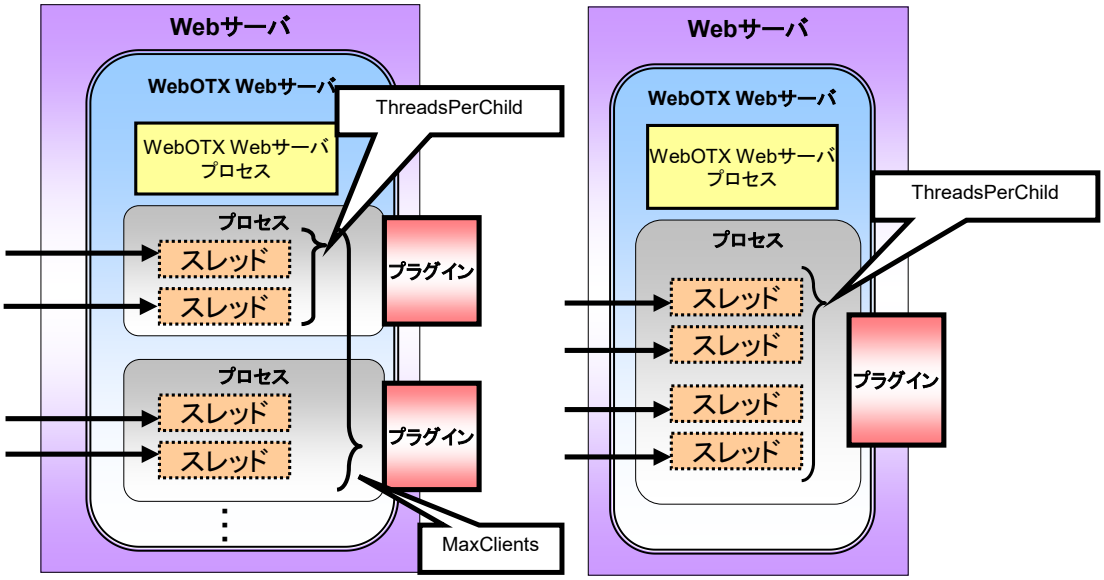
上記の算出式は単純なモデルを想定しています。使用する場合は、システム要件に合わせて変更してください。

詳細については、「2.3. 通信/タイムアウト」を参照してください。

Web サーバの同時に処理できるリクエスト数(MaxClients)を制限するには、WebOTX Web サーバが生成する子プロセス数の上限値(ServerLimit)と、子プロセスあたりの生成されるスレッド数上限値(ThreadsPerChild)を変更します。

同時リクエスト数を設定する際、OS により設定内容が異なります。

設定ファイルは<ドメイン名>/config/WebServer/httpd.conf です。



LINUX の場合　　Windows の場合

図 2.2.4.Web サーバ内部構成

- WebOTX Web サーバ 2.4(Windows)

1つの子プロセス内に ThreadsPerChild で指定されたスレッドを作成します。Windows では生成される子プロセスが1つのため、スレッド数(ThreadPerChlid)は、最大同時リクエスト数と同数に設定する必要があります。

最大同時リクエスト数(MaxClients) ＝ ThreadsPerChild × ServerLimit(1 固定) ＝ ThreadsPerChild

- WebOTX Web サーバ 2.4(LINUX)

LINUX では1つの子プロセス内に ThreadsPerChild で指定されたスレッドを作成します。空きスレッド数が不足すると ServerLimit で設定されるプロセス数まで子プロセスを生成します。そのため、最大リクエスト処理数は全ての子プロセス内の生成スレッド数の合計になります。アイドル状態になると MaSpareThreads/MinSpareThreads の数にあわせて子プロセス数が増減します。

MaxClients を超過した場合は、ListenBackLog にキューイングされます。ListenBackLog とは、クライアントから MaxClients を超えたコネクション要求を保留状態にしておきます。既定値は 511 です。

最大同時リクエスト数(MaxClients) <＝ ThreadsPerChild × ServerLimit

※Web サーバの子プロセスは、最大で MaxClients / ThreadsPerChild 分生成されます。Web サーバの子プロセス数が ServerLimit で設定した値よりも多い場合は ServerLimit の値も変更してください。

➢ ThreadsPerChild
子プロセスで生成されるスレッド数です。スレッド数の上限値(ThreadLimit)以下の値を指定します。LINUX で ThreadsPerChild の値を 64 より大きい値に設定する場合は、ThreadLimit の定義を追加してください。定義しない場合、ThreadsPerChild の値は ThreadLimit の既定値の 64 に設定されます。

ThreadsPerChild >= MaxClients ÷ ServerLimit

➢ ThreadLimit
子プロセスで動作するスレッド数の上限値です。ThreadsPerChild が ThreadLimit より小さい値であれば変更の必要はありません。ThreadLimit を定義しない場合、ThreadLimit は既定値の 64 に設定されます。

ThreadLimit >= ThreadsPerChild

➢ ServerLimit
生成される子プロセスの上限値です。Windows は 1(固定)です。

ServerLimit >= MaxClients ÷ ThreadLimit

※WebOTX Web サーバ 2.4(Windows)では MaxClients の設定は無効です。WebOTX Web サーバ 2.4(Windows)において、上記計算式を用いる場合、MaxClients に最大同時リクエスト数を使用してください。

➢ プラグインモジュールの最大リクエスト処理数

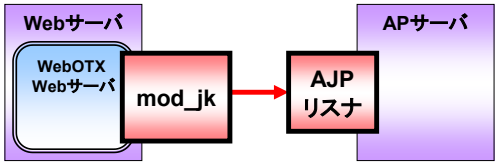


図 2.2.5.AJP リスナ使用時

Webサーバで受け付けたクライアントからのリクエストは、mod_jk を用いて、アプリケーションサーバに送信されます。

このため、同時リクエスト処理数の設定を mod_jk に対して行う必要があります。要件によって次のいずれかを満たすように設定します。

(a) 1セッションあたりの同時リクエスト数は、WebOTX Web サーバの子プロセスで動作するスレッド数の上限値と同じになるように設定する

プラグインモジュールのリクエスト処理数 = WebOTX Web サーバ の ThreadsPerChild

この場合、高負荷時に次のような現象が発生します。

- 500 エラー等は発生しにくいが、Web サーバに接続できなくなることが多くなる。
- Web サーバの接続バックログにリクエストが溜まるため、レスポンスタイムが悪化する。

(b) 1セッションあたりの同時リクエスト数は、WebOTX Web サーバの子プロセスで動作するスレッド数の上限値より小さくなるように設定する

プラグインモジュールのリクエスト処理数 < WebOTX Web サーバ の ThreadsPerChild

この場合、高負荷時に次のような現象が発生します。

- Web サーバには接続できるが、500 エラーが多く発生するようになる。
- Web サーバの接続バックログにリクエストは溜まらないので、レスポンスタイムはある程度確保できる。

同時に処理できるリクエストの数を拡大するには、connection_pool_size の値を

<ドメイン名>/config/WebCont/workers.properties に定義します。このファイルをエディタで開いて編集してください。

具体的には、次のようになります。既定値は 150 です。

worker.<worker 名>.connection_pool_size=150

※worker 名にはエージェントプロセス用の"agent-ajp"、もしくは、プロセスグループ用の"tpsystem-ajp"を指定します。プラグインの負荷分散機能を利用した場合は負荷分散用に命名した worker 名を指定します。

値を反映するには Web サーバを再起動する必要があります。使用中の接続クライアント数については「**運用編 – モニタリング** – 3.3.4 接続クライアント数のモニタリング」を参照してください。

2) アプリケーションサーバの実行多重度設定 **プロセスグループ** **エージェントプロセス**

アプリケーションサーバにおける多重度では、リスナとプロセスグループの多重度を設定します。

プロセスグループ

➢ AJP リスナの多重度

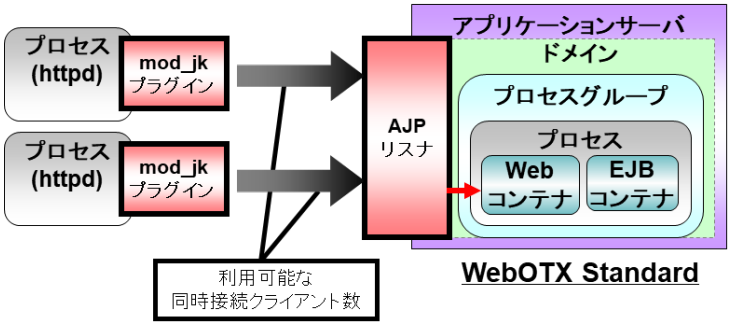


図 2.2.6.プロセスグループに配備した場合の構成

Webサーバで受け付けたクライアントからのリクエストは、mod_jk プラグインを経由し、AP サーバ側の AJP リスナに送信されます。このため、Webサーバのリクエスト数にあわせ、AJP リスナの多重度の設定を行う必要があります。

- AJP リスナ 利用可能な同時接続クライアント数
  システム拡張や障害防止を考慮して、設定値は既定値以下としないことを推奨します。

  アプリケーションサーバでの同時接続クライアント数を設定します。クライアント数(クライアント側のプロセス数)以上の値を設定します。Web サーバを使用している場合は Web サーバの数(httpd プロセスの数)以上の設定が必要となります。
  統合運用管理ツールから設定する場合は[TP システム]-[AJP Listener]-[上限設定]-[最大同時リクエスト処理数]で設定します。

  図 2.2.6.の場合、利用可能な同時接続クライアント数は2となりますが、既定値(256)以下であるため、変更の必要がありません。

➢ プロセスグループ内の実行多重度
プロセスグループ単位で多重度について設計します。

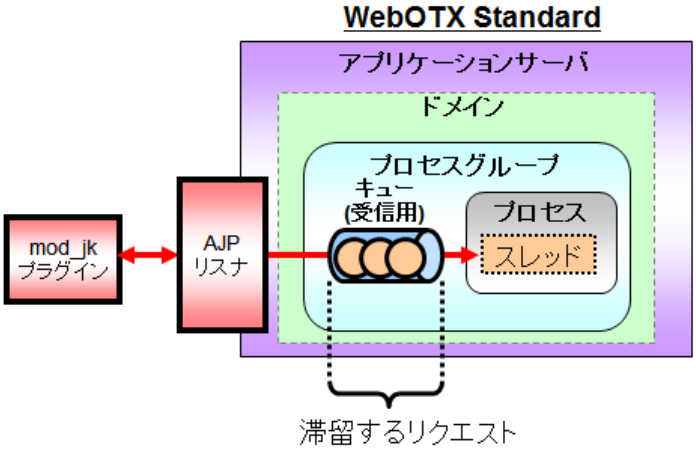


図 2.2.7.プロセスグループ構成

WebOTX ではプロセスグループ単位に実行多重度(プロセス、スレッド)を設定します。そのプロセスグループの特性に応じて実行多重度を設定できます。
プロセスグループ内の実行多重度は、プロセスグループのプロセス数と、1プロセスあたりのスレッド数で調整します。プロセス数を増やすことにより業務の安定化を図ることができ、スレッド数を増やすことにより効率よくリクエスト処理を行うことができます。

プロセスグループ内の実行多重度 = プロセス数×スレッド数

リクエスト数に対して実行多重度が少ない場合、処理待ち(キュー滞留)が発生することがあります。
このため、業務アプリケーションにて受け付けるリクエスト数を考慮した実装が必要になります。

- プロセスグループ プロセス数
  プロセスグループとは、ある特定のサービスを提供するプロセス群です。
  WebOTX は、ビジネスロジックを記述した 1 つまたは複数のコンポーネントをプロセスグループに登録し、マルチプロセスとして実行します。プロセスグループは、マルチプロセス実行させることにより、1つのプロセスでアプリケーション障害が発生しても、そのプロセスだけが異常終了し、他のプロセスは影響を受けません。可用性を高めるためは、プロセスグループのプロセス数を2以上になるように設計します。

  プロセス数 = プロセスグループ内の実行多重度 ÷ スレッド数

- プロセスグループ スレッド数
  1プロセスあたりのスレッド数です。プロセスグループ単位に設定します。リクエストは、プロセスグループの空きスレッドに割り当てられて実行されます。

  スレッド数 = プロセスグループ内の実行多重度 ÷ プロセス数

- プロセスグループ リクエストキューのサイズ
  WebOTX ではプロセスグループまたはプロセス単位にキューを生成します。同一プロセスグループ上のリクエストはすべてこのキューにキューイングされます。スレッド群はプロセスの上で動作し、単一のキューに対してデータの到着を待ち合わせます。そのため、空きスレッドに対し、効率良くトランザクションをディスパッチすることができます。
  また、このキューの滞留数を制限することで、キューに入れなかったリクエストについてはエラーとし、高負荷時でもクライアントへのレスポンス時間を保証することが可能です。

※ 注意事項

Java プロセスを増やすことは、スレッド数を増やすことに比べて、よりOSのリソースを消費します。そのため、Java プロセスの過剰な作成は避けるようにしてください。

また、以下の点に注意してください。

- 複数のプロセスから同一のファイルに対し更新・参照する場合には、正確な情報がファイルに反映できない可能性があります。
  そのため、業務アプリケーションに対し、以下のような対策を行ってください
  - 参照と更新が同時に発生するファイルをアプリケーション側で保持しない
  - 出力の際に同期処理が行われるような設計を行ってください。

エージェントプロセス

➢ Web コンテナの処理スレッド数

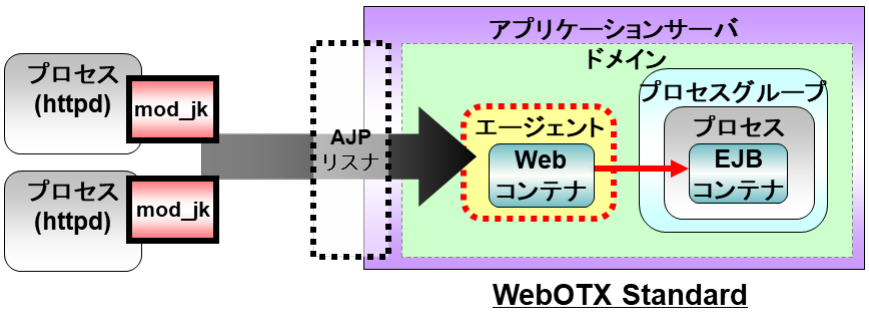


図 2.2.8.シングルプロセス構成

エージェントプロセスに配備した場合、Webサーバで受け付けたクライアントからのリクエストは、mod_jk-AJP リスナを経由し、エージェントプロセスにて受け付けられます。このため、Webサーバからのリクエスト数にあわせ、Web コ

ンテナの処理スレッド数を設定する必要があります。
このとき、otxadmin コマンドを用いて以下の設定を行います。

最小スレッド数の設定
otxadmin〉set --user〈ユーザ名〉--password〈パスワード〉--host〈ホスト名〉--port〈管理ポート〉server.thread-pools.thread-pool.<スレッドプール名>.min-thread-pool-size=<最小スレッド数>

最大スレッド数の設定
otxadmin> set --user <ユーザ名> --password <パスワード>--host <ホスト名> --port <管理ポート> server.thread-pools.thread-pool.<スレッドプール名>.max-thread-pool-size=<最大スレッド数>

初期は最小スレッド数で設定された値で処理スレッドは稼動していますが、リクエストが増加すると最大値まで徐々に増加します。

最大数は

最大スレッド数(max-thread-pool-size) **= httpd 1プロセスあたりの最大同時リクエスト数(ThreadsPerChild)×Web サーバの数(httpd プロセスの数)**

となるように設定してください。

最小スレッド数と最大スレッド数は、下記条件であてまるよう設定を行ってください。また、最大スレッド数(max-thread-pool-size) を変更した場合は、スレッド数警告のしきい値(limit-thread-pool-size) も同時に変更してください。スレッド数警告のしきい値(limit-thread-pool-size)は、スレッド数が最大スレッド数に近づいた事を示す警告メッセージを出力するためのしきい値です。

最小スレッド数(min-thread-pool-size) < 最大スレッド数(max-thread-pool-size)

➢ プロセスグループの実行多重度

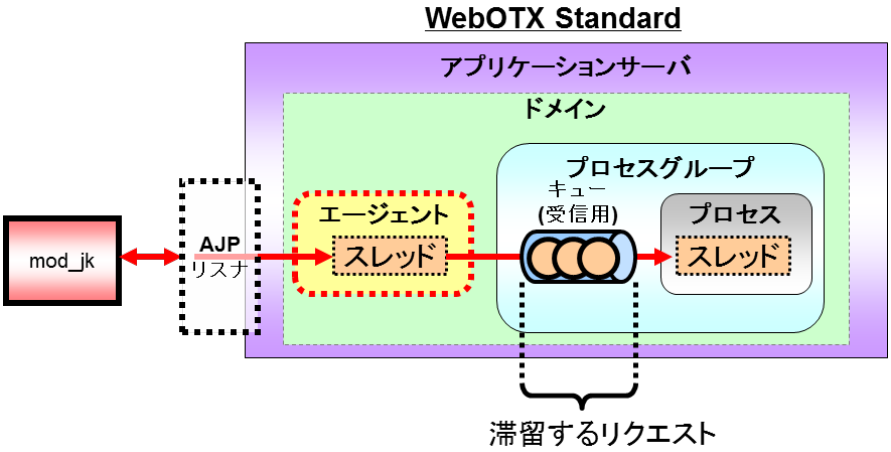


図 2.2.9.プロセスグループ構成

エージェントプロセスに配備した場合は、プロセスグループ上で動作する EJB コンテナを使用するアプリケーションに対して設定できます。エージェントプロセス上で動作する Web コンテナを使用するアプリケーションに対しては設定できません。WebOTX ではプロセスグループ単位に実行多重度(プロセス、スレッド)を設定します。そのプロセスグループの特性に応じて実行多重度を設定できます。
プロセスグループ内の実行多重度は、プロセスグループのプロセス数と、1プロセスあたりのスレッド数で調整します。プロセス数を増やすことにより業務の安定化を図ることができ、スレッド数を増やすことにより効率よくリクエスト処理を行うことができます。

プロセスグループ内の実行多重度 = プロセス数×スレッド数

リクエスト数に対して実行多重度が少ない場合、処理待ち(キュー滞留)が発生することがあります。
このため、業務アプリケーションにて受け付けるリクエスト数を考慮した実装が必要になります。

- プロセスグループ プロセス数
  プロセスグループとは、ある特定のサービスを提供するプロセス群です。
  WebOTX は、ビジネスロジックを記述した 1 つまたは複数のコンポーネントをプロセスグループに登録し、マルチプロセスとして実行します。プロセスグループは、マルチプロセス実行させることにより、1つのプロセスでアプリケーション障害が発生しても、そのプロセスだけが異常終了し、他のプロセスは影響を受けません。可用性を高めるためは、プロセスグループのプロセス数を2以上になるように設計します。

  プロセス数 = プロセスグループ内の実行多重度 ÷ スレッド数

- プロセスグループ スレッド数
  1プロセスあたりのスレッド数です。プロセスグループ単位に設定します。リクエストは、プロセスグループの空きスレッドに割り当てられて実行されます。

  スレッド数 = プロセスグループ内の実行多重度 ÷ プロセス数

- プロセスグループ リクエストキューのサイズ
  WebOTX ではプロセスグループまたはプロセス単位にキューを生成します。同一プロセスグループ上のリクエストはすべてこのキューにキューイングされます。スレッド群はプロセスの上で動作し、単一のキューに対してデータの到着を待ち合わせます。そのため、空きスレッドに対し、効率良くトランザクションをディスパッチすることができます。
  また、このキューの滞留数を制限することで、キューに入れなかったリクエストについてはエラーとし、高負荷時でもクライアントへのレスポンス時間を保証することが可能です。

※ 注意事項

Java プロセスを増やすことは、スレッド数を増やすことに比べて、よりOSのリソースを消費します。そのため、Java プロセスの過剰な作成は避けるようにしてください。

また、以下の点に注意してください。

- 複数のプロセスから同一のファイルに対し更新・参照する場合には、正確な情報がファイルに反映できない可能性があります。
  そのため、業務アプリケーションに対し、以下のような対策を行ってください
  - 参照と更新が同時に発生するファイルをアプリケーション側で保持しない
  - 出力の際に同期処理が行われるような設計を行ってください。

3) データベースコネクションプールの多重度設定 共通

- DB サーバへのコネクション
  AP サーバが利用するバックエンドサーバ(DB や ACOS や TPBASE)との TCP/IP のコネクションは 1 つのプロセス内で共有されます。よってプロセスグループを分けたり、マルチプロセスにしたりする場合はその分コネクションが生成されることになります。
  コネクションプールを定義し、コネクションプールで指定した最大プールサイズの値は、プロセス毎に最大プールサイズ数のコネクションが生成される可能性がある点に注意が必要です。

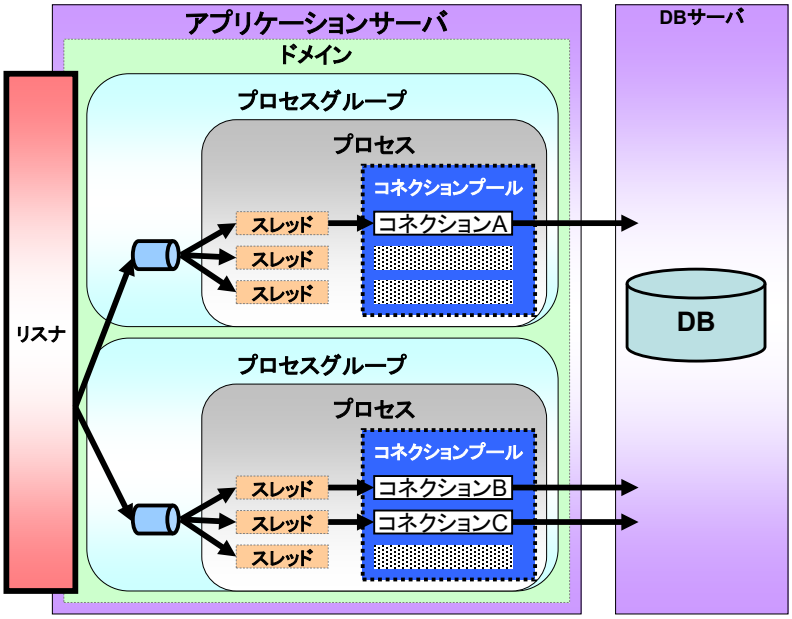


図 2.2.10.アプリケーションサーバとデータベースの関連

JDBC の設定についての詳細は、2.6 データベース連携 を参照してください。

2.2.2.設計指針

レスポンス時間を考慮して多重度を正確に見積もることは非常に困難です。ここでは、モデルを単純化し近似値による構築時の目安となる設定値を説明しています。実際には、アプリケーションの特性や、CPU 数、キューによる待ち時間などで変わるためチューニングによる多重度の再調整が必要になります。

チューニングについては、WebOTX マニュアルの「高度な管理と運用サイクルガイド － 2. チューニング」を参照してください。

以下の条件を例に設計指針について説明します。

1） Webサーバの多重度設定 共通

- クライアント数：1000台
- 同時接続クライアント数：100台
- WebOTX Web サーバ 2.4 (Linux)

2） アプリケーションサーバの実行多重度設定 プロセスグループ エージェントプロセス

- プロセスグループの1スレッドあたりのリクエスト処理数：3 件／秒
- 同時接続クライアントからのリクエスト間隔：0.2 リクエスト/秒

1） Webサーバの多重度設定 共通

➢ 同時リクエスト処理数（MaxClients,ThreadsPerChild,ServerLimit）
条件から同時接続クライアント数の上限を 100 台と仮定します。
1クライアントからの同時リクエスト数を2として、同時リクエスト処理数は以下の式より 200 となります。

同時リクエスト処理数 ＝ 同時接続クライアント数（100） × 1クライアントからの同時リクエスト数（2） ＝ 200

ただし、同時リクエスト処理数をぎりぎりに設定してしまうとゴーストセッションが残っている場合に接続できなくなる可能性がありますので、ゴーストセッションの対策をとった上である程度の余裕を持たせてください。

1回のリクエストで処理が完了するような場合には、リクエスト処理が完了してもクライアントとの接続が保持された状態となるため、設定時間が経過するまで Web サーバのスレッドは待機状態となります。

ここでは、1回のリクエストで処理が完了するような場合であると仮定します。

KeepAliveTimeout=2、1 件当たりのリクエスト処理時間≒0 とした場合、以下のように設定します。

同時リクエスト処理数 = 200 × (KeepAliveTimeout (2) + 1 件当たりのリクエスト処理時間(0)) = 400

実際には、受付リクエスト数については余裕をもった値を設定しておくことを推奨します。想定される同時リクエスト数から 1.2～1.5 倍程度の値にします。

同時リクエスト処理数 ＝ 400 × 1.5 = 600

例えば同時リクエスト処理数を Web サーバにて、最大 12 プロセス数で運用を行いたい場合、ThreadsPerChild を調整行います。

この場合、MaxClients は 600 と設定されるため、ThreadsPerChild を 50 と設定することで、動作する最大プロセス数を 12 とすることができます。

MaxClients(600) ＜＝ ThreadsPerChild(50) × ServerLimit(16)

※ServerLimit はデフォルト 16 で設定されているため、変更する必要はありません。デフォルト以上のプロセス数で運用を行う場合のみ ServerLimit の設定を変更してください

※ThreadsPerCild を ThreadLimit の既定値(64)以上で設定する場合、ThreadLimit が ThreadsPerCild より小さくならないように設定を変更してください。
ThreadLimit 設定箇所は<IfModule mpm_worker_module>の直下となります。

- プラグインモジュールの最大リクエスト処理数

ThreadsPerChildの最大数を設定します。

1 プロセスあたりの同時実行数(worker.<worker 名> .connection_pool_size) ＝ ThreadsPerChild(50) ＝ 50

※worker 名にはエージェントプロセス用の"agent-ajp"、プロセスグループ用の"tpsystem-ajp"が入ります。

2) アプリケーションサーバの実行多重度設定 プロセスグループ エージェントプロセス

プロセスグループ

- プロセスグループ プロセス数

プロセス数は予期せぬアプリケーション障害に備えての多重化とし、クライアントからの同時実行に備えての多重化はできるだけマルチスレッドで行う構成とします。
ここでは、プロセス数を 2 に設定します。

プロセス数 ＝ 2

- プロセスグループ スレッド数

MaxClients で設定された同時リクエスト処理数600の内、50%の 300 リクエストがプロセスグループAに集中するとします。

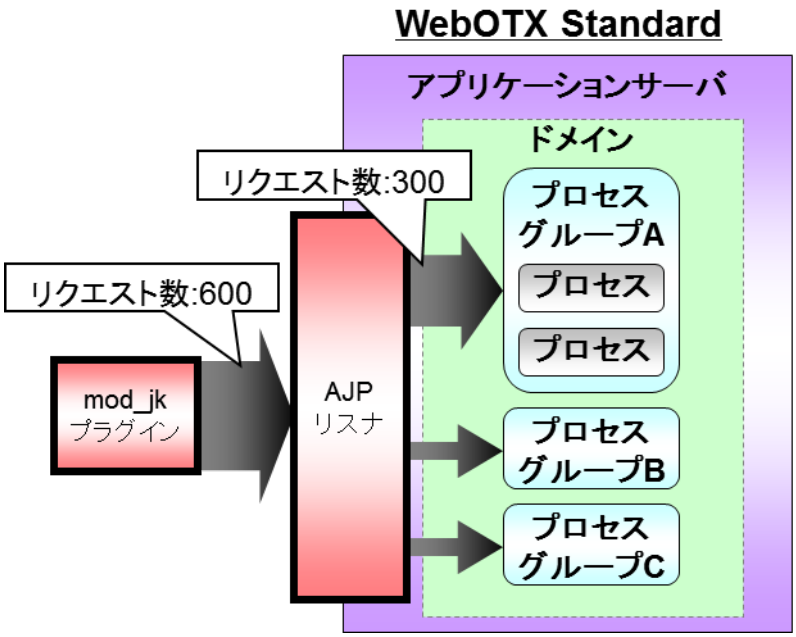


図 2.2.11.リクエスト経路

リクエスト間隔(0.2 リクエスト/秒)より、1 秒間のリクエスト処理数は、

300×0.2(リクエスト/秒) ＝ 60(リクエスト/秒)になります。

1スレッドの処理数を、3 リクエスト/秒 とした場合に、プロセスグループ内に必要な実行多重度は 20 になります。

プロセスグループ内の実行多重度 ＝ 60 ÷ 3 ＝ 20

プロセス数を2に設定した場合の1プロセスあたりに必要なスレッド数は、10 になります。

スレッド数 ＝ プロセスグループ内の実行多重度(20) ÷ プロセス数(2) ＝ 10

エージェントプロセス

- Web コンテナのスレッド数
  MaxClients で設定された同時リクエスト処理数600の内、20%の 120 リクエストがアプリケーションサーバに常にリクエストを要求するとします。この場合、最小スレッド数を以下のように設定します。
  最小スレッド数(min-thread-pool-size) = 120

  同時リクエスト数に応じて最小スレッド数(min-thread-pool-size)から最大スレッド数(max-thread-pool-size)に向けてのスレッドは増加します。

  一度に高負荷がかかる要件ではこれらの値が大きく開きがある場合、スレッド数の増加が追いつかずエラーが発生する場合があります。
  受け付けられなかったリクエストは、TCP バックログにて滞留するため、TCP バックログの値を考慮した上で最小スレッド数の設定を行ってください。
  TCP バックログについては、Web コンテナの accept-count で調整してください。

  最大スレッド数は Web サーバの接続クライアント数上限にあわせて変更します。

  最大スレッド数(max-thread-pool-size)
  = httpd 1プロセスあたりの最大同時リクエスト数(ThreadsPerChild) ×Web サーバの数(httpd プロセスの数)
  = 600

- プロセスグループ プロセス数
  プロセス数は予期せぬアプリケーション障害に備えての多重化とし、クライアントからの同時実行に備えての多重化はできるだけマルチスレッドで行う構成とします。

ここでは、プロセス数を 2 に設定します。

プロセス数 ＝ 2

- プロセスグループ スレッド数
  WEBアプリケーションから EJB アプリケーションを呼び出す場合、また JakartaEE アプリケーション(EAR) を利用した場合、リクエストは図 2.2.12.の経路を使用します。このとき、MaxClients で設定された同時リクエスト処理数 600の内、50%の 300 リクエストがプロセスグループ A に集中するとします。

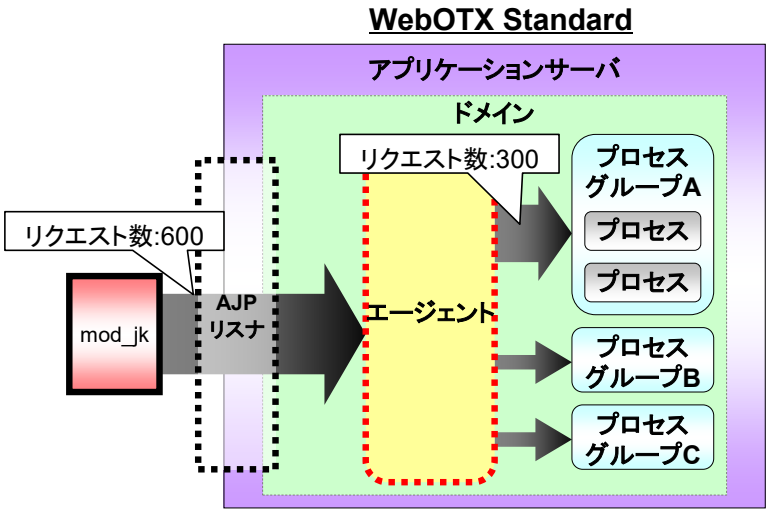


図 2.2.12.リクエスト経路

リクエスト間隔(0.2 リクエスト/秒)より、1 秒間のリクエスト処理数は、

300×0.2(リクエスト/秒) ＝ 60(リクエスト/秒)になります。

1スレッドの処理数を、3 リクエスト/秒 とした場合に、プロセスグループ内に必要な実行多重度は 20 になります。

プロセスグループ内の実行多重度 ＝ 60 ÷ 3 ＝ 20

プロセス数を2に設定した場合の1プロセスあたりに必要なスレッド数は、10 になります。

スレッド数 ＝ プロセスグループ内の実行多重度(20) ÷ プロセス数(2) ＝ 10

2.2.3.設定項目

1） Webサーバの多重度設定項目 共通

Webサーバの多重度の設計では、以下の内容を決定します。

WebOTX Web サーバ 2.4 (Windows)				
最大同時接続数 ThreadsPerChild	説明	プロセス内で動作するスレッド数となり、Web サーバが処理できる最大同時接続コネクション数を意味します。		
	既定値	400		
	推奨値	クライアント数*ブラウザ設定リクエスト数		
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf		
	超過した場合に出力されるログ	-		
子プロセスの 最大リクエスト数 MaxRequestsPerChild	説明	子プロセスが処理するリクエストの上限を指定します。本指定数のリクエストを受信すると子プロセスは終了します。0 を指定すると子プロセスは終了しなくなります。		
	既定値(= 推奨値)	0		
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf		
	超過した場合に出力されるログ	-		
	備考	WebOTX Web サーバ 2.4 では、MaxRequestsPerChild は MaxConnectionsPerChild に変更されました。ただし、MaxRequestsPerChild での指定もサポートされています。		
WebOTX Web サーバ 2.4 (LINUX)				
最大同時接続数 MaxClients	説明	Web サーバが処理できる最大同時接続コネクション数を設定します。クライアントは、この値を超えて同時に接続することはできません。		
	既定値	400		
	推奨値	クライアント数*ブラウザ設定リクエスト数		
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf		
	超過した場合に出力されるログ	-		
	備考	WebOTX Web サーバ 2.4 では、MaxClients は MaxRequestWorkers に変更されました。ただし、MaxClients での指定もサポートされています。		
子プロセスの上限値 ServerLimit	説明	子プロセスの上限値を指定します。		
	既定値(= 推奨値)	16	上限値	20000
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf		
	超過した場合に出力されるログ	-		
子プロセスのスレッドの 上限値 ThreadLimit	説明	子プロセスで動作するスレッドの上限値を指定します。		
	既定値(= 推奨値)	64	上限値	20000
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf		
	超過した場合に出力されるログ	-		
子プロセスのスレッド数 ThreadsPerChild	説明	１つのプロセス内で動作するスレッド数を指定します。		
	既定値(= 推奨値)	25		
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf		
	超過した場合に出力されるログ	-		

起動時のプロセス数 StartServers	説明	起動初期化時に生成されるプロセス数を指定します。
	既定値(= 推奨値)	2
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf
	超過した場合に出力されるログ	-
アイドルスレッド最小値 MinSpareThreads	説明	アイドル状態のスレッドの最小値を指定します。
	既定値(= 推奨値)	25
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf
	超過した場合に出力されるログ	-
アイドルスレッド最大値 MaxSpareThreads	説明	アイドル状態のスレッドの最大値を指定します。
	既定値(= 推奨値)	75
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf
	超過した場合に出力されるログ	-
子プロセスの最大リクエスト数 MaxRequestsPerChild	説明	子プロセスが処理するリクエストの上限を指定します。 本指定数のリクエストを受信すると子プロセスは終了します。0 を指定すると子プロセスは終了しなくなります。
	既定値(= 推奨値)	0
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf
	超過した場合出力されるログ	-
	備考	WebOTX Web サーバ 2.4 では、MaxRequestsPerChild は MaxConnectionsPerChild に変更されました。ただし、MaxRequestsPerChild での指定もサポートされています。

Web サーバプラグイン(mod_jk)		
プラグインモジュールの 最大リクエスト処理数 workers.<worker 名> .connection_pool_size	説明	Web サーバが Web コンテナもしくは AJP リスナに対して張るコネクションのプールサイズの上限値を指定します。 この設定値は Web サーバの子プロセスごとに適用されます。 コネクタはこの値の数だけ同時にコネクションを張ることができ、この値を超えたリクエストはエラーとなります。
	既定値	150
	推奨値	Web サーバの ThreadsPerChild に応じて値を設定してください。
	設定箇所	<ドメインのパス>/config/WebCont/workers.properties
	超過した場合に返却されるエラー	クライアント側には 503（Service Temporarily Unavailable）が返却される。

2） アプリケーションサーバの多重度設定項目 共通 プロセスグループ エージェントプロセス

アプリケーションサーバの多重度の設計では、以下の内容を決定します。

共通

プロセスグループ		
スレッド数 threadCount	説明	マルチプロセスで動作する場合のスレッド数を指定します。
	既定値	3 (プロセスグループに配備した場合の Jakarta EE プロセスグループ) / 1 (エージェントプロセスに配備した場合の Jakarta EE、CORBA プ

		ロセスグループ)
	推奨値	システム要件にしたがってください。
	設定箇所	[運用管理ツール]-[TP システム]-[アプリケーショングループ]-[アプリケーション名]-[プロセスグループ]-[プロセスグループ名]-[スレッド制御]-[スレッド数] tpsystem.applicationGroups.<アプリケーション名>.processGroups.<プロセスグループ名>.threadCount
	超過した場合に出力されるログ	-
プロセス数 processCount	説明	マルチプロセスで動作する場合のプロセス数を指定します。
	既定値	1
	推奨値	システム要件にしたがってください。
	設定箇所	[運用管理ツール]-[TP システム]-[アプリケーショングループ]-[プロセスグループ]-[プロセス制御]-[プロセス数] tpsystem.applicationGroups.<アプリケーショングループ名>.processGroups.<プロセスグループ名>.processCount
	超過した場合に出力されるログ	-

プロセスグループ

AJP リスナ		
最大同時リクエスト処理数 simultaneousConnectionClients	説明	AJP リスナの最大同時リクエスト処理数を指定します。
	既定値	256
	推奨値	システム要件にしたがってください。
	設定箇所	[運用管理ツール]-[TP システム]-[AJP リスナ]-[上限設定]-[最大同時リクエスト処理数]
	超過した場合に出力されるログ	-

