WebOTX Application Server V7.1 構築ガイド

2009 年 12 月 11 日

第 4 版

NEC

第二システムソフトウェア事業部

AP サーバ開発グループ

改訂履歴

版　数	更新日付	更新内容【概要】
1	2009/05/18	初版作成
2	2009/06/15	2.2 多重度の概要説明において、説明を追記 2.5 セキュリティの設定項目において、http-listner を無効化する設定を削除 2.6 データベース連携の概要説明、設計指針において、説明を追記
3	2009/09/14	2.2 多重度の設計指針において、アプリケーションサーバの実行多重度設計条件を変更 2.3 通信・タイムアウトにおいて CORBA リクエストタイムアウトの説明を追記 2.8 監視の設計指針において、プロセスの障害発生時の対処法の説明を追記
4	2009/12/11	タイトルを「WebOTX Application Server V7.1 構築ガイド」に変更 2.2 多重度の同時リクエスト処理数にキープアライブタイムアウトの説明を追記 2.3 通信/タイムアウトにキープアライブタイムアウトの説明を追記 3.2.3 通信/タイムアウトのパラメータ一覧にキープアライブタイムアウトを追記

目次

1.はじめに

「WebOTX Application Server 構築ガイド」は、WebOTX Application Server を導入される方を対象に、一般的な設計・構築指針をわかりやすく記載します。

利用者向けの画面のカスタマイズやシステムチューニングについては、一部を除き本書では説明しておりません。

1.1.1.「WebOTX Application Server」とは

「WebOTX Application Server」(以降、WebOTX と記載)とは、拡張性に優れた業務システム構築を実現する J2EE 対応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、長期に渡って安心いただける万全のサポートとあわせて、企業システムのインフラとして最適なミドルウェア製品です。

本構築ガイドでは、「WebOTX Application Server V7.11」を対象としております。

1.1.2.WebOTX Application Server V7.1 構築ガイドの目的

WebOTX を採用するシステム担当者が、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記載し、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 を用いたシステムを構築できることを目指します。

1.1.3.WebOTX Application Server V7.1 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) アプリケーション基盤担当者

WebOTX の設計を行います

詳細設計にて利用することを想定しております。

2) 基盤担当者

WebOTX の構築、パラメータ設定、動作確認を行います

構築・評価にて利用することを想定しております。

1.1.4.本文の記述に関して

本構築ガイドは、「マルチプロセスモード」、「シングルプロセスモード」 両方に対応しております。

マルチプロセスモードとは、WebOTX V6.5 から導入されており、これを使うことで、Web コンテナ上で動作する Web アプリケーションを、WebOTX が提供する TP モニタ上で監視可能、かつマルチプロセスでの並列処理が可能となるため、Web システムの信頼性を実現します。

シングルプロセスモードとは、V6.4 までの構成で、それらの機能を使用することができません。

マルチプロセスモードとシングルプロセスモードでは、記載内容がことなりますので、項目ごとに以下のタグを付けております。

共通

共通タグ:マルチプロセスモード、シングルプロセスモード共通項目です。

マルチ

マルチタグ:マルチプロセスモードに関する記述がされております、シングルプロセスモード選択時は不要な項目です。

シングル

シングルタグ:シングルプロセスモードに関する記述がされております、マルチプロセスモード選択時は不要な項目です。

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.設計

本章では、以下の設計項目について、考え方や設計方法を示します。

1. 性能設計

限られたリソースの中で効率よく処理を行えるように、サーバの CPU、メモリなどのパラメータの設計方法を示します。

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。

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） プロセスモードの選択

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設計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） プロセスモードの選択 マルチ シングル

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 のヒープメモリについて

3） 使用していないサービスについて 共通

1） プロセスモードの選択 マルチ シングル

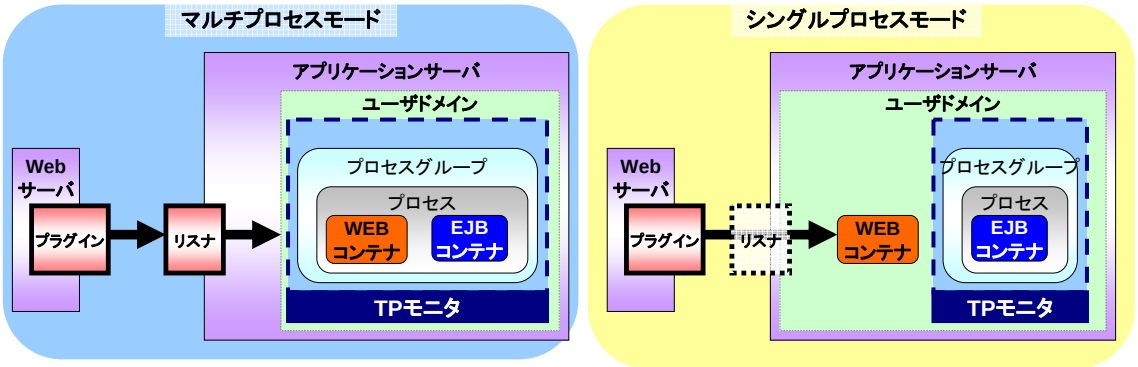


図 2.1.1. プロセスモードの差異

WebOTX は、マルチプロセスモード、シングルプロセスモードのどちらかを選択し運用を行います。

マルチ

マルチプロセスモードを選択した場合、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出来ます。
TP モニタ上で運用することにより、これらのアプリケーションに対し

- 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
- 統計情報の取得

を行うことができます。このため、信頼性、可用性が向上し、ミッションクリティカル性が向上します。
WebOTX Standard/Enterprise Edition では、マルチプロセスモードによる運用を推奨しています。

シングル

シングルプロセスモードを選択した場合、EJB コンテナ上で動作する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、リソースを節約できます。

また、シングルプロセスモードを選択した場合、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 AP サーバ間での通信速度を向上させることが出来ます。

2） WebOTX が使用するメモリについて 共通

共通

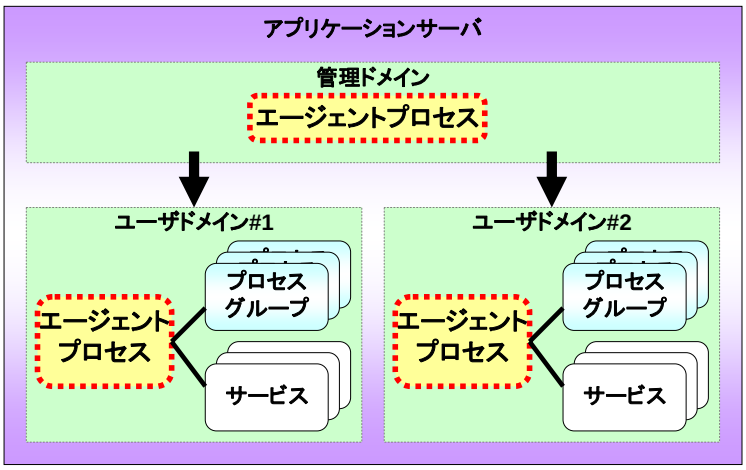


図 2.1.2.WebOTX 構成

WebOTX は管理ドメイン(WebOTXAdmin)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(WebOTXAdmin)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。インストール時、シングルドメインモードとマルチドメインモードを選択できます。

    マルチドメインモードを選択した場合、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出来ます。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出来ます。

    シングルドメインモードを選択した場合、1つのユーザドメインしか起動できません。しかし管理ドメインの起動が不要になり管理ドメインが使用するリソースを削減できます。

    管理ドメインは Java プロセスとしておよそ、64～128MByte 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、J2EE 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、主に3つの領域(ヒープ領域、パーマネント領域、C ヒープ領域)から構成されます。このうち、Java VM のメモリサイズを調整する場合、主に、ヒープ領域と、パーマネント領域を調整します。ヒープ領域は最大ヒープサイズ(-Xmx)、パーマネント領域は最大パーマネントサイズ(-XX:MaxPermSize)を Java システムプロパティで指定することで調整できます。

  - Java VM のガベージジコレクション(GC)
    Java VM はヒープの空きメモリ領域が少なくなるとガベージコレクション(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

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) プロセスモードの選択 マルチ シングル
2) WebOTX が使用するメモリサイズの設計 共通
   - シングルドメインモードとマルチドメインモードの選択
   - エージェントプロセス(Java VM)の最大ヒープサイズ(-Xmx)
   - エージェントプロセス(Java VM)の最大パーマネントサイズ(-XX:MaxPermSize)
   - GC ログの出力設定(-verbose:gc)
   - プロセスグループの最大ヒープサイズ(-Xmx)
   - プロセスグループの GC ログ出力設定 (-verbose:gc)
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) プロセスモードの選択 マルチ シングル

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

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、ガベージコレクション(GC)の頻度などを考慮した設計を行うことができます。

- シングルドメインモードとマルチドメインモードの選択
  WebOTX は、インストール時にシングルドメインモードとマルチドメインモードの選択が可能です。WebOTX は、マルチドメインモードを推奨しています。
  ユーザドメインを1つしか使用しないと確実に判断できる場合のみ、シングルドメインモードを選択してください。
  詳細は、WebOTX マニュアル「運用編 －運用管理の概要」を参照してください。
- エージェントプロセス(Java VM)の最大ヒープサイズ(-Xmx)
  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します。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 CORBA や EJB のインタフェースを指します。

統合運用管理ツール上では
<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

<EJB>
◆アプリケーション
　+ -WebOTX J2EE アプリケーション名
　　+ -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

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 のヒープ領域が必要になります。

シングルプロセスモードで動作している環境において、セッション情報を登録するアプリケーションを利用する場合、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、シングルプロセスモード、マルチプロセスモードに関わらず、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。

- エージェントプロセス(Java VM)の最大パーマネントサイズ(-XX:MaxPermSize)
  非常に多くのクラスをロードするアプリケーションの場合、パーマネント領域が足りなくなることがあります。パーマネント領域はヒープ領域とは別の領域のため、ヒープ領域に空きがあってもパーマネント領域が不足すると、OutOfMemoryError が発生することがあります。パーマネント領域が不足すると、エージェントのログファイル(/opt/WebOTX/domains/<ドメイン名> /logs/server.log)に“OutOfMemoryError:PermGen”のメッセージが出力され、配備処理失敗やエージェントプロセスが異常終了することがあります。

- GC ログの出力設定(-verbose:gc)
  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。

  以下の手順でエージェントの Java VM オプションに -verbose:gc を追加してください。

  1) WebOTX を停止します。
     /sbin/init.d/WOAgentSvc stop
  2) /opt/WebOTX/domains/<ドメイン名> /config/domain.xml の jvm-options タグに Java VM 引数を追加します。
     ～
     <jvm-options>-verbose:gc</jvm-options>
     <jvm-options>-Xmx512m</jvm-options>
     <jvm-options>-XX:MaxPermSize=128m</jvm-options>
     ～
  3) WebOTX を起動します。
     /sbin/init.d/WOAgentSvc start

  GC ログは出力先オプションを指定しない場合、<インストールディレクトリ>/domains/<ドメイン名> /logs/server.log に出力されます。

- プロセスグループの最大ヒープサイズ(-Xmx)
  業務アプリケーションが動作するプロセスグループに対して最大ヒープサイズを指定します。デフォルトでは最大ヒープサイズの指定はありません Java VM のデフォルト値（64MBtye）が適用されます。

  必要なヒープサイズはプロセスグループに登録したアプリケーションを考慮し、設定する必要があります。
  WebOTX では、最大ヒープサイズに設定できる値は Windows 32bitOS の場合、最大値が 700MByte、64bitOS の場合、使用 OS に依存しますが、事実上制限がありません。詳細は、WebOTX マニュアル「注意制限事項　－13.22 Java ヒープサイズ、メモリプールサイズ、最大同時接続数の上限」を参照してください。

また、性能を重視した設計では、最大ヒープサイズと最小ヒープサイズを揃える場合も考えられます。

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。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(/opt/WebOTX/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。

マルチプロセスモードで動作している環境において、セッション情報を登録するアプリケーションを利用する場合、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。

- プロセスグループの GC ログ出力設定（-verbose:gc)
  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
  <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。

※エージェントプロセス、プロセスグループは別々に Java VM の設定を行う必要があります。

3） 使用していないサービスを抑止する共通

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。
- RCS の起動抑止
  業務アプリケーションが Java の場合、Transaction サービスの C++ アプリケーション用 RCS の起動を抑止します。

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:全て JVM オプション(jvm-options)という属性に設定します。

設定パラメータ	説明	デフォルト値	設定値
最大ヒープサイズ -Xmx	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。	-Xmx512m	-Xmx1024m
最大パーマネントサイズ -XX:MaxPermSize	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パーマネントサイズを指定します。	-XX:MaxPermSize=128m	-XX:MaxPermSize=128m
GCログ -verbose:gc	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。	なし	-verbose:gc

プロセスグループ

設定パラメータ(属性)	説明	デフォルト値	設定値

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(Java VM の既定値)	使用するアプリケーションに合わせて設定してください
GCログ その他の引数(otherArguments)に設定する	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	-verbose:gc

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)	説明	デフォルト値	設定値
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
RCS の起動抑止 rcs-cpp-startup	C++ AP を用いてトランザクション管理を行う場合(true)は Transaction サービス起動時に C++ AP 用 RCS プロセスを自動起動します。	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） Web サーバの接続多重度
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障するために、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
レスポンスを保障するために、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 サーバの接続多重度設定 共通 マルチ シングル
Web サーバの受付リクエスト数をもとに、Web サーバの Apache 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 マルチ シングル
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。