エージェントプロセス

Web コンテナ		
スレッドプール最小数 min-thread-pool-size	説明	同時に処理すべきリクエスト数の最小スレッド数を設定します。
	既定値	25
	推奨値	スレッドプール最大数以下に設定してください。
	設定箇所	[運用管理ツール]-[アプリケーションサーバ]-[スレッドプール]-[thread-pool]-[http-thread-pool] server.thread-pools.thread-pool.http-thread-pool.min-thread-pool-size
	超過した場合に出力されるログ	-
スレッドプール最大数 max-thread-pool-size	説明	同時に処理すべきリクエスト数を拡大するために変更します。
	既定値	100
	推奨値	接続クライアント数以上に設定してください。
	設定箇所	[運用管理ツール]-[アプリケーションサーバ]-[スレッドプール]-[thread-pool]-[http-thread-pool] server.thread-pools.thread-pool.http-thread-pool.max-thread-pool-size

	最大数に達した場合に出力されるログ	OTX02160003: すべてのスレッド（n）が現在稼働中で待機しています。max-processors (n) を増やすか、そのサーブレットのステータスをチェックしてください。 ※n は max-thread-pool-size の値
スレッド数警告のしきい値 limit-thread-pool-size	説明	アクティブなリクエスト処理スレッドの数が最大数に近づいた時に出力する警告ログの閾値を設定します。
	既定値	80
	推奨値	スレッドプール最小値以上で、かつスレッドプール最大数より少し小さい値を設定してください。
	設定箇所	[運用管理ツール]-[アプリケーションサーバ]-[スレッドプール]-[thread-pool]-[http-thread-pool] server.thread-pools.thread-pool.http-thread-pool.limit-thread-pool-size
	超過した場合に出力されるログ	OTX02160003: すべてのスレッド（{アクティブなスレッド数}）が現在稼働中で待機しています。max-processors（{スレッドプール最大数}）を増やすか、そのサーブレットのステータスをチェックしてください。 OTX02160003: All threads ({アクティブなスレッド数}) are currently busy, waiting. Increase max-processors ({スレッドプール最大数}) or check the servlet status
ソケットのバックログ値 accept-count	説明	リクエスト受け付け用ソケットのバックログ値です。
	既定値	60
	推奨値	システム要件に従ってください。
	設定箇所	[運用管理ツール]-[アプリケーションサーバ]-[ネットワーク構成]-[トランスポート構成]-[transport]-[TCP] server.network-config.transports.transport.tcp.accept-count
	超過した場合に返却されるエラー	クライアント側では 503（Service Temporarily Unavailable）が返却されます。
最大接続数 max-connections	説明	コネクションの最大接続数を指定します。
	既定値	100(初期値) ※設定がない場合のデフォルトは 10000
	推奨値	接続クライアント数以上に設定してください。
	設定箇所	[運用管理ツール]-[アプリケーションサーバ]-[ネットワーク構成]-[プロトコル構成]-[protocol]-[HTTP プロトコル] -[HTTP] server.network-config.protocols.protocol.{protocol-name}.http.max-connections
	超過した場合に出力されるログ	-

2.3.通信/タイムアウト

クライアントが業務アプリケーションを実行するためには、以下の経路で通信が行われます。

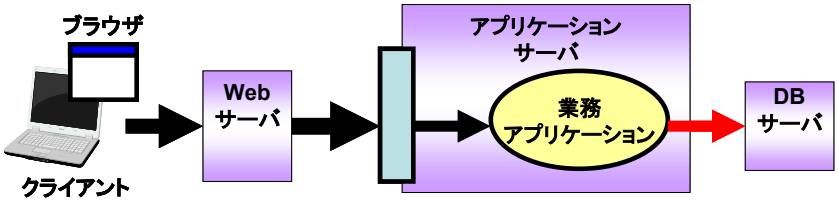


図 2.3.1.通信経路

1） クライアント

- クライアントから Web サーバへの通信

2） Webサーバ

- Web サーバからアプリケーションサーバへの通信

3） アプリケーションサーバ

- アプリケーションサーバ内で、業務アプリケーションを呼び出すための通信
- 業務アプリケーションから DB サーバへの通信(DB を使用する場合)

ここでは、各通信経路のタイムアウト時間設定について説明します。

2.3.1.概要説明

通信の際、要求したリクエストサイズが大きい、通信回線に障害がある、サーバで問題が発生する等の理由で要求への応答が遅れることがあります。その間、CPU や回線などのリソースが占有されてしまい、他の作業に影響を及ぼす可能性があります。この問題を回避するために、タイムアウトの利用が考えられます。例えば、通信元でタイムアウト時間を設定し、応答が遅れている処理を次の段階（エラー処理等）に移行させることで、問題を回避できます。

通信/タイムアウトでは、主に以下の項目について説明を行ないます。

1） クライアントで設定するタイムアウト値 共通

- クライアント(Web ブラウザ)タイムアウト

2） WebOTX Web サーバで設定するタイムアウト値 共通

- キープアライブタイムアウト
- コネクションプールタイムアウト
- ソケットタイムアウト

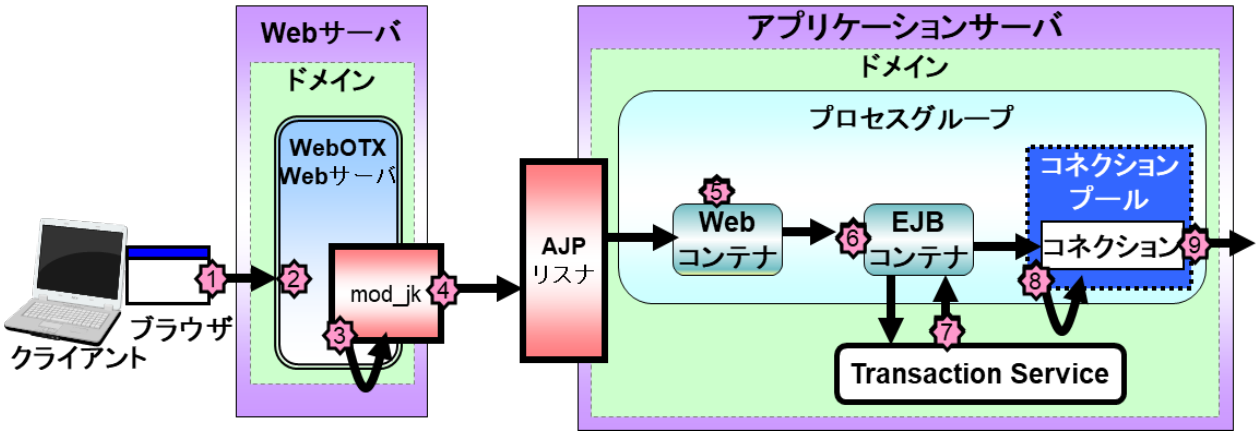
3） アプリケーションサーバで設定するタイムアウト値 共通 プロセスグループ エージェントプロセス

- CORBA リクエストタイムアウト
- トランザクションタイムアウト
- セッションタイムアウト
- JDBC コネクションタイムアウト

これら通信/タイムアウトの概念を持つモジュールについて、以下の図を例に説明を行います。なお、本章では、以降、WebOTX Web サーバを Web サーバと記述します。

※ Web サーバの Timeout 設定は、業務トランザクションのタイムアウトとは種類が異なるため、通常は設定不要です。

プロセスグループ



①クライアント(Web ブラウザ)タイムアウト

②キープアライブタイムアウト　③コネクションプールタイムアウト　④ソケットタイムアウト

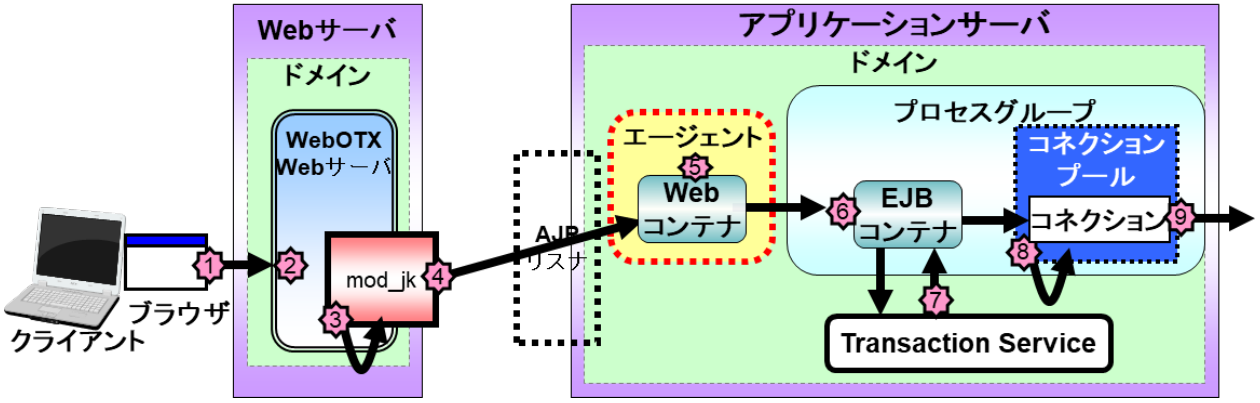
⑤セッションタイムアウト　⑥CORBA リクエストタイムアウト

⑦トランザクションタイムアウト　⑧コネクション解放までのタイムアウト

⑨ログインタイムアウト、クエリータイムアウト、ソケットの読み取りタイムアウト、空きコネクション取得時の待ち合わせ時間

図 2.3.2.a. Web サーバとアプリケーションの通信経路(プロセスグループ)

エージェントプロセス



①クライアント(Web ブラウザ)タイムアウト

②キープアライブタイムアウト　③コネクションプールタイムアウト　④ソケットタイムアウト

⑤セッションタイムアウト　⑥CORBA リクエストタイムアウト

⑦トランザクションタイムアウト　⑧コネクション解放までのタイムアウト　⑨ログインタイムアウト、クエリータイムアウト

⑨ログインタイムアウト、クエリータイムアウト、ソケットの読み取りタイムアウト、空きコネクション取得時の待ち合わせ時間

図 2.3.2.b. Web サーバとアプリケーションの通信経路(エージェントプロセス)

1)　クライアントで設定するタイムアウト値 共通

- クライアント(Web ブラウザ)タイムアウト
  Web ブラウザからリクエストを送信後、応答までのタイムアウト値です。本設定は通常ブラウザ側で行います。
  また、使用するブラウザにより、設定箇所が異なります。

Firefox を使用する場合、アドレスバーに about:config を入力し、「危険を承知の上で使用する」ボタンをクリックします。network.http.response.timeout を検索し、ダブルクリックで値（秒）を変更します。タイムアウト時間のデフォルト値は 300 秒です。

2） WebOTX Web サーバで設定するタイムアウト値 共通

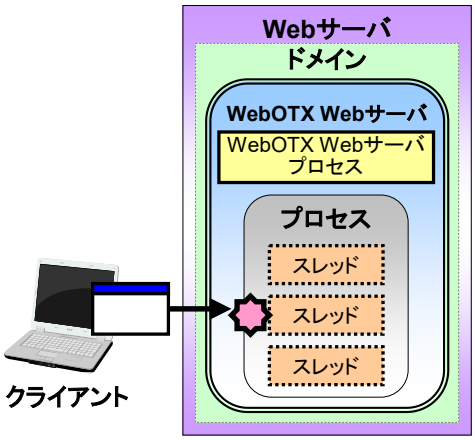


図 2.3.3.クライアント Web サーバの通信経路

➢ キープアライブタイムアウト(KeepAliveTimeout)

WebOTX Web サーバの子プロセス内のスレッドにて、クライアントからのリクエスト処理を完了した後に使用されるタイムアウト値です。リクエスト処理完了後に、同じクライアントから到達するリクエストの待機時間(秒)を設定します。設定時間を越えると、Web サーバがクライアントとの接続を切断します。

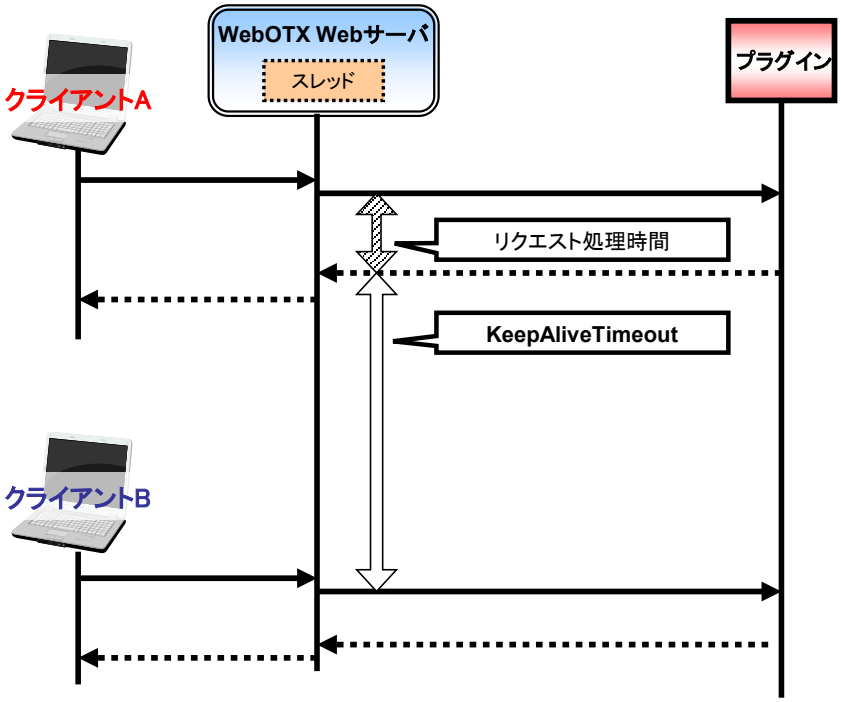


図 2.3.4.キープアライブタイムアウト

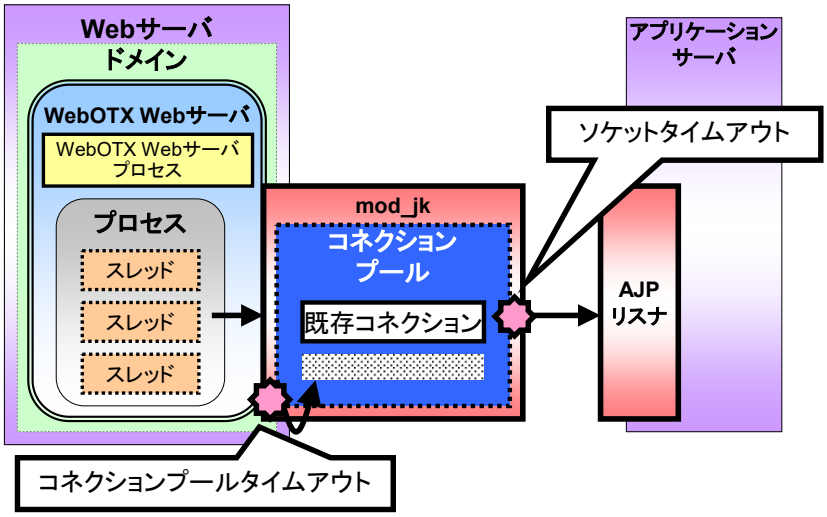


図 2.3.7.Web サーバとアプリケーションの通信経路

- コネクションプールタイムアウト(connection_pool_timeout)
  Web サーバが Web コンテナ(AJP リスナ)に対して張るコネクションのプールを維持する時間(秒)を指定します。
  既定値は 59 です。

- ソケットタイムアウト(socket_timeout)
  リモートホストとコネクタ間におけるソケット通信のタイムアウト時間(秒)を指定します。リトライ回数(retries)と関連性があります。retries の既定値は 2 です。

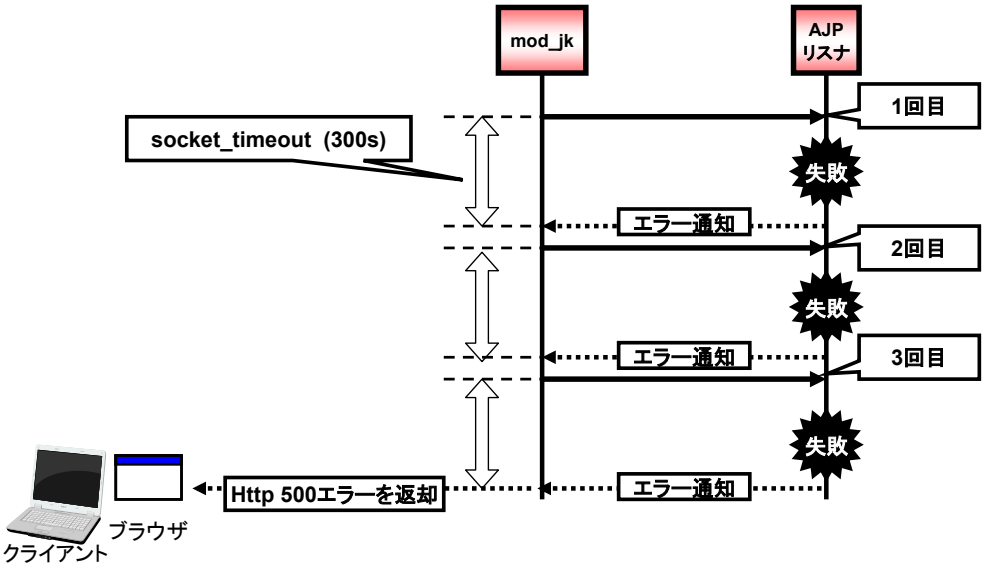


図 2.3.8 ソケットタイムアウト

既定値では socket_timeout=0 秒(タイムアウト無し)、retries=2 回で設定が行われています。
図の例は socket_timeout=300、retries=3 にて設定を行った場合を表しています。
※retries の最小値は 1 となっています。retries=0 と設定しても 1 として処理されます。

3) アプリケーションサーバで設定するタイムアウト値 共通 プロセスグループ エージェントプロセス

プロセスグループ

- ear 形式で Web アプリケーションと EJB アプリケーションを配備する構成:

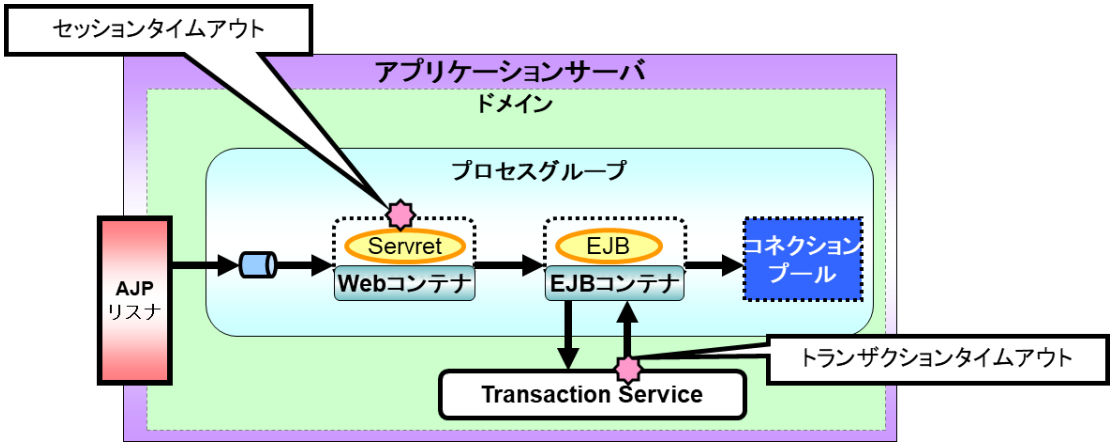


図 2.3.9.アプリケーションサーバ内部構成(プロセスグループ)

Web アプリケーションから EJB アプリケーションの呼出しはローカル呼出しとなります。
アプリケーションの設定や統計情報は ear ファイルが配備されたプロセスグループ単位で行なうため、Web アプリケーションと EJB アプリケーションに同じ設定が適用されます。

- Web アプリケーションを war 形式で、EJB アプリケーションを jar 形式で異なるプロセスグループに配備する構成:

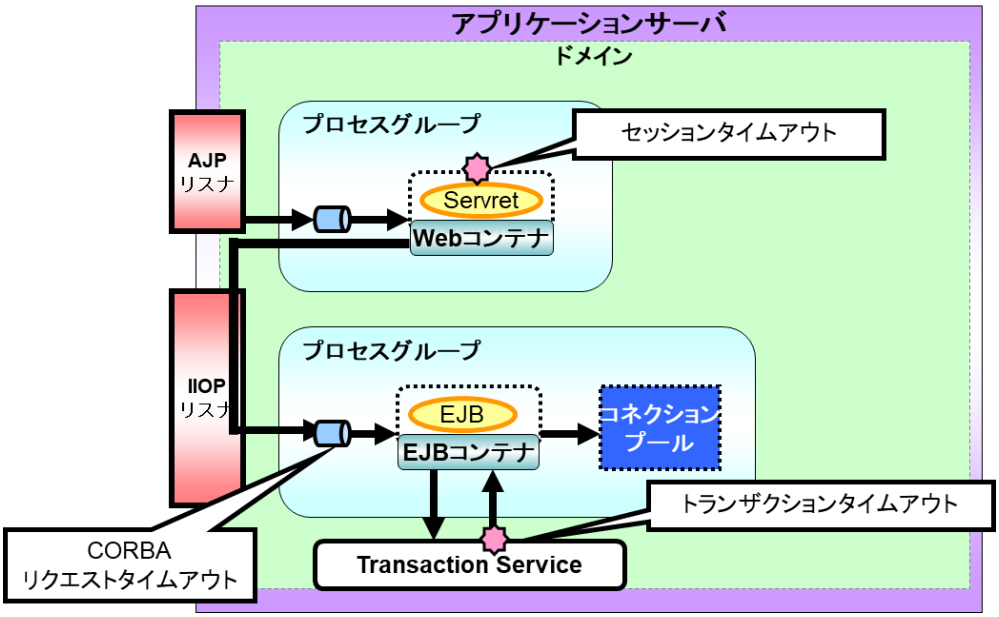


図 2.3.10.アプリケーションサーバ内部構成(プロセスグループ)

Web アプリケーションから EJB アプリケーションの呼出しはリモート呼出しとなります。
この構成ではアプリケーションの設定をプロセスグループ単位で行なうことができ、統計情報もプロセスグループごとに取得できます。

- Web アプリケーションを war 形式で、EJB アプリケーションを jar 形式で一つのプロセスグループに配備する構成:

Web アプリケーションから EJB アプリケーションの呼出しはローカル呼出しとなります。
アプリケーションの設定や統計情報はプロセスグループ単位で行なうため、Web アプリケーションと EJB アプリケー

ションに同じ設定が適用されます。
この構成ではアプリケーションの配備、実行に必要なメモリが他の構成に比べて少ないため、多くのアプリケーションを配備する場合や、使用可能なメモリリソースが制限されている状況で有効です。ただし、ほかの構成に比べて必要な実行スレッドが多いため、スレッド多重度を大きな値に設定する必要があるので注意が必要です。

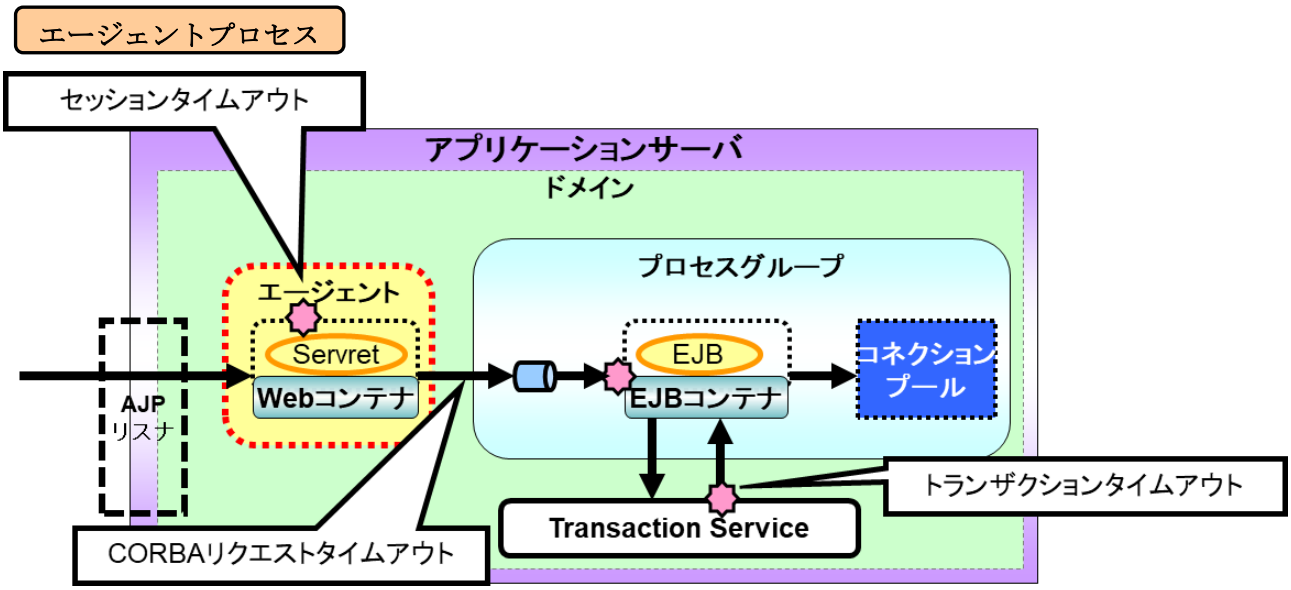


図 2.3.11.アプリケーションサーバ内部構成(エージェントプロセス)

共通

- CORBA リクエストタイムアウト（リクエスト呼び出しのタイムアウト時間）
  呼び出し元と呼び出し先のアプリケーションが異なるアーカイブファイルとしてプロセスグループに配備されている場合、または、エージェントプロセスに配備した運用の場合に、EJB アプリケーションを呼び出す際のタイムアウト値です。オペレーションが、指定された時間を過ぎてもレスポンスを返さない場合、org.omg.CORBA.NO_RESPONSE（マイナーコード:5130）例外が発生します。デフォルトでは 30 秒でタイムアウトします。

- トランザクションタイムアウト
  トランザクションのタイムアウト値です。トランザクションを開始してからこの時間が経過しても完了していない場合はトランザクションサービスが自動的にロールバック処理を実施します。トランザクションタイムアウトが発生した場合、スレッドは消えず SQLの結果が返るまで処理が実行されたままとなります。デフォルトでは 600 秒です。

- セッションタイムアウト
  業務アプリケーション内の web.xml にて設定を行います。業務アプリケーションごとに設定することを推奨します。設定されていない場合、<ドメインのパス>/config/default-web.xml 内で<session-timeout>タグを用いて指定されている値(30 分)を用います。

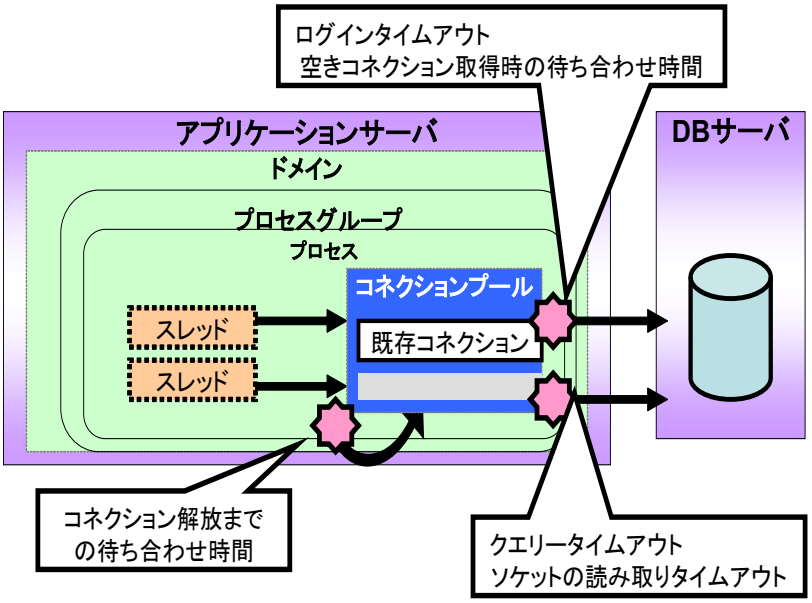


図 2.3.12.JDBC コネクションタイムアウト

- JDBC コネクションタイムアウト
  JDBCに関して設定を行うことができるタイムアウトは、以下のものが挙げられます

  1. ログインタイムアウト(loginTimeout)
     JDBC コネクションを接続する際のタイムアウトです。接続リトライ回数(connectRetryMax)、及び接続リトライ間隔(connectRetryInterval)と関連があります。
  2. コネクション解放までの待ち合わせ時間(shrinkDelayTime)
     コネクションプールに常時保持されるコネクション数を超えて払い出された JDBC コネクションを解放するまでの待ち時間(単位:秒)です。JDBC コネクションは、指定された時間が経過した後で、クローズした時、または、トランザクションが完了した時に解放されます。
  3. クエリータイムアウト(queryTimeout)
     java.sql.Statement に指定するクエリのタイムアウト時間(単位:秒)を指定します。接続リトライ回数(connectRetryMax)、及び接続リトライ間隔(connectRetryInterval)と関連があります。
  4. ソケットの読み取りタイムアウト(readTimeout)
     TCP/IP レベルの無応答を検出するまでの時間(単位:秒)を指定します。クエリータイムアウトよりも長い時間を指定します。この設定は、Oracle の JDBC ドライバを利用する際に有効です。接続リトライ回数(connectRetryMax)、及び接続リトライ間隔(connectRetryInterval)と関連があります。
  5. 空きコネクション取得時の待ち合わせ時間(waitFreeTimeout)
     最大プール数の JDBC コネクションが全て使用中の場合に、空きコネクションが取得できるまで待ち合わせる時間(単位:秒)です。接続リトライ回数(connectRetryMax)、及び接続リトライ間隔(connectRetryInterval)と関連があります。

2.3.2.設計指針

以下の箇所において、設計指針の説明を行います。

1) クライアントで設定するタイムアウト値 共通

2) Web サーバで設定するタイムアウト値 共通

3) アプリケーションサーバで設定するタイムアウト値 共通

1) クライアントで設定するタイムアウト値 共通

- クライアント(Web ブラウザ)タイムアウト
  ブラウザへの応答時間が 30 秒以上かかるケースで設定変更を検討します。

  ただし、この設定はクライアント側に変更作業が必要になるため通常は変更すべきではありません。業務を分割するなど適切なレスポンス時間になるようアプリケーション側で考慮するようにします。

2） Web サーバで設定するタイムアウト値共通

- キープアライブタイムアウト(KeepAliveTimeout)

  同じクライアントからのリクエスト要求が連続してくるような場合、本設定を用いることで、クライアントとの接続に費やす時間を削減することができます。

  しかし、1回のリクエストで処理が完了するような場合には、リクエスト処理が完了してもクライアントとの接続が保持された状態となるため、設定時間が経過するまで待機状態となります。このような運用を行う場合には、本設定値をデフォルト値よりも小さく設定してください。

- コネクションプールタイムアウト(connection_pool_timeout)
  通常は変更する必要はありません。
  アプリケーションサーバに対してのコネクションを保持するため、ネットワーク間の通信に影響を与えます。

- ソケットタイムアウト(socket_timeout)
  通常は変更する必要はありません。
  アプリケーションサーバの障害検知、Web サーバとアプリケーションサーバ間のネットワーク障害検知に影響をあたえます。

- リプライタイムアウト（reply_timeout）

  Web サーバプラグインからアプリケーションサーバに対してリクエストを送信後、レスポンスが返ってくるまでの待ち時間をミリ秒単位で設定します。既定値の 0 ではタイムアウトを行わないため、レスポンスを待ち続けます。このタイムアウトが発生した場合、クライアント（ブラウザ）側には HTTP ステータスコード 504 エラーが返却されます。

3） アプリケーションサーバで設定するタイムアウト値共通

- CORBA リクエストタイムアウト（リクエスト呼び出しのタイムアウト時間）

  EJB アプリケーションを呼び出す際のタイムアウト値を設定します。

  デフォルトでは、30 秒 です。

- トランザクションタイムアウト

  EJBのトランザクション連携設定時にトランザクションタイムアウト値を設定します。

  デフォルトでは、600 秒 です。

- セッションタイムアウト

  業務アプリケーションの web.xml で設定します。

- JDBC コネクションタイムアウト

  詳細については、「2.6 データベース連携」も参照してください。

  1. ログインタイムアウト
     - 設定値
       デフォルトでは 0 秒です。0 秒の場合は、タイムアウトは発生しません。
     - 設計指針
       使用するデータベースに応じて、無応答障害を回避するために設定してください。
  2. コネクション解放までの待ち合わせ時間
     - 設定値
       デフォルトは 15 秒です。
     - 設計指針
       最小プールサイズ、最大プールサイズが異なる場合に設定する必要があります。新規接続が頻繁に行われる場合の負荷を考慮して設定してください。詳細は、WebOTX マニュアル「構築・運用 – チューニン

グ – 7.JDBC データソース」を参照してください。

3. クエリータイムアウト

- 設定値
  デフォルトは 0 秒です。0 秒の場合は、タイムアウトは発生しません。
- 設計指針
  異常に時間がかかるクエリを取り消すために、または、無応答障害を回避するために設定してください。

4. ソケットの読み取りタイムアウト

- 設定値
  デフォルトは 0 秒です。0 秒の場合は、タイムアウトは発生しません。
- 設計指針
  Oracle の JDBC ドライバを利用する際に、TCP/IP レベルの無応答障害を検出するためにクエリータイムアウトよりも長い時間を設定してください。

5. 空きコネクション取得時の待ち合わせ時間

- 設定値
  デフォルトは 15 秒です。0 秒の場合は、既に最大プールサイズ分の JDBC コネクションが使用中の場合に直ちに例外が発生します。
- 設計指針
  アプリケーションの動作スレッド数よりも小さい値を最大プールサイズに指定する場合に設定してください。

2.3.3.設定項目

通信/タイムアウトで決定すべきパラメーター一覧です。

1） クライアントの設定項目 共通

ブラウザ		
Web ブラウザタイムアウト設定 (Firefox の場合)	説明	Firefox のタイムアウト値を設定します。 (単位:秒)
	既定値	300 秒
	推奨値	使用するブラウザに従います。
	設定箇所	ブラウザ。 アドレスバーに about:config を入力し、「危険を承知の上で使用する」ボタンをクリック。network.http.response.timeout を検索し、ダブルクリックで値(秒)を変更。
	タイムアウトした場合に出力されるログ	-

2） Web サーバの設定項目 共通

WebOTX Web サーバ		
キープアライブタイムアウト KeepAliveTimeout	説明	接続を閉じる前に、Web サーバ が次のリクエストを何秒待つかを指定します。 この時間を経過するとクライアント側に FIN を送信します。HTTP キープアライブ(KeepAlive)が On の場合に有効です。 同じクライアントからのリクエスト要求が連続してくるような場合、本設定を用いることで、クライアントとの接続に費やす時間を削減することができます。
	既定値	10 秒

	推奨値	要件にしたがって設定する必要があります。
	設定箇所	<ドメインのパス>/config/WebServer/httpd.conf
	タイムアウトした場合に出力されるログ	-

Web サーバプラグイン(mod_jk)		
コネクションプールタイムアウト connection_pool_timeout	説明	Web サーバとアプリケーションサーバ間でリクエスト完了後に使われなくなったコネクションを切断するタイミングを秒単位で設定します。 これは WebOTX 上で動作するスレッド数を抑えるために使用します。
	既定値(= 推奨値)	59 秒
	設定箇所	<ドメインのパス>/config/WebCont/workers.properties
	タイムアウトした場合に出力されるログ	-
ソケットタイムアウト socket_timeout	説明	リモートホストとコネクタ間におけるソケット通信のタイムアウト時間(秒単位)を指定します。 指定した時間以上が経過すると、エラーを返し、リトライ処理を行います。 既定値である 0 を指定すると、TCP レベルでのタイムアウトが適用されます。
	既定値(= 推奨値)	0 秒
	設定箇所	<ドメインのパス>/config/WebCont/workers.properties
	タイムアウトした場合に出力されるログ	-
レスポンス応答持ち時間 reply_timeout	説明	Web サーバプラグインからアプリケーションサーバにリクエストを送信し、レスポンスを受信するまでの待ち時間(ミリ秒)を指定します。0 を設定した場合はタイムアウトしません。
	既定値(=推奨値)	0 ミリ秒
	設定箇所	<ドメインのパス>/config/WebCont/workers.properties
	タイムアウトした場合に出力されるログ	ブラウザ側に 504 (Gateway Time-out)が返却されます。

3) アプリケーションサーバの設定項目 共通

Object Broker		
リクエスト呼び出しのタイムアウト時間 RequestTimeout	説明	CORBA や EJB アプリケーションを呼び出す際の最大待ち時間を設定します。設定時間(秒)を過ぎてもレスポンスが返却されない場合、org.omg.CORBA.NORESPONSE(5130)例外が発生します。
	既定値	30 秒
	推奨値	アプリケーションの処理時間に応じて設定する必要があります。
	設定箇所	[運用管理ツール]-[アプリケーションサーバ]-[Object Broker コンフィグ]-[共通]-[リクエスト呼び出しのタイムアウト時間]
	タイムアウトした場合に出力されるログ	-
トランザクション		
トランザクションタイムアウト tx-timeout	説明	トランザクションタイムアウト時間(秒)を指定します。
	既定値(= 推奨値)	600 秒
	設定箇所	[運用管理ツール]-[アプリケーションサーバ]-[Transaction サービス]-[TM 設定]-[トランザクションタイムアウト時間]