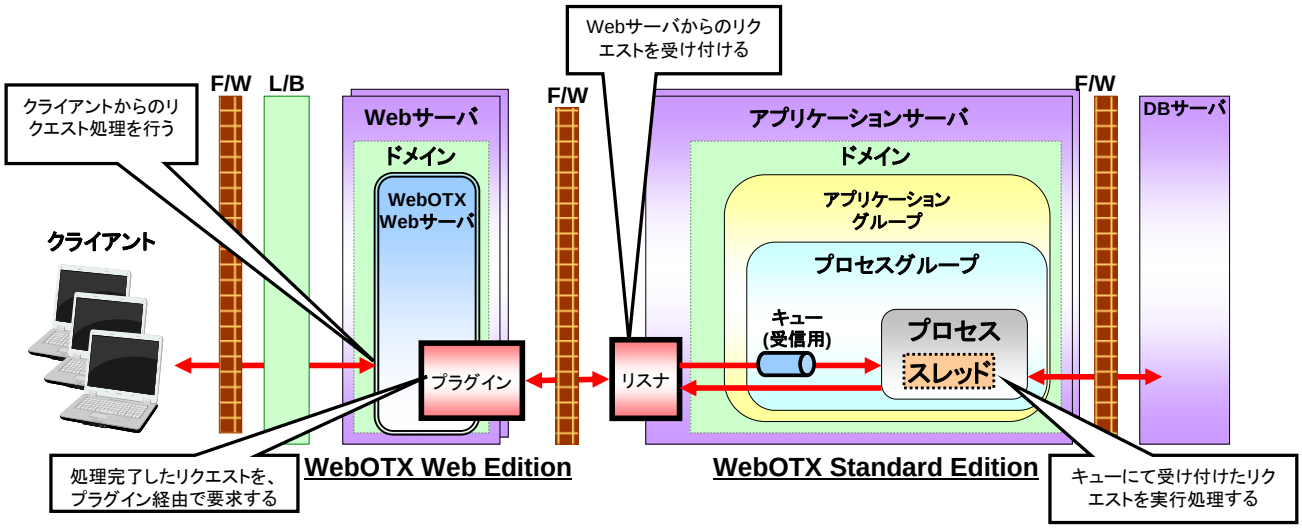


図 2.2.1.WebOTX での処理の流れ

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 マルチ シングル

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。

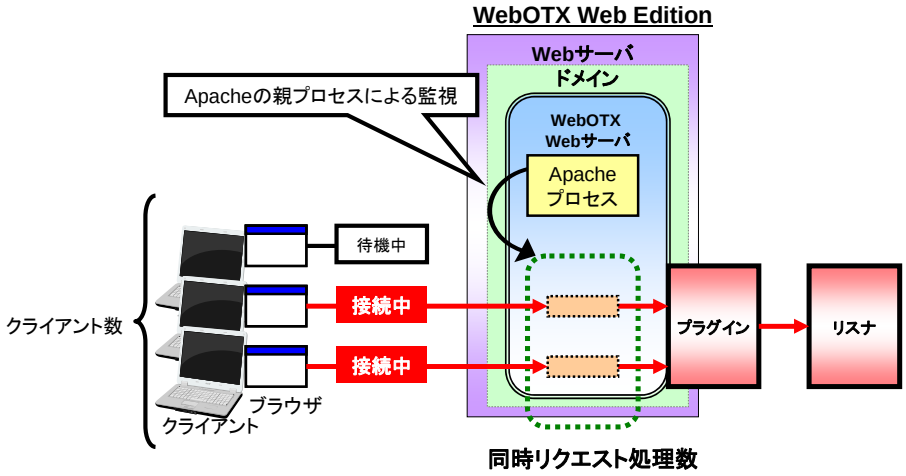


図 2.2.2.Web サーバ構成

共通

- 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。

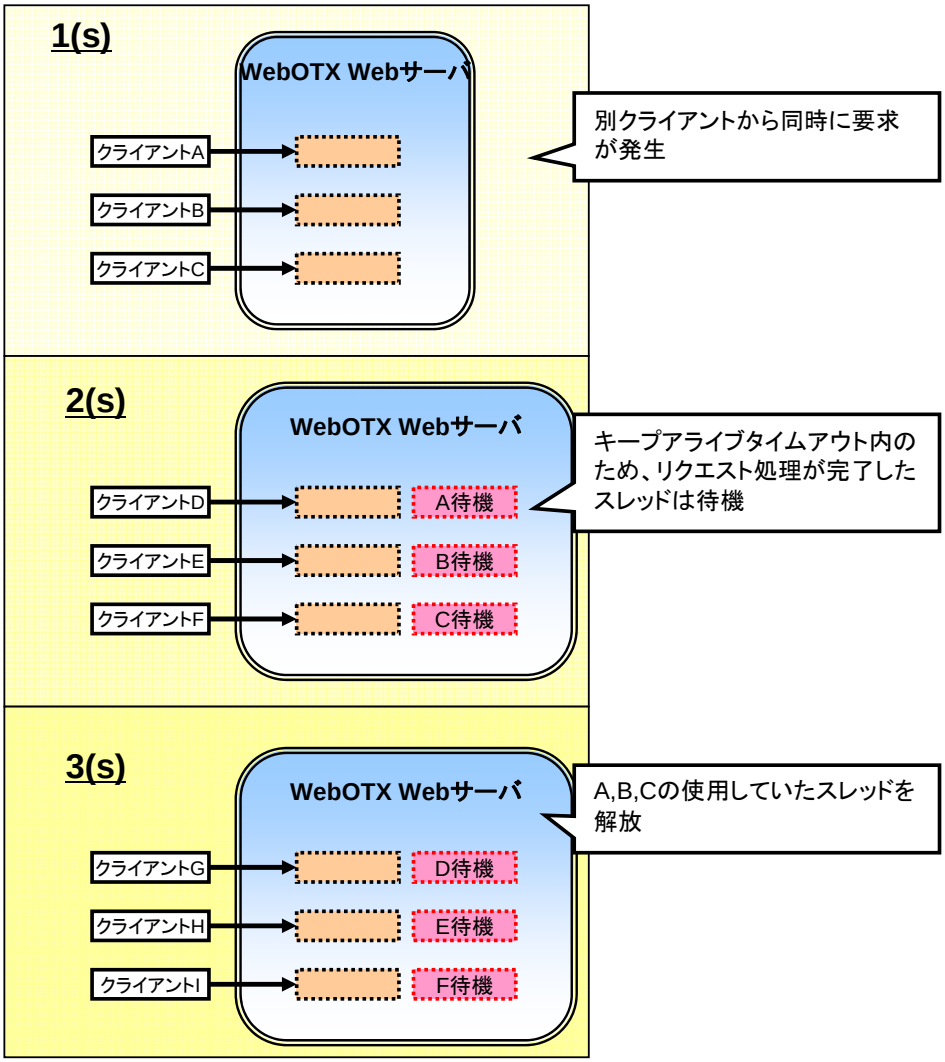
- 同時接続クライアント数
  クライアント数の内、Web サーバに同時に接続する可能性があるクライアント数です。

- 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

  Internet Explorer の場合、1クライアントからの同時リクエスト数は2になります。

  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、Apache 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 ）

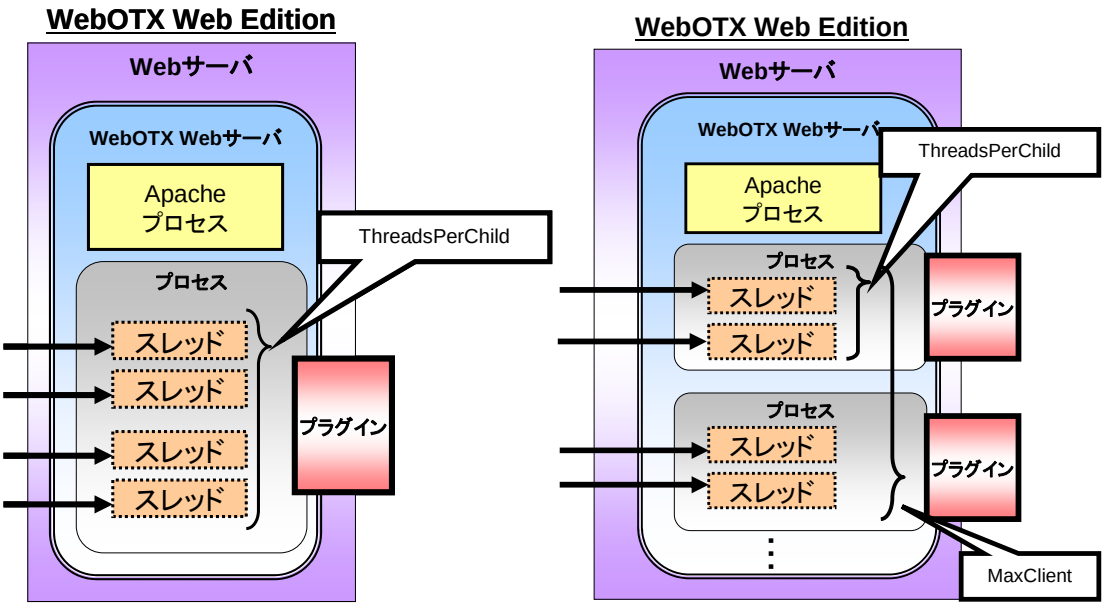
上記の算出式は単純なモデルを想定しています。使用する場合は、システム要件に合わせて変更してください。

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(MaxClient)を制限するには、Apache 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。



UNIX の場合　　Windows の場合

図 2.2.4.Web サーバ内部構成

- **Apache 2.0(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 = ThreadsPerChild × ServerLimit(1 固定) = ThreadsPerChild

- **Apache 2.0(UNIX)**

UNIX 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。

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

➢ 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。

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。TheradPerChild が ThreadLimit より小さい値であれば変更の必要はありません。

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

※Apache2.0(Windows)では MaxClient の設定は無効です。Apache2.0(Windows)において、上記計算式を用いる場合、MaxClient に最大同時リクエスト数を使用してください。

マルチ

➢ プラグインモジュールの最大リクエスト処理数

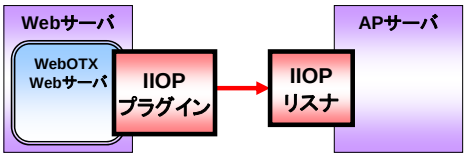


図 2.2.5 マルチプロセスモード時

IIOP プラグインとは Web サーバで受け付けたクライアントからのリクエストをアプリケーションサーバに送信するためのプラグインです。Apache の子プロセス単位に 1 つのセッションが作成されます。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、Apache 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

プラインモジュールのリクエスト処理数 = Apache の ThreadPerChild

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、Apache 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

プラインモジュールのリクエスト処理数 < Apache の ThreadPerChild

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、cachesize の値を変更します。cachesize は、ドメインディレクトリの config/WebCont/ior_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。"otxiiop" については通常固定と考えてください(プラグインの負荷分散機能を利用した場合は可変となります)。デフォルト値は 150 になっています。

worker.otxiiop.cachesize=150

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 WebOTX マニュアル「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。

シングル

- ➢ プラグインモジュールの最大リクエスト処理数

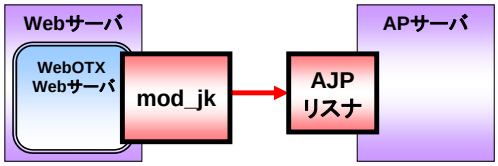


図 2.2.6 シングルプロセスモード時

シングルプロセスモードを選択した場合、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 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、Apache 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

プラインモジュールのリクエスト処理数 = Apache の ThreadPerChild

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、Apache 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

プラインモジュールのリクエスト処理数 ＜ Apache の ThreadPerChild

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、cachesize の値を 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。"worker.ajp13" については通常固定と考えてください(プラグインの負荷分散機能を利用した場合は可変となります)。デフォルト値は 150 になっています。

worker.ajp13.cachesize=150

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多重度 マルチ シングル

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。

マルチ

➢ IIOP リスナの多重度

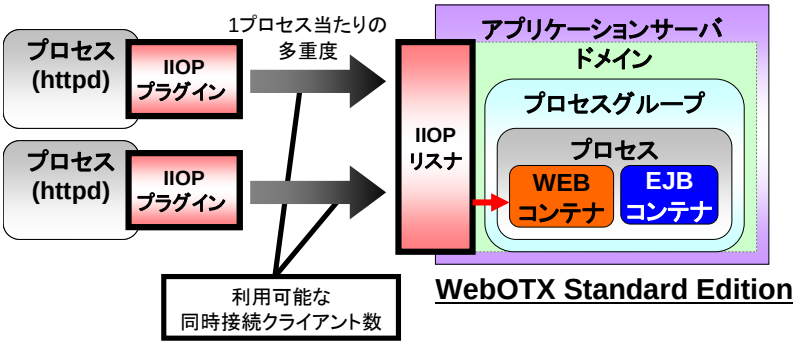


図 2.2.7.マルチプロセスモード構成

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、IIOP プラグインを経由し、AP サーバ側の IIOP 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、IIOP リスナの多重度の設定を行う必要があります。

- IIOP リスナ 1プロセス当たりの多重度
  Web サーバの同時リクエスト処理数(ThreadPerChild)と同じ数になるように設定します。

  Web サーバと接続している場合、Web サーバから1つのセッションを通して多重にリクエスト要求が発行されます。このため、クライアント数やトランザクション処理件数に応じた 1 セッション当たりの同時実行数を設定する必要があります。
  WebOTX では、IIOP リスナの**『1 プロセス当たりの多重度』**の設定で、1 セッション当たりの同時実行数を設定することができます。
  統合運用管理ツールから設定する場合は[TP システム]-[IIOP リスナ]-[クライアント制御]-[1 プロセスあたりの多重度]で設定します。

IIOP リスナ　1プロセス当たりの多重度　＝　同時リクエスト処理数(ThreadPerChild)

Web サーバが複数台ある場合、多重度の設定は個々の Web サーバに対してそれぞれ適用されます。全ての Web サーバの中で、もっともリクエストの多重度が大きい値を設定してください。

- IIOP 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]-[IIOP リスナ]-[上限設定]-[利用可能な同時接続クライアント数]で設定します。

  図 2.2.7.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(100)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。

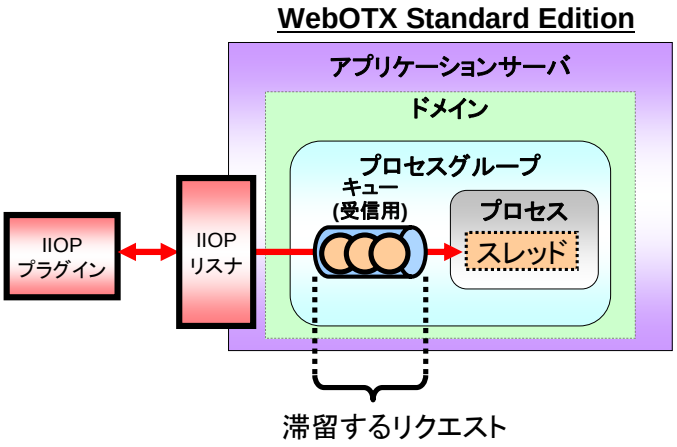


図 2.2.8.プロセスグループ構成

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多重度と、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　＝　プロセス多重度×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。

- プロセスグループのプロセス多重度

  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多重度を2以上になるように設計します。

- 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　÷　プロセス多重度

※プロセスを増やすことは、同時に Java のプロセスを生成するため、スレッド数の増加より OS のリソースを消費します。このため、プロセスの過剰な作成は避けるようにしてください。

- キュー
  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障することが可能です。

シングル

➢ WEB コンテナのプロセッサ数

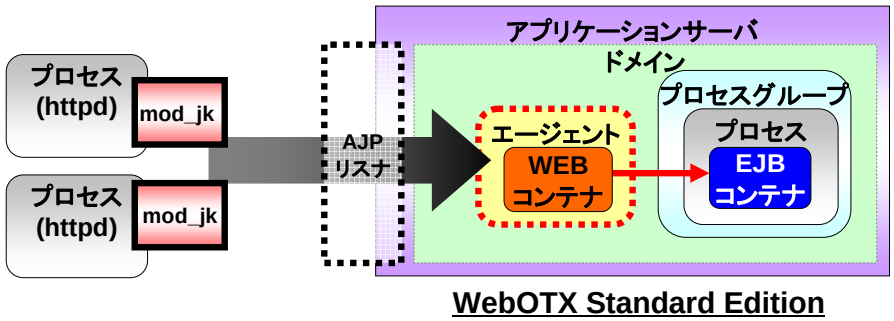


図 2.2.9.シングルプロセス構成

シングルプロセスモード時、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-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 コンテナのプロセッサ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.http-service.http-listener.ajp-listener-1.min-processors=<最小数>

最大数の設定
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.http-service.http-listener.ajp-listener-1.max-processors=<最大数>

初期は最小数で設定された値でプロセッサ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

最大プロセッサ数(max-processors)
= 同時リクエスト処理数(ThreadPerChild)×Web サーバの数(httpd プロセスの数)+ 5

となるように設定してください。「5」は内部的な管理用スレッド数です。

またプロセッサ数の設定を変更した場合、限界プロセッサ数のチューニングが必要となります。
限界プロセッサ数は処理プロセッサ数を増やしたことをログに出力させる閾値です。デフォルトは 15 であり、処理プロセッサ数が 15 以上になるとログに出力されます。

限界プロセッサ数の設定
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.http-service.http-listener.<リスナ ID>.limit-processors=<閾値>

下記条件であてまるよう設定を行ってください。
最小プロセッサ数(min-processors) < 限界プロセッサ数(limit-processors) < 最大プロセッサ数

(max-processors)

➢ プロセスグループの実行多重度

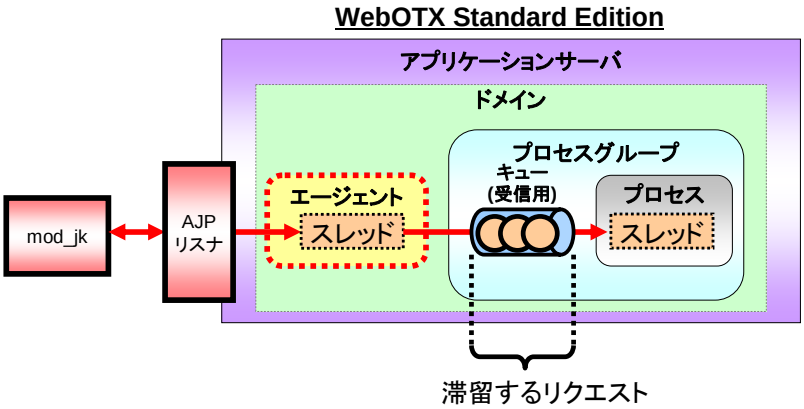


図 2.2.10.プロセスグループ構成

- プロセスグループのプロセス多重度
  シングルプロセスモード時、EJB コンテナを使用するアプリケーションに対して、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 ＝ プロセス多重度×スレッド数

- スレッド数
  WEBコンテナを使用するアプリケーション・EJBコンテナを使用するアプリケーションにより、スレッドを作成するプロセスが異なります。

  ➢ WEB コンテナを使用する場合
  プロセッサ数の値が反映されます。

  ➢ EJB コンテナを使用する場合
  EJB コンテナを使用するアプリケーションの場合、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 ÷ プロセス多重度

- キュー
  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障することが可能です。

3) データベース接続の多重度設定 共通

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)へのコネクションは 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。

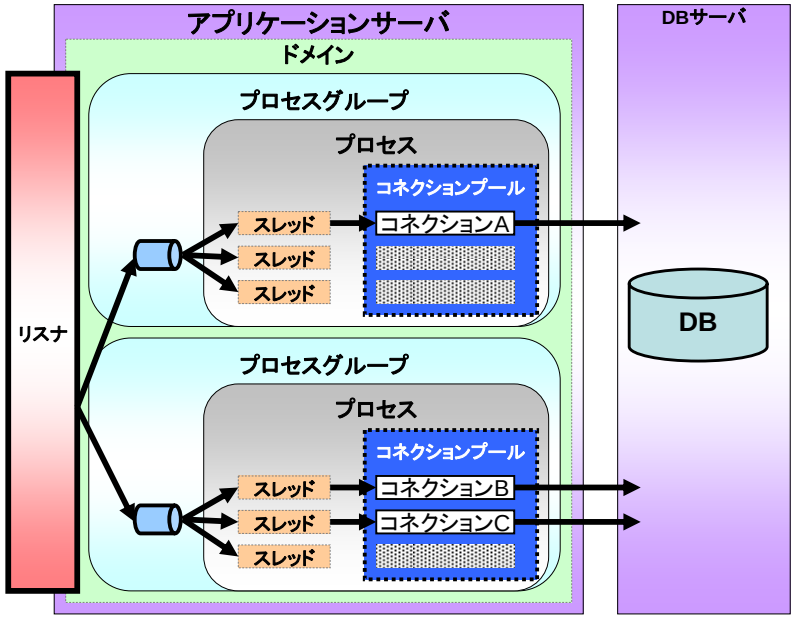


図 2.2.11.アプリケーションサーバとデータベースの関連

詳細設定、及び JDBC の設定については、2.4 DB 連携設定 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「運用編 －チューニング編」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サーバの接続多重度設計条件 共通 マルチ シングル

・ クライアント数：1000台

- 同時接続クライアント数：200台
- Web サーバ Apache 2.0 (Unix)

2） アプリケーションサーバの実行多重度設計条件 マルチ シングル

1. 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
2. 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 マルチ シングル

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 400 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。ゴーストセッションの対策については WebOTX マニュアル「運用編 － トラブルシューティング(障害解析) － 機能別 － TP モニタ (Standard/Enterprise Edition) － クライアント接続数オーバへの対応」を参照してください。

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 のデフォルト値(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 worker.c>の直下となります。

マルチ

- 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 セッションあたりの同時実行数(worker.otxiiop.cachesize) ＝ ThreadsPerChild(50) ＝ 50

シングル

- 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 セッションあたりの同時実行数(worker.ajp13.cachesize) ＝ ThreadsPerChild(50) ＝ 50

2) アプリケーションサーバの多重度設定 マルチ シングル

マルチ

- IIOP リスナの 1 セッション当たりの同時実行数(1プロセス当たりの多重度)

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。

IIOP リスナ 1 セッション当たりの同時実行数(1 プロセス当たりの多重度) = ThreadsPerChild(50) = 50

- プロセス多重度

プロセス多重度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、プロセス多重度を 2 に設定します。

プロセス多重度 ＝ 2

- スレッド数

MaxClient 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。

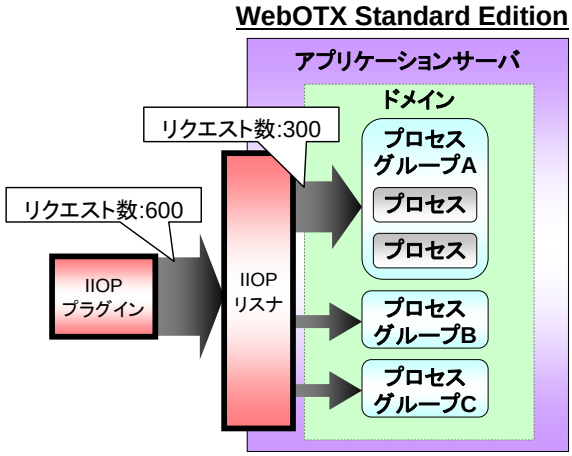


図 2.2.12.リクエスト経路

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

プロセス多重度を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) ÷ プロセス多重度(2) ＝ 10

シングル

- WEB コンテナのプロセッサ数

MaxClient 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、最小プロセッサ数を以下のように設定します。

最小プロセッサ数(min-processors) = 120 + 内部管理スレッド数(5) = 125

同時リクエスト数に応じて最小プロセッサ数(min-processors)から最大プロセッサ数(max-processors)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プロセッサ数の設定を行ってください。
TCP バックログについては、対象 OS を確認してください。

最大プロセッサ数は 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。

最大プロセッサ数(max-processors)
= リクエスト処理数(ThreadPerChild) ×Web サーバの数(httpd プロセスの数) + 内部管理スレッド数(5)
= 605

このとき、MaxClient で設定された同時リクエスト処理数 600 の内、80%の 480 リクエストを超えた場合、ログにプロセッサ数を増やしたことを表示させたいとします。
その場合、限界プロセッサ数を以下のように設定します。

限界プロセッサ数(limit-processors) = 480 + 内部管理スレッド数(5) =485

- プロセス多重度

プロセス多重度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、プロセス多重度を 2 に設定します。

プロセス多重度 ＝ 2

➢ 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 JavaEE アプリケーション(EAR) を利用した場合、リクエストは図 2.2.13.の経路を使用します。このとき、MaxClient 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。

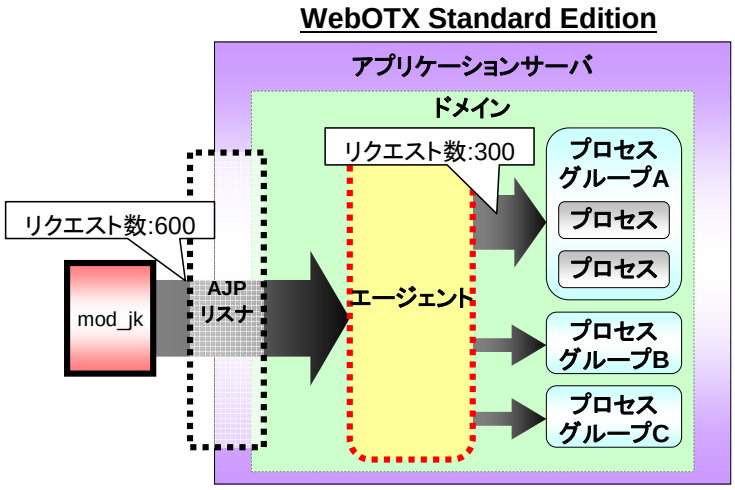


図 2.2.13.リクエスト経路

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

2.2.3.設定項目

1) Web サーバの多重度設定項目

Web 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

Windows Apache

設定パラメータ(属性)	説明	デフォルト値	設定値
最大同時接続数 ThreadsPerChild	子プロセスが作成するスレッドの総数を指定します。4096 まで指定可能です。	50 (1.3) 250 (2.0)	クライアント数*ブラウザ設定リクエスト数
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。	0	0

共通

UNIX Apache2.0

設定パラメータ(属性)	説明	デフォルト値	設定値
最大同時接続数 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。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。	150	クライアント数*ブラウザ設定リクエスト数
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。	16	16

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。15000 まで指定可能です。	64	64
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。	25	25
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。	2	2
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。	25	25
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。	75	75
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。	0	0

マルチ

IIOP プラグイン

設定パラメータ(属性)	説明	デフォルト値	設定値
プラグインモジュールの最大リクエスト処理数 otxiiop.cachesize	IIOP プラグインの同時実行数を指定します。	150	Web サーバの ThreadsPerChild に応じて値を設定してください。

シングル

mod_jk

設定パラメータ(属性)	説明	デフォルト値	設定値
プラグインモジュールの最大リクエスト処理数 ajp13.cachesize	mod_jk の同時実行数を指定します。	150	Web サーバの ThreadsPerChild に応じて値を設定してください。

2) アプリケーションサーバの多重度設定項目

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

設定パラメータ(属性)	説明	デフォルト値	設定値
スレッド数 threadCount	スレッド数を指定します	マルチプロセスモード:3 シングルプロセスモード:1	システム要件にしたがってください。
プロセス数 processCount	プロセス数を指定します	1	システム要件に従ってください。

マルチ

IIOP リスナ

設定パラメータ(属性)	説明	デフォルト値	設定値
同時接続クライアント数 simultaneousConnectionClients	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 WebOTX Application Server に接続する場合に設定してください。	100	接続クライアント数以上
1 プロセス当たりの多重度 multiplexprocess	1 プロセス当たりのリクエスト処理同時実行多重度を指定します	128	IIOP プラグインのコネクションプールサイズに応じて値を設定してくださ

			い。

シングル

AJP リスナ

設定パラメータ(属性)	説明	デフォルト値	設定値
最小プロセッサ数 min-processors	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。	5	最大プロセッサ数以下
最大プロセッサ数 max-processors	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。	20	接続クライアント数以上
限界プロセッサ数 limit-processors	Web コンテナは、アクティブなリクエスト受け付けプロセッサの数が最大数に近づいた時に運用管理コンソールに警告を通知します。その閾値を設定します。	15	最小プロセッサ数以上 最大プロセッサ数以下

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。

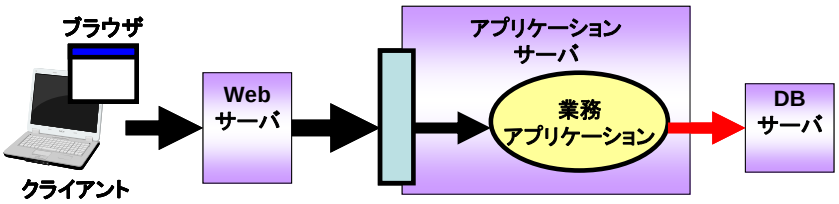


図 2.3.1.通信経路

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） Web サーバで設定するタイムアウト値 共通 マルチ シングル