	タイムアウトした場合に出力されるログ	-
セッション		
セッションタイムアウト <session-timeout>タグ	説明	セッションのタイムアウト時間(分)を指定します。
	既定値	30 分
	推奨値	業務アプリケーションごとに設定してください。
	設定箇所	業務アプリケーション内の web.xml にて設定を行います。業務アプリケーションごとに設定することを推奨します。 設定されていない場合、<ドメインのパス>/config/default-web.xml 内で<session-timeout>タグを用いて指定されている値を用います。
	タイムアウトした場合に出力されるログ	-
データソース		
ログインタイムアウト loginTimeout	説明	コネクション接続時のタイムアウト値(秒)を指定します。0 を指定した場合、監視は行われません。
	既定値	0 秒
	推奨値	15 秒程度
	設定箇所	[運用管理ツール]-[リソース]-[JDBC データソース]-[<JDBC データソース名>]-[コネクション制御]-[ログインタイムアウト]
	タイムアウトした場合に出力されるログ	-
コネクション解放までの待ち合わせ時間 shrinkDelayTime	説明	最小プールサイズを超えて払い出されたコネクションを解放するまでの待ち時間(秒)を指定します。値が 0 の場合待ち合わせは行われません。
	既定値	15 秒
	推奨値	60 秒
	設定箇所	[運用管理ツール]-[リソース]-[JDBC データソース]-[<JDBC データソース名>]-[コネクション制御]-[コネクション解放までの待ち合わせ時間]
	タイムアウトした場合に出力されるログ	-
	備考	指定した時間が経過した段階では、コネクションは解放されません。未使用状態が指定した時間が継続された後に該当コネクションが使用され、クローズした時、または、トランザクションが完了した時に解放されます。
クエリータイムアウト queryTimeout	説明	java.sql.Statement に指定するクエリのタイムアウト時間（単位：秒）を指定します。0 を指定した場合、監視は行われません。
	既定値	0 秒
	推奨値	300 秒(実際のクエリの実行時間に応じて調整してください。)
	設定箇所	[運用管理ツール]-[リソース]-[JDBC データソース]-[<JDBC データソース名>]-[コネクション制御]-[クエリータイムアウト]
	タイムアウトした場合に出力されるログ	-
ソケットの読み取りタイムアウト readTimeout	説明	TCP/IP レベルの無応答を検出するまでの時間(単位：秒)を指定します。 この設定は、Oracle の JDBC ドライバを利用する際に有効です。0 秒の場合は、タイムアウトは発生しません。
	既定値	0 秒
	推奨値	クエリータイムアウトよりも 10 数秒程度長い時間に設定してください。
	設定箇所	[運用管理ツール]-[リソース]-[JDBC データソース]-[<JDBC データソース名>]-[コネクション制御]-[ソケットの読み取りタイムアウト]
	タイムアウトした場合に出力されるログ	-

空きコネクション取得時の待ち合わせ時間 waitFreeConnTimeout	説明	最大プール数の JDBC コネクションが全て使用中の場合に、空きコネクションが取得できるまで待ち合わせる時間（単位：秒）を指定します。 0 秒の場合は、既に最大プールサイズ分の JDBC コネクションが使用中の場合に直ちに例外が発生します。
	既定値	15 秒
	推奨値	アプリケーションの処理時間に応じて設定する必要があります。
	設定箇所	[運用管理ツール]-[リソース]-[JDBC データソース]-[<JDBC データソース名>]-[コネクションプール]-[空きコネクション取得時の待ち合わせ]
	タイムアウトした場合に出力されるログ	-

2.4.通信/セッション

クライアントが、サーバ上の業務アプリケーションにリクエストを送信する際、サーバ内で一時的にクライアントの情報を保持する仕組みがあります。この仕組みを、セッション管理と呼びます。
また、複数のプロセスやサーバで構成されたシステムにおいて、セッション情報を共有することができます。複数のプロセスやサーバでセッション情報を共有することで、一部のサーバに障害が発生しても、セッション情報の消失を防ぐことができます。
本節では、セッション管理の仕組みについて、プロセスグループに配備した構成の場合や複数のサーバを用いたときのセッション管理について、説明を行います。

2.4.1.概要説明

主に以下の項目に対し、説明を行います。

1） セッション管理 共通

- セッション
- セッションの保持方法
- セッションタイムアウト
- セッションレプリケーション

2） プロセスグループを利用する場合のセッションレプリケーション プロセスグループ

1） セッション管理 共通

セッション
セッションとは、Web サイトを訪れたユーザが、サイト内で行う一連の行動のことです。たとえば、電子商取引においては、ログインからログアウトまでを 1 セッションとしています。セッションを管理するには、セッション ID と呼ばれる識別子を利用します。セッション ID はユーザごとに与えられ、セッション ID からどのユーザがサーバにリクエストを送信しているのかを判別することができます。このセッション ID を利用することで、あるユーザが一連の行動中に複数の Web ページを読み込んだとしても、同じユーザからの要求として処理することが可能となります。

- セッションの保持方法
  セッションの情報の最適な保持方法は、システムの構成や要件によって異なります。WebOTX ではセッションの保持方法として以下の方法を提供しています。システムの構成や要件により、適する保持方法を選択する必要があります。
  一般的に、Webコンテナへの保持はアプリケーションサーバが 1 台である場合や、クライアントが常に接続するアプリケーションサーバが同じ場合に利用されます。JNDI サーバへの保持は、プロセスグループを利用する場合や、複数のアプリケーションサーバを用いて負荷分散を行う場合などに利用されます。

  - Web コンテナに保持する
    Web コンテナのメモリにセッション情報を保持します。複数の Web コンテナが存在する場合には、クライアントは常に同じ Web コンテナに接続する必要があります。また、Web コンテナに障害が起こった場合、セッション

情報は消えてしまいます。このため、プロセスグループを利用する場合や、アプリケーションサーバが複数あり負荷分散を行う場合などには、Web コンテナへの保持は適切ではありません。

- JNDI サーバに保持する
  WebOTX では、Web コンテナのクラスタ対応機能を利用することで、Web コンテナに加えて、JNDI サーバにセッション情報を保持することができます。JNDIサーバに保持することで、プロセスグループを利用する場合に複数のプロセスで同じセッション情報を共有することができます。また、複数の JNDI サーバが存在する場合にはレプリケーションを行うことが可能です。アプリケーションサーバが複数ある場合には、セッションレプリケーションを利用することで負荷分散を行うことができます。

➢ セッションタイムアウト
いつまでもセッション情報を保持しておくことは、メモリの無駄な消費に繋がるため、セッション情報の保持にはタイムアウトを設定する必要があります。WebOTX では、デフォルトのタイムアウト値は 30 分となっています。ユーザに求められる条件(セッションを長時間保持することが必須など)により、タイムアウトの値を設定する必要があります。

➢ セッションレプリケーション
セッションレプリケーションとは、セッション情報を複数のアプリケーションサーバやプロセスで保持しておく機能のことです。

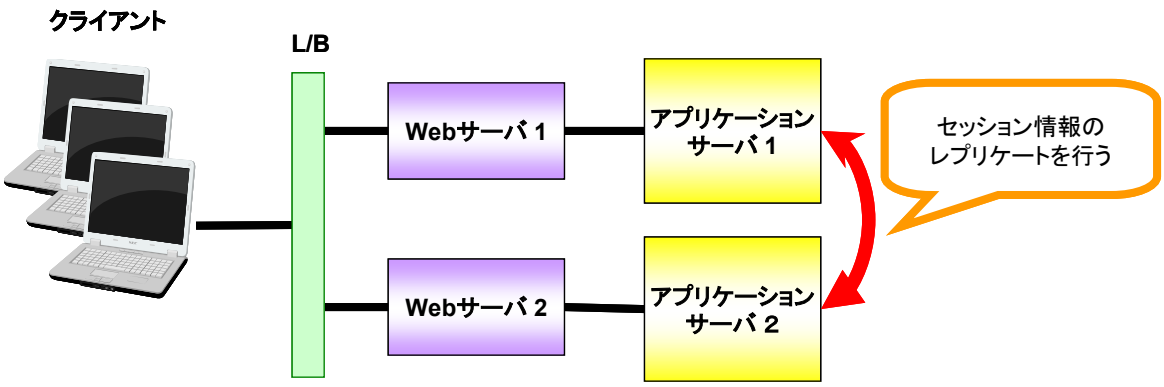


図 2.4.1.セッションレプリケーション(複数のアプリケーションサーバの場合)

セッションレプリケーションは、次のような場合に有効です。

- ロードバランサにより負荷分散が行われている場合
  負荷分散を行っている場合、クライアントがロードバランサによりどのサーバにリクエストを行っても同じセッション情報を取得することができます。
- 一部のサーバに障害が発生した場合
  一部のサーバに障害が発生しても、他のサーバの情報を元に復元することができるためセッション情報の消失を防ぐことができます。
- プロセスグループを利用する場合
  プロセスグループを利用する場合、プロセス毎に Web コンテナが存在します。JNDI サーバにセッションを保持させることにより、複数の Web コンテナでセッション情報を共有することができます。詳細は、「(2)プロセスグループを利用する場合のセッションレプリケーション」を参照して下さい。

2) プロセスグループを利用する場合のセッションレプリケーション プロセスグループ

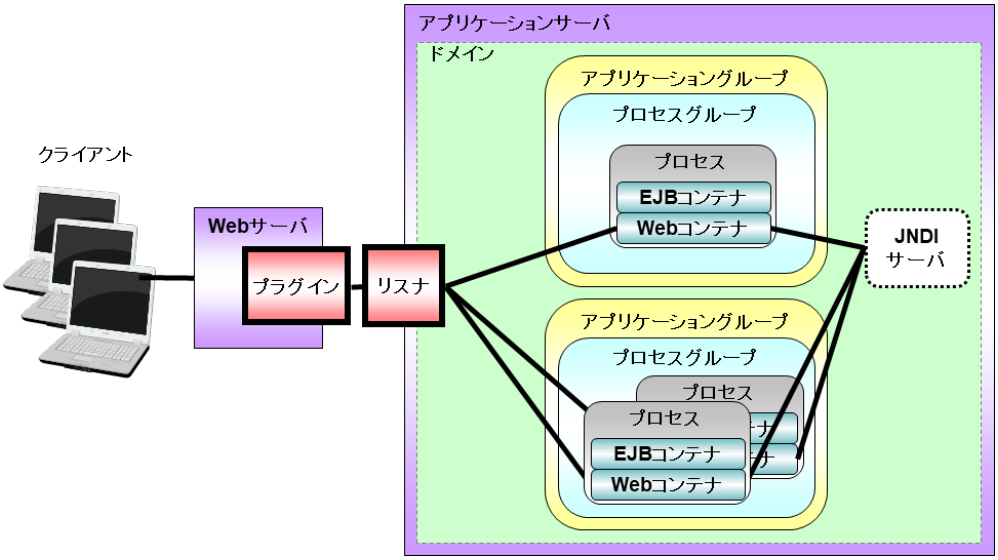


図 2.4.2.セッションレプリケーション(プロセスグループを利用する場合)

プロセスグループを利用した場合、プロセスグループのプロセス上でそれぞれ Web コンテナが動作します。このとき、セッションレプリケーションを利用してセッション情報を JNDI サーバに保持しておくことで、同じセッションを複数のプロセス間、プロセスグループ間で共有することができます。プロセスグループ上で動作するアプリケーションは、Web コンテナを通して、JNDI サーバにセッション情報の登録・取得を行います。そのため、一部のプロセスに障害が発生しても、途中まで実行された処理の続きを他の正常なプロセスで行うことができます。

2.4.2.設計指針

1） セッション管理 共通

- セッションの保持方法
  概要説明で述べたように、セッション情報の保持には、Web コンテナに保持する場合と、Web コンテナだけではなく JNDI サーバにも保持する場合があります。セッションレプリケーションを行う場合は、JNDI サーバに保持させる必要があります。JNDI サーバに保持させる場合は、Web アプリケーションの web.xml に<distributable>タグの記述を追加します。
  また、セッションオブジェクトはシリアライズされている必要があります。JNDI サーバへの更新/参照にはシリアライズ／デシリアライズが行われるため複雑なコレクションクラスの処理には時間がかかります。そのため、セッション情報に使用するオブジェクトはシンプルなものにしてください。

- セッションタイムアウト
  Web アプリケーション内の web.xml の<web-app>-<session-config>-<session-timeout>タグにて設定を行います。この値は、クライアントが 1 セッション内で行う操作がどのくらい時間を要するかにより、適切な値を設定する必要があります。値を設定しない場合は、既定値の 30 分となります。

  設定例）
  たとえば、ショッピングサイトにて、ユーザが商品を閲覧するために1時間画面を保留することがあるとします。この場合、タイムアウト値が 30 分だとすると、閲覧後に操作しようとすると再度ログインを行う必要が発生します。際ログインの必要をなくすためには、タイムアウトの時間は余裕を持って1時間以上を設定する必要があります。

- セッションレプリケーション
  WebOTX では、JNDI サーバに保持する方法を用いる場合に、セッションレプリケーションの機能を提供しています。概要説明でも述べたとおり、JNDI サーバに保持することは、プロセスグループを利用する場合や複数のアプリケーションサーバが存在する場合に有用です。
  複数のアプリケーションサーバが存在する場合、JNDI サーバに保持されているセッション情報の同期を取ることでセッション情報を共有し、システムの信頼性を向上させることができます。以下で、複数のアプリケーションサーバ間でセッションレプリケーションを実施している環境において、JNDI サーバに対しセッション情報を登録・削除する場合、JNDI サーバからセッション情報を取得する場合、JNDI サーバに障害が起こった場合の動作を説明します。

- JNDI サーバに対し、セッション情報を登録・削除する場合

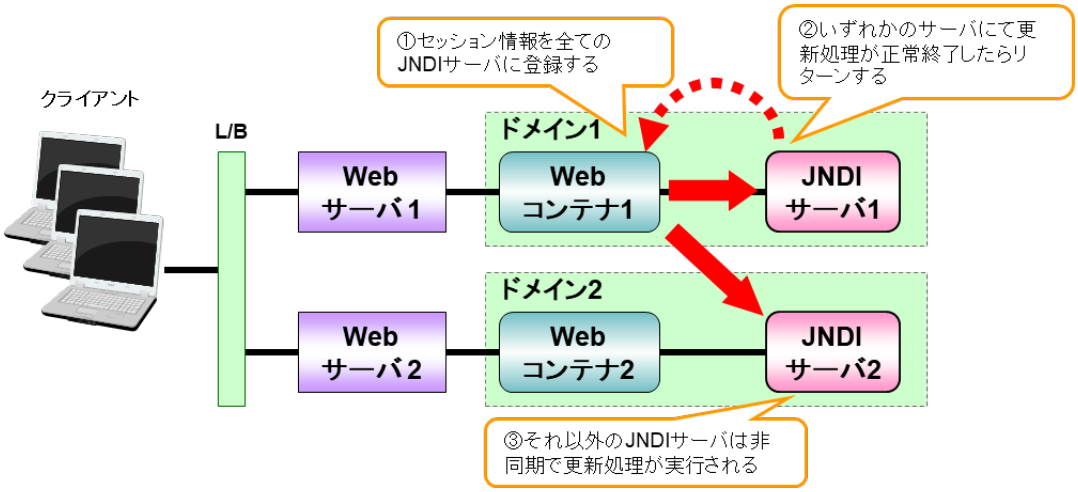
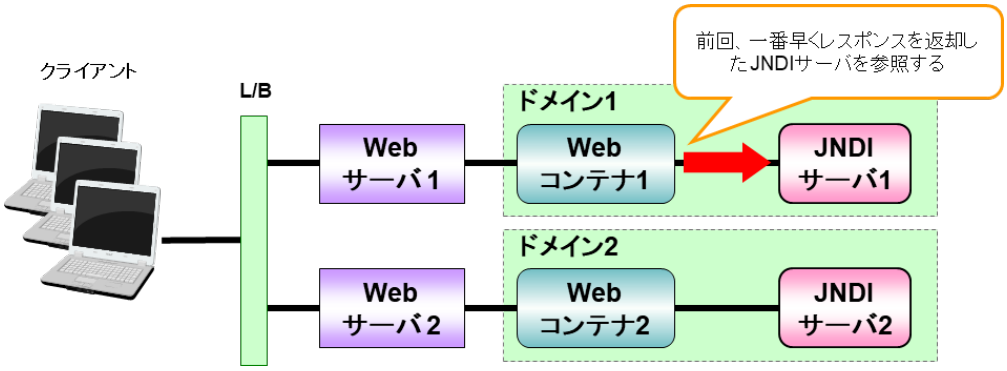


図 2.4.3.セッション情報の登録

クライアントがアプリケーションサーバにアクセスすると、Web コンテナはセッション情報を作成し、レプリケーションを行う全ての JNDI サーバにセッション情報を登録します。そのためセッションレプリケーション時には、レプリケーションを行う全ての JNDI サーバの URL を各 Web コンテナに登録しておくことが必要となります。

- JNDI サーバからセッション情報を取得する場合



※参照に失敗した場合、エラー内容に従い、他の JNDI サーバ対してリトライを行います

図 2.4.4.セッション情報の参照

Web コンテナが JNDI サーバに登録されたセッション情報を取得する場合は、全ての JND
I サーバのうちどれか 1 つからセッション情報を取得します。

- JNDI サーバに障害が起こった場合

JNDI クライアントは、レプリケーション対象である JNDI サーバの死活監視を定期的に行います。

死活監視により、対象の JNDI サーバの障害を検知した場合には、その JNDI サーバを一時的に更新・参照の対象から外します(縮退運転)。縮退運転中も死活監視は行われており、障害が発生した JNDI サーバが再起動などで復旧した場合には、これを検知して、再び更新・参照の対象とします。この際、JNDI クライアントが、復旧した JNDI サーバに対して、他の JNDI サーバが持つセッション情報を取り込むように指示します。これにより、復旧した JNDI サーバは、他の JNDI サーバと同様、最新のセッション情報を保持します。

同期中の JNDI サーバは Web コンテナからの参照のみを受け付けます。セッション情報が多い場合は同期に時間がかかる可能性があります。同期に時間がかかると、その間Webコンテナからのセッション情報の更新ができないため、待ち時間が発生してしまう場合があります。

複数サーバでセッションレプリケーションを行う場合は、セッション情報の登録・削除を JNDI サーバに対して行うため、Web コンテナのみに保持する場合に比べて、処理時間とメモリ使用量が増大します。さらに、セッション情報のサイズが大きい場合や、アプリケーションサーバの台数が多い場合は、レプリケーションにより多くの時間がかかります。大規模なシステムにおいて、複数台サーバによるセッションレプリケーションを行う場合は、性能の劣化に注意してください。

2) プロセスグループを利用する場合のセッションレプリケーション プロセスグループ

プロセスグループを利用する構成で、複数のプロセスで同じ業務を行う場合に、セッション情報を JNDI サーバに保持しておくことで、複数のプロセス間でセッション情報を共有することができます。JNDI サーバに保持させる場合、クライアントからセッション情報の登録・参照・削除の要求があると、Web コンテナは同じアプリケーションサーバ内の JNDI サーバと通信して、セッション情報の登録・参照・削除を行います。
必要となる設定は、JNDI サーバに保持させるための、web.xml への<distributable>タグの記述を追加と、セッションオブジェクトをシリアライズ可能なものとすることです。

2.4.3.設定項目

通信/セッションで決定すべきパラメーター一覧です。

1) セッション管理について 共通

web.xml

設定パラメータ(属性名)	説明	既定値	推奨値
セッションのクラスタ化 <distributable>タグ	JNDI サーバにセッションを保持する場合に指定します。web.xml に記述します。	記述されていない	<distributable>タグを記載してください。
セッションタイムアウト <session-timeout>タグ	セッションのタイムアウト時間(分)を指定します。web.xml 内に記述します。	30	システム要件にしたがってください。

Web コンテナ

設定パラメータ(属性名)	説明	既定値	推奨値
JNDI サーバの URL session-replication-jndi-url	セッションレプリケーション時の JNDI サーバの URL を指定します。複数の URL を記載する場合は、カンマで区切って指定してください。	記述されていない	セッションレプリケーションを行う JNDI サーバの URL をカンマ区切りで記載してください。

エージェントプロセスのシステムプロパティ

設定パラメータ(属性名)	説明	既定値	推奨値
セッションレプリケーションの監視間隔 webotx.jndi.listinginterval	セッションレプリケーション時の JNDI サーバの監視間隔を指定します。	10	システム要件にしたがってください

2.5.セキュリティ

本節では、通信時のセキュリティ強化について、説明を行います。また、他の WebOTX のセキュリティ設定についても説明します。

2.5.1.概要説明

セキュリティでは、以下の項目について説明を行います。

1） 通信時のセキュリティを強化する 共通

- ファイアーウォールの設定
- SSL の利用
- 不正アクセス・攻撃の防止

2） その他のセキュリティ設定 共通

- パスワードの管理について

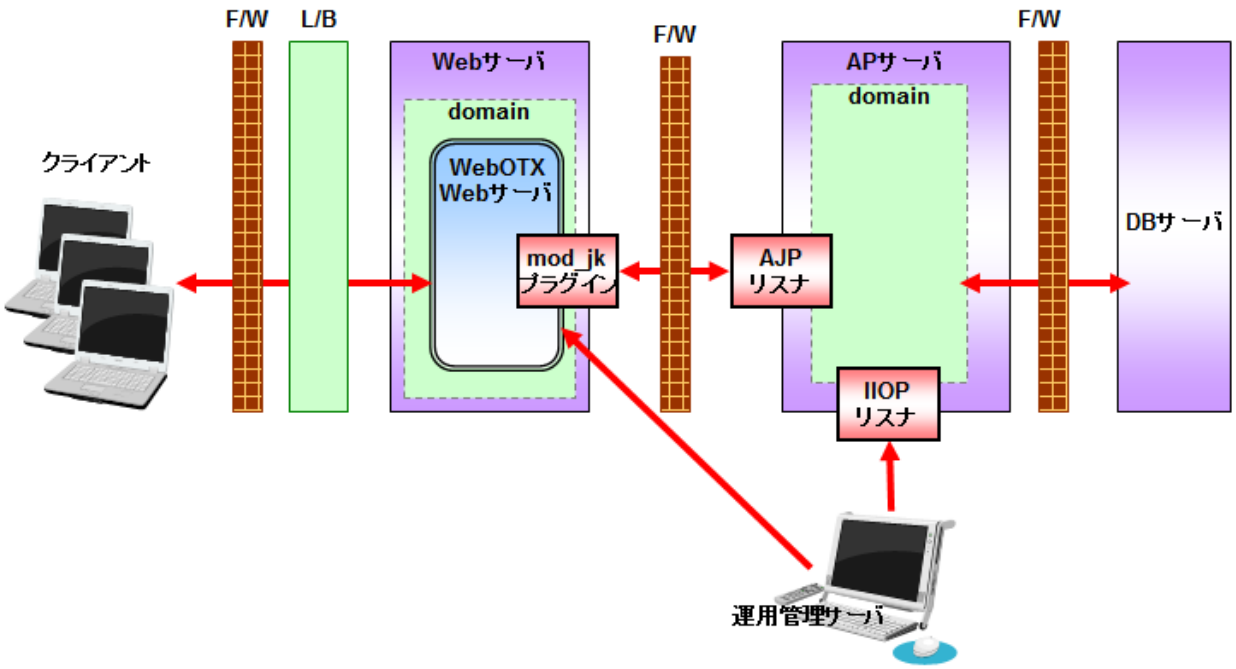


図 2.5.1.全体構成

1） 通信時のセキュリティを強化する 共通

- ファイアーウォールの設定
  クライアント、Web サーバ、アプリケーションサーバがお互いに通信を行うために、数種類のポートを使用します。このため、必要なポートが使用できるように、ファイアーウォールの設定を行う必要があります。図 2.5.1 の F/W が、ポートの設定が必要となるファイアーウォールです。

- SSL の利用
  通信時の情報が漏洩することを防ぐために、クライアントと Web サーバ間の通信にて、SSL を利用することができます。WebOTX Web サーバでは、OpenSSL ライブラリを利用した mod_ssl モジュールを使用して、SSL 3.0 および TLS 1.0/1.1/1.2/1.3 を利用し、かつ 128Bit 以上の暗号化方式をサポートしたセキュアな Web サイトを構築することができます。また、SSL クライアント認証機能も利用可能です。

- 不正アクセス・攻撃の防止
  クライアントからのアクセスを受け付ける Web サーバは、不正アクセスや、侵入・攻撃を受ける可能性があります。これを防ぐために、クライアントに公開する情報は必要最低限のものとし、不要な情報は非公開とする必要があります。

2） その他のセキュリティ設定 共通

そのほか、一般的なセキュリティの設定として以下のものがあります。

➢ パスワードの管理について
WebOTXには、管理パスワードが存在します。インストール時には既定値となっていますので、インストール後に必ず変更するようにして下さい。

2.5.2.設計指針

セキュリティを考慮する上で必要となる指針を記載します。

1） 通信時のセキュリティを強化する 共通 プロセスグループ エージェントプロセス

2） その他のセキュリティ設定 共通

1） 通信時のセキュリティを強化する 共通 プロセスグループ エージェントプロセス

共通

➢ ファイアーウォールの設定
WebOTX では、次のポートを使用します。必要に応じて、ファイアーウォールの設定を行って下さい。

- クライアント － Webサーバ間

	説明	既定値
http 通信用ポート	HTTP サーバが利用するポート番号です。	80
SSL(HTTPS)通信用ポート	SSL 用のポート番号です。	443

- クライアント － アプリケーションサーバ間

	説明	既定値
IIOP リスナ通信ポート (平文)	IIOP リスナが使用するポート番号です。平文を利用する場合に利用します。	5151
IIOP リスナ通信ポート (SSL クライアント認証あり)	IIOP リスナが使用するポート番号(SSL クライアント認証ありポート)SSL クライアント認証ありを利用する場合に利用します。	なし
IIOP リスナ通信ポート (SSL クライアント認証なし)	IIOP リスナが使用するポート番号(SSL クライアント認証なしポート)SSL クライアント認証なしを利用する場合に利用します。	なし
IIOP リスナ通信ポート (JNDI サーバとの通信用)	エージェントプロセス上で動作する IIOP リスナのポートです。JNDI サーバとの通信を行います。	7780
IIOPAsync のポート番号	クライアント管理ライブラリや VB クライアント、CORBA Gateway を使用している場合、クライアントと通信を行うために使用します。	5220

- Webサーバ/アプリケーションサーバ － 運用管理サーバ間

	説明	既定値
管理ドメイン運用管理ポート (※)	管理ドメインのエージェントが利用する RMI のポート番号	6202
ユーザドメイン運用管理ポート (※)	ユーザドメインのエージェントが利用する RMI のポート番号	6212
運用管理コンソールポート	エージェントが利用する管理用ポートの番号。Web 管理コンソールとの通信に利用	5858

(※) RMI 通信では一時ポートを使用します。Web サーバ/アプリケーションサーバの一時ポートに、運用管理サーバから接続できるようファイアーウォールの設定を行なってください。一時ポートの範囲は、OS の設定を確認してください。

- アプリケーションサーバ － DBサーバ間

	説明	既定値
データベース用ポート	データベースサーバのポート番号です。	なし

➢ SSL の利用
クライアントと Web サーバ間の通信にて SSL を利用するためには、Web サーバの SSL 利用の設定をONにすることと、SSL 通信用のポートを確保することが必要となります。秘密鍵の生成、証明書の取得、秘密鍵や証明書の設定は、環境により異なりますので、要件に合わせて利用してください。

➢ 不正アクセス・攻撃の防止
不正アクセスや、侵入・攻撃を防ぐために、具体的に以下の設定を行う必要があります。

1. Web サーバのディレクトリリスティング機能無効化
2. クライアントへの応答のヘッダ/フッタに含める情報の制限
3. Web サーバのマニュアル/コンテキストの削除

プロセスグループ

➢ ファイアーウォールの設定
WebOTX では、次のポートを使用します。必要に応じて、ファイアーウォールの設定を行って下さい。

- Webサーバ － アプリケーションサーバ間

	説明	既定値
AJP リスナ (プロセスグループ用)	AJP リスナが使用するポート番号です。外部の HTTP サーバと AJP リスナが AJP によって連携を行う際のポートの番号。プロセスグループを利用する構成で HTTP サーバを起動して、AJP リスナを使用する場合利用します。	20102

エージェントプロセス

➢ ファイアーウォールの設定
WebOTX では、次のポートを使用します。必要に応じて、ファイアーウォールの設定を行って下さい。

- Webサーバ － アプリケーションサーバ間

	説明	既定値
AJP リスナ (エージェントプロセス用)	AJP リスナが使用するポート番号です。外部の HTTP サーバと Web コンテナが AJP によって連携を行う際のポートの番号。HTTP サーバ、Web コンテナを起動した場合利用します。	8099

2) その他のセキュリティ設定 共通

➢ パスワードの管理について
WebOTX 管理者のパスワードは、インストールした時点では既定値となっています。インストール後に変更するようにしてください。
WebOTX の管理ユーザは、ドメイン単位に登録されています。パスワードの変更は、管理ドメインとユーザドメインそれぞれに対して行います。管理ドメインとユーザドメインの管理者パスワードは運用の便宜性から同一にしておきます。管理ユーザのパスワードは、8文字以上としてください。

2.5.3.設定項目

セキュリティで決定すべきパラメーター覧です。

1） 通信時のセキュリティを強化する 共通 プロセスグループ エージェントプロセス

共通

ファイアーウォールの設計では以下の内容を決定します。

アプリケーションサーバ

設定パラメータ(属性名)	説明	既定値
管理ドメイン運用管理ポート番号 port	管理ドメインのエージェントプロセスが利用する RMI 通信ポート番号を指定します。	6202
ユーザドメイン運用管理ポート番号 port	ユーザドメインのエージェントプロセスが利用する RMI 通信を行うためのポート番号を指定します。	6212

※上記のパラメータはドメイン作成時のプロパティファイルにて設定を行います。

Web コンテナ

設定パラメータ(属性名)	説明	既定値
HTTP 通信用ポート番号 port	HTTP サーバが利用する HTTP ポートの番号を指定します。HTTP サーバを起動した場合に利用します。	80
SSL(HTTPS)通信用ポート番号 ssl-port	SSL で保護された HTTPS ポートの番号を指定します。HTTP サーバを起動した場合に利用します。	443
運用管理コンソールポート番号 port	エージェントが利用する管理用ポートの番号を指定します。Web管理コンソールとの通信に利用します。	5858

IIOP リスナ

設定パラメータ(属性名)	説明	既定値
IIOP リスナ通信ポート番号(平文) listenerPortNumber	TP システム上の IIOP リスナが使用するポート番号を指定します。平文を利用する場合に利用します。	5151
IIOP リスナ通信ポート番号(SSL) sslPortNumberCert	TP システム上の IIOP リスナが使用するポート番号を指定します。SSL クライアント認証ありを利用する場合に利用します。	なし
IIOP リスナ通信ポート番号(SSL) sslPortNumberNoCert	TP システム上の IIOP リスナが使用するポート番号を指定します。SSL クライアント認証なしを利用する場合に利用します。	なし
IIOP リスナ通信ポート番号(JNDI) port	エージェントプロセス上で動作する IIOP リスナのポートを指定します。JNDI サーバとの通信に利用します。	7780
iiopAsync ポート番号	クライアント管理ライブラリや VB クライアント、CORBA Gateway を使用している場合、クライアントと通信を行うために使用するポート番号を指定します。	5220

プロセスグループ

AJP リスナ

設定パラメータ(属性名)	説明	既定値
AJP リスナ(プロセスグループ用) 通信ポート番号 Port	AJP リスナが使用するポート番号です。外部の HTTP サーバと AJP リスナが AJP によって連携を行う際のポートの番号。プロセスグループを利用する構成で HTTP サーバを起動して、AJP リスナを使用する場合利用します。	20102

エージェントプロセス

AJP リスナ

設定パラメータ(属性名)	説明	既定値
AJP リスナ(エージェントプロセス用) 通信ポート番号 Port	AJPリスナが使用するポート番号です。外部の HTTP サーバと Web コンテナが AJP によって連携を行う際のポートの番号。HTTP サーバ、Web コンテナを起動した場合利用します。	8099

JDBC データソース

設定パラメータ(属性名)	説明	既定値

データベースサーバポート番号 portNumber	データベースサーバのポート番号を指定します。	なし

SSL の設計では、以下の内容を決定します。

設定パラメータ(属性名)	説明	既定値
SSL 通信の利用 security-enabled	SSL 通信を利用するかを指定します。 ※運用管理コンソールにて SSL 通信を利用するには Admin プロトコルの「SSL 通信の利用の有無」をチェックします。	false
SSL(HTTPS)通信用ポート番号 ssl-port	SSL で保護された HTTPS ポートの番号です。HTTP サーバを起動した場合に利用します。	443

不正アクセス・攻撃の防止では、以下の内容を決定します。

httpd.conf(WebOTX Web サーバの設定)

設定パラメータ	説明	既定値	推奨値
Directory ディレクティブ (ディレクトリのアクセス権の設定)	指定したディレクトリに対してのアクセス制限、アクセス許可、アクセス拒否を設定します。Indexes を削除することで、ディレクトリ表示を無効とします。	Option FollowSymLinks	Option FollowSymLinks
ServerSignature ディレクティブ	エラーメッセージ出力時のフッタを表示するか否かの設定を行います。	On	Off
ServerTokens ディレクティブ	クライアントへの応答ヘッダや、サーバが生成するドキュメント(エラードキュメント等)に含める情報を指定します。ProductOnly とすると、WebOTX Web サーバの名前のみをヘッダに含めます。	Full	ProductOnly
Directory ディレクティブ (マニュアル、コンテキスト情報を削除)	Web サーバのマニュアル、コンテキスト情報を指定します。これらの部分をコメントアウトすることで、マニュアル、コンテキスト情報を削除できます。	AliasMatch 略 <Directory "/opt/WebOTX/WebServer24/manual"> 略 </Directory>	# AliasMatch 略 # <Directory "/opt/WebOTX/WebServer24/manual"> # 略 # </Directory>

default-web.xml(Web コンテナの設定)

設定パラメータ	説明	既定値	推奨値
<servlet> <servlet-name>default</servlet-name> ・・・ <init-param> <param-name>listings</param-name> <param-value>false</param-value> </init-param> </servlet>	Web コンテナ上で動作する、全ての Web アプリケーションのディレクトリリスティングの無効化を行います。	false	false

2) その他のセキュリティ設定 共通

その他のセキュリティの設計では、以下の内容を決定します。

パスワードの管理では、以下の内容を決定します。

設定パラメータ(属性名)	説明	既定値
運用ユーザ（admin）パスワード	update-file-user コマンドで指定します。8 文字以上で設定してください。	adminadmin

2.6.データベース連携

本節では、データベース連携について説明します。

AP サーバが DB サーバの情報を参照/更新するためには、AP サーバから DB サーバへの接続の設定や、アプリケーション実行時の性能向上、データベース障害に対する対処を考える必要があります。

APサーバからDBサーバへアクセスするデータベース連携処理では、大きく次の3つの処理に分けて設計します。

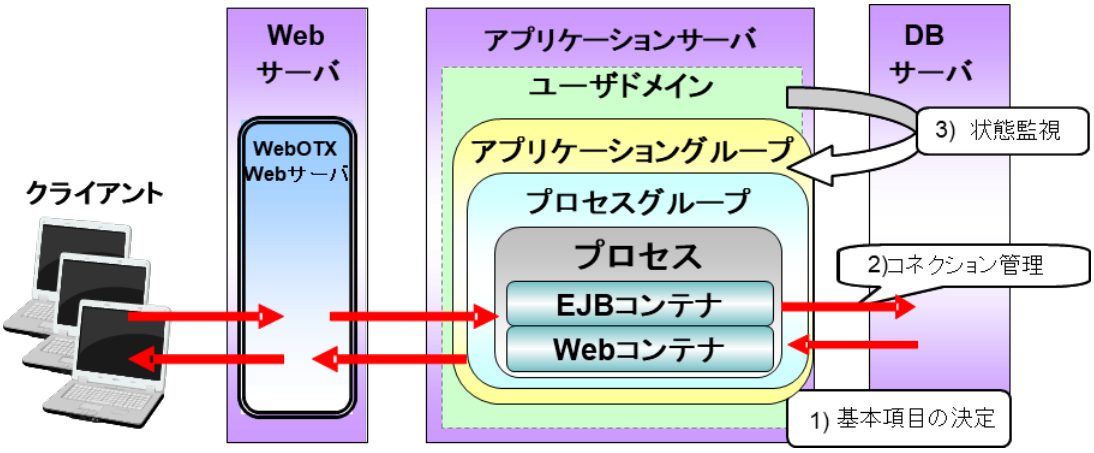


図 2.6.1 データベース連携処理概略図

1） 基本項目の決定

データベースへアクセスするための基本事項を決定します。

2） コネクション管理

データベースへのアクセス速度の向上のために、コネクションの保持の方法を設計します。

3） 状態監視

データベース障害発生時の動作を検討します。

ここでは、それら項目に対する詳細設定を示します。

なお本節では、データソースとしてJDBCを前提に説明いたします。

2.6.1.概要説明

データベース連携について、以下の3項目に関して説明します。

1） 基本項目の決定 共通

ここでは、データベースと接続する際に必要な基本項目を決定します。

WebOTX ではJDBCデータソースを使用してデータベースにアクセスすることを推奨します。

JDBCデータソースとは、プログラムとデータベースへの接続との間のインタフェースで、データベースへの接続を取得するために使用されます。

また、トランザクション処理ではACID特性を保証する必要があります。トランザクション処理では、一貫した状態のデータを維持するよう設計されており、相互依存のある複数の操作が全て完了するか、全てキャンセルされることを保証します。1 つのトランザクションが複数のデータベースに対してアクセスする分散トランザクションでは、JDBC データソースのデータソースタイプとして、2 フェーズコミット用の JDBC Optional Package API (JDBC_xxxx)を選択する必要がありますが、2 フェーズコミットを行わない場合に、2 フェーズコミット用のデータソースタイプを選択すると処理が遅くなるため注意が必要です。

複数 DB 接続を行う場合は、2 フェーズコミットを行う必要があるかどうか(トランザクション処理の ACID 特性を保証する必要があるか)の検討結果をふまえて、データソースタイプを選択します。

2) コネクション管理 共通

WebOTX では、ドメイン起動時、またはプロセスグループ起動時にデータベースとの接続を行っておくことができます。それにより、アプリケーションでは、接続にかかる処理コストを気にすることなく、業務処理速度を向上させることができます。