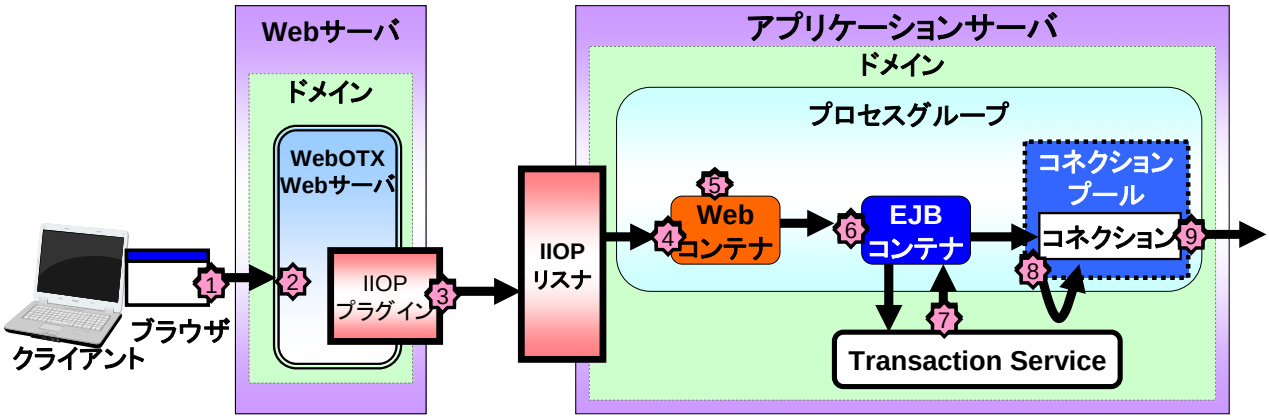
- キープアライブタイムアウト
- IIOP メソッド呼び出しタイムアウト

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

- 実行時間タイムアウト
- 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。

マルチ



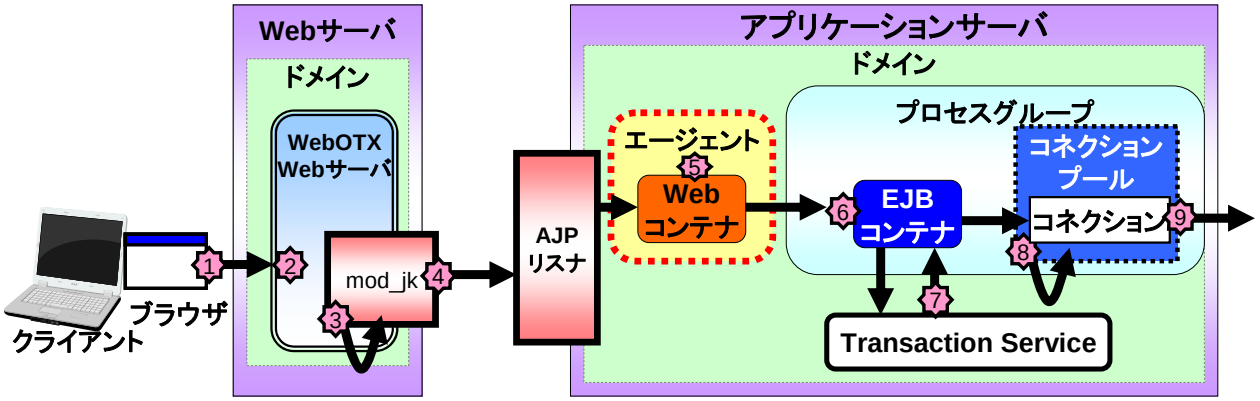
①クライアント(Web ブラウザ)タイムアウト

②キープアライブタイムアウト　③IIOP メソッド呼び出しタイムアウト

④実行タイムアウト　⑤セッションタイムアウト　⑥実行タイムアウト・CORBA 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

シングル



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　⑥実行タイムアウト・CORBA 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

図 2.3.2.Web サーバとアプリケーションの通信経路

1)　クライアントで設定するタイムアウト値 共通

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。
  Internet Explorer を使用した場合、レジストリキーに対し ReceiveTimeout という DWORD 値を追加して、値のデータを <秒数>*1000 に設定します。Internet Explorer 7 の場合、タイムアウト時間は 30 秒です。

2） Web サーバで設定するタイムアウト値共通マルチシングル

共通

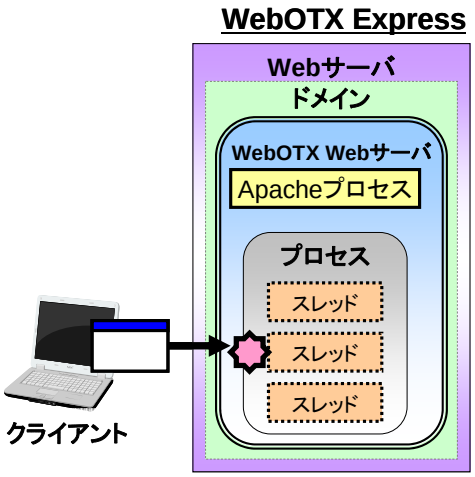


図 2.3.3.クライアント Web サーバの通信経路

- キープアライブタイムアウト(KeepAliveTimeout)

  Apache 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。

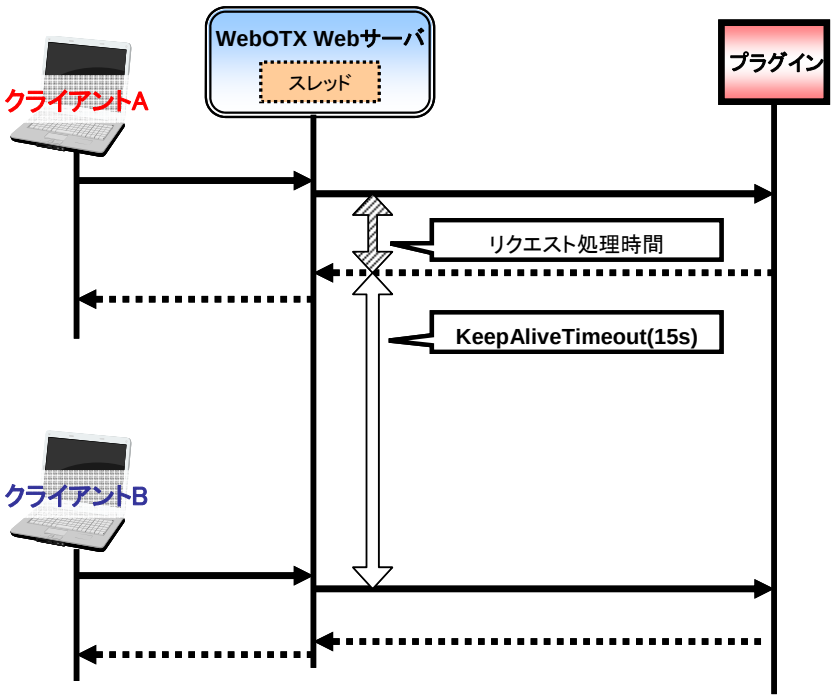


図 2.3.4.キープアライブタイムアウト

マルチ

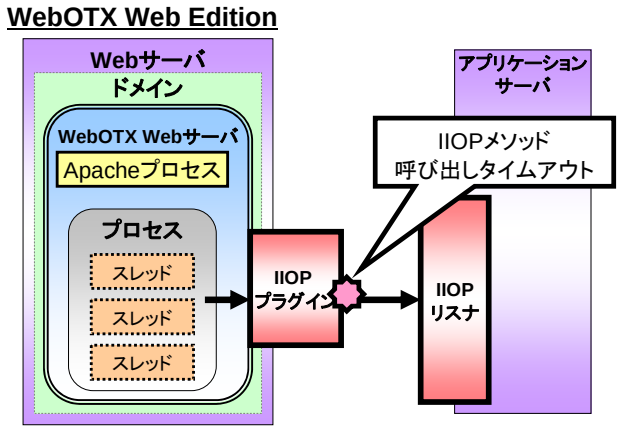


図 2.3.5.Web サーバとアプリケーションの通信経路

- IIOP メソッド呼び出しタイムアウト(iiop_timeout)
  IIOP プラグインからアプリケーションサーバを呼び出す際、設定されるタイムアウト値です。
  リトライ回数(iiop_retry_count)と、リトライインターバル時間(iiop_retry_interval)と関連性があります

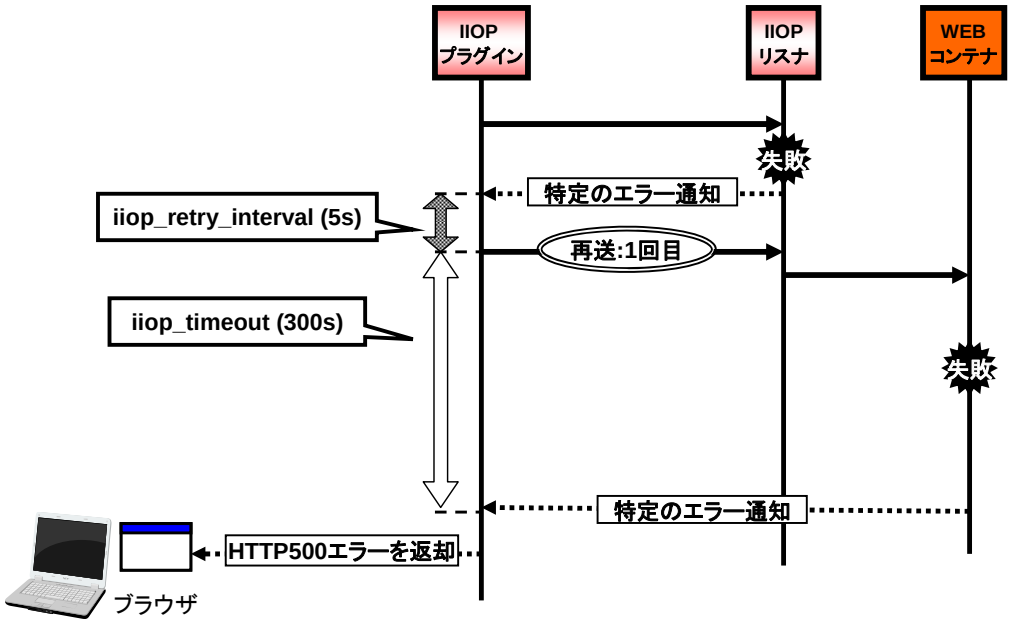


図 2.3.6.IIOP メソッド呼び出しタイムアウト

デフォルトでは iiop_timeout=300 秒、iiop_retry_interval=5 秒　iiop_retry_count=3 回の設定が行なわれています。
下記に示した特定の通信エラーが発生した場合のみ、iiop_timeout を待たず、iiop_retry_interval で指定された時間待機した後、iiop_retry_count 分、再送を行います。

1. CORBA::NO_RESPONSE または CORBA::COMM_FAILURE
2. CORBA::INV_OBJREF
3. CORBA::TRANSIENT
4. CORBA::NO_RESOURCES

詳細は、WebOTX マニュアル「メッセージ編 - CORBA - 2. 例外のマイナーコード一覧 - 2.3.WebOTX Object Broker C++」及び「メッセージ編 - CORBA - 3. Standard Edition、Enterprise Edition の CORBA 例外」を参照してください。

また、iiop_timeout は、WEB コンテナにて処理に時間がかかる等の理由で発生します。この場合、リクエストの再送は行ないません。リクエストの送信が失敗した場合、最終的にブラウザに HTTP 500 エラーが返却されます。

シングル

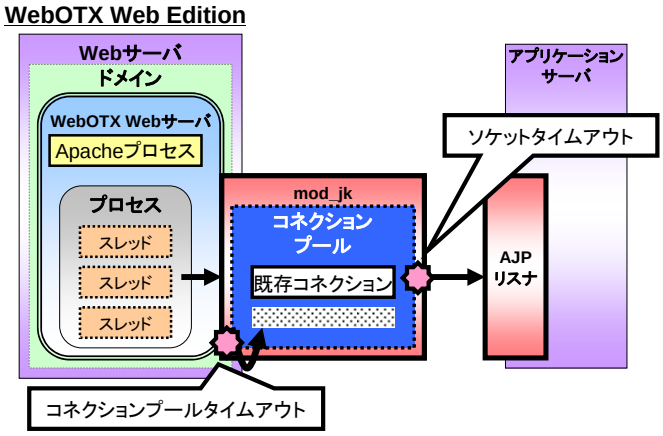


図 2.3.7.Web サーバとアプリケーションの通信経路

- 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。
  デフォルト値は 0(タイムアウト無し)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。

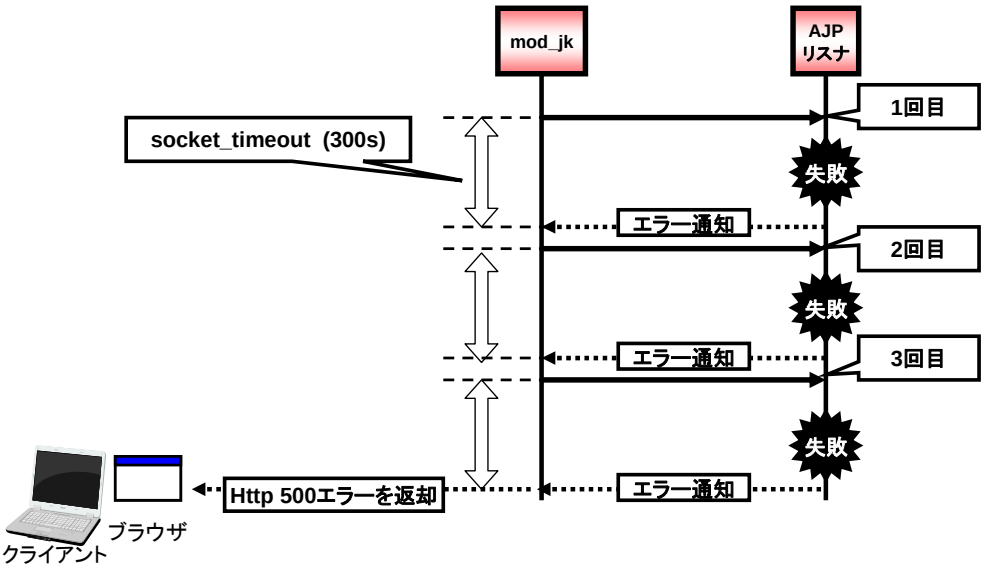


図 2.3.8 ソケットタイムアウト

デフォルト値では socket_timeout=0 秒(タイムアウト無し)、retries=3 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

マルチ

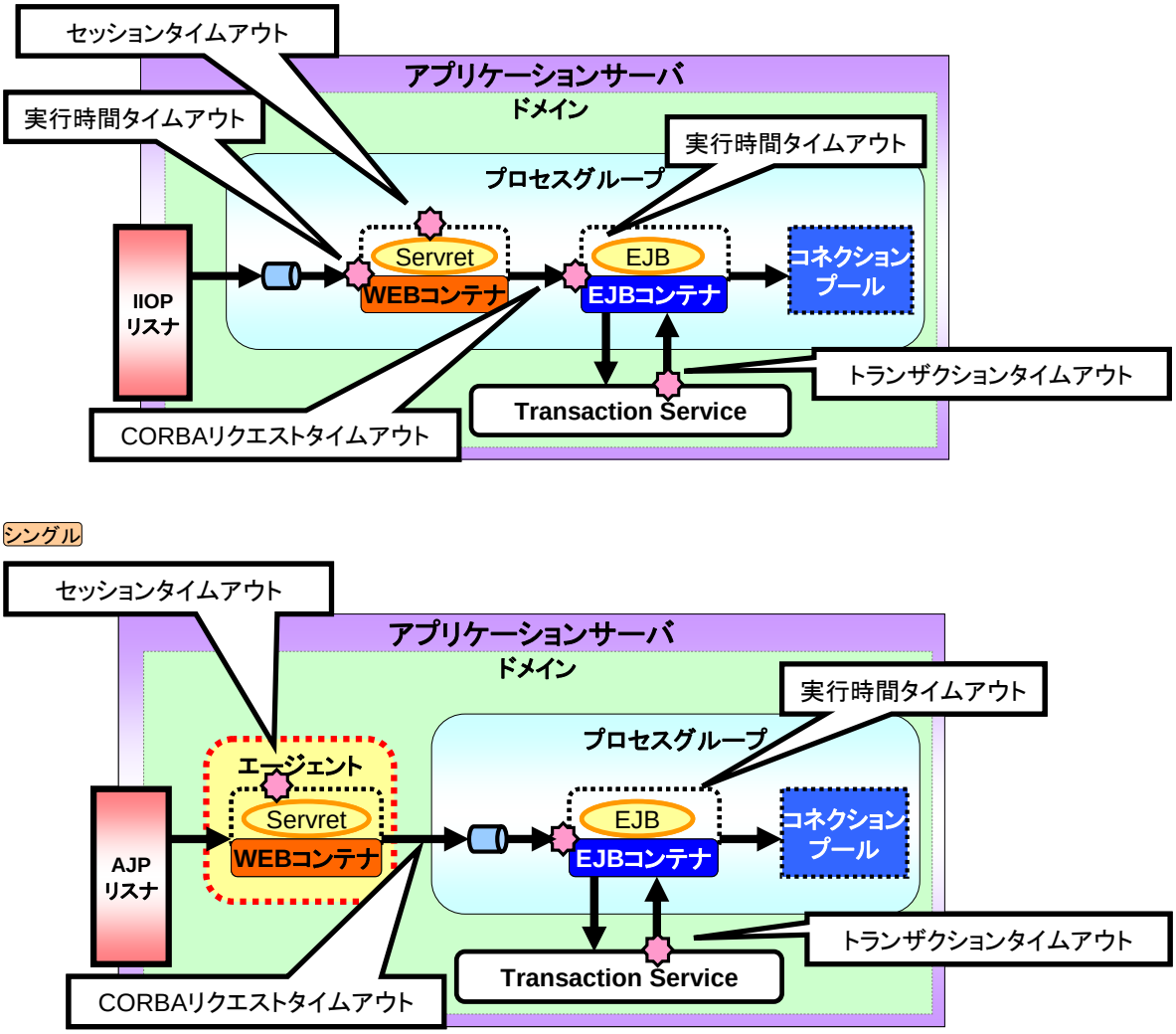


図 2.3.9.アプリケーションサーバ内部構成

共通

- 実行時間タイムアウト(実行時間上限)
  サーバに配備されたサーバアプリケーションのオペレーションに対して実行時間の上限を設定します。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、オペレーション処理を中断します。デフォルトではタイムアウト値を設定していません。

- 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**）
  マルチプロセスモードで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配備されている場合、または、シングルプロセスモードの場合に、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、<WebOTX インストールディレクトリ>/domains/<ドメイン名>/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。

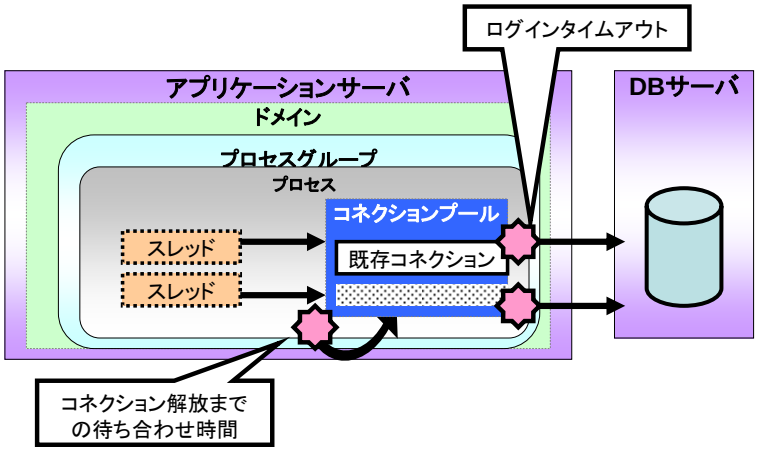


図 2.3.10.JDBC コネクションタイムアウト

- 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
     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。

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 マルチ シングル

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

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 マルチ シングル

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除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。

マルチ

- IIOP メソッド呼び出しタイムアウト(iiop_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。

シングル

- 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
  通常は変更する必要はありません。(socket_timeout)
  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。

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

共通

- 実行時間タイムアウト

  アプリケーションの CPU ループやデッドロック、データベースサーバからの無応答によるストール障害に対して、アプリケーションが使用していたリソースを解放します。アプリケーションの各オペレーション単位に設定することができます。

  タイムアウトが発生すると該当するプロセスの再起動が行われます。プロセス上の他のスレッドで実行中のアプリケーションについては処理の完了を待ち合わせます。(最大 10 分)

  デフォルトではタイムアウトは行われません。

  設計ポリシーとしていずかを選択します。

  - 実行時間タイムアウトを個々のオペレーション単位に設定する

  - システムで一括して実行時間タイムアウト上限を設定する

    全アプリケーションで共通的に実行時間上限を決定します。オペレーション数が大きい場合、個々に設定するのは非常に大変です。そのため、全アプリケーションに対して共通の実行時間上限を設定することもできます。
    全アプリケーションに一括で設定するには、以下のコマンドを実行してください。「*」は、任意の文字列を表しています。

    otxadmin> set applications.*.exetimeMax=＜実行時間タイムアウト値＞

  実行時間タイムアウト時間を適切に設定するために、運用アシスタントの実行時間上限の適正値算出機能を利用し、配備時に設定を行ってください。詳細は、WebOTX「運用編 TP モニタの運用操作 2.40.4 実行時間上限の適正値算出」を参照してください。
  また、再配備時に、設定の初期化が行われるため、再度適正値を設定してください。

- CORB 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」も参照して下さい。

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、タイムアウトは設定されていないと見なされます。
   - 設計指針
     使用する DB に合わせて、必要であれば設定して下さい。
2. コネクション解放までのタイムアウト
   - 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。DB アクセスを考慮して設定してください。また、詳しい WebOTX マニュアル「運用編 －チューニング －8.JDBC データソースのチューニング」も参照してください。

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

設定パラメータ(属性)	レジストリキー	説明	デフォルト値	設定値
Web ブラウザタイムアウト設定 (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)ます。	IE5.x、6 の場合:60 分 IE 7 の場合:30 秒	使用するブラウザに従います。

2） Web サーバの設定項目

共通

設定パラメータ(属性名)	説明	既定値	推奨値
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。	15 秒	要件にしたがって設定する必要があります。

マルチ

設定パラメータ(属性)	説明	デフォルト値	設定値
iiop_timeout	IIOP プラグインのメソッド呼び出しを行う際のタイムアウト値を設定します	300 秒	300 秒

シングル

設定パラメータ(属性)	説明	デフォルト値	設定値
connection_pool_timeout	Web サーバから Web コンテナへのコネクションを	0 秒(設定なし)	0 秒(設定なし)

	保持する時間を変更する場合、設定を変更します。		
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を指定します。	0 秒(設定なし)	0 秒(設定なし)

3) アプリケーションサーバの設定項目

共通

設定パラメータ(属性)	説明	デフォルト値	設定値
実行時間上限	オペレーションの実行時間に上限を設定する場合に設定します。実行時間が設定時間(秒)を過ぎてもレスポンスが返却されない場合オペレーション処理を中断します。	－1(制限なし)	要件にしたがって設定する必要があります。
リクエスト呼び出しのタイムアウト時間	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。	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。	30 秒
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。	600 秒	600 秒
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。	30min	業務アプリケーションごとに設定してください
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。	0 秒(設定なし)	0 秒(設定なし)
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。	15 秒	15 秒

2.4.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、マルチプロセスモード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） マルチプロセスモードを利用する場合のセッションレプリケーション マルチ

4） セッション管理 共通

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、マルチプロセスモードを利用する場合や、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、マルチプロセスモードの場合や、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、マルチプロセスモードの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。

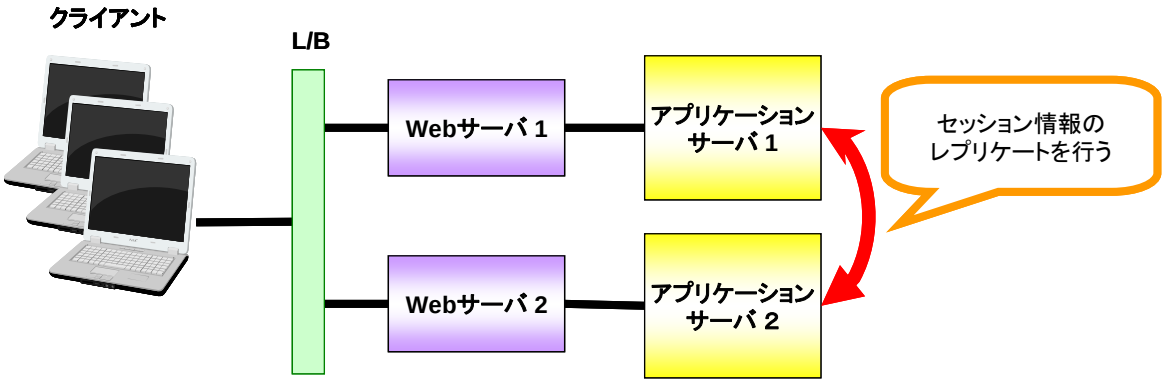


図 2.4.1.セッションレプリケーション(複数のアプリケーションサーバの場合)

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。
- マルチプロセスモードを利用する場合
  マルチプロセスモードを利用する場合、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)マルチプロセスモード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。

5) マルチプロセスモードを利用する場合のセッションレプリケーション マルチ

マルチ

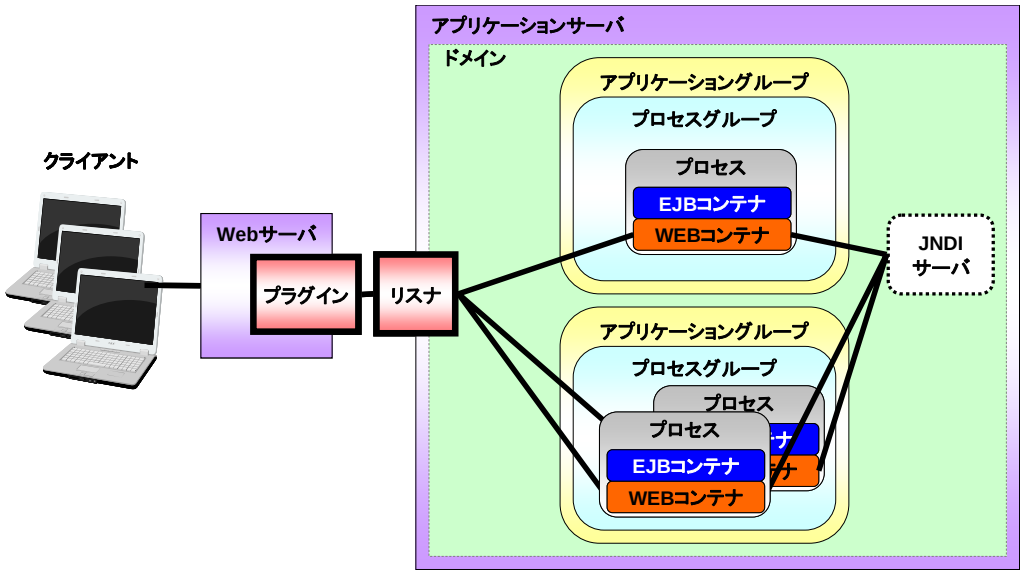


図 2.4.2.セッションレプリケーション(マルチプロセスモードの場合)

マルチプロセスモードを利用した場合、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

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するようにしてください。

- 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がかかるかにより、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、デフォルト値の 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で、マルチプロセスモード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。以下に、複数のアプリケーションサーバでセッションレプリケーションを実施するときの、JNDI サーバに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について説明します。

  - JNDI サーバに対し、セッション情報が登録・削除される場合

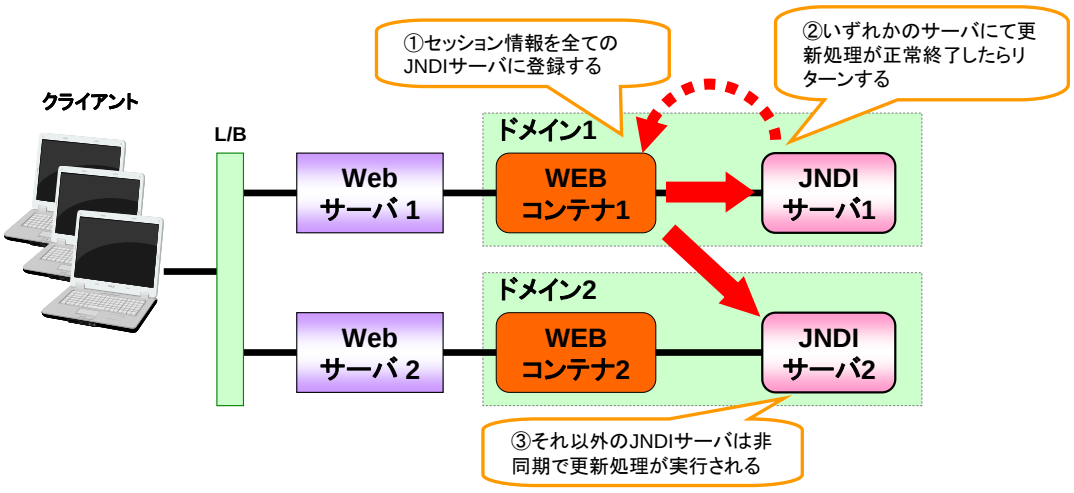
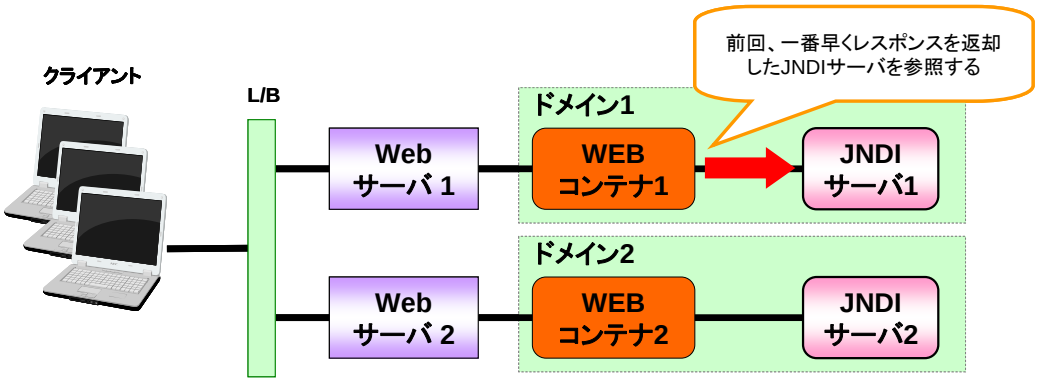