3) 状態監視 共通

WebOTX では、データベースの状態監視機能を提供します。データベースサーバの状態監視コマンドを発行することでデータベースサーバの状態を確認し、無効となったコネクションを破棄することができます。また、データベースサーバ復旧時に破棄されたコネクションを接続し直すこともできます。

そのため、状態監視は”2)コネクション管理”と密接に関係します。

2.6.2.設計指針

APサーバからDBサーバへアクセスするDB連携処理は、大きく次の3つの処理についての設計指針を記載します。

1) 基本項目の決定 共通
2) コネクション管理 共通
3) 状態監視 共通

1) 基本項目の決定

基本項目の決定では、以下の内容を決定します。

- データベースの接続設定
- プロセスグループ単位の設定
- 複数DB接続の検討
- カスタマイズテンプレートの採用検討

➢ データベースの接続設定

データベースと接続する上で決めなければならない基本項目について決定します。

- データソースの種別

  データソースの種別とは、JDBC ドライバベンダが提供するインタフェースの種別を表わす文字列です。

  以下のフローにてデータソースの種別を決めることができます。

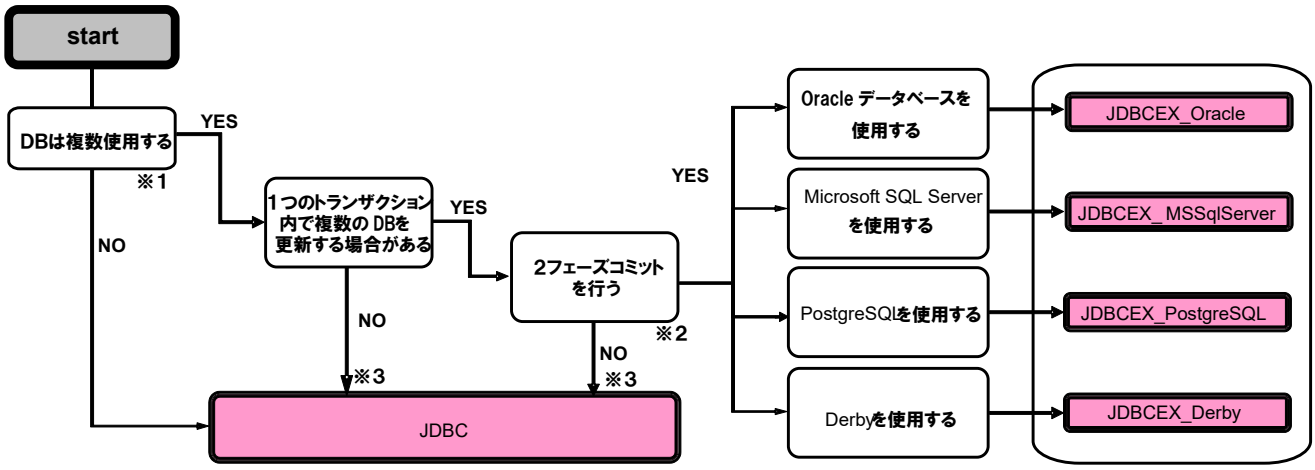


図 2.6.2.データソース種別決定フロー

※1）データベースを複数使用する場合は後述の「複数DB接続の検討」を参照してください。

※2）2フェーズコミットを行う場合は、データベースのバージョンが条件を満たしている必要があります。

※3)参照にのみ使用するデータソースの JTA 連携は false で設定をし、JTA 連携を行わないようにします。

- データベースサーバと接続するための文字列

  データベースサーバと接続するための文字列はデータソースの種別ごとに異なります。

データソースの種別	データベースサーバと接続するための文字列	例
JDBC	JDBC データベース URL jdbc:oracle:thin:@<ホスト名>:<リスナのポート番号>:<Oracle SID> Oracle の SID の代わりに Oracle Net Service のアドレス記述を行うこともできます。詳細は、Oracle のリファレンスを参照してください。	jdbc:oracle:thin:@host:1521:orcl
JDBCEX_Oracle	JDBC データベース URL	jdbc:oracle:thin:@host:1521:orcl
JDBCEX_MsSqlServer	MsSqlServer のデータベース名	“MsSql001”
JDBCEX_PostgreSQL	PostgreSQL のデータベース名	“PostgreSQL001”
JDBCEX_Derby	Derby のデータベース名	“Derby001”

- JDBC 仕様のメジャーバージョン（対応する JDK のバージョン）

  JDBC ドライバのメジャーバージョンを指定します。

  JDBC2.0、3.0 を使用する必要がないのであれば、4 に設定してください。

JDBC	JDK	設定例
JDBC2.0	JDK1.2 からサポート	2
JDBC3.0	JDK1.4 からサポート	3
JDBC4.0	JDK1.6 からサポート	4
JDBC4.1	JDK1.7 からサポート	4
JDBC4.2	JDK8 からサポート	4
JDBC4.3	JDK9 からサポート	4

- JDBC 仕様のマイナーバージョン

  JDBC ドライバのマイナーバージョンを指定します。

  JDBC 仕様のバージョンが 4.1 の場合に 1 を設定します。

- ポート番号

  データベースサーバのポート番号を指定します。

  データソースの種別ごとに設定可否が異なります。種別ごとのマニュアルを確認してください。

- ユーザ名

  データベースと接続するためのユーザ名を指定します。

  データベースのログインユーザ名を設定してください。

- パスワード

  データベースと接続するためのパスワードを指定します。

  データベースのログインパスワードを設定してください。できるだけわかりにくい文字列にしてください。

- JNDI サーバへの登録名

  データベースに接続する時は、JNDI サーバから javax.sql.DataSource を取得して利用します。その取得の際に指定する名前です。この名前は JDBC データソースを一意に表わす名前として使用されます。

  “jdbc/”で始まる名前で登録してください。

  登録例) jdbc/TestDataSource

- トランザクション管理の検討(JTA 連携の有無)

  JTA(Java Transaction API)との連携を行うかどうかを指定します。データベースを参照するだけであれば、JTA 連携の必要はありません。EJB コンテナでトランザクションを管理する場合や、UserTransaction を利用したトランザクション制御を行う場合に必要になります。

➢ プロセスグループ単位の設定

Web コンテナをエージェントプロセスで動作させた場合、EJB コンテナは Web コンテナとは別の Java VM 上で実行されます。また、プロセスグループが異なる EJB コンテナも別々の Java VM 上で実行されます。

ドメインに定義された JDBC データソースは既定値ではコンテナの動作している全 Java プロセス上でロードされます。(Web コンテナをエージェントプロセス上で動作させた場合、そのコンテナが動作している Java プロセスではデフォルトでロードされません)。例えば、後述する JDBC データソースの初期接続数を 1 に設定していても、Web コンテナ+ EJB コンテナのプロセスグループのプロセス数分だけ初期接続が張られることになり、そのリソースを使用していないプロセスでは無駄に接続が張られることになってしまいます。

そこで、JDBC データソースをプロセスグループ単位でロードするように設定することができます。

- リソースのプロセス単位のロード設定

  EJB コンテナが動作する全てのプロセスグループで使用するか、または各プロセスグループの属性でロードする、しないの指定を行なうかどうかを選択します。

  デフォルトすべてのプロセスグループでロードされる設定になっていますが、プロセスグループ単位でロードするように設定する場合は、無効にしてください。プロセスグループごとに使用する JDBC データソースを指定する場合は、プロセスグループの設定で、datasourceList にそのプロセスグループでロードする JDBC データソースの JNDI 名を指定してください。運用管理ツールでは、リソースタブの「使用するデータソース」で、追加ボタンを押して JNDI 名を入力してください。

➢ 複数DB接続の検討

1 つのトランザクションが複数のデータベースに対してアクセスするトランザクション(分散トランザクション)は、通常の設定では原子性を保証できません。以下に説明するような対応が必要になります。

ただし、1 つのトランザクションから複数のデータベースに接続するが、データベース更新の対象が1つだけの場合は検討不要です。

① データソースごとに別のトランザクションとして分割する

トランザクションをデータソースごとに複数の EJB に分割し、EJB 毎に異なる接続先データベースの設定を行い、各 EJB を呼び出すようにします。更新処理の同期は取れませんが、2 フェーズコミット用のプ

ロトコル（XA プロトコル）を使用しないため、処理速度は 2 フェーズコミットよりも向上することが期待されます。

② 2フェーズコミットを使用する

2フェーズコミットとは、複数のデータベースの内容を更新するトランザクション処理において、処理が矛盾しないよう整合性(ACID)を保証する手法のことです。2フェーズコミットを行う場合は RCS サーバを利用して、異常時に原子性が保たれるようにします。更新処理の同期は取れますが、AP サーバと DB サーバの間で最低4回の交信を行う必要があるため、無視できない長さの通信オーバーヘッドが生じてしまいます。そのため、システム全体の処理速度が遅くなります。

➢ カスタマイズテンプレートの採用検討

JDBCデータソースの複数の属性を一括してカスタマイズするためのカスタマイズテンプレートを選択することができます。テンプレートには、性能重視の Performance と信頼性重視の Reliability、性能と信頼性の両方を重視する PerformanceAndReliability があります。

カスタマイズの内容はそれぞれ次の通りです。最小および最大プールサイズの 20 は、Web コンテナの動作スレッド数のデフォルト値です。初期プールサイズは、JavaEE のプロセスグループの動作スレッド数の初期値プラスアルファです。実際の動作スレッド数に応じて調整してください。また、接続リトライ間隔とログインタイムアウトは、EJB のリクエストタイムアウトのデフォルト値 30 秒以上との想定でカスタマイズします。これに対し、クエリータイムアウトやソケットの読み取りタイムアウト時間は、リクエストタイムアウト時間を多少長くしても、調整不要な時間にカスタマイズします。

属性	Performance	Reliability
初期プールサイズ	5	—
最小プールサイズ	20	—
最大プールサイズ	20	—
コネクション解放までの待ち合わせ時間	60	—
空きコネクション取得時の待ち合わせ時間	20	—
最大ステートメントサイズ	20	—
接続リトライ回数	—	1
接続リトライ間隔	—	10
データベースサーバの監視オプション	—	monitor
データベースサーバの監視間隔	—	60
データベースサーバの監視コマンド	—	connect
クエリータイムアウト	—	300
ソケットの読み取りタイムアウト	—	315
ログインタイムアウト	—	15

これらの設定を基本にして、その他の設計の結果から、必要なものだけを調整し直してください。

2） コネクション管理

WebOTX では、ドメイン起動時、またはプロセスグループ起動時にデータベースとコネクションを接続し、トランザクションにかかるコストを低減することができます。

DB サーバとコネクションを接続する際、プロセス数×スレッドの数だけコネクションが作成される可能性があります。そのため、コネクションプールに対し、最大コネクションプール数を設定することで、DB サーバのライセンス数を考慮すること、また過度なメモリの使用を抑制することができます。

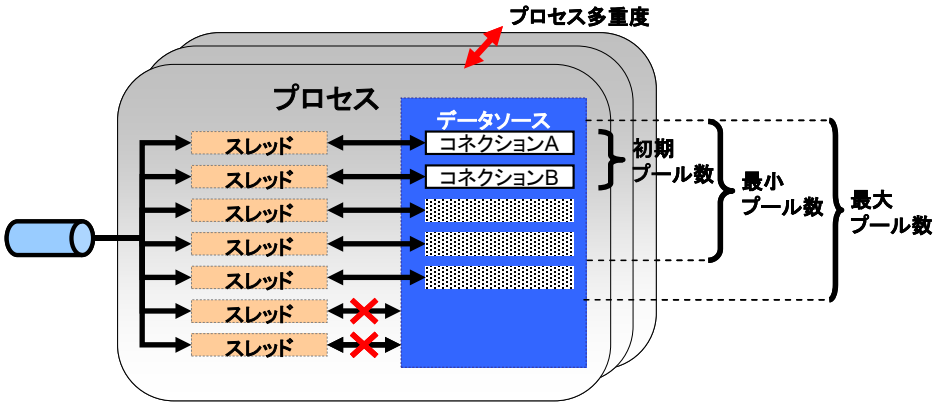


図 2.6.3.スレッドとコネクションの関係

例えば、図のようなプロセス構成の場合、実行スレッド7に対して、初期コネクションプール数を2、最小コネクションプール数を4、最大コネクションプール数を5と設定している場合、全スレッドから同時にコネクション要求があると、最大で5本分の接続要求が成功し、2本分の接続要求が失敗します。

マルチプロセス構成とした場合、コネクションの作成数は最大コネクションプール数×プロセス数で決定されます。

したがって、図のようなプロセス数が3の場合は

最大コネクションプール数(5) × プロセス数(3) = １５

APサーバとDBサーバの間で、最大で15本のコネクションが作成される可能性があります。

- 初期プールサイズ

  ドメイン起動時、またはプロセスグループ起動時に作成されるコネクション数を指定します。

  0 を設定した場合は、アプリケーションがコネクション取得 API を呼び出した際に、コネクションプール内に未使用のコネクションがない場合に、コネクション接続が行われます。

  初期接続の接続リトライ有無(デフォルト有効)を有効にしていると、データベースの復旧が確認された時点で、初期接続の復旧（再接続）が行われます。初期接続は、再接続処理で、1本以上の接続に成功すると完了とみなされます。

  パフォーマンスに問題がないようであれば、最大プールサイズと同等数を指定します。

- 最小プールサイズ

  プールに常時保持されるコネクション数を指定します。

  最小プールサイズ数を越えて接続されたコネクションは、コネクション解放までの待機時間経過後に解放されます。パフォーマンスに問題がないようであれば、最大プールサイズと同等数を指定します。

- 最大プールサイズ

  プールに保持される最大のコネクション数を指定します。

  0 を設定した場合は、制限はありません。それ以外は最小プールサイズ以上の値を設定してください。

  通常、データベースにアクセスするスレッド数と同じ値を指定します。データベースサーバの状態監視で、定期監視を行う場合は、スレッド数+1 を指定してください。データベース側で、接続数の上限が規定されている場合には、上限を考慮して値を決定してください。最大プールサイズにデータベースにアクセスするスレッド数より小さい値を指定した場合に、コネクションが空くまで待ち合わせることができます。

プールサイズは以下の関係で設定してください。

初期プールサイズ <= 最小プールサイズ <= 最大プールサイズ

- コネクション解放までの待機時間

  最少プールサイズを越えて払い出されたコネクションを解放するまでの待ち時間(単位:秒)を指定します。

  コネクションは指定された時間が経過した後でクローズした時、またはトランザクションが完了した時に解放されます。

  最少プールサイズと最大プールサイズが同数で無い場合、DB アクセス頻度を考慮して設定します。

- コネクションの一括破棄可否

  データベース異常発生時にプールの全コネクションを一括破棄するかどうかを指定します。

  DB サーバ状態監視異常時、あるいは業務で使用中のコネクションで障害となった場合にコネクションの一括破棄を行います。

  EJB コンテナのトランザクション制御下においては、データベースに対してコミット等のトランザクション制御の要求を発行した際に障害が検出されます。その際、JDBC ドライバから「致命的なエラー」が通知された場合にコネクションの一括破棄が行われます。

  一般的に、コネクションの一括破棄可否は有効にしてください。

3) 状態監視

WebOTX ではコネクションをプールしておきますが、データベースに異常が発生した場合、プール済みの異常なコネクションをアプリケーションに返却することを防ぐため、状態監視を行い、異常を検知した不要なコネクションを破棄することができます。また、データベースの復旧後、初期接続の復旧(再接続)が行われます。

- データベース正常時

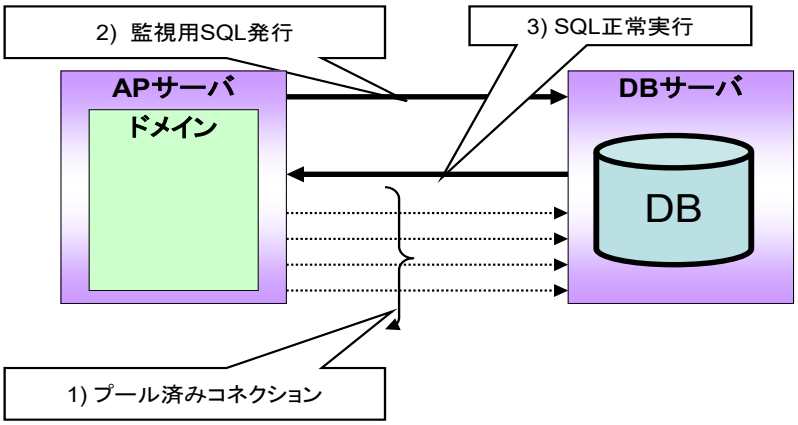


図 2.6.4.状態監視(データベース正常時)

データベースが正常動作時のデータベース死活監視の動作は上記のようなイメージになります。

1. 業務アプリケーションでデータベースを使用しているため、すでにコネクションがプールされています。
2. 設定した監視間隔ごとに監視用の SQL が発行されます
3. 発行した SQL が正常に実行されます

● データベース異常時

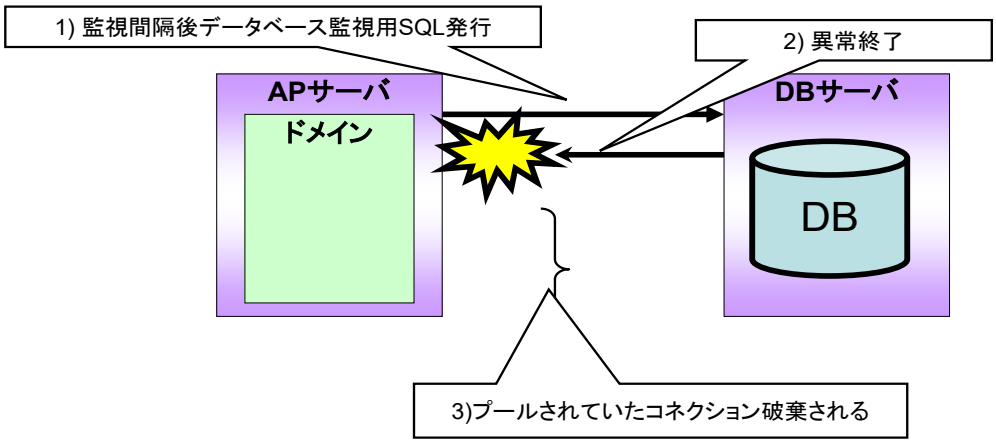


図 2.6.5.状態監視(データベース異常時)

データベースが異常状態時のデータベース死活監視の動作は上記のようなイメージになります。

1. データベース監視用 SQL が発行されます
2. 発行した SQL が正常に実行されずに、エラーを返却
3. プールされていたコネクションが破棄されます
4. 設定した監視間隔で監視用 SQL の発行を行います

補足
監視に失敗した場合に加えて、EJB コンテナのトランザクション制御下においてデータベースに対してコミット等のトランザクション制御の要求を発行した際に障害が検出された場合や、JDBC ドライバから「致命的なエラー」が通知された場合にコネクションの一括破棄が行われます。

● データベース復旧時

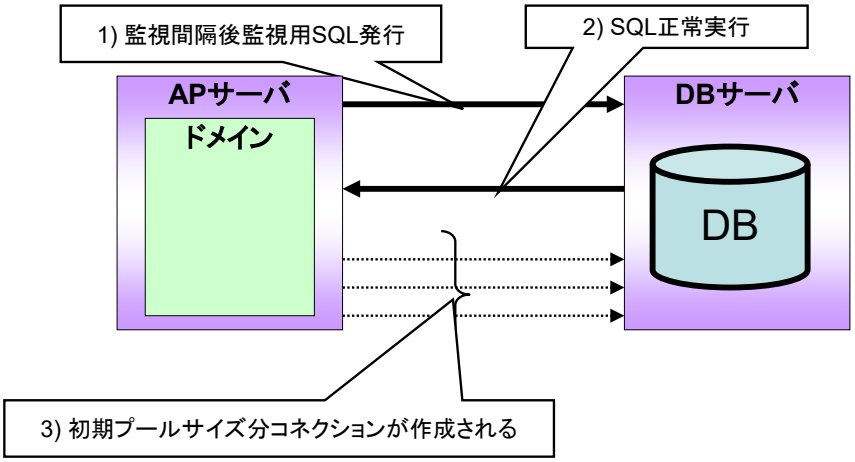


図 2.6.6.状態監視(データベース復旧時)

データベースが異常状態から復旧したときのデータベース死活監視の動作は上記のようなイメージになります。

1. データベース監視用 SQL が発行されます
2. データベースが復旧しているため、SQL が正常に実行されます
3. JDBC データソースの初期プールサイズ分のコネクションが作成されます

- データベースサーバの状態監視コマンド

  負荷が少なく、ロック競合が発生しない、かつ失敗の可能性がすくないクエリを指定することを推奨します。ただし、高速に障害を検出しなければならない場合は、本コマンドに”connect”を指定し、ログインタイムアウト(loginTimeout)で障害検出までの時間を指定してください。また、接続および切断が頻繁に繰り返されないよう

に、データベースの状態監視オプションに”monitor”を指定して定期監視を行ってください。加えて、コネクションの一括破棄可否を有効にしてください。

- データベースサーバの状態監視間隔

  SQL 発行や新規接続時の負荷を考慮して決定してください。

- データベースサーバの状態監視オプション

  データベースサーバの状態監視のタイミングは 3 通り(「なし」、「定期的」、「コネクションを払い出すたび」)存在しますが、負荷を極力抑えるため、定期的に監視することを推奨します。
  monitor ： 定期的にデータベースサーバの状態を確認します。
  method ：JDBC コネクションを払い出す度に、データベースサーバの状態を確認します。
  none ： この機能を無効にします。

- 接続リトライ回数

  データベースへの接続失敗時のリトライ回数を指定します。要件にあわせ設定してください。

- 接続リトライ間隔

  データベースへの接続失敗時にリトライを行う間隔を指定します。

2.6.3.コネクションの状態遷移

以下に、コネクションが生成、使用、破棄、復旧される一連の流れを記載いたします。

- ドメイン起動時、プロセスグループ起動時

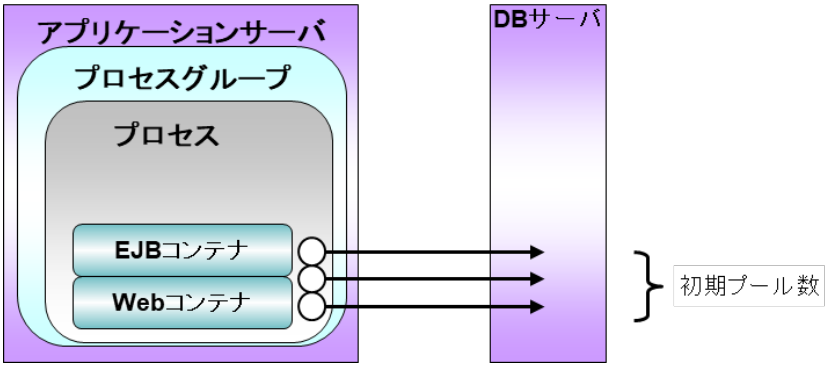


図 2.6.7.ドメイン起動時、またはプロセスグループ起動時

ドメイン起動時、またはプロセスグループ起動時に初期プールサイズに設定した本数分のコネクションを作成します。

コネクションが作成できなかった時を考慮して、初期接続の接続リトライ有無を有効にしておくと、初期接続の接続リトライが行われます。データベースサーバの状態監視間隔で指定した間隔で、初期接続の復旧が成功するまでリトライが行われます。

● コネクション要求時

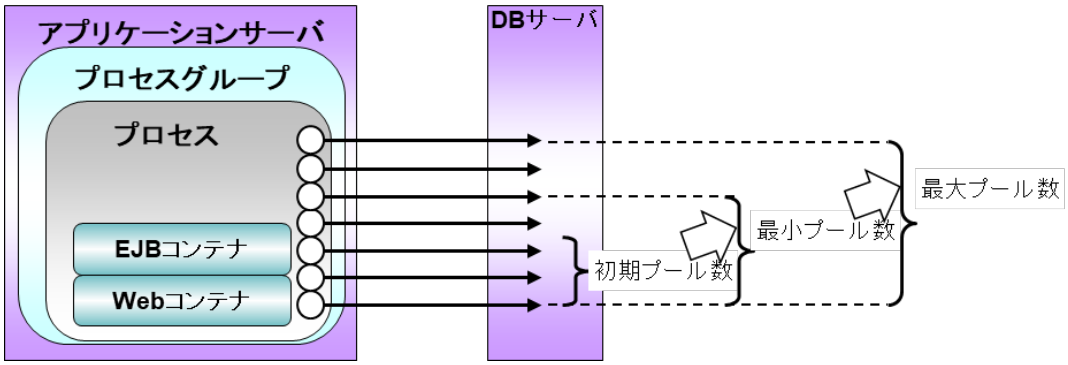


図 2.6.8.コネクション要求時

データベースにアクセスするプロセス内で使用中のコネクションが初期プールサイズ以上の場合、最大プールサイズまで作成します。最大プールサイズ以上のコネクション要求があった場合は、空きコネクション取得時の待ち合わせ時間に応じて、コネクションが空くのを待ち合わせます。待ち合わせ時間が 0 の場合は、リトライを行わず、すぐにエラーを返します。

また、接続リトライ間隔と接続リトライ回数を設定している場合、コネクション取得に失敗した場合にリトライを行います。

● 処理完了時

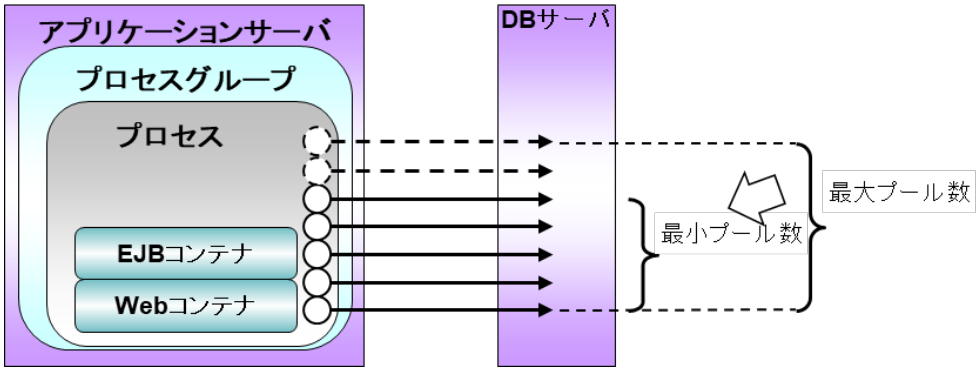


図 2.6.9.プロセス終了時

データベースアクセス処理終了後、プールにデータベースコネクションを返却します。
プールされているコネクション数が最小プールサイズ以上の場合、コネクション解放までの待機時間経過後、使用されていないコネクションを最小プールサイズになるようにコネクションが解放されます。

● データベース異常時(状態監視なし)

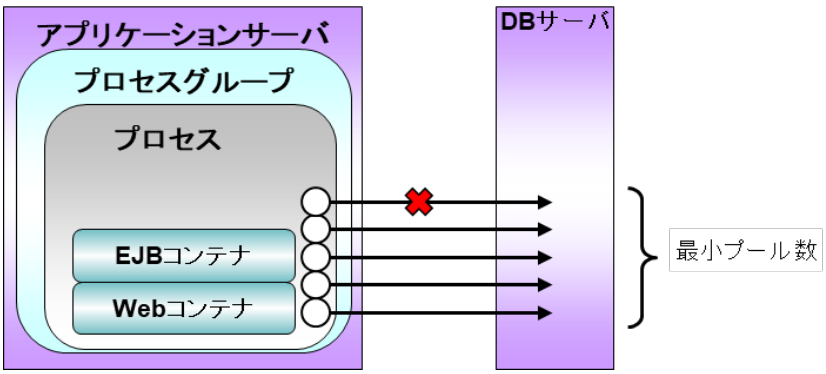


図 2.6.10.データベース異常時(状態監視なし)

データベース異常は、EJB コンテナのトランザクション制御下においてデータベースに対してコミット等のトランザクション制御の要求を発行した際に障害が検出された場合や、JDBC ドライバから「致命的なエラー」が通知された場合に検知されます。そのため、JTA 連携なしで障害を検出できない場合や、JDBC ドライバからの通知次第となる場合があります。

- データベース異常時(状態監視あり、一括破棄否)

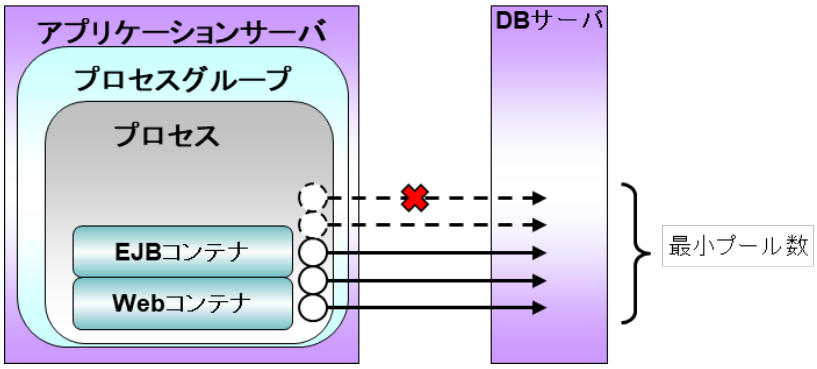


図 2.6.11.データベース異常時(状態監視あり、一括破棄否)

コネクションの一括破棄可否が無効の場合、未使用の全コネクションに対して監視を行います。監視の結果がエラーの場合、そのコネクションだけを破棄します。

- データベース異常時(状態監視あり、一括破棄可)

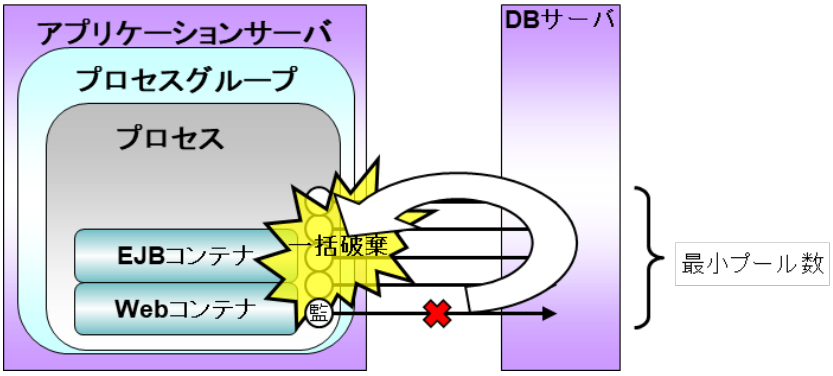


図 2.6.12.データベース異常時(状態監視あり、一括破棄可)

コネクションの一括破棄可否が有効の場合、状態監視はいずれか1つのコネクションを監視用に使用します。データベースサーバの状態監視コマンドとして”connect”が指定されている場合は、監視の都度、プール数にカウントされない新規コネクションを使用します。

監視の結果がエラーの場合、全コネクションを破棄します。

● データベース復旧時

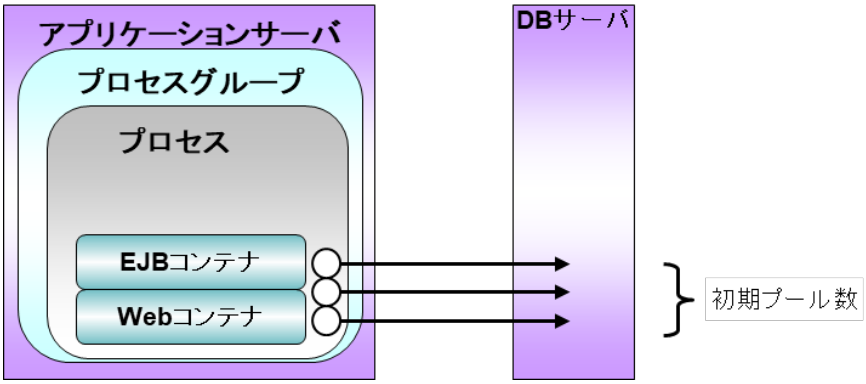


図 2.6.13.データベースの復旧時

破棄されたコネクションは状態監視により、初期接続の接続リトライ有無が有効の場合、初期プールサイズ分復旧されます。ドメイン起動時と同じく、リトライ間隔はデータベースサーバの状態監視間隔で指定した値で、初期接続が復旧するまでリトライが行われます。

初期接続の接続リトライ有無が無効の場合は、データベースアクセス時にコネクションを作成し、トランザクション終了後にコネクションをプールします。

コネクションを再接続するために AP サーバを再起動する必要はありません。

2.6.4.設定項目

DB連携処理で決定すべきパラメータ一覧です。

1） JDBCタイプの設定項目 共通

JDBCタイプの設計では、以下の内容を決定します。

設定パラメータ(属性名)	説明	既定値	推奨値
データソース種別 dataSourceType	データソースの種別を指定します。	なし	JDBC
データソース名 dataSourceName	JDBC URL またはデータベース名、データソース名を指定します。	なし	dataSourceType に依存
JDBC メジャーバージョン jdbcMajorVersion	JDBC 仕様のメジャーバージョンを指定します。	4	dataSourceType および JDK に依存
JDBC マイナーバージョン jdbcMinorVersion	JDBC 仕様のマイナーバージョンを設定します。 JDBC 仕様のバージョンが 4.1 の場合に 1 を設定します。	0	dataSourceType および JDK に依存
データベースサーバ名 serverName	データベースサーバのサーバ名を指定します。	なし	※1
データベースポート番号 portNumber	データベースサーバのポート番号を指定します。	なし	※1
データベース接続ユーザ名 userName	データベースと接続するためのユーザ名を指定します。	なし	※2
データベース接続パスワード password	データベースと接続するためのパスワードを指定します。	なし	※2
JNDI サーバへの登録名 jndiName	JNDI サーバへの登録名を指定します。	なし	※2
JTA 連携の有無 useJTA	JTA と連携するかどうかを指定します。	true	システム要件にしたがってください
カスタマイズテンプレート customizeTemplate	JDBCデータソースの複数の属性を一括してカスタマイズするためのテンプレート名を指定します。性能重視か信頼性重視かによって Performance、Reliability、PerformanceAndReliability のいずれかを選択できます。	none	PerformanceAndReliability

※1：データソースタイプが JDBC の場合は指定不要です。管理ツールからは指定できません。

※2：データベースの情報により決定します

設定パラメータ(属性名)	説明	既定値	推奨値
リソースのプロセス単位のロード設定 use-all-ejb-processgroups	リソースのプロセス単位のロード設定をするかどうかを指定します。	true	true

2） コネクション管理共通

コネクション管理の設計では、以下の内容を決定します

設定パラメータ(属性名)	説明	既定値	推奨値
最小プールサイズ minPoolSize	プールに常時保持されるコネクション数を指定します。	4	maxPoolSize
最大プールサイズ maxPoolSize	プールに保持される最大のコネクション数を指定します。値が 0 の場合、無制限になります。	0	スレッド数×プロセス数＋(プロセス毎に+1)
初期プールサイズ initialPoolSize	プール生成時に作成されるコネクション数を指定します。	0	maxPoolSize
コネクションの一括破棄可否 resetAllConnectionsOnFailure	コネクションの一括破棄を行うかどうかを指定します。	true	true
コネクション解放までの待ち合わせ時間 shrinkDelayTime	最少プールサイズ以上のコネクションがある時、解放までの待機時間を指定します。	15	任意
接続リトライ回数 connectRetryMax	JDBC コネクションの取得に失敗した場合の、接続リトライ回数を指定します。	0	0
接続リトライ間隔 connectRetryInterval	JDBC コネクションの取得に失敗した場合の、接続リトライ間隔（単位：秒）を指定します。	10	10

3） 状態監視共通

状態監視の設計では、以下の内容を決定します。

設定パラメータ(属性名)	説明	既定値	推奨値
状態監視コマンド checkServerCommand	データベースサーバの状態監視コマンドとして使用する SQL 命令を指定します。	commit	commit
状態監視間隔 checkServerInterval	データベースサーバの状態監視間隔を指定します。（単位：秒）	180	任意
状態監視オプション checkServerOption	データベースサーバの状態監視契機を指定します。	none	monitor

2.7.ログ

WebOTX はプログラムの実行状況を逐一ログファイルに出力します。このログファイルは以下の目的のために使用することができます。

- サーバ・業務アプリケーションの状況確認
- 障害発生時の原因調査
- クライアントからのリクエスト情報の管理・監視

本節では、これらの目的のために有用なログファイルについて説明します。

2.7.1.概要説明

以下の点について説明します。

1） WebOTX の出力するログ
   - ➢ ログの概要
   - ➢ ログの設定項目

2） 業務アプリケーションの出力するログ

3） クライアントからのリクエスト情報の管理・監視

1） WebOTX の出力するログ

➢ ログの概要

WebOTX は運用領域をドメインと呼ばれる単位でグルーピングして管理しています。ドメインは各種機能を提供するサービス群から構成されています。Standard では、各サービスは以下の 2 種類のサービスに分類することができます。

- エージェントプロセス内のモジュールとその制御モジュールから構成されるサービス
- エージェントプロセス外部のプロセス群とエージェントプロセス内の制御モジュールから構成されるサービス

各サービスの状況は、エージェントプロセスにて出力される以下のログを閲覧することで、確認することができます。

ログファイル名	ディレクトリ	説明
server.log	<ドメイン名>/logs	エージェントのJava VM上の標準出力(System.out)、標準エラー出力(System.err)に出力された内容を保持します。
agent.log	<ドメイン名>/logs	エージェントプロセス内で動作するコンポーネントのログを出力します。

ただし、サービスによっては agent.log とは別のファイルに詳細なログを出力している場合があります。

ログファイル名	ディレクトリ	説明
jdbc.log	<ドメイン名>/logs/jdbc	JDBCデータソースに関するログ。
jta.log	<ドメイン名>/logs/TS	JTAに関するログ
objava.log	<ドメイン名>/logs/ ObjectBroker	WebOTX Object Broker Java ライブラリに関するログ。
jms.log	<ドメイン名>/logs/jmq	JMSリソースアダプタ、及びクライアントに関するログ。

➢ ログの設定項目

各サービスのログは、設定項目として以下の項目を持ちます。

- ローテーション方式
  ログファイルサイズに指定した最大サイズでのローテーション、もしくは指定した時間間隔でのローテーションを選択します。
- ログファイルサイズ
  ローテーション方式にログファイルサイズでのローテーションを選択した場合に、出力を行うログの最大サイズを設定します。
- ローテーション間隔
  ローテーション方式に時間間隔でのローテーションを選択した場合に、その時間を設定します。
- 世代数
  ログファイルの世代数を設定します。
- ログレベル
  ログの出力を行うレベルを設定します。設定できるログレベルは OFF, ERROR, WARN, INFO,CONFIG, DEBUG, DETAIL, TRACE となっています。

閾値	説明
OFF	全ての出力を行いません
ERROR	システムを継続運用するのに支障をきたす障害以上を出力します。
WARN	システムを継続運用することはできるレベルの障害以上を出力します。
INFO	SLOGINFO(*)以外の一般的な情報メッセージ以上を出力します。
CONFIG	構成の追加や変更などコンフィグレーションに関するメッセージ以上を出力します。
DEBUG	デバックレベルのメッセージ以上を出力します。
DETAIL	詳細なデバックレベルメッセージ(WebOTX オリジナルのレベル)以上を出力します。
TRACE	詳細なデバックレベルメッセージ以上を出力します。

(*) SLOGINFO は WebOTX 独自のログレベルで設定することはできません。このレベル以上のものを OS のシステムログに出力します。

ログの設定方法については WebOTX マニュアル「構築・運用 - ログ」を参照してください。

2) 業務アプリケーションの出力するログ
業務アプリケーションのログ出力には、JDK で標準提供されている JDK Logger や Log4j などのサードパーティのロギングライブラリを使用することができます。サードパーティのロギングライブラリを使用する場合は、以下のディレクトリにライブラリの jar ファイルを配置する必要があります。

- 各 WEB アプリケーションから出力する場合

  warファイルを作成時に WEB-INF/lib ディレクトリにライブラリの jar ファイルを配置します。詳細については、WebOTX マニュアル「構築・運用 - ログ - 4. 業務アプリケーションのログ」を参照してください。

- EJB アプリケーションから出力する場合
  <ドメイン名>/lib ディレクトリにライブラリの jar ファイルを配置します。jar ファイルの配置は次の手順で行ってください。

  1. ドメインの停止
  2. <ドメイン名>/lib へのファイル格納
  3. ドメインの起動

  実行方法については WebOTX マニュアル「構築・運用 - ログ - 4. 業務アプリケーションのログ」を参照してください。

また、業務アプリケーションで標準出力と標準エラー出力を利用した場合、以下のファイルに出力されます。

アプリケーションの配備先	出力先
エージェントプロセス	<ドメイン名>/logs/server.log
プロセスグループ	<ドメイン名>/logs/tpsystem/<アプリケーショングループ名>/<プロセスグループ名>/<プロ

	セスグループ名>.<プロセスID>.log

3） クライアントからのリクエスト情報の管理・監視
リクエスト情報の管理・監視を行う場合、WebOTX Web サーバが以下のログファイルに出力するログを確認してください。

ログファイル名	ディレクトリ	説明
access.log (Windows) access_log (Linux)	<ドメイン名>/logs/web	ブラウザのアクセス情報を出力するアクセスログ。1 行で1リクエストに対応。
ssl_request.log (Windows) ssl_request_log (Linux)	<ドメイン名>/logs/web	HTTPS 通信時のブラウザのアクセス情報と、SSL プロトコルのバージョンと暗号化アルゴリズムの情報を出力するアクセスログ。1 行で1リクエストに対応。

2.7.2.設計指針

ログの設計指針では、以下の項目について注意する必要があります。

1） ログのローテート機能

2） TP モニタのトレースログ

1） ログのローテート機能
ログファイルは時間が経過するにつれ、容量が肥大化し、ディスクを圧迫します。ログファイルの肥大化を抑えるために、古くなり不要となったログは削除する必要があります。WebOTX では、ファイルサイズによるログローテート機能、および指定した時間間隔でのログローテート機能を提供しています。
ローテート機能の設定に加えて、システム要件におけるログの保存期間に合うよう、保存しておくログファイルの世代数を設定してください。

2） TP モニタのトレースログ
WebOTX では、プロセスグループのサーバアプリケーションのトレースファイルを、以下のディレクトリに退避しています。退避のタイミングは、プロセスまたはプロセスグループの終了時となります。

<ドメイン名>/logs/tpsystem/<アプリケーショングループ名>/<プロセスグループ名>/save

この save ディレクトリに保存されたトレースログは既定値 30 日で自動で削除されます。

プロセスグループのトレース設定でローテーションやバックアップ方式の指定が可能です。詳細は、WebOTX マニュアル「構築・運用 - ドメインの構築 – 4. TP システム – 4.3. 操作・状態確認(プロセスグループ) – 4.3.3. 設定値の説明」を参照してください。

2.8.監視

WebOTX は、複数のプロセスによりサービスを提供しています。また、そのサービスの動作状況をログとして出力しています。それらのプロセスや、ログの状況を監視することで、障害発生を検知し、早期に業務を復旧させることができます。

ここでは、以下の項目について説明を行ないます。

- 監視すべきプロセス
- ログ内で監視すべきメッセージ

2.8.1.概要説明

1） 監視すべきプロセス 共通 プロセスグループ

- WebOTX Jakarta EE サーバで動作するプロセス
- TP モニタで動作するプロセス

2） ログ内で監視すべきメッセージ 共通

1） 監視すべきプロセス 共通 プロセスグループ

WebOTX のプロセス群は、Jakarta EE サーバとして動作しているものと、TP モニタ上で動作しているものに分けられます。
TP モニタとは実行中の業務アプリケーション等を監視し、実際のそれらの運用を行うモジュールです。

ここでは、障害発生時に業務アプリケーションの実行に影響を与えるプロセスについて、Jakarta EE で動作するものと、TP モニタにて動作するものを説明します。

- WebOTX Jakarta EE サーバで動作するプロセス

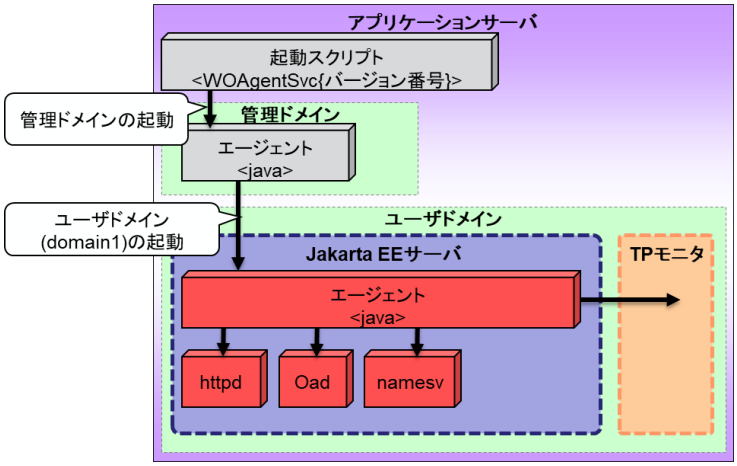


図 2.8.1.WebOTX アプリケーションサーバ内 監視プロセス

共通

図 2.8.1.	プロセス名	説明
httpd	httpd.exe (Windows) httpd (Linux)	エージェントプロセスに管理されている HTTP サーバのデーモンです。インストール時に WebOTX Web サーバを選択した場合に、Web サーバを起動すると動作します。 親プロセス(監視プロセス)と子プロセス(HTTP サービスデーモン)が存在しており、子プロセスに異常が発生したとき、親プロセスは子プロセスの再起動を行います。

		そのため、親プロセスの監視が必要となります。親プロセスの判別はプロセス ID(WebOTX/domains/<ドメイン名>/logs/web/httpd.pid)を参照してください。
oad	oad.exe (Windows) oad (Linux)	エージェントプロセスに管理されている CORBA オブジェクト活性化デーモンです。Object Broker を起動すると動作します。 異常終了時は新たな IIOP 通信(RMI/IIOP)が行えなくなります。 デフォルトでは、自動起動しません(V9.2 より)。Object Broker の irsv/corbaloc/cnamesv/oadj や、Transaction サービスの C++ AP 用 RCS プロセスを利用する場合に必要となるプロセスです。
namesv	namesv.exe (Windows) namesv (Linux)	エージェントプロセスに管理されているCORBA 名前サーバデーモンです。Object Broker を起動すると動作します。 異常終了時はオブジェクトの取得が行えなくなり IIOP 通信ができなくなります。
domain1	javaw.exe (Windows) java (Linux)	エージェントプロセスの Java VM です。異常終了した場合、該当ドメインにアクセスができなくなります。 エージェントプロセスに業務アプリケーションを配備していた場合、異常終了すると業務アプリケーションにアクセスできなくなります。 funcid=agent domain.name=<ドメイン名>を用いて、該当プロセスの判別を行ってください

※ 業務アプリケーションはユーザドメインで動作するため、管理ドメイン(admin)は監視不要です。

➢ TP モニタで動作するプロセス

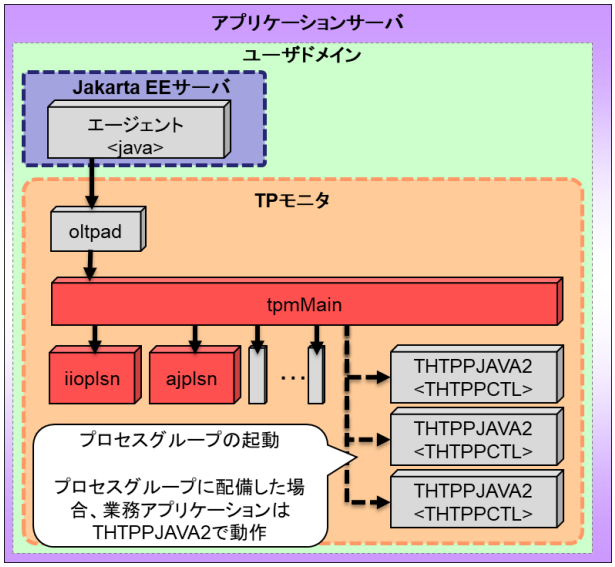


図 2.8.2.TP モニタ内監視プロセス

プロセスグループ

図 2.8.2.	プロセス名	説明
tpmMain	tpmMain.exe (Windows) tpmMain (Linux)	アプリケーションプロセスの障害検出や異常終了時の自動復旧など、WebOTX の高信頼性を実現させているプロセスです。 異常終了した場合、TP モニタの機能が利用できなくなるため、プロセス監視を行なう場合は、監視対象にしてください。

iioplsn	iioplsn.exe （Windows） iioplsn （Linux）	TPモニタ上で動作するEJB、CORBAアプリケーションを呼び出すクライアントとの接続切断や、電文の送受信を行っているプロセスです。 異常終了時は IIOP リスナとの通信が不可となります。全てのクライアントからのアクセスができなくなります。
ajplsn	ajplsn.exe (Windows) ajplsn (Linux)	TPモニタ上で動作するWebアプリケーションを呼び出すクライアント(Webサーバ)との接続切断や、電文の送受信を行っているプロセスです。 異常終了時はAJPリスナとの通信が不可となります。全てのクライアントからのアクセスができなくなります。

2） ログ内で監視すべきメッセージ共通
監視すべき対象としては、イベントログ、システムログが上げられます。イベントログ、システムログには以下のレベルのメッセージが出力されます。、

- エージェントプロセス

  エージェントプロセスは、ログレベルにおける SLOGINFO 以上(SLOGINFO、WARN、ERROR)のものをイベントログ、システムログに出力しています。

ERROR	システムを継続運用するのに支障をきたす障害を出力します。
WARN	システムを継続運用することはできるレベルの障害を出力します。
SLOGINFO(*)	サービスの起動・停止などイベントログとして表示すべき運用状況を示す情報以上を出力します。

(*) SLOGINFO は WebOTX 独自のログレベルで設定することはできません。このレベル以上のものを OS のシステムログに出力します。

- Transaction Service

  Transaction Service を使用した場合、トレースレベル1、トレースレベル2のものをイベントログ、システムログに対して出力しています。

トレースレベル 1	[障害]レベルを表示します
トレースレベル 2	[警告]レベルを表示します。

これらのレベルで出力される情報を監視してください。

2.8.2.設計指針

監視設計をする上で必要となる指針を記載します。

1） プロセス障害発生時の対処法共通

2） ログ内で監視すべきメッセージ共通

1） プロセスの障害発生時の対処法共通
プロセス ID、プロセス名を用いて、対象プロセスを選択し監視を行ってください。また、Linux の場合、プロセスのコマンドラインを取得できるためコマンドの情報を用いて監視することが可能です。
監視対象プロセスは以下となります。

プロセス名	コマンドラインオプション
httpd.exe (Windows) httpd(Linux)	-f /opt/WebOTX/domains/${domain.name}/config/WebServer/httpd.conf
oad.exe(Windows) oad(Linux)	-WOdomain /opt/WebOTX/domains/${domain.name}
namesv.exe(Windows) namesv(Linux)	-WOdomain /opt/WebOTX/domains/${domain.name}

javaw.exe(Windows) java(Linux)	funcid=agent domain.name=${domain.name}
tpmMain.exe（Windows） tpmMain（Linux）	-n${domain.name}
iioplsn.exe（Windows） iioplsn（Linux）	-sg_fname /opt/WebOTX/domains/${domain.name}/config/tpsystem/iiop.sg
ajplsn.exe(Windows) ajplsn(Linux)	-sg_fname /opt/WebOTX/domains/${domain.name}/config/tpsystem/ajp.sg

2） ログ内で監視すべきメッセージ共通

イベントログ、システムログに出力されるメッセージに対し、監視を行ってください。

2.9.配備

配備とは、アプリケーションを WebOTX の管理下に置く作業のことです。アプリケーションを配備することで、クライアントはサーバ上のアプリケーションにアクセスできるようになります。また、アプリケーション間で共通に利用するライブラリがある場合、適切にライブラリを配置する必要があります。

本節では、配備の仕組み、クラスローダの構造、ライブラリの配置について説明します。

2.9.1.概要説明

ここでは、主に以下の項目に対し、説明を行います。

1） 配備 共通 プロセスグループ エージェントプロセス

- 配備とは
- 配備先

2） クラスローダ(ライブラリ配置) 共通

1） 配備 共通 プロセスグループ エージェントプロセス

共通

- 配備とは

  配備とは、アプリケーションを WebOTX の管理下に置く作業のことです。アプリケーションの配備を行うことで、クライアントがサーバ上のアプリケーションを使用することができるようになります。WebOTX ではアプリケーションに対して以下のような運用操作を行うことができます。配備も以下の運用操作の 1 つです。

  - パッケージング
    開発したアプリケーションを、各アプリケーションが準拠している仕様に基づきアーカイブします。
  - 配備
    作成したアプリケーションを WebOTX の管理下におく作業です。アプリケーションを配備するためには、ドメインが起動されている必要があります。配備するアプリケーションは、あらかじめ ear や war 等の形式でパッケージングしておきます。
  - 開始と停止
    デフォルトでは配備後のアプリケーションは開始した状態にあります。開始した状態とは、配備されたアプリケーションが動作しており、クライアントからの要求を受け付けることができる状態です。また、停止した状態とは、アプリケーションが動作していない状態のことで、クライアントからの要求は受け付けることができません。停止した状態のアプリケーションも WebOTX の管理下にあります。簡単な運用操作によって、アプリケーションの開始状態と停止状態を切り替えることができます。
  - 再配備
    配備済みのアプリケーションを置換する作業を再配備と呼びます。再配備はアプリケーションの開発時に頻繁に行われる作業です。
  - 置換
    インタフェースやオペレーション単位に設定した情報を引き継いだまま、配備済みのアプリケーションを置換する作業を置換と呼びます。 置換はすでに運用されている環境において、アプリケーションの小さな修正を適用したいときに有効です。 インタフェースやオペレーションが再配備前後ですべて一致している必要があり、異なっている場合置換することができません。
  - 配備解除
    アプリケーションを WebOTX の管理下から除外する作業です。

共通

- 配備先

  アプリケーションが動作するプロセスは「エージェントプロセス」と「プロセスグループ」の 2 種類が存在します。

エージェントプロセス

- エージェントプロセス

エージェントプロセスとは、WebOTX のコアとなるプロセスのことです。配備したアプリケーションをこのプロセス上で動作させることもできます。アプリケーションを配備する際に、配備先(アプリケーショングループとプロセスグループ)を指定しない場合、アプリケーションはエージェントプロセスに配備され、エージェントプロセス上で動作します。

プロセスグループ

- プロセスグループ
  配備したアプリケーションは、WebOTX のコアとなるプロセス(エージェントプロセス)から分離された個別のプロセス上で動作させることができます。このプロセスは多重化することができ、多重化したプロセス群のことを「プロセスグループ」と呼んでいます。プロセスグループは、更に業務単位にまとめることができます。業務単位でまとめたプロセスグループ群を「アプリケーショングループ」と呼びます。プロセスグループで動作させるには、アプリケーションを配備する際に、どのアプリケーショングループおよびプロセスグループに対して配備するのかを指定する必要があります。

2) クラスローダ(ライブラリ配置) 共通

アプリケーションが外部のライブラリを利用する場合、WebOTX はそのライブラリを読み込む必要があります。WebOTX では、クラスローダというモジュールを利用して、ライブラリの読み込みを行っています。
クラスローダとは、その名のとおりロードしたクラスを管理するためのモジュールです。またクラスローダは 1 つではなく、複数のクラスローダが存在しており、親子関係のある階層構造となっています。親子関係により、どのクラスが優先して読み込まれるのかが、異なります。

2.9.2.設計指針

アプリケーションの設計の際には、クラスローダの構造について知っておくことが重要です。以下ではクラスローダについて説明します。

1) クラスローダ(ライブラリ配置) 共通

- クラスローダの構造
- クラスローダの読み込みの順番について
- 用途によるライブラリ配置場所

1) クラスローダ(ライブラリ配置) 共通

- クラスローダの構造
  クラスローダはツリー構造になっています。WebOTX では、次の図のようなツリー構造となっています。

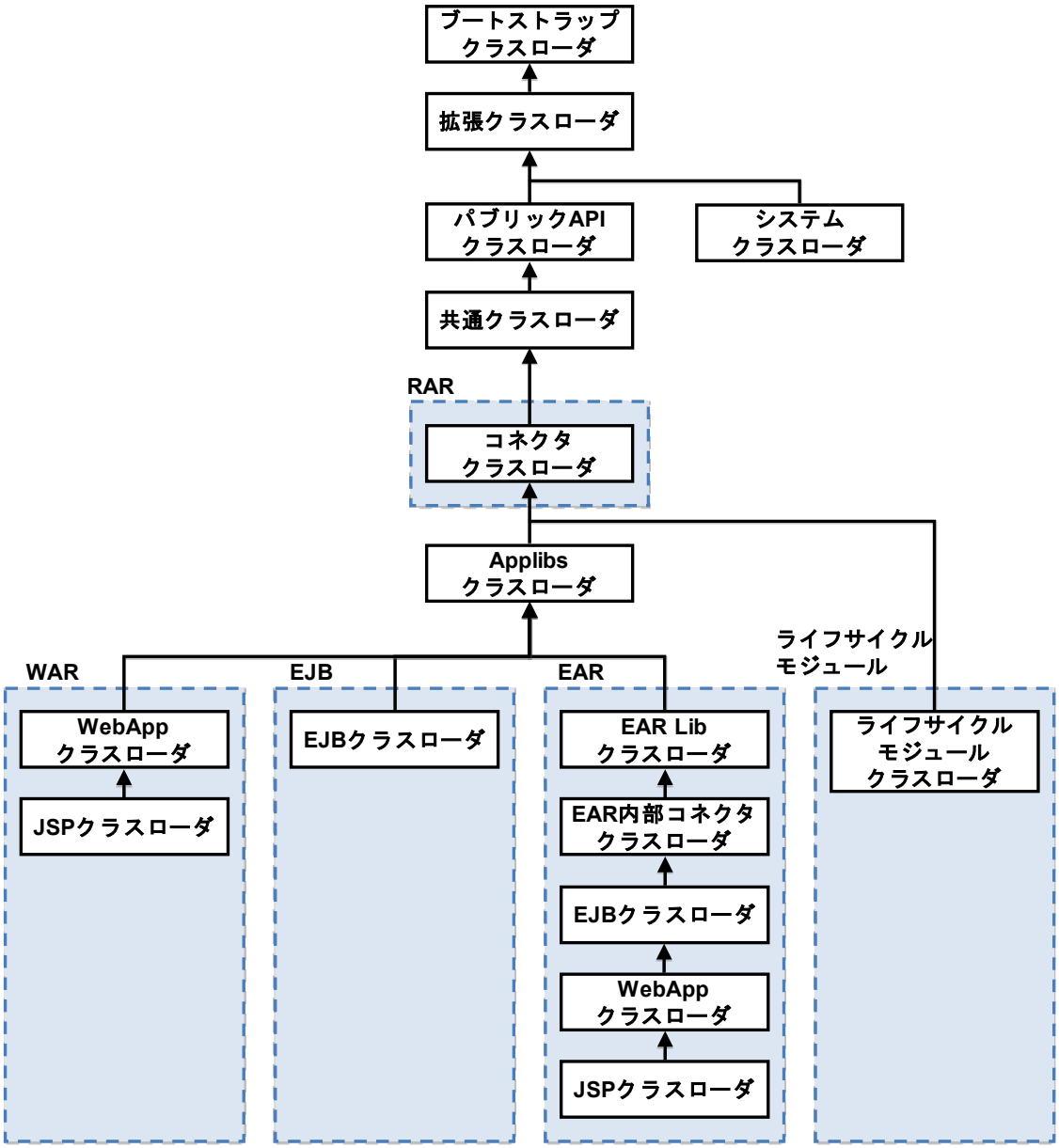


図 2.9.1.クラスローダ階層構造

クラスローダ名	説明
ブートストラップクラスローダ	Java の標準のクラスライブラリをロードするクラスローダです。
拡張クラスローダ	Java の拡張機能を読み込むクラスローダです。JDK の標準のディレクトリ下とドメインの lib/ext 下に置かれた jar または zip ファイルをロードします。 JDK 9 以降では拡張機能メカニズムが廃止されたことにより、拡張クラスローダは廃止されました。 そのため、上記の場所に Jar ライブラリを配置しても読み込まれません。
システムクラスローダ	WebOTX の起動用ライブラリ、system-classpath、CLASSPATH 環境変数で指定されたライブラリをロードするクラスローダです。システムクラスローダでロードされるクラスはアプリケーションから参照できません。 system-classpath はエージェントプロセスでのみ有効です。エージェントプロセスで CLASSPATH 環境変数を有効にするには、env-classpath-ignored を false に設定する必要があります(既定値: true)。プロセスグループの CLASSPATH 環境変数は、各プロセスグループの「環境変数」で設定します。
パブリック API クラスローダ	配備されたアプリケーション用に WebOTX が明示的にエクスポートしているクラスを使用可能にします。この中には、Jakarta EE API などが含まれます。
共通クラスローダ	ドメイン内で動作するアプリケーションで共通に利用するクラスをロードします。ドメインの lib ディレクトリに置かれた jar ファイルと lib/classes 下に置かれたクラスファイル、server-classpath で指定されたライブラリ、システムプロパティ com.nec.webotx.enterprise.common-classpath で指定されたライブラリをロードするクラスローダです。システムプロパティはエージェントプロセスと各プロセスグループごとに個別に設定できます。
コネクタクラスローダ	コネクタクラスローダは JCA に準拠したリソースアダプタを読み込むためのクラスローダです。EAR に含まれないスタンドアローンのリソースアダプタがコネクタクラスローダでロードされます。

	EAR に含まれるリソースアダプタは EAR 内部コネクタクラスローダでロードされます。リソースアダプタは他のアプリケーションから利用されるという性質上、コネクタクラスローダはドメイン内で共有されます。 class-loading-policy に derived (既定値)を指定した場合、アプリケーションからはアプリケーションで指定したリソースアダプタのみが参照可能になります(Java EE 6 以降に準拠した動作)。class-loading-policy に global を指定した場合、全てのリソースアダプタがアプリケーションから参照可能になります(Java EE 5 に準拠した動作)。
Applibs クラスローダ	アプリケーション配備時の libraries オプションで指定したライブラリをロードするクラスローダです。libraries オプションでは、ドメインの lib/applibs からの相対パスで指定します。ライブラリはドメイン内で共有されますが、アプリケーションからは配備時に指定したライブラリしか参照できません。なお、Applibs クラスローダでロードされる jar ファイルのクラスは、Applibs クラスローダでロードされる他の jar ファイルのクラスやリソースを参照することはできません。
WebApp クラスローダ	Jakarta EE アプリケーションまたは Web アプリケーションに含まれる Servlet 用のクラスをロードするクラスローダです。/WEB-INF/classes に配置したクラスファイルと/WEB-INF/lib に配置した Jar ライブラリを読み込みます。
JSP クラスローダ	コンパイルされた JSP のクラスをロードするクラスローダです。
EJB クラスローダ	Jakarta EE アプリケーションまたは EJB モジュールに含まれるクラスをロードするクラスローダです。
EAR Lib クラスローダ	EAR のライブラリディレクトリ(既定では lib)の jar ファイルをロードするクラスローダです。EAR 内の全てのモジュールから参照可能です。
EAR 内部コネクタクラスローダ	EAR に含まれるリソースアダプタをロードするクラスローダです。同じ EAR 内のモジュールからのみ参照可能です。
ライフサイクルモジュールクラスローダ	ライフサイクルモジュールをロードするクラスローダです。

ブートストラップクラスローダから Applibs クラスローダまでは、同一 Java VM 内で共通に利用するクラスローダです。破線で囲まれたクラスローダは、アプリケーションをロードするクラスローダです。WAR、EJB、EAR、ライフサイクルモジュールはクラスローダによって分離されているため、お互い干渉することなく動作します。RAR は他のアプリケーションから利用されるという性質を持ちますので、コネクタクラスローダはドメイン内で共通に利用されます。

- クラスローダの読み込みの順番について
  一般に、下位のクラスローダでロードしたクラスは上位のクラスローダでロードしたクラスを参照できますが、その逆はできません。つまり、上位のクラスローダでロードしたクラスは、下位のクラスローダでロードしたクラスを参照できず、NoClassDefFoundError または ClassNotFoundException の原因となります。アプリケーションを構築する上で、この原則に従うようにライブラリを配置する必要があります。
  Java では一般的にクラスローダはロードする際に委譲モデルを使用します。委譲モデルでは、クラスローダが個々のクラスやリソースのロード要求を受けた場合、まずは要求を親のクラスローダに委譲し、親のクラスローダが要求されたクラスやリソースを見つけられなかった場合だけ自分のリポジトリから検索するように動作します。WebOTX の提供するクラスローダも、基本的には委譲モデルで動作します。

  - delegate オプション
    Web アプリケーションでは、委譲モデルを使用するかどうかを、WAR ファイルに含まれる nec-web.xml で定義されている class-loader 要素の delegate の値で設定することができます。この設定を変更することで、クラスをロードする際の優先順位が変わります。同じ名前のライブラリがある場合、優先順位が高いほうのクラスローダにより、必要なクラスがロードされます。Servlet 2.3 以前では既定値が false、Servlet 2.4 以降では既定値が true となります。

  - delegate=true の場合
    同じ名前のクラスがある場合、Web アプリケーション側(WEB-INF/lib)よりも、WebOTX のシステムライブラリやドメインの lib ディレクトリなどに配置したライブラリを優先して読み込みます。WebOTX に含まれるライブラリと重なっている場合、delegate=true に設定することで問題を回避することができます。読み込む順番は以下のようになります。
    以下の例では、配備されたアプリケーションが、Web アプリケーション、EJB アプリケーションを含んでいる場合となります。どちらか片方のみである場合は、それぞれ必要となるクラスローダのみが利用されます。

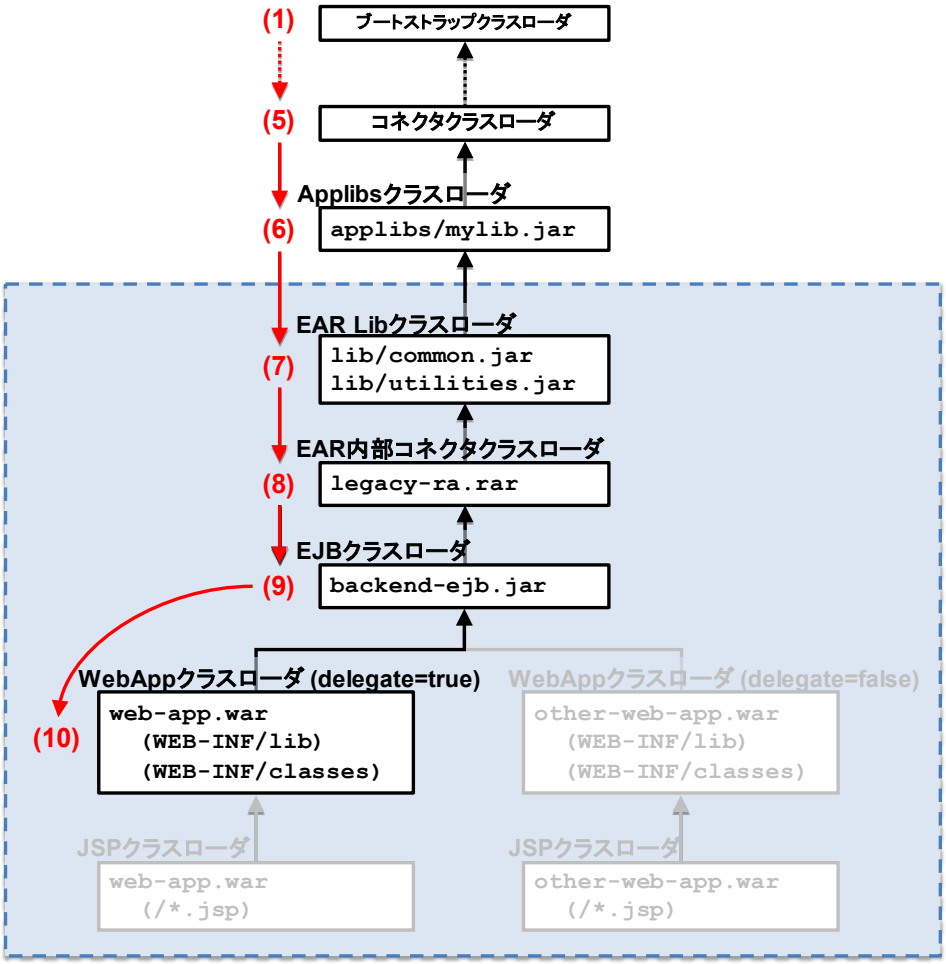


図 2.9.2.delegate=true の場合

ただし、JSP を読み込む場合は、必ず最初に JSP クラスローダが呼ばれるため、以下のような順番となります。

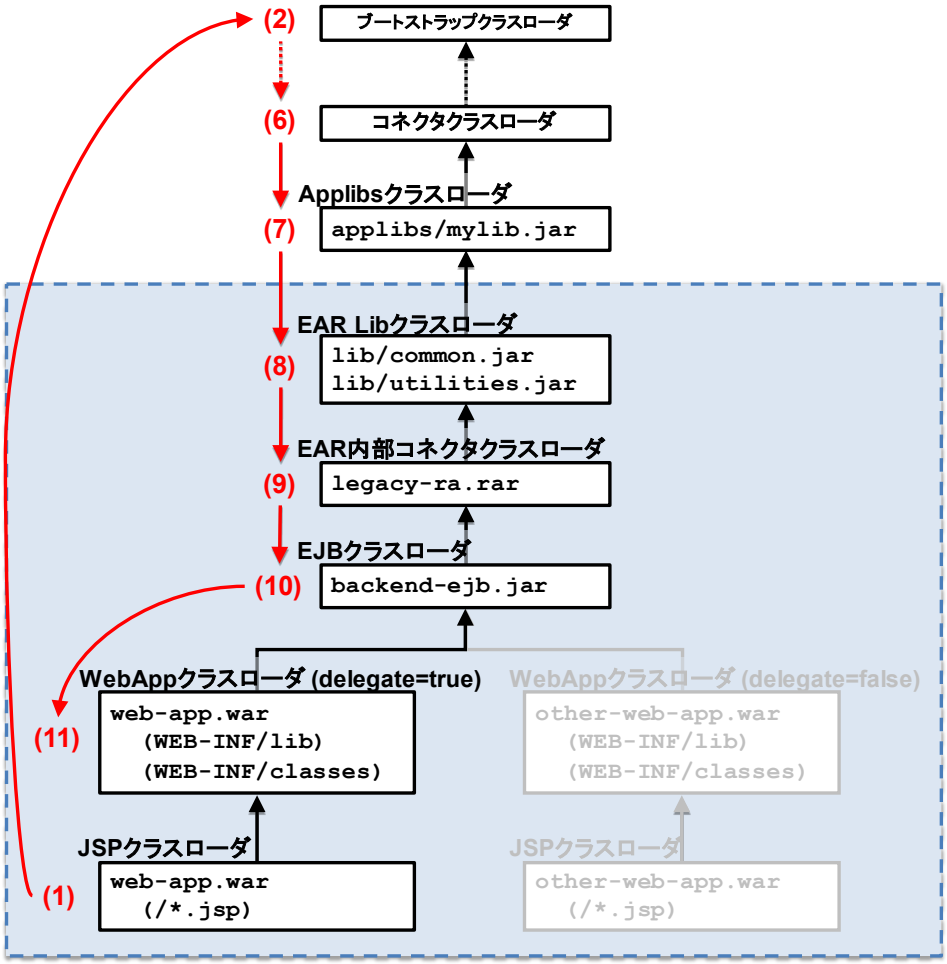


図 2.9.3 delegate=true の場合(JSP クラスローダ)

- delegate=false の場合
  同じ名前のライブラリがある場合、Web アプリケーション側(WEB-INF/lib)に配置したライブラリを優先して読み込みます。読み込む順番は以下のようになります。

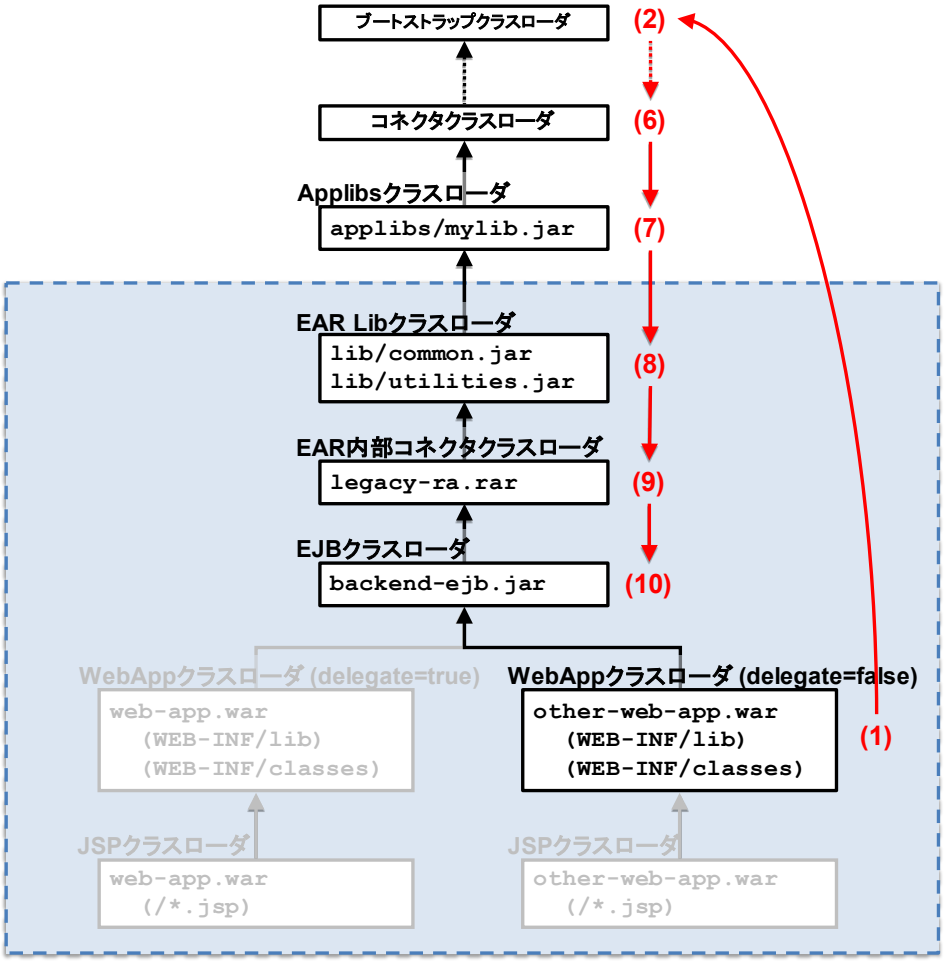


図 2.9.4.delegate=false の場合

また、JSP を読み込む場合は次の図のような順番となります。

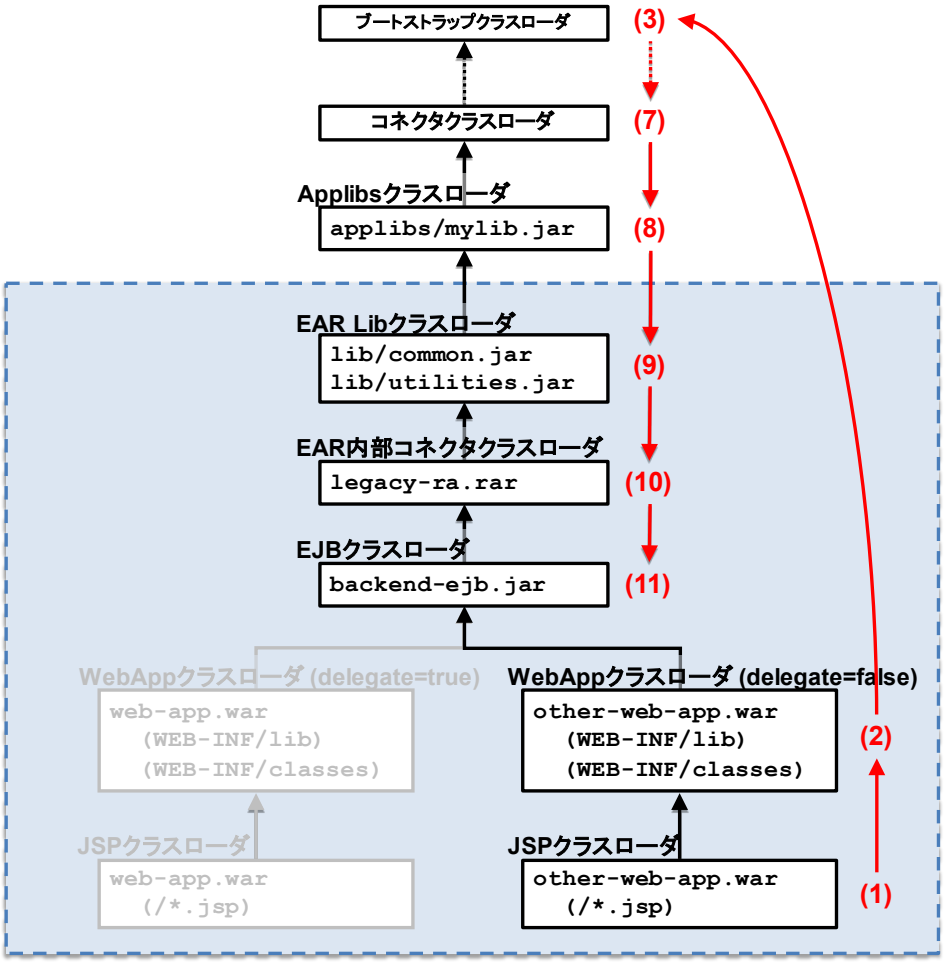


図 2.9.5.delegate=false の場合(JSP クラスローダ)