図 2.4.3.セッション情報の登録

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 JNDI 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つけます。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、処理時間がかかります。また、処理の量も多くなるため、多くのリソースを使用します。そのため、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) マルチプロセスモードを利用する場合のセッションレプリケーション マルチ

マルチ

マルチプロセスモードでは、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

共通

web.xml

設定パラメータ(属性)	説明	デフォルト値	設定値
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)	説明	デフォルト値	設定値
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。	なし	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。

システムプロパティ

設定パラメータ(属性)	説明	デフォルト値	設定値
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。	なし	10

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

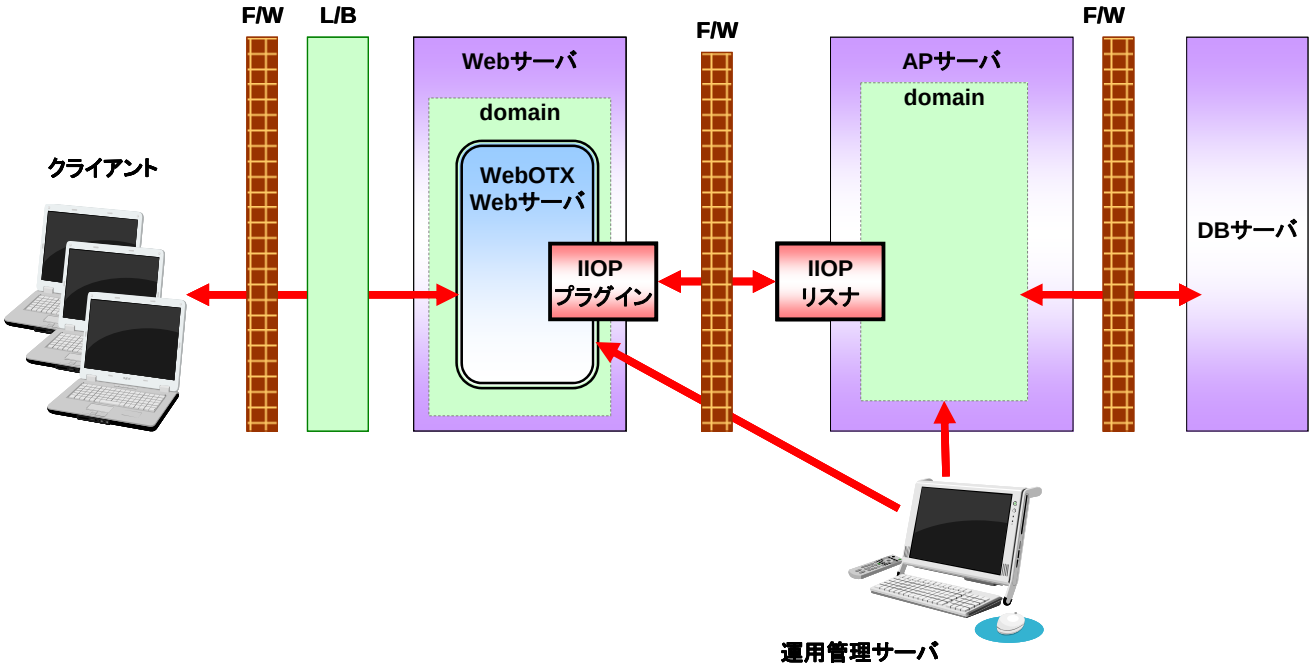


図 2.5.1.全体構成

6） 通信時のセキュリティを強化する 共通

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、SSL2.0/3.0、TLS1.0 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。

1） その他のセキュリティ設定 共通

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デフォルト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 シングル

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	デフォルト値
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 通信用ポート	SSL 用のポート番号です。	443

  - Web サーバ - アプリケーションサーバ間

	説明	デフォルト値
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。	5751
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
IIOP リスナアライブチェックポート	IIOP リスナのアライブチェックを行うためのポートです。「5220+TP モニタのシステム ID」の値となります。	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	デフォルト値
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 JMXMP 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 JMXMP 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	4848

  - アプリケーションサーバ – DB サーバ間

	説明	デフォルト値
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。

- Web サーバのディレクトリリスティング機能無効化
- 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
- Web サーバのマニュアル/コンテキストの削除

シングル

② ファイアーウォールの設定
シングルプロセスモード選択時、共通に加え、次のような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	デフォルト値
AJP リスナ通信ポート	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。	8009

2) その他のセキュリティ設定 共通

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デフォルト値となっています。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 シングル

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)	説明	デフォルト値
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。	6202
ユーザドメイン運用管理ポート番号 domain.admin.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。	6212

Web コンテナ

設定パラメータ(属性)	説明	デフォルト値
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 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。	4848

IIOP リスナ

設定パラメータ(属性)	説明	デフォルト値
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。	5751
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
IIOP リスナアライブチェックポート番号 /etc/services ファイル webotx-mess	IIOP リスナのアライブチェックを行うためのポート番号を指定します。「5220+TP モニタのシステム ID」の値となります。	5220

JDBC データソース

設定パラメータ(属性)	説明	デフォルト値
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)	説明	デフォルト値
SSL 通信の利用 security-enabled	SSL 通信を利用するかを指定します。	false
SSL 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(Web サーバの設定)

設定パラメータ	説明	デフォルト値	設定値
Directory ディレクティブ	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に含める情報を指定します。ProductOnly とすると、Apache の名前のみをヘッダに含めます。	Full	ProductOnly
Directory ディレクティブ	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/We bServer2/manual "> 略 </Directory>	# AliasMatch 略 # <Directory "/opt/WebOTX/We bServer2/manual "> # 略 # </Directory>

default-web.xml(Web コンテナの設定)

設定パラメータ	説明	デフォルト値	設定値
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。	false	<servlet> <init-param> <param-name>listings</param-name> <param-value>false</param-value> </init-param> </servlet>

シングル

AJP リスナ

設定パラメータ(属性)	説明	デフォルト値
AJP リスナ通信ポート番号 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。	8009

2) その他のセキュリティ設定 共通

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)	説明	デフォルト値
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

ディレクトリリスティング無効化では、以下の内容を決定します。
default-web.xml

設定パラメータ	説明	デフォルト値
<servlet> <init-param> <param-name>listings</param-name> <param-value>false</param-value> </init-param> </servlet>	default-web.xml 内に、パラメータを追記してください。	false

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。

データベース連携とは、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。

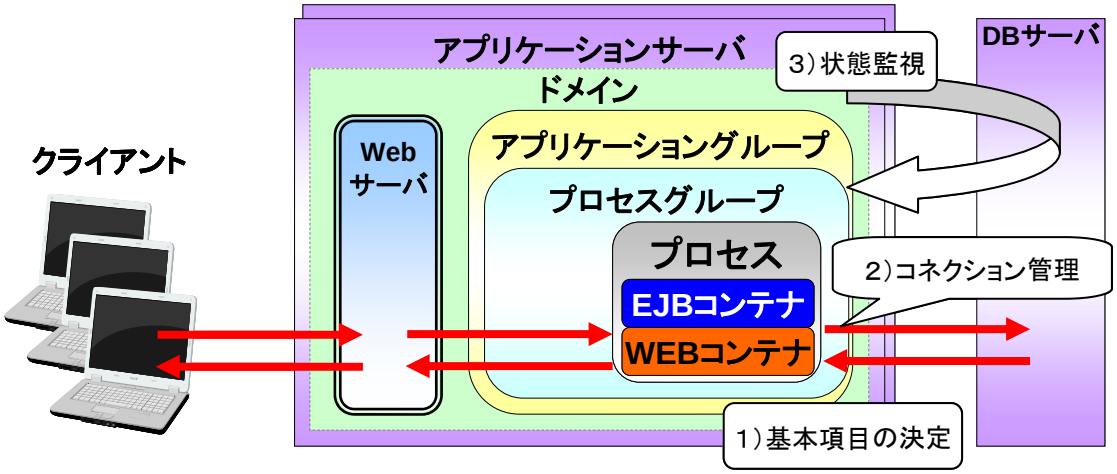


図 2.6.1 データベース連携処理概略図

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。

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。また、この章の内容はプロセスの動作モードに関わらず全て共通の内容となります。

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

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とは、既知の一貫した状態のデータベース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することです。

しかし、

① 複数のデータベースを使用するか

② 1つのトランザクションにおいて、複数のデータベースに対して更新を行うことがあるか

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)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する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)の検討をし、検討結果に合わせて、データソースタイプを選択します。

2) コネクション管理

WebOTX ではデータベースとの接続は逐次接続するのではなく、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、データベースにアクセスする際に既に接続されたコネクションを使用することで、アプリケーションは、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

WebOTX ではデータベースの状態監視機能が用意されております。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なおすことも可能です。

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
2) コネクション管理
3) 状態監視

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

➢ 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。以下のフローを参考に、データソースの種別を決定してください。

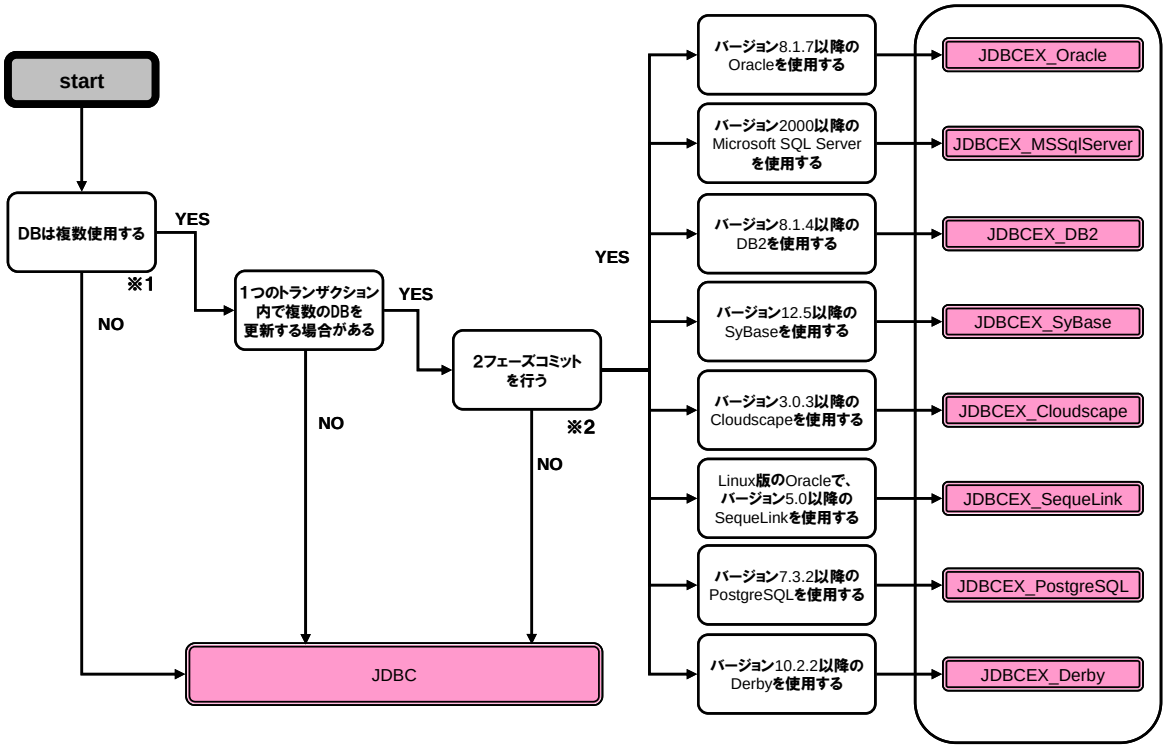


図 2.6.2.データソース種別決定フロー

※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、使用する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。

- 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_DB2	DB2 のデータベース名	”DB2001”
JDBCEX_SyBase	SyBase のデータベース名	”SyBase 001”
JDBCEX_Cloudscape	Cloudscape のデータベース名	“Cloud001”
JDBCEX_SequeLink	SequeLink サーバ側の JDBC データソース名	デフォルトは”Default”
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

  JDBC ドライバのメジャーバージョンを指定します。

  JDBC3.0、4.0 を使用する必要がないのであれば、2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

JDBC 仕様のマイナーバージョンは特に決める必要はありません。

- 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と同じもの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と同じものを設定してください。出来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ときに、JNDI に登録した javax.sql.XXXX オブジェクトを取得します。そのため、JNDI サーバへ接続情報を設定した javax.sql.XXXX オブジェクトを登録します。

  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 で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

Standard/Enterprise Edition では、EJB 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 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。(シングルプロセスモードでは、Web コンテナはエージェント上で動作しますが、そのコンテナの動作している 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 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出来ます。

- リソースのプロセス単位のロード設定

  全ての EJB 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。

  デフォルト有効になっていますが、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。また、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。統合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リスト」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)では、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 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低くなります。

2) コネクション管理

WebOTX ではデータベースとの接続は逐次接続するのではなく、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出来ます。

DB サーバとコネクションを接続する際、プロセス多重度×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。