- 用途によるライブラリ配置場所
  アプリケーションにより、利用するライブラリが異なります。WebOTX では、ライブラリにより配置場所が異なります。以下の表にその違いをまとめました。

ライブラリの種類	説明	ライブラリ配置場所
アプリケーションが参照するクラス、リソース	一般的に、アプリケーションが参照するクラス、リソースがjarファイルにアーカイブされている場合はそのファイルを<ドメインディレクトリ>/lib ディレクトリに配置します。アーカイブされていない場合はそのファイルを<ドメインディレクトリ>/lib/classes ディレクトリにそのクラス、リソースのパッケージ名をディレクトリ構造に反映させた形で配置します。変更した場合はプロセスを再起動する必要があります。	<ドメイン名>/lib <ドメイン名>/lib/classes
JDBCドライバの配置	JDBCドライバは WebOTX 本体からアクセスできる必要があります。そのため、JDBCドライバは拡張クラスローダか共通クラスローダで読み込むように配置する必要があります。	<ドメイン名>/lib/ext <ドメイン名>/lib

配備の操作方法など、詳細については WebOTX マニュアルの「配備 > アプリケーション配備 > 4.4. 配備・再配備」を参照して下さい。

2.10.分散

大規模なシステムでは同一アプリケーションの動作する複数台の筐体でサービスを提供し、クライアントからのリクエスト負荷を分散させる構成をとることがあります。ここでは WebOTX を利用する際の負荷分散の方式について、説明を行います。

2.10.1.概要説明

WebOTX における負荷分散について、主に以下の項目に対し、説明を行います。

(1) WebOTX における負荷分散について

- 負荷分散のパターンと設計方針

(2) AP サーバ呼出の負荷分散機能(マルチサーバラウンドロビン負荷分散運用)

- マルチサーバラウンドロビン負荷分散の機能について
- 使用するメリット

(3) マルチサーバラウンドロビン負荷分散運用の場合の挙動

- 通常動作時の挙動
- 障害時の挙動
- 障害復旧時の挙動

(4) AP サーバ呼出の負荷分散機能(キャッシュ名前サービス)

- キャッシュ名前サービスの機能について
- 使用するメリット

(5) キャッシュ名前サービスを使用した場合の挙動

- 起動時の挙動
- 通常動作時の挙動
- 障害時の挙動
- 障害復旧時の挙動
- 片系正常停止時の挙動
- 片系正常起動時の挙動

以下、(1)～(5)についての説明をします。

(1) WebOTX における負荷分散について

- 負荷分散のパターンと設計方針

アプリケーションサーバの負荷分散構成は、一般的に以下のようなものがあげられます。

- WEB 層での負荷分散

  同一のアプリケーションを搭載したマシンを複数台並べ、ロードバランサ機器で負荷を割り振る。クライアントから受けたリクエストは、基本的にひとつの筐体で Web 層(Web サーバ+Web コンテナ)と AP 層(EJB)の処理を完結させる。

  (ex. ラウンドロビン/スティッキー/重み付けラウンドロビン)

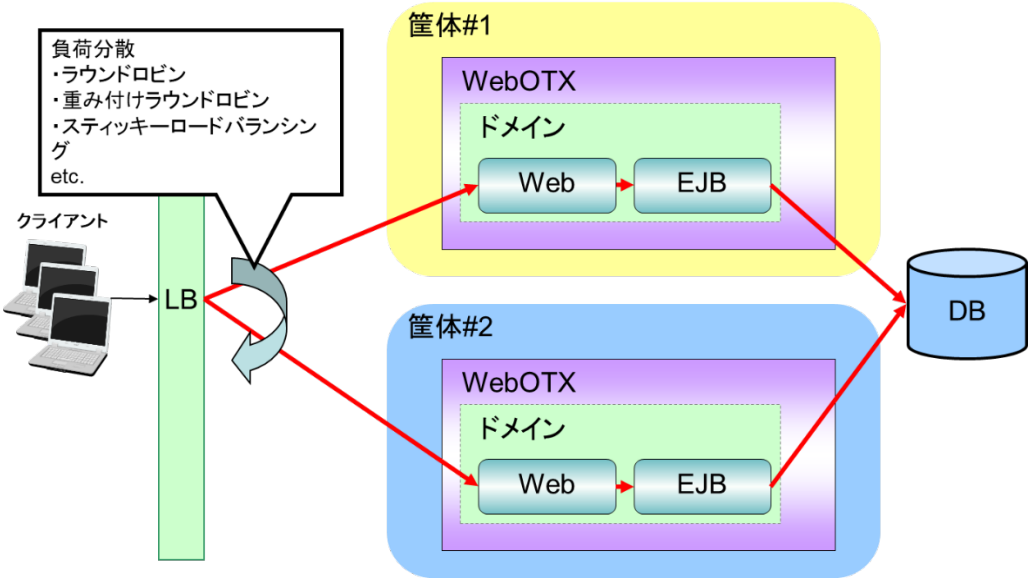


図 2.10.1 WEB 層での負荷分散

ロードバランサの負荷分散方式については、ここでは詳細を述べることはしません。ロードバランサのマニュアル等で採用すべき負荷分散方式を選択してください。

---> この構成を採用するケースについて

複数の筐体により、一台が障害発生によりダウンしても他マシンでのサービス提供続行を行いたい

WEB 層と AP 層の筐体を、セキュリティ等の理由から分ける必要が無い

- AP 層での負荷分散

  Web 層(Web サーバ+Web コンテナ)と AP 層(EJB)を別筐体に分離し、AP 層を複数台並べる構成。WEB 層のマシンが複数台存在する場合には、ロードバランサ機器で負荷を割り振る。AP 層の負荷分散はロードバランサ機器ではなく、WebOTX の負荷分散サービス(マルチサーバラウンドロビン負荷分散)を使用して負荷分散する。

  (ex. ラウンドロビン/スティッキー/重み付けラウンドロビン)

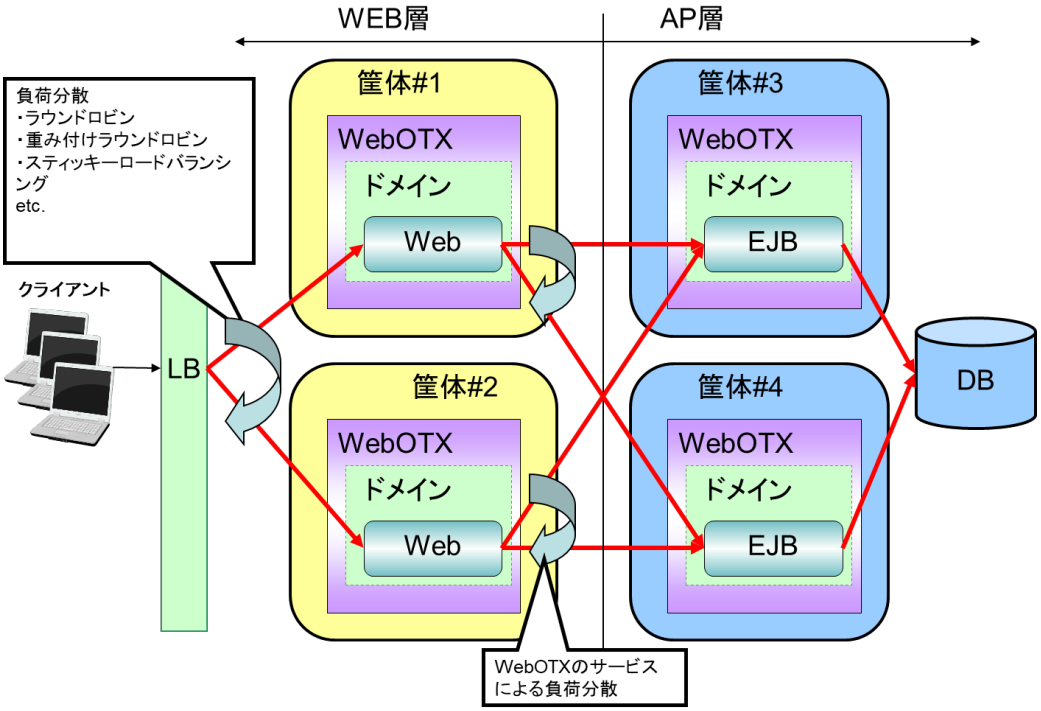


図 2.10.2. AP 層での負荷分散

---> この構成を採用するケースについて

複数の筐体により、一台が障害発生によりダウンしても他マシンでのサービス提供続行を行いたい
WEB 層と AP 層の筐体を、セキュリティ等の理由から分ける必要がある

(※ WEB 層と AP 層が同一筐体の場合でも、複数台の環境(筐体もしくは WebOTX ドメイン)が存在すれば、AP 層の負荷分散は可能です。ですが、複雑な構成となるため、推奨はしません)

(2) AP サーバ呼出の負荷分散機能(マルチサーバラウンドロビン負荷分散運用)

➢ マルチサーバラウンドロビン負荷分散の機能について

マルチサーバラウンドロビン負荷分散は以下の機能を提供します。

- 複数ドメイン間の負荷分散運用

➢ 使用するメリット

AP サーバ筐体またはドメインが複数存在し、AP 呼び出しを負荷分散させる必要がある場合に使用します。マルチサーバラウンドロビン負荷分散の設定を行うことで、WEB-AP 間の通信を負荷分散させることが可能になります。

CORBA などオブジェクトを通してクライアント-サーバ間のリクエスト処理を実装する方式では、クライアントが利用するオブジェクトリファレンス(IOR)内にリクエストを処理するサーバの情報(サーバ名、ポート番号)が埋め込まれており、クライアントがリクエストを要求した場合、オブジェクト内で指定されているサーバに要求が行われます。

クライアントがオブジェクト取得を要求したときに返却するオブジェクトリファレンス内にあらかじめ複数のサーバ名が指定されており、クライアントリクエスト要求毎に、要求先のサーバ情報をそのオブジェクト内で指定されているサーバ情報よりラウンドロビンに取得することにより負荷分散を実現します。

(3) マルチサーバラウンドロビン負荷分散運用の場合の挙動

➢ 通常動作時の挙動

全ての AP が正常に動作している場合の挙動は以下となります。

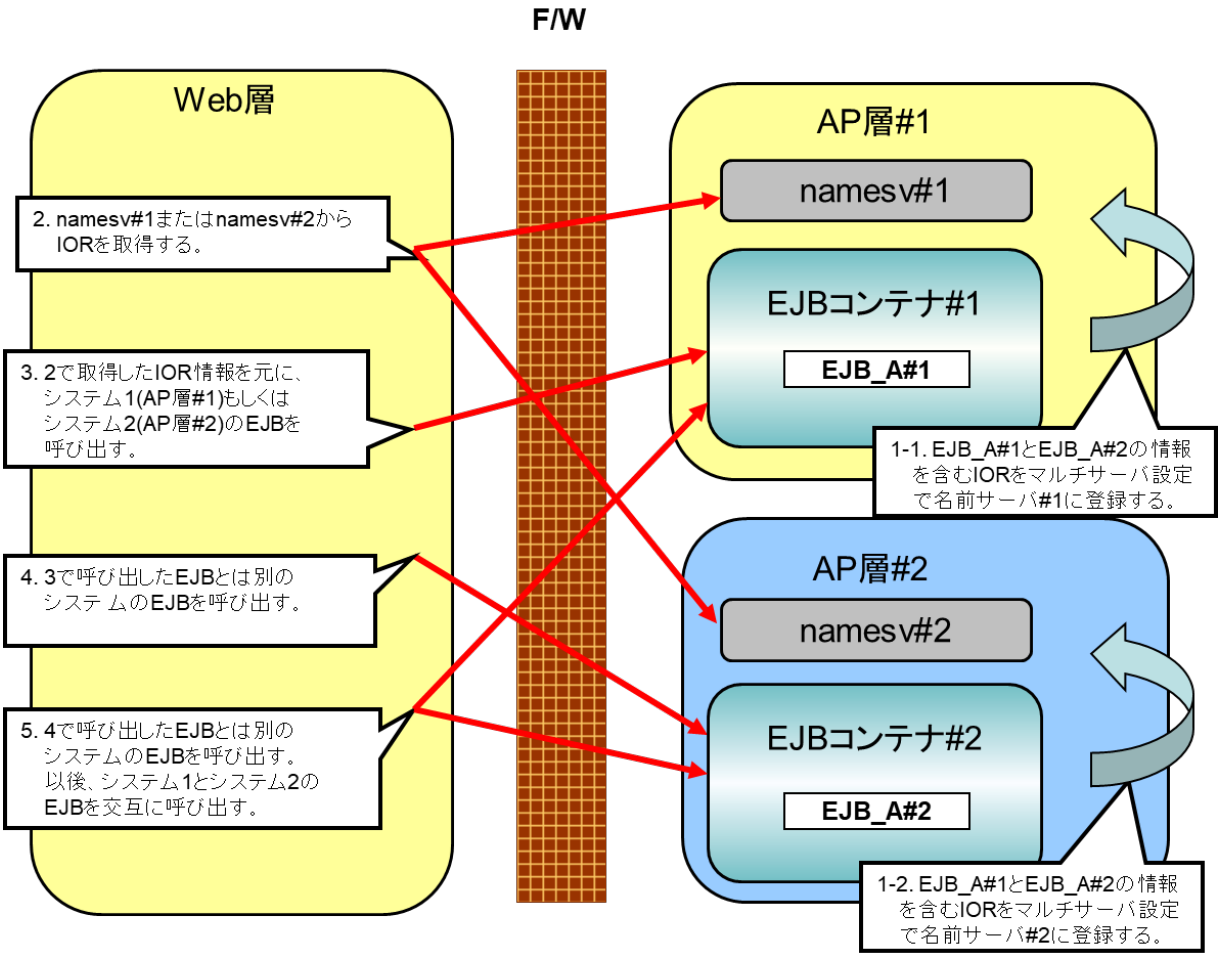


図 2.10.3 マルチサーバラウンドロビン負荷分散運用(通常動作時)

1-1. システム1の EJB とシステム 2 の EJB の情報を含む IOR をマルチサーバの設定で名前サーバ#1 に登録する。

1-2. システム1の EJB とシステム 2 の EJB の情報を含む IOR をマルチサーバの設定で名前サーバ#2 に登録する。
名前サーバ#1 と名前サーバ#2 に登録する IOR は必ず同じ情報のものを登録する。
複数の名前サーバを使用するのは、ひとつの名前サーバで異常が発生しても業務を継続させるためである。

2. クライアントは名前サーバ#1 または名前サーバ#2 から IOR を取得する。複数の名前サーバから任意のひとつの名前サーバを参照できるようにするためには、クライアント側では string_to_object()で、複数アドレスを記載した corbaname URL を指定する必要がある。

3. "2"で取得した IOR にはシステム 1 とシステム 2 の EJB を呼び出すための情報が埋め込まれており、このうち一方の EJB が呼び出される。仮にシステム1の EJB が呼ばれたとする。

4. "3"でシステム 1 のEJBが呼び出されたので、次の呼び出しではシステム 2 の EJB が呼び出される。

5. EJB の呼び出しはシステム1とシステム 2 で交互に行われる。

注意事項)

CORBA アプリケーションで Factory ありの場合、サーバオブジェクトをオペレーション呼び出しの度に生成するようにしないと負荷分散は行われません。

➢ 障害時の挙動

複数のうちひとつのシステムのサービスが停止した場合の挙動は以下となります。

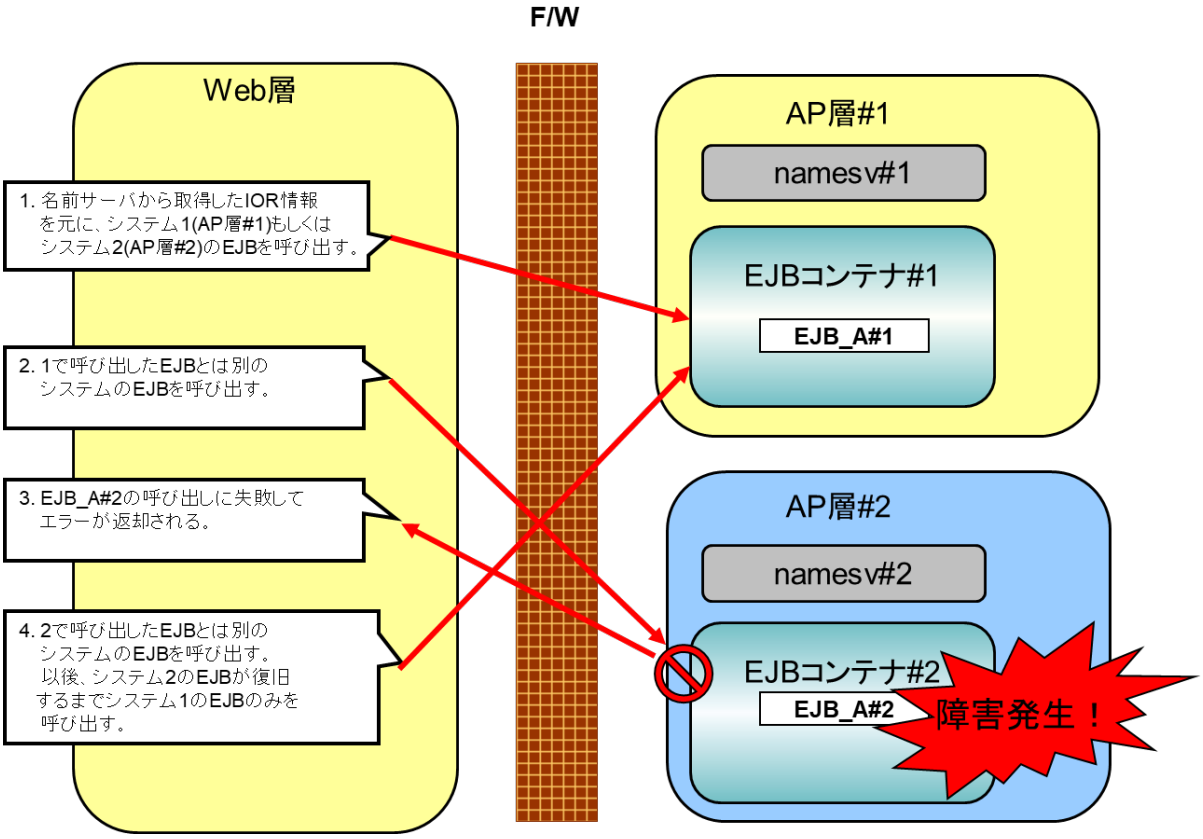


図 2.10.4 マルチサーバラウンドロビン負荷分散運用(障害時)

1. 名前サーバから取得した IOR にはシステム 1 とシステム 2 の EJB を呼び出すための情報が埋め込まれており、このうち一方の EJB が呼び出される。仮にシステム1の EJB が呼ばれたとする。
2. ″1″でシステム 1 のEJBが呼び出されたので、次の呼び出しではシステム 2 の EJB が呼び出される。
3. システム 2 の EJB で障害が発生していた場合、EJB の呼び出しに失敗してクライアントにエラーが返却される。エラー返却時にシステム1の EJB を再度自動で呼び出しにいくかどうかはクライアントアプリケーションの実装に依存する。
4. 次に EJB を呼び出した場合、″2″で呼び出した EJB とは別のシステム(つまりシステム 1)の EJB を呼び出す。以後システム2の EJB が復旧するまでシステム 1 の EJB のみを呼び出し、縮退運転となる。

注意事項)

オペレーションが閉塞している場合、クライアントは閉塞状態を検出することはできません。この場合、閉塞しているオペレーションが呼び出される度にエラーが返却されます。

➢ 障害復旧時の挙動

複数のうちひとつのシステムのサービスが停止し、そのサービスが復旧した場合の挙動は以下となります。

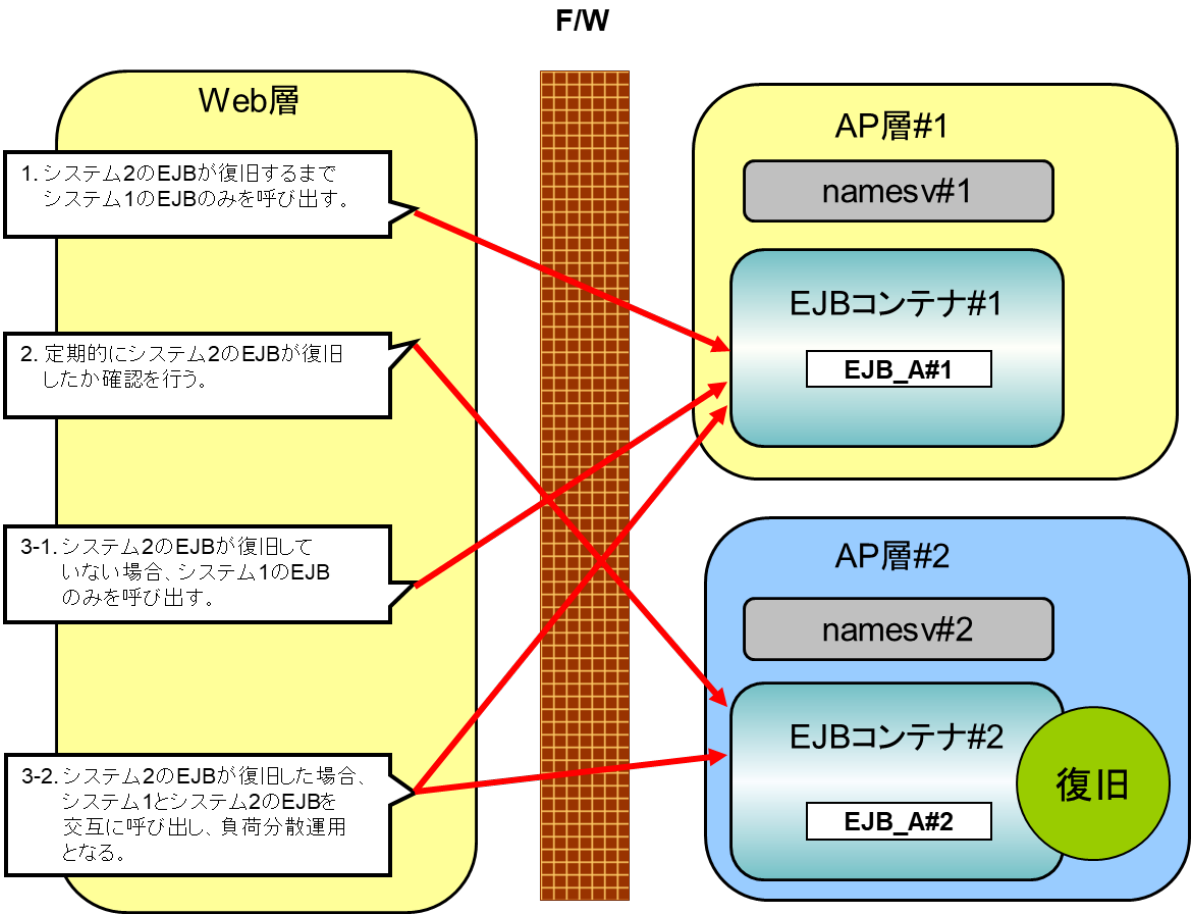


図 2.10.5 マルチサーバラウンドロビン負荷分散運用(障害復旧時)

1. システム 2 の EJB で障害が発生している場合、システム2の EJB が復旧するまでシステム 1 の EJB のみを呼び出し、縮退運転となる。
2. クライアントは定期的にシステム 2 の EJB が復旧したか確認を行う(既定値では 300 秒間隔)。チェック間隔の設定変更はクライアント側で実施し、Java の場合は jp.co.nec.orb.ConnectionAutoRetryInterval、C++の場合は ConnectionWatchInterval で設定する。

3-1. 状態チェック時にシステム 2 の EJB が復旧していない場合、クライアントは引き続きシステム1の EJB のみを呼び出す。

3-2. 状態チェック時にシステム 2 の EJB が復旧していた場合、クライアントはシステム 1 とシステム 2 の EJB を交互に呼び出し、負荷分散運用となる。システム起動時の正常運用状態に戻る。

(4) AP サーバ呼出の負荷分散機能(キャッシュ名前サービス)

➢ キャッシュ名前サービスの機能について

CNS は以下の機能を提供します。

- AP 呼び出しのためのオブジェクト(IOR)のキャッシュ機能
- 同期対象筐体の障害検知機能

➢ 使用するメリット

AP サーバ筐体が複数存在し、AP(EJB)呼び出しを負荷分散させる必要がある場合に、キャッシュ名前サービス(CNS)を使用します。CNS は WebOTX のサービス群のうちのひとつです。このサービスを起動するように設定し、WEB 層からの呼び出し設定を行うことで、WEB-AP 間の通信を負荷分散させることが可能になります。

CNS を使用しなくても負荷分散は可能ですが、AP 層のマシンの障害時に AP(EJB)呼び出しエラーが発生するため、CNS を使用する必要があります。

IOR とは、Web アプリケーションが EJB を呼び出す際に必要なオブジェクト(Interoperable Object Reference)であり、ここに

は呼び出し先 EJB のホスト情報、インタフェース情報が記載されています。

WEB 層に CNS を導入することで、IOR 取得のための WEB 層-EJB 層通信回数を減らすことができます。しかし、本資料では AP サーバに CNS を導入することを前提にします。その場合、AP 呼出のためのオブジェクト(IOR)のキャッシュ機能のメリットはありません。

（※ CNS を WEB 層に導入することにより、EJB 呼出のオブジェクト(IOR)のキャッシュ効果が得られますが、ここでは AP 層に CNS を配置することを前提に記述します）

(5) キャッシュ名前サービスを使用した場合の挙動

➢ 起動時の挙動

キャッシュ名前サーバが起動する際の挙動は以下となります。以下の説明は、WEB 層からの AP 層の CORBA 通信問合せは、複数 AP 層に対してパラレルで送信をし、レスポンスを速く返却した CNS を使用するという設定(CorbalocAskWithMT=true)がされていることを前提としています。

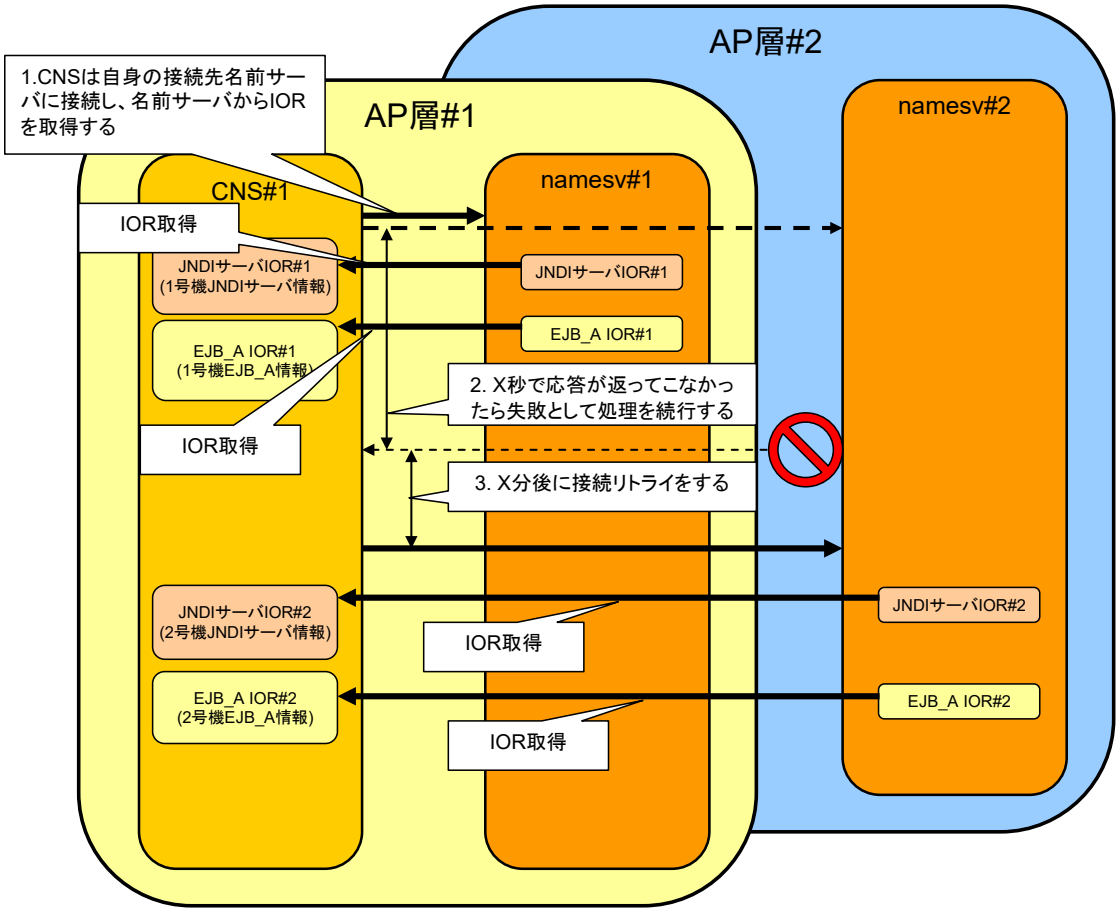


図 2.10.6 キャッシュ名前サービスの使用

1. キャッシュ名前サーバは、起動をすると、通信先として設定してある名前サーバ(namesv)に対して接続を行う。
2. デフォルト 30 秒待ち、名前サーバに対して接続ができなかった場合には、接続を中断し、起動作業を継続する。
3. デフォルト 5 分間隔で、通信先名前サーバに対して死活監視を行う。
4. 停止していた名前サーバに対して接続ができたら、その名前サーバ内の IOR を取得する。

➢ 通常動作時の挙動

キャッシュ名前サーバが正常稼動している状態の挙動は以下となります。

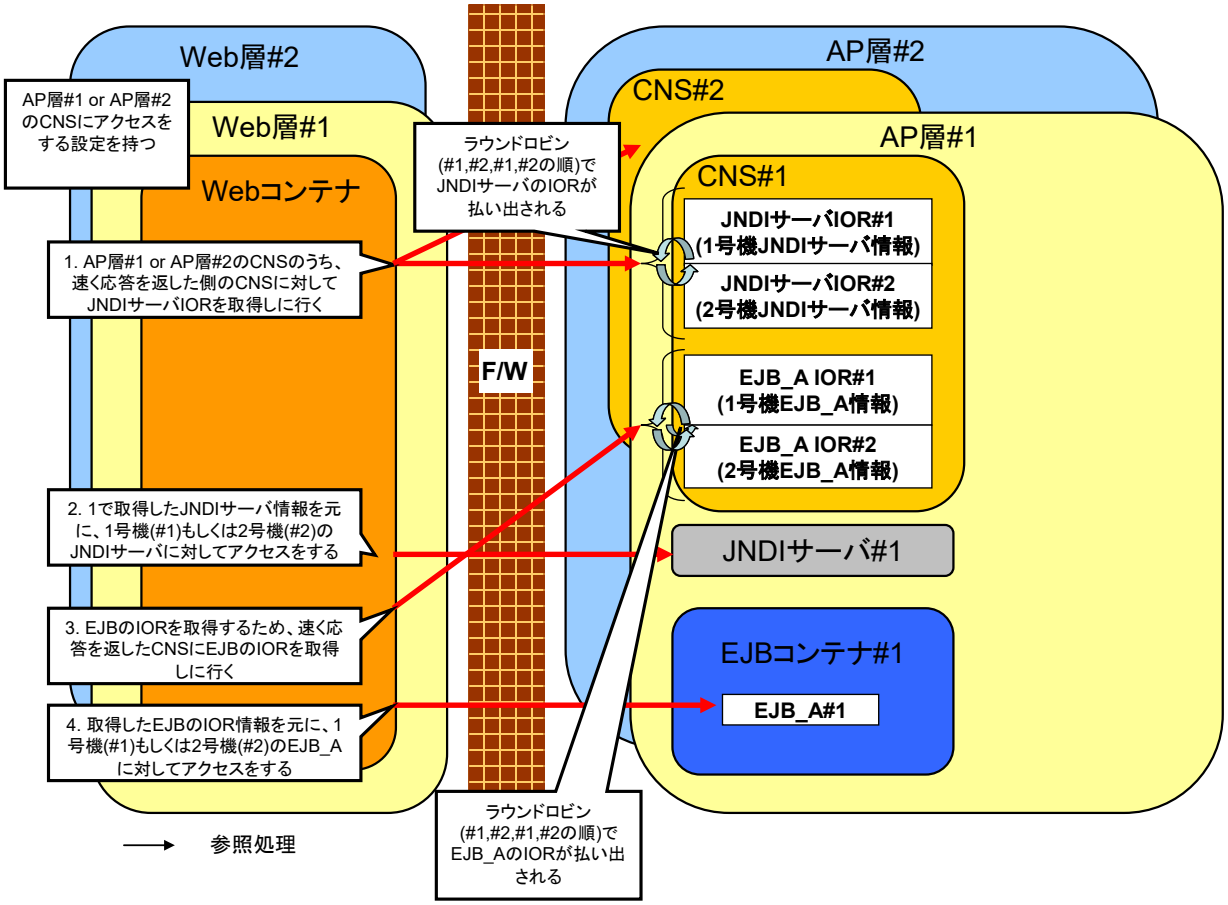


図 2.10.7 キャッシュ名前サーバ(正常稼動)

1. Web コンテナは、参照先として設定してある CNS に対してパラレルでアクセスをし、そのうちで一番応答が速かった CNS と通信を行い、JNDI サーバの情報を取得する。接続先 JNDI サーバは CNS からラウンドロビンで払い出される。つまり、CNS に登録されている JNDI サーバの IOR が 1 号機と 2 号機の筐体のものである場合、1 号機->2 号機->1 号機->2 号機の順番で IOR を取得することになる。

2. Web コンテナは、CNS から返却された JNDI サーバに対して、EJB の名前取得要求を行う。

3. 名前取得後、Web コンテナは、CNS から該当 EJB の IOR を取得する。この IOR はラウンドロビンで払い出される。

4. 払い出された IOR を使用して、Web コンテナは EJB 呼出を行う。

5. Web コンテナは、使用した CNS を優先して使用するように内部で接続先 CNS の情報を保持する。

➢ 障害時の挙動

キャッシュ名前サーバの障害時の挙動は、状況により挙動が異なります。

(1) 複数のうちの一台の AP 層マシンのサービスが停止した場合(OS は起動している)

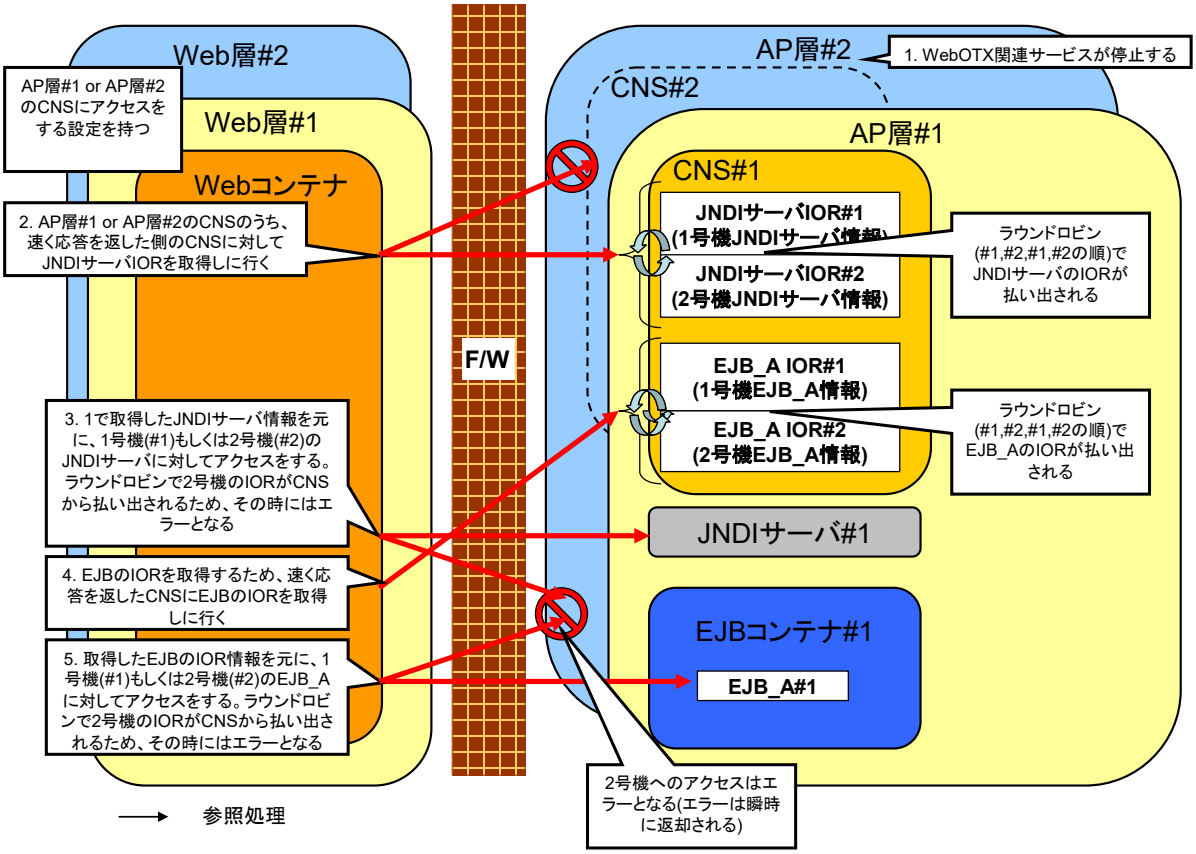


図 2.10.8 キャッシュ名前サーバ(障害1)