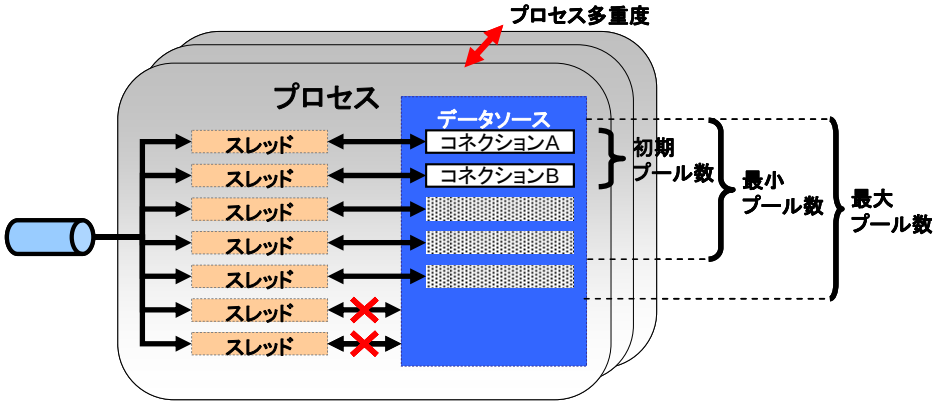


図 2.6.3.スレッドとコネクションの関係

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。

また、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×プロセス多重度で決定されます。

したがって、図のようなプロセス多重度が3の場合は

最大コネクションプール数(5) × プロセス多重度(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。また、データベース障害から復旧時に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、初期プールサイズ分のコネクションを作成します。

  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。データベースにアクセスするスレッド数+αを指定します。αには、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、プロセス毎に+1 にしてください。

また、プールサイズは以下の関係を満たすように設定してください。

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 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、プールしてある異常なコネクションを利用してデータベースに接続するのを防ぐため、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、データベースが復旧した場合、コネクションのプールが行われます。

- データベース正常時

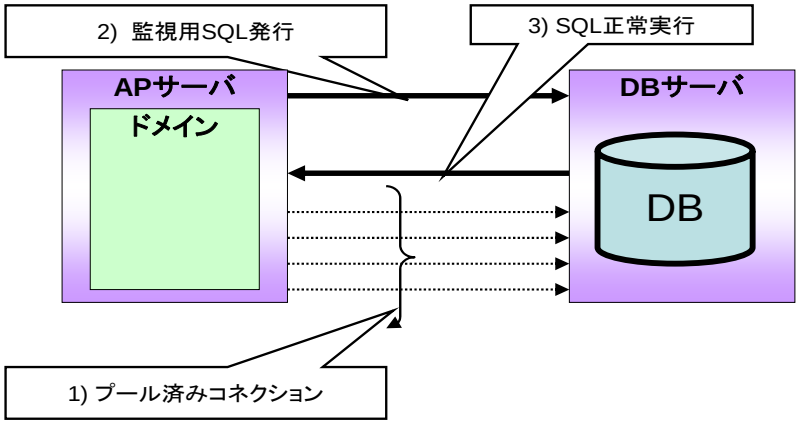


図 2.6.4.状態監視(データベース正常時)

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

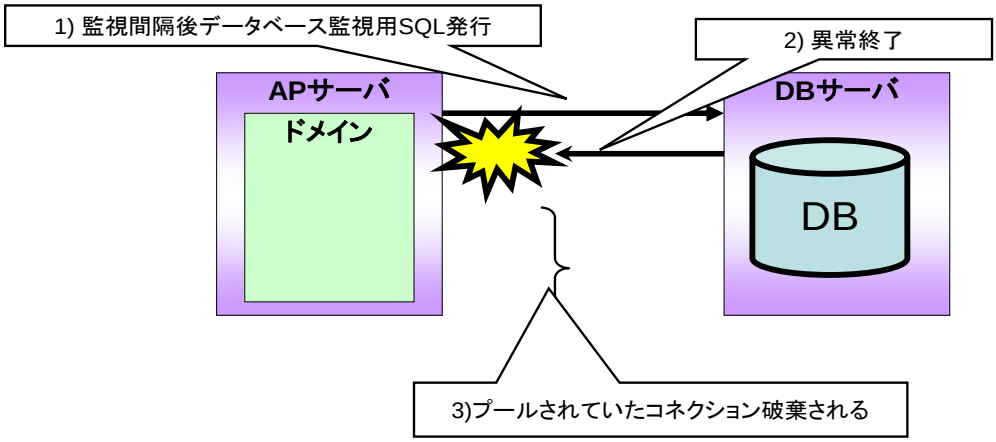


図 2.6.5.状態監視(データベース異常時)

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 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、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。

- データベース復旧時

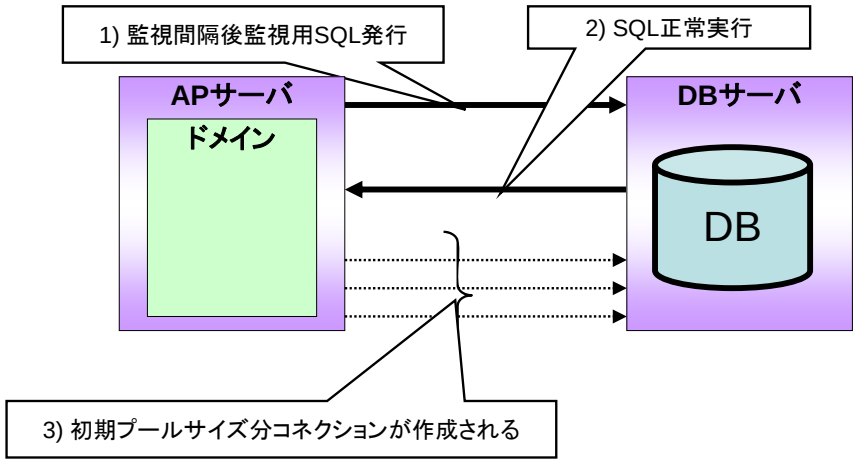


図 2.6.6.状態監視(データベース復旧時)

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

データベース監視 SQL 発行前に、業務アプリケーション経由でデータベースへの接続に成功した場合は、監視用 SQL の実行に関係なく、コネクションプールの再作成が行われます。

- 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。
- データベースサーバの状態監視間隔

  SQL 発行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

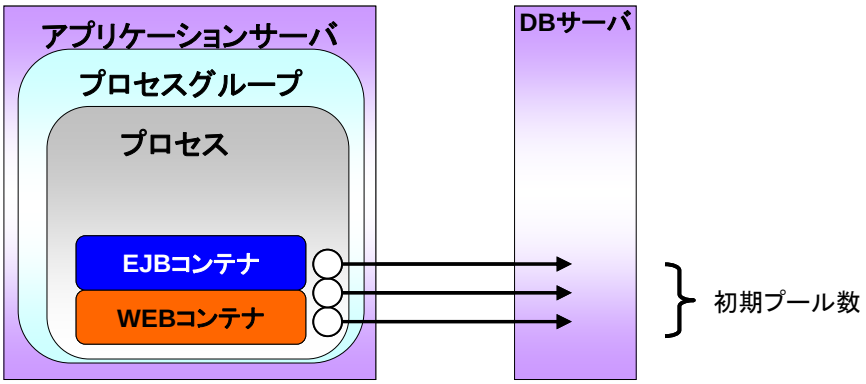


図 2.6.7.ドメイン起動時、またはプロセスグループ起動時

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、接続リトライが行われます。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、初期プールサイズが作成されるまで永続的にリトライが行われます。

- コネクション要求時

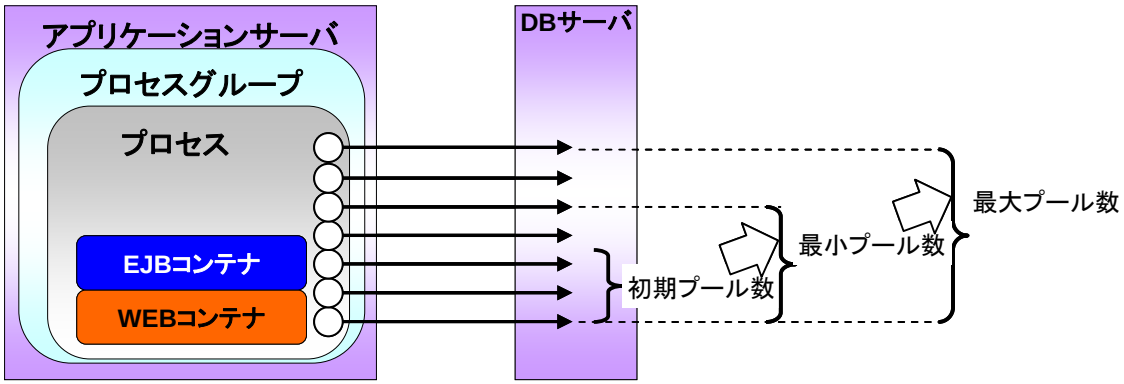


図 2.6.8.コネクション要求時

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、エラーを返します。状態監視を行う場合、接続さ

れたコネクションのうち1本を状態監視用として使用します。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と、コネクション取得に失敗した場合はリトライを行います。

- プロセス終了時

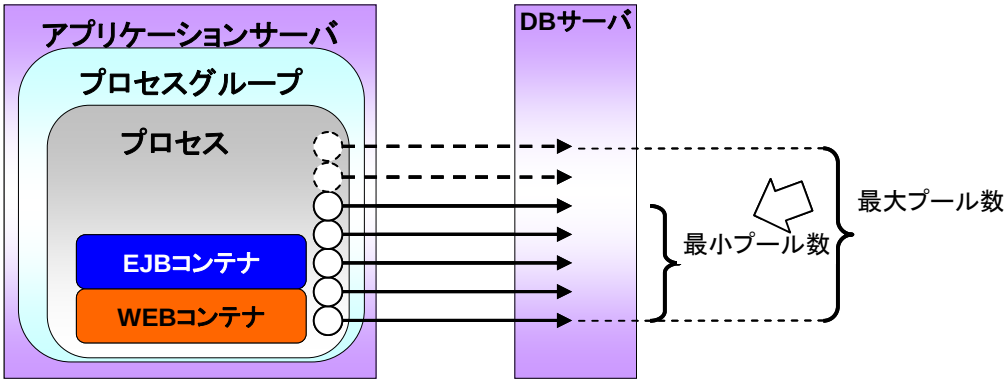


図 2.6.9.プロセス終了時

プロセス終了時、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)

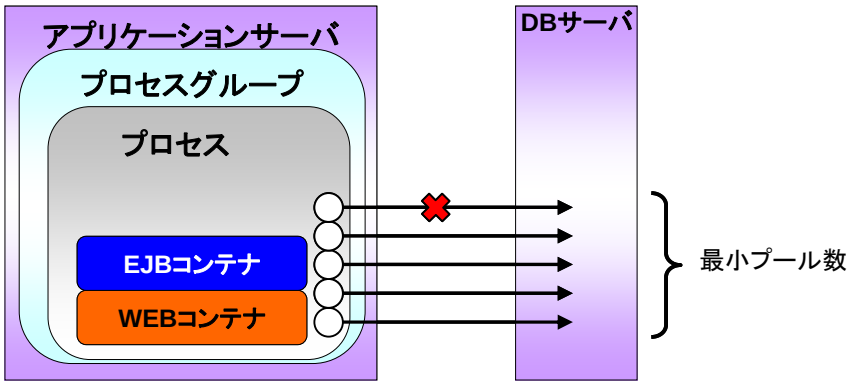


図 2.6.10.データベース異常時(状態監視なし)

データベース異常時には、対象のコネクションは破棄されます。

データベース異常の検知は、業務で実行されるコマンドが、SQLException を返却した場合です。

コネクションの一括破棄可否を有効にすると、データベース異常時に全コネクションを破棄することができます。コネクションが一括破棄される際、業務仕掛かり中のコネクションは業務終了後にクローズされ破棄されます。再利用はされません。

データベースから応答がない場合、TCP/IP レベルでのタイムアウトが有効となります。または、アプリケーションで、java.sql.Statement にクエリタイムアウトを指定した場合は、その値が優先します。

- データベース異常時(状態監視あり、一括破棄否)

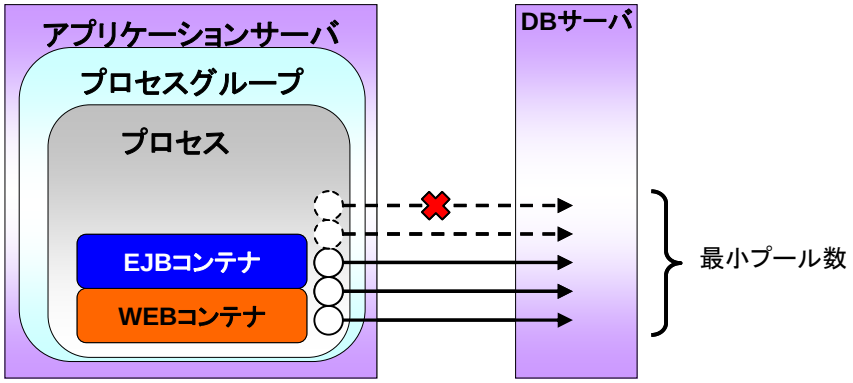


図 2.6.11.データベース異常時(状態監視あり、一括破棄否)

コネクションの一括破棄可否が無効の場合、状態監視は未使用のコネクションに対してのみ、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)

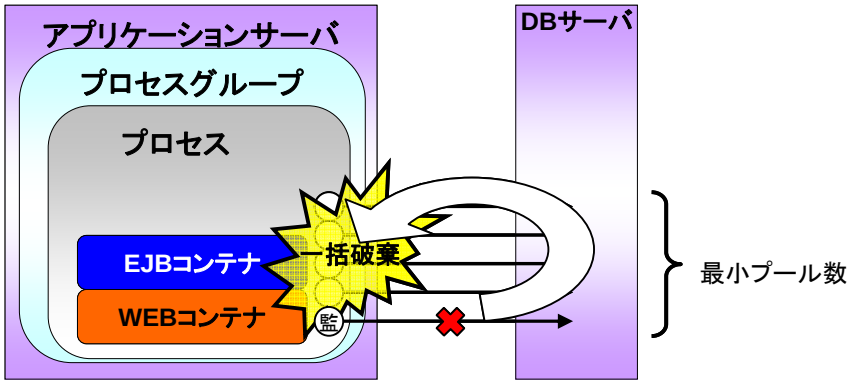


図 2.6.12.データベース異常時(状態監視あり、一括破棄可)

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。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

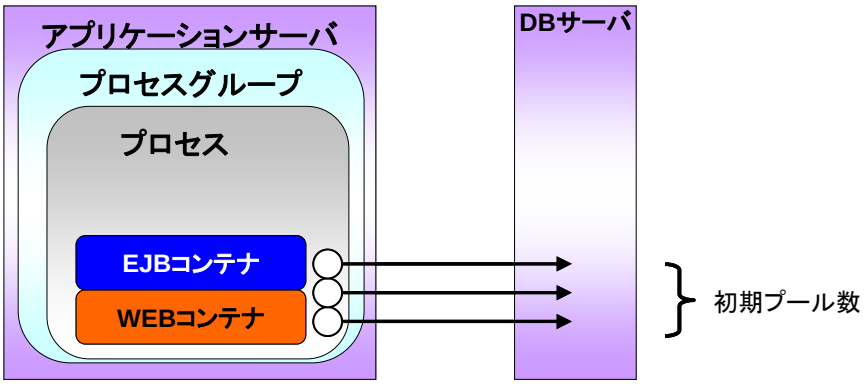


図 2.6.13.データベースの復旧時

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、初期プールサイズが作成されるまで永続的に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ー一覧です。