1. WebOTX のサービスがダウンする(OS は起動している)。

2. Web コンテナは AP 層#1 or AP 層#2 の CNS のうち、速く応答を返した側の CNS に対して JNDI サーバ IOR を取得しに行く。

3. CNS から払い出された IOR が、起動しているサーバの IOR であれば業務処理は続行する。

4. CNS から払い出された IOR が、停止しているサーバの IOR であれば、JNDI サーバ呼出時、もしくは EJB 呼出時にエラーが発生する。エラーは瞬時に返却される。

5. この後、CNS の死活監視が動作するまでは上記 3,4 の状態が続く。

6. CNS の死活監視が反応し、通信先サーバの名前サーバが停止していることを検知したら、自身のキャッシュ情報のうち、停止先サーバから登録された IOR を削除する。

(2) 複数のうちの一台の AP 層マシンの OS が停止した場合

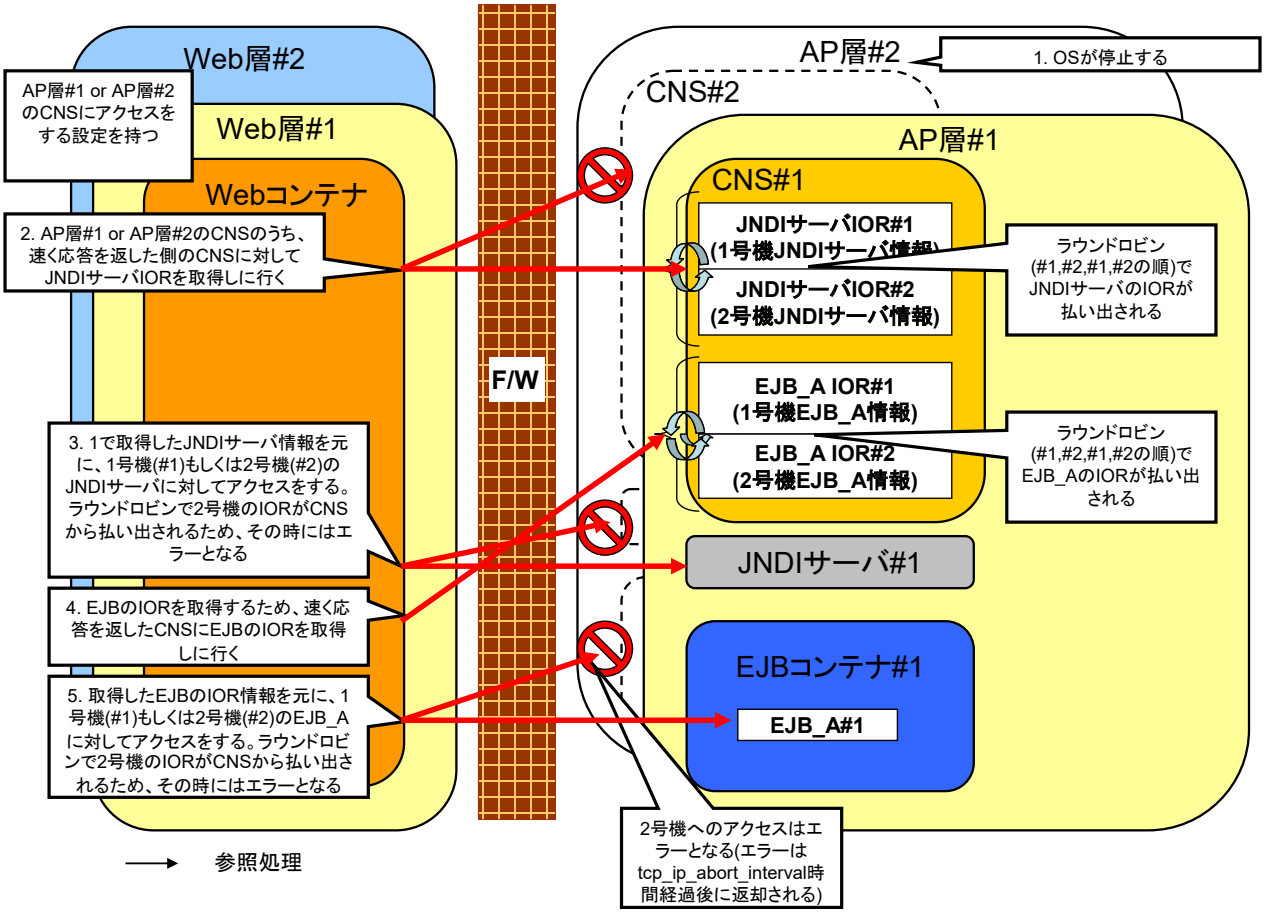


図 2.10.9 キャッシュ名前サーバ(障害 2)

1. OS がダウンする。

2. Web コンテナは AP 層#1 or AP 層#2 の CNS のうち、速く応答を返した側の CNS に対して JNDI サーバ IOR を取得しに行く。

3. CNS から払い出された IOR が、起動している筐体の IOR であれば業務処理は続行する。

4. CNS から払い出された IOR が、停止している筐体の IOR であれば、JNDI サーバ呼出時、もしくは EJB 呼出時にエラーが発生する。呼び出し元は tcp_ip_abort_interval まで待たされ、その後エラーとして処理を続ける。この時点でクライアントにエラーが返却される。この間、この WEB 層に入ってきたリクエストは全て待ち状態となる。

5. この後、CNS の死活監視が動作するまでは上記 4 の状態が続く。ただし、タイムアウトまでの時間は connect のタイムアウトとなる。

6. CNS の死活監視が反応し、通信先サーバの名前サーバが停止していることを検知したら、自身のキャッシュ情報のうち、停止先サーバから登録された IOR を削除する。

また、CNS は一定間隔で通信先への死活監視を行っています。通信先のサービス提供が不可能であることを検知すると、CNS は自分自身に保持されている情報のうち、死んだサービスから登録された情報を削除します。これにより、アクセスできない EJB の IOR を払い出すことを防ぎます。

その時の挙動は以下となります。

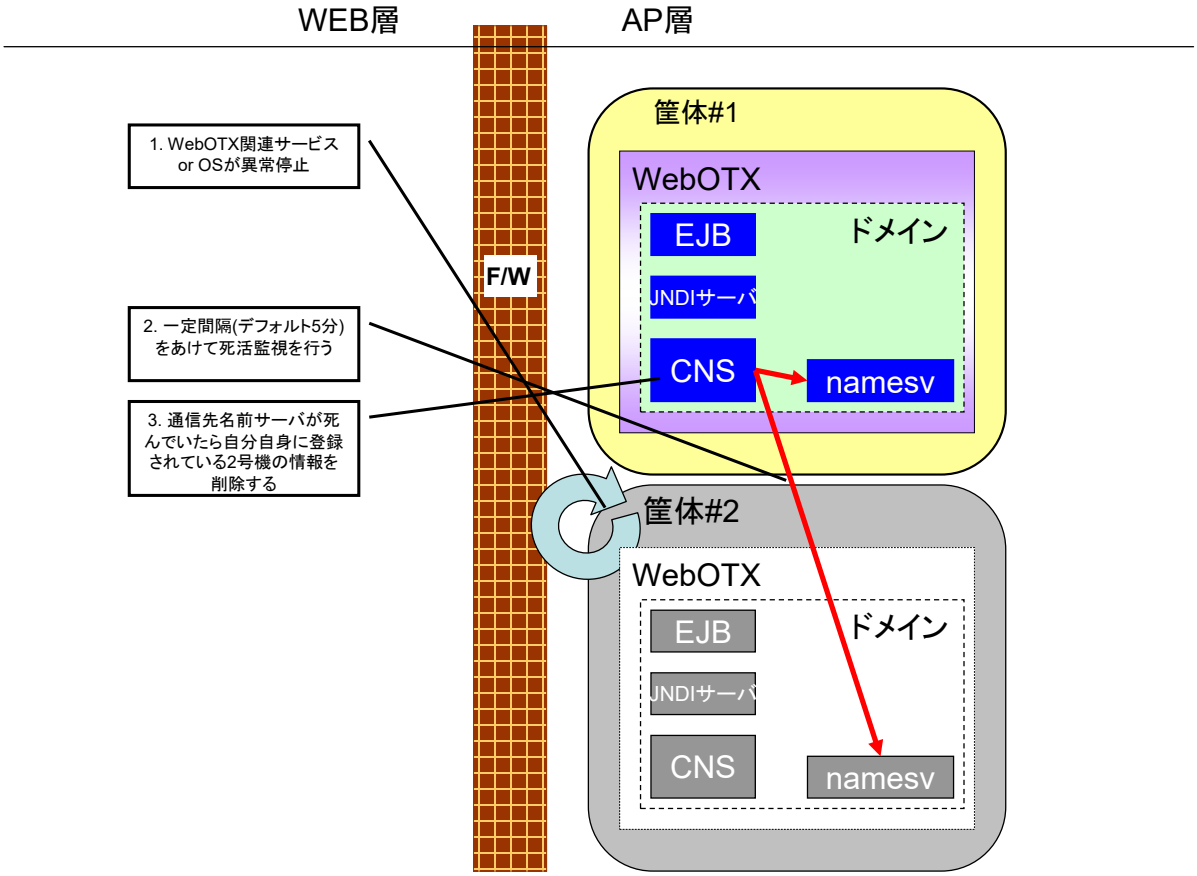


図 2.10.10 キャッシュ名前サーバ(障害 3)

1. OS、または WebOTX のサービスがダウンする。

2. CNS は一定間隔で通信先名前サーバへの死活監視を行う。

3. 通信先名前サーバからの応答がない場合、自身にキャッシュしてある IOR のうち、死んでいるサービスから登録された情を削除する。

➢ 障害復旧時の挙動

キャッシュ名前サーバが障害から復旧する際の挙動は以下となります。

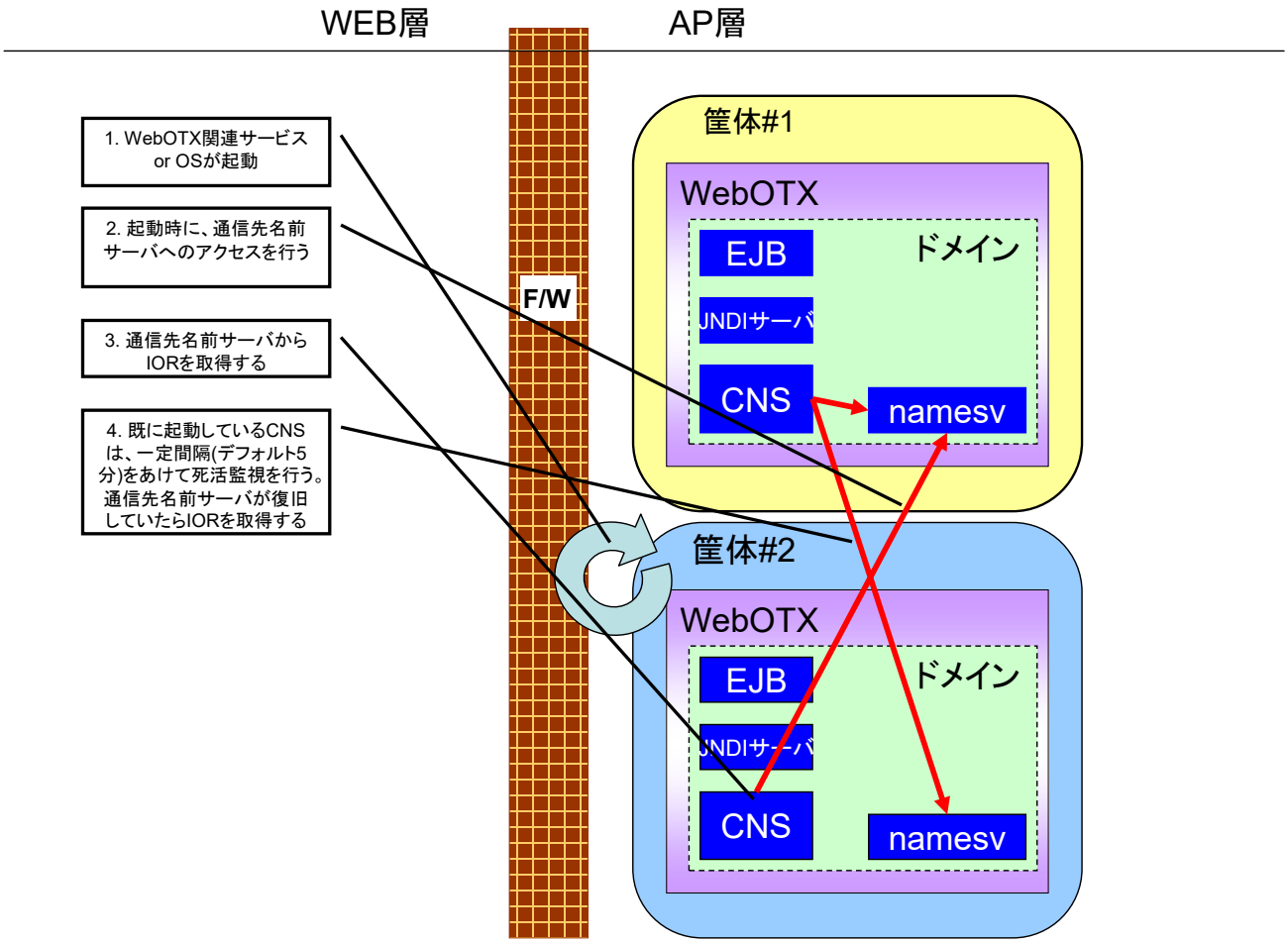


図 2.10.11 キャッシュ名前サーバ(障害復旧時)

1. OS、または WebOTX のサービスが起動する。

2. 障害が発生した側の CNS が起動する際、通信先の名前サーバへアクセスを行う。

3. 通信先名前サーバへのアクセスができたら、その名前サーバから IOR を取得する。

4. 最初から起動している側のサーバが、障害から復旧したサーバの名前サーバが起動していることを死活監視機能で確認したら、その時点で障害復旧側のサーバから IOR を取得する。

5. 4 にて、最初から起動している側のサーバが通信先の CNS の生存を確認するまでは、サーバ復旧後でもまだ死んでいるものとみなし、割り振り対象として認識をしない。

➢ 片系正常停止時の挙動

複数台構成の AP 層のマシンのうち、一台を停止する際の挙動は以下となります。

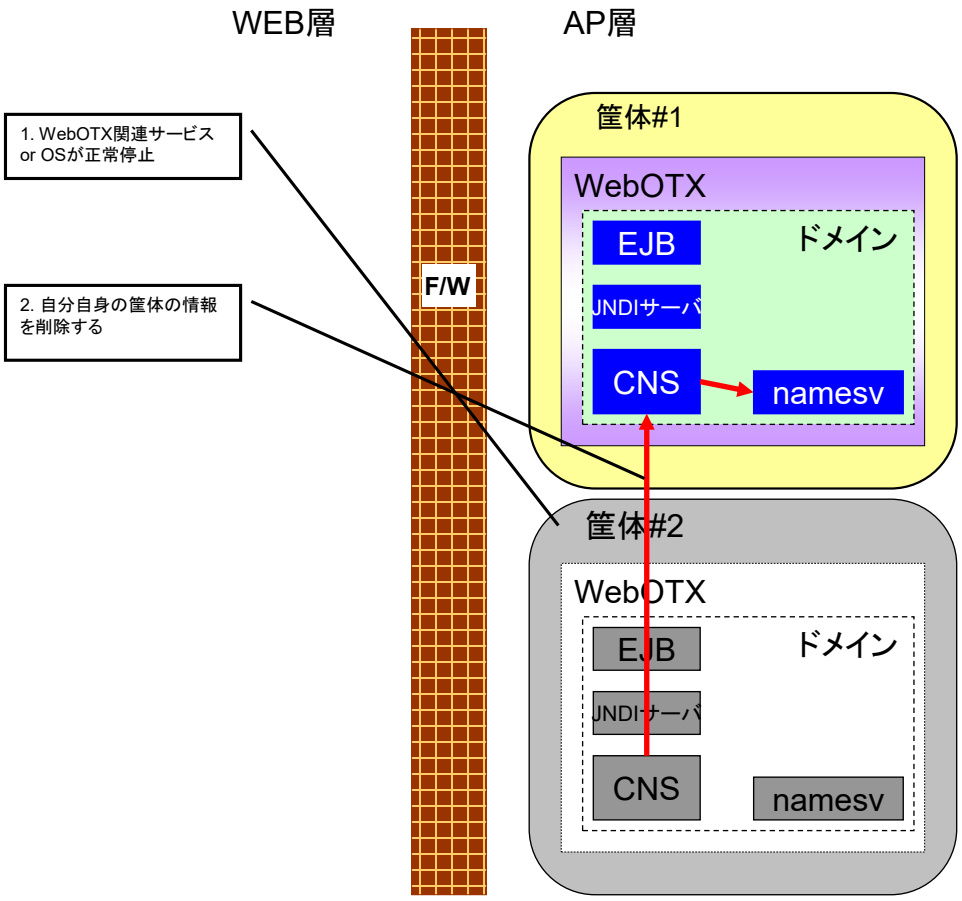


図 2.10.12 キャッシュ名前サーバ(片系正常停止時)

1. 片側のサービスを停止する。
2. CNS は停止時に、通信先の CNS から、自身の筐体から登録した IOR を削除する。

➢ 片系正常起動時の挙動

片側の筐体が既に起動しており、その後もう片側の筐体を OS 起動した場合の挙動は、障害復旧時の挙動と同様となります。

2.10.2.設定項目

マルチサーバラウンドロビン負荷分散運用を利用した分散の設計では、WebOTX マニュアル「運用編-3.運用と操作-TPモニタの運用操作-2.32 マルチサーバラウンドロビン負荷分散運用の設定」を参照して設定を行ってください。

3.その他

3.1.使用上の条件の確認

3.1.1.WebOTX V11.1

■ 目的

WebOTX を利用するために必要な条件について説明します。

リソース及び対応するソフトウェアはリリースされた時期やエディションによって異なる場合があります。
本節では WebOTX Media V11 Release 1 を利用して WebOTX Application Server V11.1 をインストールする場合の使用上の条件に関して記載します。
WebOTX Application Server V11.1 をインストールする場合の使用条件に関する詳細な情報や最新の情報については以下を参照してください。

・製品に添付されるインストールガイド
・利用バージョンの WebOTX のマニュアルに同梱されるインストールガイド
・WebOTX Web サイト 動作環境
https://jpn.nec.com/webotx/env/index.html

■ 概要

● リソース
WebOTX を利用する際に必要なメモリ容量、ハード ディスク空き容量について説明します。
● ソフトウェア
動作対象オペレーティング・システムや Java、データベース・サーバなどのソフトウェアについて説明します。

■ 説明

リソースについて

ここでは、インストールするために必要なハードディスク空き容量と、インストール中、およびインストール後の初期動作で必要なメモリ容量について説明します。
下記に示すメモリ容量は、インストール時に選択する必要のあるオプションは、すべて既定値を選択して動作させた場合を表しています。

・インストール時に必要なディスク容量、メモリ容量

WebOTX Application Server Standard

プラットフォーム	ハード ディスク	メモリ容量
Windows(x64)	440MB 以上	最小 1 GB、推奨 1.5 GB 以上
Linux(x64)	440MB 以上	最小 1 GB、推奨 1.5 GB 以上

WebOTX Application Server Standard Extended Option

プラットフォーム	ハード ディスク	メモリ容量
Windows(x64)	560MB 以上	最小 1 GB、推奨 1.5 GB 以上
Linux(x64)	560MB 以上	最小 1 GB、推奨 1.5 GB 以上

ソフトウェアについて

製品がサポートするオペレーティング・システム(OS)とハードウェア、および、製品を利用するために必要な関連ソフトウェアについて説明します。

・オペレーティング・システム(OS)
製品の動作対象であるオペレーティング・システムとハードウェアの対応を以下に示します

プラットフォーム	動作対象オペレーティング・システム
Windows(x64)	Windows Server® 2022 Standard (*1,2) Windows Server® 2022 Datacenter (*1,2) Windows Server® 2019 Standard (*1,2) Windows Server® 2019 Datacenter (*1,2) Windows Server® 2016 Standard (*1,2) Windows Server® 2016 Datacenter (*1,2)
Linux(x64)	Red Hat Enterprise Linux 8 Server (8.1 以降)(*3) Red Hat Enterprise Linux 7 Server (7.1 以降)(*3) Oracle Linux 8 (Red Hat Compatible Kernel) (8.1 以降)(*3,4,5) Oracle Linux 7 (Red Hat Compatible Kernel) (7.7 以降)(*3,4,5)

(*1) Server Core をサポートします。
(*2) Nano Server としてインストールした場合は未サポートとなります。
(*3) SELinux 有効化をサポート。
(*4) Oracle JDK 8/11/17 のみサポート
(*5) カーネルは Red Hat Compatible Kernel のみサポート。

・Java SE Development Kit
WebOTX は、実行時に Java™ Platform, Standard Edition の SDK を必要とします。サポートする SDK バージョンは次のとおりです

サポートする SDK バージョン
[Windows プラットフォーム] Oracle Java SE Development Kit 8 (Update 202 以降) Oracle Java SE Development Kit 11 (11.0.15 以降) LTS 版 (*1) Oracle Java SE Development Kit 17 (17.0.3 以降) LTS 版 OpenJDK 8 (Update 202 以降) (*2, 3) OpenJDK 11 (11.0.15 以降) (*2, 4) OpenJDK 17 (17.0.3 以降) (*2, 5) [Linux プラットフォーム] Oracle Java SE Development Kit 8 (Update 202 以降) Oracle Java SE Development Kit 11 (11.0.15 以降) LTS 版 (*1) Oracle Java SE Development Kit 17 (17.0.3 以降) LTS 版 OpenJDK 8 (Update 201 以降) (*2) OpenJDK 11 (11.0.15 以降) (*2) OpenJDK 17 (17.0.3 以降) (*2)

(*1) Java SE Subscription(有償)契約ユーザのみ取得可能
(*2) 各ディストリビュータからリリースされている OpenJDK をサポート
(*3) Eclipse Temurin JDK with Hotspot 8 について製品出荷前の Update にて評価済み
(*4) Eclipse Temurin JDK with Hotspot 11 について製品出荷前の Update にて評価済み
(*5) Eclipse Temurin JDK with Hotspot 17 について製品出荷前の Update にて評価済み
(*6) Red Hat リリース版をサポート

（注意） WebOTX 製品は、Windows、Linux に対応した Oracle 社製の Java SDK をバンドルしていますが、Java SDK 自体の保守は行っていませんので、ご了承ください。

・Web サーバ

WebOTX は次の Web サーバに対応しています。

対応する Web サーバ
WebOTX Web サーバ 2.4.54 以降 Apache HTTP Server 2.4.54 以降 Internet Information Services (IIS) 8.0(*1)、8.5(*1)、 10.0(*1)

(*1) 64bit 環境での、IIS 32bit モードはサポートしていません。

・データベース・サーバ

WebOTX がサポート対象とするデータベース・サーバは、プログラミング言語、オペレーティング・システムによって次の製品に対応しています。

・Java

WebOTX AS は、JDBC 2.0 から JDBC4.3 の仕様に準拠している JDBC ドライバを介して任意の DBMS への接続をサポートするように設計されています。アプリケーションが独自の方式でデータベース・サーバに接続したり、WebOTX が提供する JDBC データソースによる接続、あるいは、WebOTX の Transaction サービス機能と連携した JTA トランザクションを使用する場合には、データベース・サーバ製品にバンドルされる JDBC ドライバを入手して、セットアップしなければなりません。

WebOTX では以下の JDBC ドライバについて動作確認を行っております。

JDBC ベンダー	JDBC ドライバ・タイプ	サポートするデータベース・サーバ	備考
Oracle	Type 2、4	Oracle Database 11g Release 2 (11.2.0.4) Oracle Database 12c Release 1 (12.1.0.1.0) Oracle Database 12c Release 1 (12.1.0.2) Oracle Database 12c Release 2 (12.2.0.1.0) Oracle Database 18c(18.3.0) Oracle Database 19c(19.3.0.0.0) Oracle Database 19c(19. 4 .0.0.0) Oracle Database 19c(19.7.0.0.0) Oracle Database 19c(19.9.0.0.0) Oracle Database 19c(19.15.0.0.0) Oracle Database 21c(21.3.0.0.0)	
Oracle UCP	Type 2、4	Oracle Database 11g Release 2 以降、Oracle Database 21c まで	
Microsoft	Type 4	Microsoft SQL Server 2014 Microsoft SQL Server 2016 Microsoft SQL Server 2017 Microsoft SQL Server 2019	
DataDirect	Type 4	「Connect for JDBC 3.3 以降」経由による Oracle 接続	
PostgreSQL Development	Type 4	PostgreSQL 8.1（JDBC ドライバ 8.1 Build 401）～ PostgreSQL 14.4（JDBC ドライバ 42.4.0）	

Group			
Apache Derby	Type 4	Apache Derby 10.2.2 ～ 10.11.1.2	
Amazon RDS Maria DB	Type 4	Maria DB 10.0.24（JDBC ドライバ Maria DB Connection/J 2.0.2）～ MariaDB 10.6.7(JDBC ドライバ MariaDB Connector/J 2.7.5)	
Amazon RDS Aurora MySQL	Type 4	Aurora（MySQL-Compatible）5.6.10a（JDBC ドライバ mysql-connector-java-5.1.42）～Aurora MySQL 3.02.0 （compatible with MySQL8.0.23)(JDBC ドライバ mysql-connector-java-8.0.29)	
Amazon RDS Aurora PostgreSQL	Type 4	Aurora PostgreSQL(Compatible with PostgreSQL 14.3)(JDBC ドライバ 42.4.0)	

その他の製品についても、例えば MySQL Connector/J 5.0 など、JDBC 2.0 から JDBC4.3 の仕様に準拠している JDBC ドライバであれば、WebOTX AS と連携して使用することができます。ただし、十分な評価を行ってください。

3.2.パラメーター覧

2 章で述べたパラメータを、統合運用管理ツール/運用管理コマンドから設定する方法を一覧にまとめました。

3.2.1.性能

1） WebOTX が使用するメモリサイズの設定項目

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
エージェントプロセスの最大ヒープサイズ max-heap-size	ドメインのエージェントプロセスの JavaVM オプションとして最大ヒープサイズを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]- [JVM 構成]-[一般]-[最大 Java ヒープサイズ]を変更
JVM オプション jvm-options	JVM オプションを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]- [JVM 構成]-[JVM オプション]を変更
エージェントプロセスの GC ログ verbose-gc-enabled	ドメインのエージェントの JavaVM オプションとして GC ログオプションを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]- [JVM 構成]-[GC]-[GC 情報の出力]を変更
プロセスグループの最大ヒープサイズ maxHeapSize	プロセスグループの JavaVM オプションとして指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[TP システム]-[アプリケーショングループ]-[アプリケーショングループ名]-[プロセスグループ]-[プロセスグループ名]-[JavaVM オプション]-[最大ヒープサイズ] を変更
その他の引数 otherArguments	プロセスグループの JavaVM オプションのその他引数として指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[TP システム]-[アプリケーショングループ]-[アプリケーショングループ名]-[プロセスグループ]-[プロセスグループ名]-[JavaVM オプション]-[その他の引数] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
エージェントプロセスの最大ヒープサイズ max-heap-size	ドメインのエージェントプロセスの JavaVM オプションとして最大ヒープサイズを指定します。	otxadmin> set server.java-config.max-heap-size=
JVM オプション jvm-options	JVM オプションを指定します。	otxadmin> create-jvm-options "-XX¥:MaxMetaspaceSize=<size>"
エージェントプロセスの GC ログ verbose-gc-enabled	ドメインのエージェントの JavaVM オプションとして GC ログオプションを指定します。	otxadmin> set server.java-config.verbose-gc-enabled=true
プロセスグループの最大ヒープサイズ maxHeapSize	プロセスグループの JavaVM オプションとして指定します。	otxadmin> set tpsystem.applicationGroups.<アプリケーショングループ名>. processGroups.<プロセスグループ名>. maxHeapSize
その他の引数 otherArguments	プロセスグループの JavaVM オプションのその他引数として指定します。	otxadmin> set tpsystem.applicationGroups.<アプリケーショングループ名>. processGroups.<プロセスグループ名>. otherArguments="-XX:MaxMetaspaceSize=<size> -verbose:gc"

2） サービス抑止の設定項目

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
JMS サービスの起動の有効化 enabled	サーバ起動時における JMS サービスの起動可否を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[内部ライフサイクルモジュール]-[JMS プロバイダ]-[起動の有効化] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
JMS サービスの起動の有効化 enabled	サーバ起動時における JMS サービスの起動可否を指定します。	otxadmin> set server.internal-lifecycle-module.JMSProvider.enabled=false

3.2.2.多重度

1） Webサーバの多重度設定項目

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
最大同時接続数 Windows:ThreadsPerChild Linux: MaxClients	最大同時接続数を指定します。子プロセスで動作するスレッドの総数となります。Linux では、本値を変更する場合には、ThreadsPerChild、ServerLimit、ThreadLimit の各値を調整する必要があります。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[WebServer]-[最大同時接続数] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
最大同時接続数 Windows:ThreadsPerChild Linux: MaxClients	最大同時接続数を指定します。子プロセスで動作するスレッドの総数となります。Linux では、本値を変更する場合には、ThreadsPerChild、ServerLimit、ThreadLimit の各値を調整する必要があります。	otxadmin> set server.WebServer.MaxClients=250

ファイルを直接編集する

設定パラメータ(属性名)	説明	設定方法
子プロセスの最大リクエスト数 MaxRequestsPerChild	子プロセスが処理するリクエストの上限を指定します。本指定数のリクエストを受信すると子プロセスは終了します。0 を指定すると子プロセスは終了しなくなります。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebServer/httpd.conf に指定
子プロセスの上限値 ServerLimit	子プロセスの上限値を指定します。20000 まで指定可能です。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebServer/httpd.conf に指定
子プロセスのスレッドの上限値 ThreadLimit	子プロセスで動作するスレッドの上限値を指定します。Windows の場合は 15000 まで、Linux の場合は 20000 まで指定可能です。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebServer/httpd.conf に指定
子プロセスのスレッド数 ThreadsPerChild	子プロセスで生成されるスレッド数を指定します。ThreadLimit 以下の値を設定します。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebServer/httpd.conf に指定
起動時のプロセス数 StartServers	起動初期化時に生成されるプロセス数を指定します。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebServer/httpd.conf に指定
アイドルスレッド最小値 MinSpareThreads	アイドル状態のスレッドの最小値を指定します。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebServer/httpd.conf に指定
アイドルスレッド最大値 MaxSpareThreads	アイドル状態のスレッドの最大値を指定します。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebServer/httpd.conf に指定

ファイルを直接編集する

設定パラメータ(属性名)	説明	設定方法
プラグインモジュールの最大リクエスト処理数 connection_pool_size	mod_jk の同時実行数を指定します。	${INSTANCE_ROOT}/config/WebCont/workers.properties に指定

2） アプリケーションサーバの多重度設定項目

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法

スレッド数 threadCount	スレッド数を指定します。クライアントからの同時実行に備えて多重化します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [TPシステム]-[アプリケーショングループ]-[<アプリケーショングループ名>]-[プロセスグループ]-[<プロセスグループ名>]-[スレッド制御]-[スレッド数]を変更
プロセス数 processCount	プロセス数を指定します。マルチプロセス実行させることにより、アプリケーションの一時的な障害に対してサービスの継続が可能になります。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [TPシステム]-[アプリケーショングループ]-[<アプリケーショングループ名>]-[プロセスグループ]-[<プロセスグループ名>]-[プロセス制御]-[プロセス数]を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
スレッド数 threadCount	スレッド数を指定します。クライアントからの同時実行に備えて多重化します。	otxadmin> set tpsystem.applicationGroups.<アプリケーショングループ名>. processGroups.<プロセスグループ名>. threadCount=3
プロセス数 processCount	プロセス数を指定します。マルチプロセス実行させることにより、アプリケーションの一時的な障害に対してサービスの継続が可能になります。	otxadmin> set tpsystem.applicationGroups.<アプリケーショングループ名>. processGroups.<プロセスグループ名>. processCount=2

プロセスグループ

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
最大同時リクエスト処理数 simultaneousConnection Clients	利用可能な同時接続クライアント数を指定します。複数のクライアントから同時に TP モニタ上で動作する Web アプリケーションに接続する場合に設定してください。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [TPシステム]-[AJPListener]-[上限設定]-[最大同時リクエスト処理数] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
同時リクエスト処理数 simultaneousConnection Clients	利用可能な同時接続クライアント数を指定します。複数のクライアントから同時に TP モニタ上で動作する Web アプリケーションに接続する場合に設定してください。	otxadmin> set tpsystem.AJPListener.simultaneousConnectionClients=100

エージェントプロセス

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
最小プロセッサ数 min-thread-pool-size	同時に処理すべきリクエスト数の最小スレッド数を設定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [アプリケーションサーバ]-[スレッドプール]-[thread-pool]- [<スレッドプール名>]- [スレッドプール最小値]を変更
最大プロセッサ数 max-thread-pool-size	同時に処理すべきリクエスト数を拡大するために変更します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [アプリケーションサーバ]-[スレッドプール]-[thread-pool]- [<スレッドプール名>]- [スレッドプール最大値]を変更
スレッド数警告のしきい値 limit-thread-pool-size	アクティブなリクエスト処理スレッドの数が最大数に近づいた時に出力する警告ログの閾値を設定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [アプリケーションサーバ]-[スレッドプール]-[thread-pool]- [<スレッドプール名>]- [スレッドプールサイズの制限値]を変更
最大接続数 max-connections	コネクションの最大接続数を指定します。	[運用管理ツール]-[アプリケーションサーバ]-[ネットワーク構成]-[プロトコル構成]-[protocol]-[HTTP プロトコル] -[HTTP]
ソケットのバックログ値 accept-count	リクエスト受け付け用ソケットのバックログ値です。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [アプリケーションサーバ]-[ネットワーク構成]-[トランスポート構成]-[transport]-[TCP]-[バックログ]を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
最小プロセッサ数 min-thread-pool-size	同時に処理すべきリクエスト数の最小スレッド数を設定します。	otxadmin> set --user <ユーザ名> --password <パスワード> --host <ホスト名> --port <管理ポート> server.thread-

		pools.thread-pool.http-thread-pool.min-thread-pool-size =<最小数>
最大プロセッサ数 max-thread-pool-size	同時に処理すべきリクエスト数を拡大するために変更します。	otxadmin> set --user <ユーザ名> --password <パスワード> --host <ホスト名> --port <管理ポート> server.thread-pools.thread-pool.http-thread-pool.max-thread-pool-size =<最大数>
スレッド数警告のしきい値 limit-thread-pool-size	アクティブなリクエスト処理スレッドの数が最大数に近づいた時に出力する警告ログの閾値を設定します。	otxadmin> set --user <ユーザ名> --password <パスワード> --host <ホスト名> --port <管理ポート> server.thread-pools.thread-pool.http-thread-pool.limit-thread-pool-size =<しきい値>
最大接続数 max-connections	コネクションの最大接続数を指定します。	otxadmin> set --user <ユーザ名> --password <パスワード> --host <ホスト名> --port <管理ポート> server.network-config.protocols.protocol.{protocol-name}.http.max-connections
ソケットのバックログ値 accept-count	リクエスト受け付け用ソケットのバックログ値です。	otxadmin> set --user <ユーザ名> --password <パスワード>--host <ホスト名> --port <管理ポート> server.network-config.transports.transport.tcp. accept-count=<バックログ数>

3.2.3.通信/タイムアウト

1） クライアントの設定項目

共通

ファイルを直接編集する

設定パラメータ(属性名)	説明	設定方法
Web ブラウザ設定タイムアウト(IE の場合) ReceiveTimeout(DWORD)	Internet Explorer のタイムアウト値を設定します。 (単位:ミリ秒)	レジストリ HKEY_CURRENT_USER¥Software¥Microsoft¥Windows¥CurrentVersion¥Internet Settings で設定
セッションタイムアウト <session-timeout>タグ	セッションのタイムアウト時間(分)を指定します。web.xml 内に記述します。	web.xml の<web-app>-<session-config>-<session-timeout>タグ内にタイムアウト時間を指定

2） Web サーバの設定項目

共通

ファイルを直接編集する

設定パラメータ(属性名)	説明	設定方法
キープアライブタイムアウト KeepAliveTimeout	同じクライアントから到達するリクエストの待機時間を設定します。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebServer/httpd.conf で設定

ファイルを直接編集する

設定パラメータ(属性名)	説明	設定方法
コネクションプールタイムアウト worker. <worker 名> .connection_pool_timeout	Web サーバから Web コンテナ(AJP リスナ)へのコネクションを保持する時間を変更する場合、設定を変更します。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebCont/workers.properties で設定
ソケットタイムアウト worker. <worker 名> .socket_timeout	Web コンテナ(AJP リスナ)と Web サーバ間におけるソケット通信のタイムアウト時間を指定します。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebCont/workers.properties で設定

3） アプリケーションサーバの設定項目

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
リクエスト呼び出しのタイムアウト時間 RequestTimeout	CORBA や EJB アプリケーションを呼び出す際の最大待ち時間を設定します。設定時間(秒)を過ぎてもレスポンスが返却されない場合、org.omg.CORBA.NORESPONSE(5	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [アプリケーションサーバ]-[ORB コンフィグ]-[リクエスト呼び出しのタイムアウト時間] を変更

	130)例外が発生します。	
トランザクションタイムアウト tx-timeout	トランザクションタイムアウト時間(秒)を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[Transaction サービス]-[TM 設定]-[トランザクションタイムアウト時間] を変更
ログインタイムアウト loginTimeout	コネクション接続時のタイムアウト値(秒)を指定します。0 を指定した場合、監視は行われません。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[JDBC データソース]-[<JDBC データソース名>]-[コネクション制御]-[ログインタイムアウト] を変更
コネクション解放までの待ち合わせ時間 shrinkDelayTime	最小プールサイズを超えて払い出されたコネクションを解放するまでの待ち時間(秒)を指定します。値が 0 の場合待ち合わせは行われません。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[JDBC データソース]-[<JDBC データソース名>]-[コネクション制御]-[コネクション解放までの待ち合わせ時間] を変更
クエリータイムアウト queryTimeout	java.sql.Statement に指定するクエリのタイムアウト時間(単位:秒)を指定します。0 を指定した場合、監視は行われません。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[JDBC データソース]-[<JDBC データソース名>]-[コネクション制御]-[クエリータイムアウト] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
リクエスト呼び出しのタイムアウト時間 RequestTimeout	CORBA や EJB アプリケーションを呼び出す際の最大待ち時間を設定します。	otxadmin> set server.orb-config.RequestTimeout=30
トランザクションタイムアウト tx-timeout	トランザクションタイムアウト時間(秒)を指定します。	otxadmin> set server.transactionservice.tx-timeout=600
ログインタイムアウト loginTimeout	コネクション接続時のタイムアウト値(秒)を指定します。0 を指定した場合、監視は行われません。	otxadmin> set server.resources.jdbc-datasource.<データソース名>.loginTimeout=0
コネクション解放までの待ち合わせ時間 shrinkDelayTime	最小プールサイズを超えて払い出されたコネクションを解放するまでの待ち時間(秒)を指定します。値が 0 の場合待ち合わせは行われません。	otxadmin> set server.resources.jdbc-datasource.<データソース名>. shrinkDelayTime =15
クエリータイムアウト queryTimeout	java.sql.Statement に指定するクエリのタイムアウト時間(単位:秒)を指定します。0 を指定した場合、監視は行われません。	otxadmin> set server.resources.jdbc-datasource.<データソース名>. queryTimeout=0

3.2.4.セッション

1) セッション管理について

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
JNDI サーバの URL session-replication-jndi-url	セッションレプリケーション時の JNDI サーバの URL を指定します。複数の URL を記載する場合は、カンマで区切って指定してください。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[Web コンテナ]-[JNDI サーバの URL] を変更
セッションレプリケーションの監視間隔 webotx.jndi.client.listinginterval	セッションレプリケーション時の JNDI サーバの監視間隔を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]- [JVM 構成]-[JavaVM オプション] に設定

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
JNDI サーバの URL session-replication-jndi-url	セッションレプリケーション時の JNDI サーバの URL を指定します。複数の URL を記載する場合は、カンマで区切って指定してください。	otxadmin> set server.web-container. session-replication-jndi-url=(JNDI サーバの URL をカンマ区切りで指定)
セッションレプリケーションの監視間隔 webotx.jndi.client.listingi	セッションレプリケーション時の JNDI サーバの監視間隔を指定します。	otxadmin> create-jvm-options -Dwebotx.jndi.client.listinginterval=10

nterval		

ファイルを直接編集する

設定パラメータ(属性名)	説明	設定方法
セッションのクラスタ化 <distributable>タグ	JNDI サーバにセッションを保持する場合に指定します。web.xml に記述します。	web.xml の<web-app>タグ内に指定
セッションタイムアウト <session-timeout>タグ	セッションのタイムアウト時間(分)を指定します。web.xml 内に記述します。	web.xml の<web-app>-<session-config>-<session-timeout>タグ内にタイムアウト時間を指定

3.2.5.セキュリティ

1） 通信時のセキュリティを強化する

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
ユーザドメイン運用管理ポート番号 port	ユーザドメインのエージェントプロセスが利用する JMXMP 通信を行うためのポート番号を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[管理サービス]-[JMX コネクタ]-[system]-[設定項目]-[ポート] を変更
HTTP 通信用ポート番号 port	HTTP サーバが利用する HTTP ポートの番号を指定します。HTTP サーバを起動した場合に利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[WebServer]-[定義情報]-[ポート番号] を変更
SSL(HTTPS)通信用ポート番号 ssl-port	SSL で保護された HTTPS ポートの番号を指定します。HTTP サーバを起動した場合に利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[WebServer]-[定義情報(SSL)]-[HTTPS 通信用のポート番号] を変更
運用管理コンソールポート番号 port	エージェントが利用する管理用ポートの番号を指定します。Web 管理コンソールとの通信に利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[ネットワーク構成]-[ネットワークリスナ構成]-[network-listener]-[Admin リスナ]-[ポート番号] を変更
IIOP リスナ通信ポート番号(平文) listenerPortNumber	TP システム上の IIOP リスナが使用するポート番号を指定します。平文を利用する場合に利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [TP システム]-[IIOP リスナ]-[リスナ]-[平文ポート番号] を変更
IIOP リスナ通信ポート番号(SSL) sslPortNumberCert	TP システム上の IIOP リスナが使用するポート番号を指定します。SSL クライアント認証ありを利用する場合に利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [TP システム]-[IIOP リスナ]-[SSL]-[SSL ポート番号クライアント認証あり] を変更
IIOP リスナ通信ポート番号(SSL) sslPortNumberNoCert	TP システム上の IIOP リスナが使用するポート番号を指定します。SSL クライアント認証なしを利用する場合に利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [TP システム]-[IIOP リスナ]-[SSL]-[SSL ポート番号クライアント認証なし] を変更
IIOP リスナ通信ポート番号(JNDI) port	エージェントプロセス上で動作する IIOP リスナのポートを指定します。JNDI サーバとの通信に利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[組み込み IIOP サービス]-[ポート番号(暗号化なし)] を変更
IIOPAsync のポート番号 iiopAsyncPort	クライアント管理ライブラリや VB クライアント、CORBA Gateway を使用している場合、クライアントと通信を行うために使用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [TP システム]-[IIOP リスナ]-[リスナ]-[IIOPAsync のポート番号] を変更
データベースサーバポート番号 portNumber	データベースサーバのポート番号を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[ポート番号] を変更
SSL 通信の利用 security-enabled	SSL 通信を利用するかを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[WebServer]-[定義情報(SSL)]-[SSL(HTTPS 通信)の使用の有無] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
ユーザドメイン運用管理ポート番号 port	ユーザドメインのエージェントプロセスが利用する JMXMP 通信を行うためのポート番号を指定します。	otxadmin> set server.admin-service.jmx-connector.system.port=6212

HTTP 通信用ポート番号 port	HTTP サーバが利用する HTTP ポートの番号を指定します。HTTP サーバを起動した場合に利用します。	otxadmin> set server.WebServer.port =80
SSL(HTTPS)通信用ポート番号 ssl-port	SSL で保護された HTTPS ポートの番号を指定します。HTTP サーバを起動した場合に利用します。	otxadmin> set server.WebServer.ssl-port=443
運用管理コンソールポート番号 http-listener.admin-listener.port	エージェントが利用する管理用ポートの番号を指定します。Web 管理コンソールとの通信に利用します。	otxadmin> set server.network-config.network-listeners.network-listener.admin-listener.port=5858
IIOP リスナ通信ポート番号(平文) listenerPortNumber	TP システム上の IIOP リスナが使用するポート番号を指定します。平文を利用する場合に利用します。	otxadmin> set tpsystem.IIOPListener.listenerPortNumber=5151
IIOP リスナ通信ポート番号(SSL) sslPortNumberCert	TP システム上の IIOP リスナが使用するポート番号を指定します。SSL クライアント認証ありを利用する場合に利用します。	otxadmin> set tpsystem.IIOPListener.sslPortNumberCert =<ポート番号>
IIOP リスナ通信ポート番号(SSL) sslPortNumberNoCert	TP システム上の IIOP リスナが使用するポート番号を指定します。SSL クライアント認証なしを利用する場合に利用します。	otxadmin> set tpsystem.IIOPListener.sslPortNumberNoCert =<ポート番号>
IIOP リスナ通信ポート番号(JNDI) embedded-iiop-service.port	エージェントプロセス上で動作する IIOP リスナのポートを指定します。JNDI サーバとの通信に利用します。	otxadmin> set server.embedded-iiop-service.port=7780
IIOPAsync のポート番号 iiopAsyncPort	クライアント管理ライブラリや VB クライアント、CORBA Gateway を使用している場合、クライアントと通信を行うために使用します。	otxadmin> set tpsystem.IIOPListener.iiopAsyncPort=5220
データベースサーバポート番号 portNumber	データベースサーバのポート番号を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.portNumber=6789
SSL 通信の利用 security-enabled	SSL 通信を利用するかを指定します。	otxadmin> set server.WebServer.security-enabled=true

ファイルを直接編集する

設定パラメータ(属性名)	説明	設定方法
管理ドメイン運用管理ポート番号 admdomain.admin.port	管理ドメインのエージェントプロセスが利用する JMXMP 通信ポート番号を指定します。	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/domain.xml に指定

プロセスグループ

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
AJP リスナ(プロセスグループ用)通信ポート番号 Port	AJP リスナが使用するポート番号です。外部の HTTP サーバと AJP リスナが AJP によって連携を行う際のポートの番号。プロセスグループで HTTP サーバを起動して、AJP リスナを使用する場合利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[TP システム]-[AJPListener]- [リスナ]- [ポート番号] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
AJP リスナ(プロセスグループ用)通信ポート番号 Port	AJP リスナが使用するポート番号です。外部の HTTP サーバと AJP リスナが AJP によって連携を行う際のポートの番号。プロセスグループで HTTP サーバを起動して、AJP リスナを使用する場合利用します。	otxadmin> set tpsystem.AJPListener.listenerPortNumber=20102

エージェントプロセス

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
AJP リスナ(エージェントプロセス用)通信ポート番号 Port	AJP リスナが使用するポート番号です。外部の HTTP サーバと Web コンテナが AJP によって連携を行う際のポートの番号。HTTP サーバ、Web コンテナを起動した場合利用します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[ネットワーク構成]-[ネットワークリスナ構成]-[network-listener]-[AJP リスナ]-[ポート番号]を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
AJP リスナ(エージェントプロセス用)通信ポート番号 Port	AJP リスナが使用するポート番号です。外部の HTTP サーバと Web コンテナが AJP によって連携を行う際のポートの番号。HTTP サーバ、Web コンテナを起動した場合利用します。	otxadmin> set server.network-config.network-listeners.network-listener. agent-ajp-listener.port=8099

2) その他のセキュリティ設定

共通

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
運用ユーザ (admin) パスワード	WebOTX 管理ユーザのパスワードを指定します。8 文字以上で設定してください。	otxadmin> update-file-user (新パスワード) (ユーザ名)

httpd.conf(WebOTX Web サーバの設定)

設定パラメータ	説明	推奨値	既定値
Directory ディレクティブ (ディレクトリのアクセス権の設定)	指定したディレクトリに対してのアクセス制限、アクセス許可、アクセス拒否を設定します。Indexes を削除することで、ディレクトリ表示を無効とします。	Option FollowSymLinks	Option FollowSymLinks
ServerSignature ディレクティブ	エラーメッセージ出力時のフッタを表示するか否かの設定を行います。	Off	On
ServerTokens ディレクティブ	クライアントへの応答ヘッダや、サーバが生成するドキュメント(エラードキュメント等)に含める情報を指定します。ProductOnly とすると、WebOTX Web サーバの名前のみをヘッダに含めます。	ProductOnly	Full
Directory ディレクティブ (マニュアル、コンテキスト情報の削除)	Web サーバのマニュアル、コンテキスト情報を指定します。これらの部分をコメントアウトすることで、マニュアル、コンテキスト情報を削除できます。	# AliasMatch 略 # <Directory "/opt/WebOTX/WebServer24/manual">#略 # </Directory>	AliasMatch 略 <Directory "/opt/WebOTX/WebServer24/manual">略 </Directory>

default-web.xml(Web コンテナの設定)

設定パラメータ	説明	推奨値	既定値
ディレクトリリスティングの無効化	Web コンテナ上で動作する、全ての Web アプリケーションのディレクトリリスティングの無効化を行います。	<servlet> <init-param> <param-name>listings</param-name> <param-value>false</param-value> </init-param> </servlet>	false

3.2.6.DB 連携

1） データソース

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
データソースの種別 dataSourceType	データソースの種別を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[データソースの種別] を変更
データソース名 dataSourceName	JDBC URL またはデータベース名、データソース名を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[JDBC URL またはデータベース名、データソース名] を変更
JDBC 仕様のメジャーバージョン jdbcMajorVersion	JDBC 仕様のメジャーバージョンを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[JDBC 仕様のメジャーバージョン] を変更
JDBC 仕様のマイナーバージョン [jdbcMinorVersion]	JDBC 仕様のマイナーバージョンを設定します。 JDBC 仕様のバージョンが 4.1 の場合に 1 を設定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[JDBC 仕様のマイナーバージョン] を変更
サーバ名 serverName	データベースサーバのサーバ名を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[サーバ名] を変更
ポート番号 portNumber	データベースサーバのポート番号を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[ポート番号] を変更
ユーザ名 userName	データベースと接続するためのユーザ名を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[ユーザ名] を変更
パスワード password	データベースと接続するためのパスワードを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[パスワード] を変更
JNDI サーバへの登録名 jndiName	JNDI サーバへの登録名を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[JNDI サーバへの登録名] を変更
JTA 連携有無 useJTA	JTA と連携するかどうかを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[JTA 連携の有無] を変更

プロセスグループ単位でデータソースを設定する場合

設定パラメータ(属性名)	説明	設定方法
リソースのプロセス単位のロード設定 use-all-ejb-processgroups	リソースのプロセス単位のロード設定をするかどうかを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[一般]-[全ての EJB プロセスグループで使用するかどうか] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
データソースの種別 dataSourceType	データソースの種別を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.dataSourceType = <データソース種別>
データソース名 dataSourceName	JDBC URL またはデータベース名、データソース名を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.dataSourceName = <データソース名>
JDBC 仕様のメジャーバージョン jdbcMajorVersion	JDBC 仕様のメジャーバージョンを指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.jdbcMajorVersion = 2
JDBC 仕様のマイナーバージョン [jdbcMinorVersion]	JDBC 仕様のマイナーバージョンを設定します。 JDBC 仕様のバージョンが 4.1 の場合に 1 を設定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.jdbcMinorVersion = 1
サーバ名 serverName	データベースサーバのサーバ名を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.serverName = <DB サーバ名>
ポート番号 portNumber	データベースサーバのポート番号を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.portNumber = <DB ポート番号>
ユーザ名 userName	データベースと接続するためのユーザ名を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.userName = <ユーザ名>

パスワード password	データベースと接続するためのパスワードを指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.password = <パスワード>
JNDI サーバへの登録名 jndiName	JNDI サーバへの登録名を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.jndiName =<JNDI サーバ登録名>
JTA 連携有無 useJTA	JTA と連携するかどうかを指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.useJTA = <true/false>

プロセスグループ単位でデータソースを設定する場合

設定パラメータ(属性名)	説明	設定方法
リソースのプロセス単位のロード設定 use-all-ejb-processgroups	リソースのプロセス単位のロード設定をするかどうかを指定します。	[otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.use-all-ejb-processgroups= <true/false>

2） コネクション管理

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
最小プールサイズ minPoolSize	プールに常時保持されるコネクション数を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[最小プールサイズ] を変更
最大プールサイズ maxPoolSize	プールに保持される最大のコネクション数を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[最大プールサイズ] を変更
初期プールサイズ initialPoolSize	プール生成時に作成されるコネクション数を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[初期プールサイズ] を変更
コネクションの一括破棄可否 resetAllConnectionsOnFailure	コネクションの一括破棄を行うかどうかを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[コネクションの一括破棄可否] を変更
初期接続の接続リトライ使用有無 reconnectInitialPool	初期接続時の接続リトライを行うかどうかを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[初期接続時の接続リトライ有無] を変更
コネクション取得失敗時リトライ回数 connectRetryMax	JDBC コネクションの取得に失敗した場合の、接続リトライ回数を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクション制御]-[接続リトライ回数] を変更
コネクション取得失敗時リトライ間隔 connectRetryInterval	JDBC コネクションの取得に失敗した場合の、接続リトライ間隔（単位：秒）を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクション制御]-[接続リトライ間隔] を変更
コネクション解放までの待ち合わせ時間 shrinkDelayTime	最小プールサイズを越えて払い出された JDBC コネクションを解放するまでの待ち時間（単位：秒）を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクション制御]-[コネクション解放までの待ち合わせ時間] を変更
クエリータイムアウト queryTimeout	java.sql.Statement に指定するクエリのタイムアウト時間（単位：秒）を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクション制御]-[クエリータイムアウト] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
最小プールサイズ minPoolSize	プールに常時保持されるコネクション数を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.minPoolSize = 10
最大プールサイズ maxPoolSize	プールに保持される最大のコネクション数を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.maxPoolSize = 10
初期プールサイズ initialPoolSize	プール生成時に作成されるコネクション数を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.initialPoolSize = 4
コネクションの一括破棄可否 resetAllConnectionsOnFailure	コネクションの一括破棄を行うかどうかを指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.resetAllConnectionsOnFailure = <true/false>
初期接続の接続リトライ使用有無 reconnectInitialPool	初期接続時の接続リトライを行うかどうかを指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.reconnectInitialPool = <true/false>
コネクション取得失敗時リトライ回数	JDBC コネクションの取得に失敗した場合の、接続リトライ回数を指定	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>. connectRetryMax = 0

connectRetryMax	します。	
コネクション取得失敗時リトライ間隔 connectRetryInterval	JDBC コネクションの取得に失敗した場合の、接続リトライ間隔（単位：秒）を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>. connectRetryInterval = 10
コネクション解放までの待ち合わせ時間 shrinkDelayTime	最小プールサイズを越えて払い出された JDBC コネクションを解放するまでの待ち時間（単位：秒）を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>. shrinkDelayTime = 15
クエリータイムアウト queryTimeout	java.sql.Statement に指定するクエリのタイムアウト時間（単位：秒）を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>. queryTimeout = 120

3） 状態監視

共通

状態監視に関する設定方法を記述します

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
データベースサーバの状態監視コマンド checkServerCommand	データベースサーバの状態監視コマンドとして使用する SQL 命令を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[拡張]-[データベースサーバの状態監視コマンド] を変更
データベースサーバの状態監視間隔 checkServerInterval	データベースサーバの状態監視間隔を指定します。（単位：秒）	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[拡張]-[データベースサーバの状態監視間隔] を変更
データベースサーバの状態監視オプション checkServerOption	データベースサーバの状態監視契機を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[拡張]-[データベースサーバの状態監視オプション] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
データベースサーバの状態監視コマンド checkServerCommand	データベースサーバの状態監視コマンドとして使用する SQL 命令を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.checkServerCommand = “commit”
データベースサーバの状態監視間隔 checkServerInterval	データベースサーバの状態監視間隔を指定します。（単位：秒）	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.checkServerInterval = 30
データベースサーバの状態監視オプション checkServerOption	データベースサーバの状態監視契機を指定します。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.checkServerOption = “monitor”

3.2.7.ログ

パラメータはありません。

3.2.8.監視

パラメータはありません。

3.2.9.配備

パラメータはありません。

3.2.10.分散

共通

WebOTX 統合運用管理ツールを使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
キャッシュ名前サーバの URL	キャッシュ名前サーバを経由して JNDI サーバを使用する場合にキャ	[WebOTX 管理ドメイン[<ホスト名>]-[<ドメイン名>]-[アプリケーションサーバ]-[JNDI サービス]-[キャッシュ名前サー

cnsurl	ッシュ名前サーバの URL のリストをカンマで区切って指定します。 例 corbaname://host1:9829,corbaname://host2:9829。	バ連携]-[キャッシュ名前サーバ(CNS)の URL] を変更
キャッシュ名前サーバとの連携 NameServiceUseCNS	この設定は、名前サーバがキャッシュ名前サーバと連携するかどうかを指定します。この設定を"true"にすると、キャッシュ名前サーバとの連携を受け付けます。キャッシュ名前サーバと連携させる名前サーバが動作しているホストは"true"に指定してください。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[CORBA サービス]-[namesv]-[キャッシュ名前サーバとの連携指定] を変更
スレッド方針 CacheNameServiceThreadPolicy	キャッシュ名前サーバのスレッド方針を指定します。本値を「pool」と設定すると、キャッシュ名前サーバはマルチスレッドで動作します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[CORBA サービス]-[cnamesv]-[スレッド処理方針] を変更
キャッシュ元となる名前サーバのリスト nmsvlist	キャッシュ元となる名前サーバのリストを指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[CORBA サービス]-[cnamesv]-[キャッシュ元となる名前サーバのリスト]
連携している名前サーバの起動確認間隔 CacheSynchroInterval	キャッシュ名前サーバと連携している名前サーバの起動確認間隔(秒)を指定します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[CORBA サービス]-[cnamesv]-[連携している名前サーバの起動確認間隔] を変更
リクエストタイムアウト RequestTimeout	メソッド呼び出しにかかる最大待ち時間(秒)です。0 に設定すると永久待ちになります。キャッシュ名前サービス利用時は 0 に設定してはいけません。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]- [ORB コンフィグ]-[リクエスト呼び出しのタイムアウト時間] を変更
サービススタート時の自動起動 CacheNameServiceStartup	ドメイン起動時、CORBA サービス MO からのサービススタート時にキャッシュ名前サーバを起動するか指定します。cnamesvCacheNameService MO からのキャッシュ名前サーバ起動時は、この値に関係なく起動します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[CORBA サービス]-[cnamesv]-[サービススタート時の自動起動] を変更

運用管理コマンド(otxadmin)を使用して設定する場合

設定パラメータ(属性名)	説明	設定方法
キャッシュ名前サーバの URL cnsurl	キャッシュ名前サーバを経由して JNDI サーバを使用する場合にキャッシュ名前サーバの URL のリストをカンマで区切って指定します。 例 corbaname://host1:9829,corbaname://host2:9829。	otxadmin> set server.jndi-service.cnsurl=(CNS の URL をカンマ区切りで指定)
キャッシュ名前サーバとの連携 NameServiceUseCNS	この設定は、名前サーバがキャッシュ名前サーバと連携するかどうかを指定します。この設定を"true"にすると、キャッシュ名前サーバとの連携を受け付けます。キャッシュ名前サーバと連携させる名前サーバが動作しているホストは"true"に指定してください。	otxadmin> set server.corba-service.namesv.NameServiceUseCNS =true
スレッド方針 CacheNameServiceThreadPolicy	キャッシュ名前サーバのスレッド方針を指定します。本値を「pool」と設定すると、キャッシュ名前サーバはマルチスレッドで動作します。	otxadmin> set server.corba-service.cnamesv.CacheNameServiceThreadPolicy =pool
キャッシュ元となる名前サーバのリスト nmsvlist	キャッシュ元となる名前サーバのリストを指定します。	otxadmin> set server.corba-service.cnamesv.nmsvlist=(名前サーバの URL をカンマ区切りで指定)
連携している名前サーバの起動確認間隔 CacheSynchroInterval	キャッシュ名前サーバと連携している名前サーバの起動確認間隔(秒)を指定します。	otxadmin> set server.corba-service.cnamesv.CacheSynchroInterval=300
リクエストタイムアウト RequestTimeout	メソッド呼び出しにかかる最大待ち時間(秒)です。0 に設定すると永久待ちになります。キャッシュ名前サービス利用時は 0 に設定してはいけません。	otxadmin> set server.orb-config.RequestTimeout =30

サービススタート時の自動起動 CacheNameServiceStartup	ドメイン起動時、CORBA サービス MO からのサービススタート時にキャッシュ名前サーバを起動するか指定します。cnamesv MO からのキャッシュ名前サーバ起動時は、この値に関係なく起動します。	otxadmin> set server.corba-service.cnamesv.CacheNameServiceStartup=true

3.3.WebOTX で動作するプロセス

WebOTX 上で動作するプロセスの一覧です。

サービスプロセス

プロセス名	説明
WOAgentSvc.exe (Windows)	サービス管理用のプロセスです。 「WebOTX Agent Service」を起動すると動作し、停止すると終了します。

管理ドメインで動作するプロセス

プロセス名	説明
javaw.exe(Windows) java(Linux)	エージェントの JavaVM です。 異常終了した場合、管理ドメインへのアクセスができなくなりますが、一般ドメインは利用できます。復旧は WebOTX サービス再起動を行う必要があります。 引数に「funcid=agent domain.name=admin」という文字列が指定されますのでそれが目印になります。 Linux におけるカレントディレクトリは${INSTANCE_ROOT}/config です。

一般ドメインで動作するプロセス
ドメインを起動すると以下のプロセスが起動します。複数ドメインを起動した場合は、同一名のプロセスが複数起動します。

プロセス名	説明
java.exe(Windows) java(Linux)	エージェントの JavaVM です。 異常終了した場合、該当ドメインへのアクセスができなくなります。速やかにドメイン再起動を行う必要があります。 引数に「funcid=agent domain.name=${DOMAIN_NAME}」という文字列が指定されますのでそれが目印になります。Linux におけるカレントディレクトリは${INSTANCE_ROOT}/config です。
httpd.exe(Windows) httpd(Linux)	HTTP サーバのデーモンプロセス。インストール時に「WebOTX Web サーバ」を選択した場合に有効です。Web サーバを起動すると動作します。 常時親プロセス(監視プロセス)と子プロセス(HTTP サービスデーモン)で構成されており、子プロセスが異常終了した場合、親プロセスは子プロセスを再起動します。 異常終了時は HTTP サーバの再起動が必要です。 Linux におけるカレントディレクトリは${INSTANCE_ROOT}/です。
oad.exe (Windows) oad (Linux)	CORBA オブジェクト活性化デーモンプロセス。 CORBA サービスを起動すると動作します。異常終了時は新たな IIOP 通信(RMI/IIOP)が行えなくなります。 異常終了時は CORBA サービスの再起動が必要です。なお再起動方法については「※１」を参照ください。 Linux におけるカレントディレクトリは${INSTANCE_ROOT}/config です。 デフォルトでは、自動起動しません(V9.2 より)。CORBA サービスの irsv/corbaloc/cnamesv/oadj や、Transaction サービスの C++ AP 用 RCS プロセスを利用する場合に必要なプロセスです。 Express/Standard/ Standard + Extended Option　で動作します。
namesv.exe (Windows) namesv (Linux)	CORBA 名前サーバデーモンプロセス。 CORBA サービスを起動すると動作します。異常終了時はオブジェクトの取得が行えなくなり IIOP 通信ができなくなります。 異常終了時は CORBA サービスの再起動が必要です。なお再起動方法については「※１」を参照ください。 Linux におけるカレントディレクトリは${INSTANCE_ROOT}/config です。 Express/Standard/ Standard + Extended Option　で動作します。
irsv.exe (Windows) irsv (Linux)	CORBA インターフェースリポジトリデーモンプロセス。 CORBA サービスを起動すると動作します。異常終了時はオブジェクトのイ

	ンタフェース情報の取得が行えなくなりますが WebOTX では irsv を利用していないため影響はありません。 異常終了時は CORBA サービスの再起動が必要です。なお再起動方法については「※１」を参照ください。 デフォルトでは、自動起動しません。 Standard + Extended Option で動作します。
corbaloc.exe (Windows) corbaloc (Linux)	CORBALOC サーバデーモンプロセス。 CORBA サービスを起動すると動作します。異常終了時は CORBALOC サーバとして利用している場合、新たな IIOP 通信(RMI/IIOP)が行えなくなります。 異常終了時は CORBA サービスの再起動が必要です。なお再起動方法については「※１」を参照ください。 デフォルトでは、自動起動しません。 Standard + Extended Option で動作します。
cnamesv.exe (Windows) cnamesv (Linux)	キャッシュ名前サーバデーモンプロセス。 CORBA サービスを起動すると動作します。異常終了時はキャッシュ名前サーバとして利用している場合、新たな IIOP 通信(RMI/IIOP)が行えなくなります。 異常終了時は CORBA サービスの再起動が必要です。なお再起動方法については「※１」を参照ください。 デフォルトでは、自動起動しません。 Standard + Extended Option で動作します。
java.exe (Windows) java (Linux)	CORBA Java 自動起動デーモンプロセス。 CORBA サービスを起動すると動作します。異常終了時は Java アプリケーションの自動起動ができなくなります。また、ライセンスの取得もできなくなるため、IIOP 通信のコネクション数に制限が発生します。 異常終了時は CORBA サービスの再起動が必要です。なお再起動方法については「※１」を参照ください。 引数に「funcid=oadj domain.name=${DOMAIN_NAME}」という文字列が指定されますのでそれが目印になります。 Linux におけるカレントディレクトリは${INSTANCE_ROOT}/config です。 デフォルトでは、自動起動しません。 Express/Standard/ Standard + Extended Option　で動作します。
tpmMain.exe (Windows) tpmMain (Linux)	アプリケーションプロセスの障害検出や異常終了時の自動復旧など、WebOTX の高信頼性を実現させているプロセスです。 Standard/Standard + Extended Option　で動作します。 異常終了した場合 TP モニタの機能が利用できなくなるため、プロセス監視を行なう場合は、監視対象にしてください。 復旧方法は TP システムの再起動です。それでも復旧しない場合はマシン再起動を行なってください。 カレントディレクトリは${INSTANCE_ROOT}/logs/tpsystem です。
oltpad.exe (Windows) oltpad (Linux)	WebOTX エージェントプロセス(javaw)からの要求を受けるなど、TP モニタの運用操作に必要なプロセスです。 Standard/Standard + Extended Option　で動作します。 異常終了した場合、一時的に Standard の一部運用操作がエラーとなりますが、自動的にプロセスが再起動され復旧します。 カレントディレクトリは${INSTANCE_ROOT}/logs/tpsystem です。
tpssendtpp.exe（Windows） tpssendtpp（Linux）	クライアントへのメッセージ送信等を行うプロセスです。 Standard/Standard + Extended Option で動作します。 異常終了した場合、統合運用管理ツールからクライアントへメッセージ送信や動的ログレベル変更などが不可となります。復旧方法は TP システムの再起動です。
systpp.exe (Windows) systpp (Linux)	アプリケーショングループの起動や停止などを実行するためのプロセスです。 Standard/Standard + Extended Option　で動作します。 異常終了した場合、一部運用操作(起動、停止など)が不可となります。復旧方法は TP システムの再起動です。
jnlwrt.exe (Windows) jnlwrt (Linux)	ジャーナルを収集しているプロセスです。 Standard で動作します。 異常終了した場合、ジャーナルの採取が不可となります。復旧方法は TP シ

<table>
<tr><td></td><td>ステムの再起動です。</td></tr>
<tr><td>olftplsn.exe（Windows）
olftplsn（Linux）</td><td>。TP モニタの OLF/TP リスナプロセスです。
Standard/Standard + Extended Option で動作します。
異常終了時は OLF リスナとの通信が不可となります。復旧方法は TP システムの再起動です。</td></tr>
<tr><td>iioplsn.exe (Windows)
iioplsn (Linux)</td><td>クライアントとの接続切断や、電文の送受信を行っている IIOP リスナのプロセスです。
Standard/Standard + Extended Option で動作します。
異常終了時は IIOP リスナとの通信が不可となります。全てのクライアントから TP モニタ上で動作する EJB、CORBA アプリケーションへアクセスができなくなります。復旧方法は TP システムの再起動です。
カレントディレクトリは${INSTANCE_ROOT}/logs/tpsystem です。</td></tr>
<tr><td>ajplsn.exe (Windows)
ajplsn (Linux)</td><td>AJP リスナのプロセスです。
Standard/Standard + Extended Option で動作します。
異常終了時は AJP リスナとの通信が不可となります。全てのクライアントからの TP モニタ上で動作する Web アプリケーションへアクセスができなくなります。復旧方法は TP システムの再起動です。
カレントディレクトリは${INSTANCE_ROOT}/logs/tpsystem です。</td></tr>
<tr><td>iiopAsync.exe(Windows)
iiopAsync(Linux)</td><td>iioplsn の子プロセスです。
Standard/Standard + Extended Option で動作します。
クライアント管理ライブラリや VB クライアント、CORBA Gateway を使用している場合、クライアントとの間で制御的なコネクションを確立するプロセスです。
異常終了時は、クライアントのアライブチェック、非同期メッセージ送信が不可となります。
復旧方法は TP システムの再起動が必要です。</td></tr>
<tr><td></td><td></td></tr>
<tr><td>wosystpp.exe (Windows)
wosystpp (Linux)</td><td>トレースレベルとサイズの変更、モジュール動的追加と削除、サーバプロセスメッセージ通知、モジュールの活性化と非活性化、スタックトレースの採取を行うためのプロセスです。
Standard/Standard + Extended Option で動作します。
異常終了時は上記機能が使用できなくなります。復旧方法は TP システムの再起動です。</td></tr>
<tr><td>THTPPCTL_2005.exe
THTPPCTL_2008.exe
THTPPCTL_2010.exe
(THTPPCTL_2012.exe
THTPPCTL_2013.exe
THTPPCTL_2015.exe
THTPPCTL_2017.exe
THTPPCTL_2019.exe
（Windows）
THTPPCTL9
THTPPJAVA2
（Linux）</td><td>プロセスグループに配備したサーバアプリケーションが動作するプロセスです。
Standard/Standard + Extended Option で動作します。言語や利用機能によりプロセス名が変わります。

（Windows）
<table>
<tr><th>プロセス名</th><th>言語・利用機能</th><th>プロセスグループに配備可能な AP のバージョン</th></tr>
<tr><td>THTPPCTL_2005.exe</td><td>C++(VC 2005)</td><td>V8,V9,V11</td></tr>
<tr><td>THTPPCTL_2008.exe</td><td>C++(VC 2008)</td><td>V8,V9,V11</td></tr>
<tr><td rowspan="3">THTPPCTL_2010.exe</td><td>Jakarta EE</td><td>全バージョン</td></tr>
<tr><td>CORBA Java</td><td>全バージョン</td></tr>
<tr><td>C++(VC 2010)</td><td>V9、V11</td></tr>
<tr><td>THTPPCTL_2012.exe</td><td>C++(VC 2012)</td><td>V9,V11</td></tr>
<tr><td>THTPPCTL_2013.exe</td><td>C++(VC 2013)</td><td>V9,V11</td></tr>
<tr><td>THTPPCTL_2015.exe</td><td>C++(VC 2015)</td><td>V9,V11</td></tr>
<tr><td>THTPPCTL_2017.exe</td><td>C++(VC 2017)</td><td>V9,V11</td></tr>
<tr><td>THTPPCTL_2019.exe</td><td>C++(VC 2019)</td><td>V9,V11</td></tr>
</table>

（Linux）</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>プロセス名</td><td>言語・利用機能</td><td>プロセスグループに配備可能な AP のバージョン</td></tr>
<tr><td>THTPPCTL9</td><td>C++</td><td>V9,V11</td></tr>
<tr><td rowspan="2">THTPPJAVA2</td><td>Jakarta EE</td><td>全バージョン</td></tr>
<tr><td>CORBA Java</td><td>全バージョン</td></tr>
</table>
異常終了時はそのプロセスグループが機能しなくなります。ただし、マルチプロセス構成や再起動設定を行うことで、自動的に復旧することができます。
再起動設定に関しては、WebOTX マニュアル 「構築・運用 – ドメインの構築 - 4. TP システム - 4.1. 操作・状態確認(TP システム) - 4.1.2. 設定値の説明 - [プロセス障害時の再起動回数][プロセスを正常と仮定する間隔]」を参照してください。
カレントディレクトリは${INSTANCE_ROOT}/logs/tpsystem です。
</td></tr>
</table>

単体で動作するプロセス
ツールなど単独で動作可能なプロセスです。複数起動した場合は、同一名のプロセスが複数起動します。

プロセス名	説明
java.exe(Windows) java(Linux)	運用管理コマンド(otxadmin)のプロセスです。 引数に「funcid=otxadmin」という文字列が指定されますのでそれが目印になります。
javaw.exe(Windows)	統合運用管理ツール(otxadmingui)のプロセスです。 引数に「com.nec.webotx.admingui.AdminGUIStandAlone」という文字列が指定されますのでそれが目印になります。
javaw.exe(Windows) javaw(Linux)	JNDI 管理ツール(jndiadm)のプロセスです。 引数に「funcid= jndiadm」という文字列が指定されますのでそれが目印になります。

※1 Object Broker サービスの起動･停止方法
起動
otxadmin> invoke server.corba-service.start
停止
otxadmin> invoke server.corba-service.stop

※2 Transaction サービスの起動･停止方法
起動
otxadmin> invoke server.transactionservice.start
または
otxadmin> start-transaction-service
停止
otxadmin> invoke server.transactionservice.stop
または
otxadmin> stop-transaction-service