1） JDBC タイプの設定項目

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)	説明	デフォルト値	設定値
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。	2	dataSourceType 、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

※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)	説明	デフォルト値	設定値
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

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)	説明	デフォルト値	設定値
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×プロセス多重度+(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廃棄可否 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
初期接続リトライ reconnectInitialPool	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
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。	0	0
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。	10	10

3） 状態監視

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)	説明	デフォルト値	設定値
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

ログは以下のような状況で、閲覧されます。

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

本節では、これらの状況下で確認すべきログについて説明を行ないます。また、ログの運用を行なう上での注意すべきことについて説明を行ないます。

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 共通
   - ➢ ログの概要
   - ➢ ログの設定内容

2） 業務アプリケーションの出力するログ 共通 マルチ シングル

3） クライアントからのリクエスト情報の管理・監視 共通

1） WebOTX の出力するログ 共通

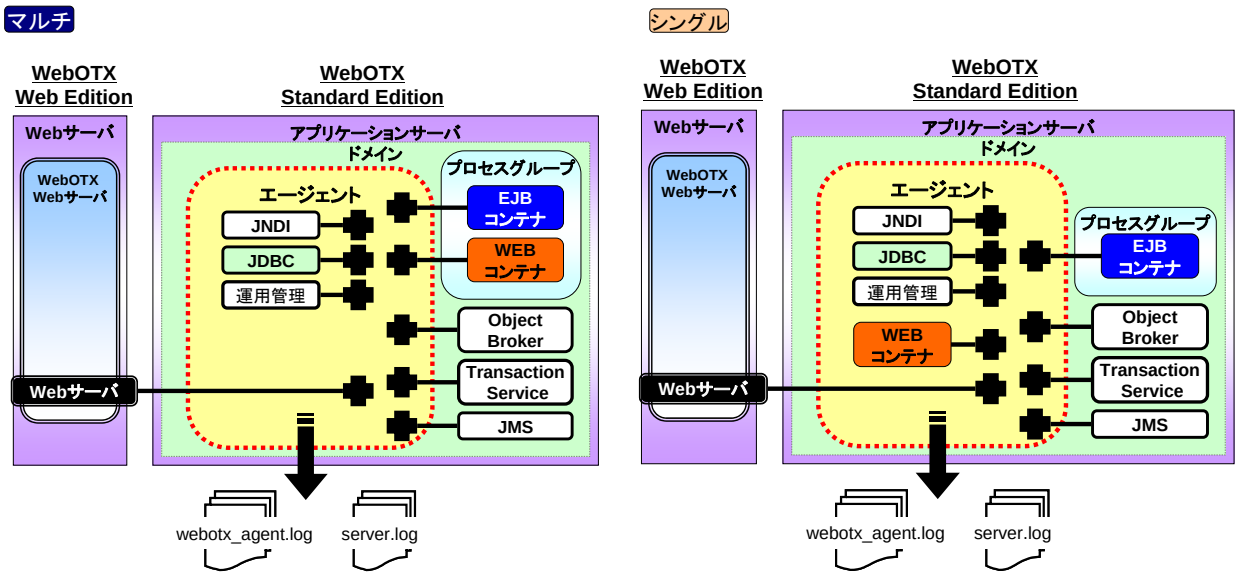


図 2.7.1.ログ出力イメージ

共通

- ➢ 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 Edition 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、確認することが出来ます。

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。
webotx_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。

webotx_agent.log に出力されないサービスに関しては、各サービスが出力している詳細なログを確認する必要があります。主に使用されるサービスであり、かつ webotx_agent.log 内に出力されないログは、以下のものとなります。

ログファイル名	ディレクトリ	説明
wojdbc.log	<ドメイン名>/logs/jdbc	JDBCデータソースに関するログ。
wojta.log	<ドメイン名>/logs/jdbc	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。
wojms.log	<ドメイン名>/logs/wojms	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。

- ログサイズ
  出力を行うログの最大サイズを設定します。
- 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、SEVERE, WARNING, CONFIG, INFO, FINE, FINEST となっています。

閾値	説明
OFF	全ての出力を行いません
SEVERE	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。
WARNING	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。
SLOGINFO (*1)	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。
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。
FINE	デバックレベルのメッセージ以上を出力します。
FINER(*2)	FINEと同等のレベルです。
FINEST	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。

※1 SLOGINFO を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。

※2 FINER を設定した場合、FINE を設定した状態と同じとなります。

- ログローテート
  ログは、ログサイズで設定した最大サイズを超えた場合、もしくは指定した時間経過した場合、ローテーションを行います。
  より詳しい内容については WebOTX マニュアル「運用編 – ロギング編」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 共通 マルチ シングル

共通

業務アプリケーションからログを出力する場合、WebOTX は標準でバンドルしている Log4j を使用するか、または業務アプリケーションに含まれる log4j を使うことも可能です。
バンドルされていない Log4j を使用する場合、以下のディレクトリに配置する必要があります。

- WEB アプリケーションから出力する場合
  <ドメイン名>/applications/… 配下で配備時に作成される WEB-INF/lib ディレクトリに配置することで、使用すること

が可能です。

- EJB アプリケーションから出力する場合
  <ドメイン名>/lib/ext に配置することで、業務アプリケーションの log4j を使用することが可能です

標準出力を利用した場合、以下のファイルにログが出力されます。

マルチ

ログファイル名	ディレクトリ	説明
<プロセスグループ名>.<プロセスID>.log	<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>	サーバアプリケーションのトレースログ。Webアプリケーション及びEJBアプリケーションのログを出力します。

シングル

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。 シングルプロセスモード選択時、WEBアプリケーションのログはここに出力されます。
<プロセスグループ名>.<プロセスID>.log	<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>	サーバアプリケーションのトレースログ。EJBアプリケーションのログを出力します。

より詳しい内容については WebOTX マニュアル「運用編 – ロギング編」を参照してください。

3) クライアントからのリクエスト情報の管理・監視 共通

共通

リクエスト情報の管理、監視を行う場合 Web サーバにて出力が行われるログを閲覧することが必要になります。

ログファイル名	ディレクトリ	説明
access_log	<ドメイン名>/logs/WebServer	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	<ドメイン名>/logs/WebServer	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。
このため、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 共通 マルチ シングル

2) TP モニタのトレースログ 共通

1) ログのローテート機能 共通 マルチ シングル
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を提供しています。
システム要件におけるログの保存期間に合うよう、ログファイルの上限サイズと、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。

ただし、以下のログについてはデフォルトではローテート機能を提供していない、またはデフォルトでローテート設定が行われておりません。
ローテート設定を行う場合、WebOTX マニュアル「運用編 – ロギング編」の対象ログ名を参照してください。

共通

- Web サーバのログ
  access_log(access.log)
  error_log(error.log)
  ssl_request_log(ssl_request.log)

- アプリケーションサーバのログ
  server.log

マルチ

- IIOP プラグインのログ
  mod_jk_om.log
  mod_jk_om-20.log
- IIOP プラグインが動作している場合、Object Broker が出力するログ(*1)
  Windows：<インストールディレクトリ>¥ObjectBroker¥log 配下
  Linux：/opt/ObjectSpinner/log 配下

※1 サイズ指定によるローテート機能を提供していますが、デフォルトではサイズ制限がありません。

シングル

- mod_jk のログ
  mod_jk.log
  mod_jk-20.log

2) TP モニタのトレースログ 共通

共通

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自動では削除されません。定期的に削除したい場合はタスクスケジューラ(Windows の場合)や cron(UNIX の場合)を利用して、削除する必要があります。

- Windows の場合
  タスクスケジューラは[コントロールパネル]-[タスク]-[スケジュールされたタスクの追加]で使用できます。save ディレクトリ削除(退避)用バッチを作成してタスクスケジューラに登録してください。

- UNIX の場合
  cron は crontab ファイルの命令に従って実行されます。HP-UX の場合、crontab ファイルは実行ユーザ名と同じで、/usr/lib/cron/cron.allow ファイルに該当する名前がある場合に crontab を実行できます。

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と障害発生時の対処法
- アライブチェックについて
- ログ内で監視すべきメッセージと障害発生時の対処法

2.8.1.概要説明

1) 監視すべきプロセス 共通 マルチ シングル
   - WebOTX アプリケーション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) アライブチェックについて 共通
3) 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と障害発生時の対処法 共通 マルチ シングル

WebOTX のプロセス群は、WebOTX アプリケーション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、WebOTX アプリケーションサーバで動作するものと、TP モニタにて動作するものを説明します。

- WebOTX アプリケーションサーバで動作するプロセス

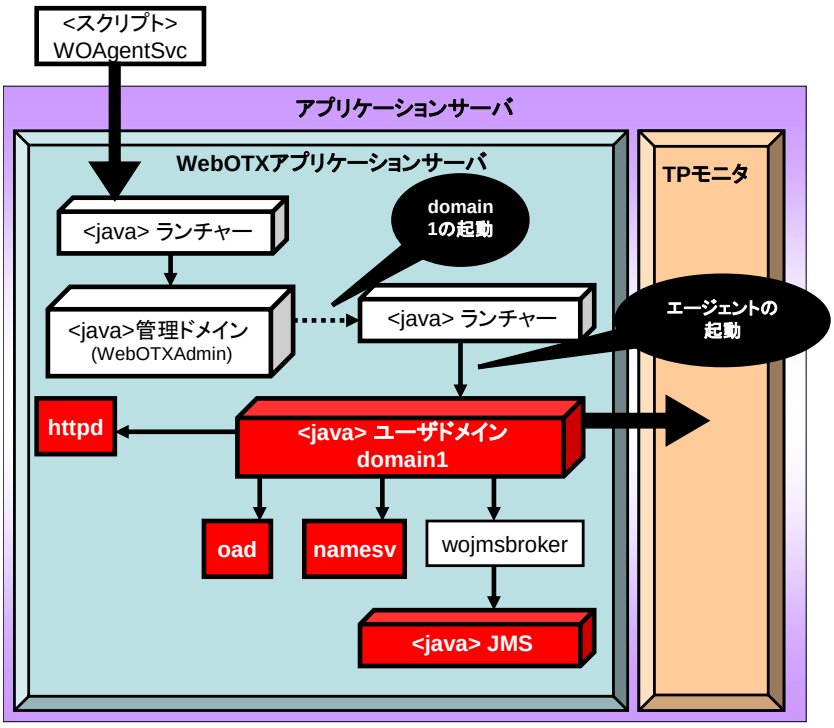


図 2.8.1.WebOTX アプリケーションサーバ内　監視プロセス

共通

図 2.8.1.	プロセス名	説明
httpd	apache.exe (Windows) httpd (UNIX)	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 サーバ選択した場合、使用します。 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/WebServer/httpd.pid)を参照してください。
oad	oad.exe (Windows) oad (UNIX)	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。
namesv	namesv.exe (Windows) namesv (UNIX)	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。
JMS	java.exe (Windows) java (UNIX)	エージェントプロセスに管理されているJMS デーモンプロセスです。JMS を起動すると動作します。 引数に「funcid=jms domain.name=<ドメイン名>」という文字列が指定されますのでそれを用いて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(WebOTXAdmin)は監視不要です。

マルチ

図 2.8.1.	プロセス名	説明
domain1	java.exe (Windows) java (UNIX)	ランチャーにより起動される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。 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

シングル

図 2.8.1.	プロセス名	説明
domain1	java.exe (Windows) java (UNIX)	ランチャーにより起動される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、及び Web コンテナを使用する業務アプリケーションにアクセスができなくなります。 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

➢ TP モニタで動作が行われるプロセス

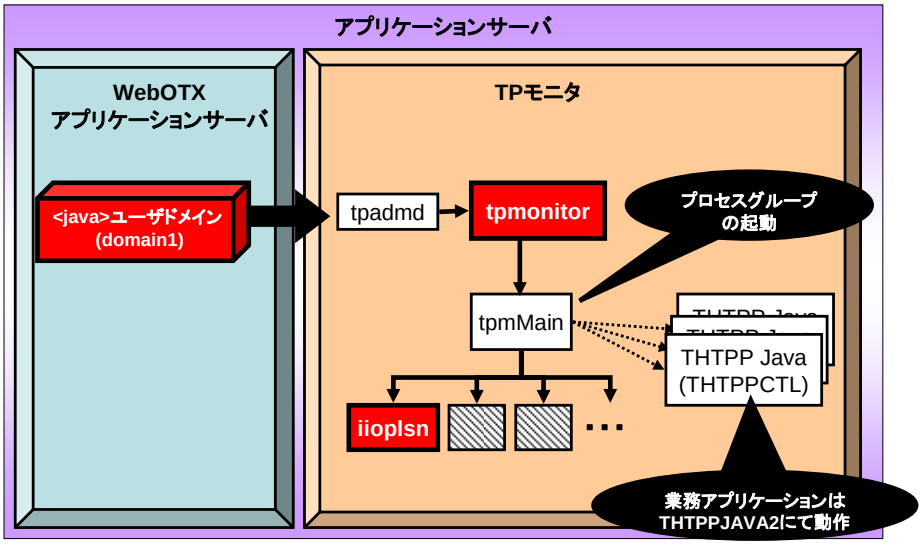


図 2.8.2.TP モニタ内　監視プロセス

共通

図 2.8.2.	プロセス名	説明
tpmonitor	tpmonitor.exe （Windows） tpmonitor （UNIX）	tpmMain プロセスの親プロセスで、主な役割はtpmMain プロセスの監視です。 プロセスグループの監視はtpmMainにて行われますが、親プロセスであるtpmonitorが監視しているため、こちらを監視対象にしてください。 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 （UNIX）	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。

2）　アライブチェックについて共通

共通

WebOTX 上で動作するサービスやサーバ AP プロセスなど状態管理を行なっているリソースに対してアライブチェックを行なうことができます。
アライブチェックは一定時間ごとに行なわれ、生存確認が行なえなくなった場合エラーログを出力します。デフォルトの設定で問題ありませんが、使用していないプロセスのアライブチェックを外すことで、使用していないプロセスのエラー通知を除くことができます。

自動起動サーバ機能(oadj)等を使う予定がない場合、このプロセスのアライブチェックを外してください。

3）　ログ内で監視すべきメッセージを障害発生時の対処法共通

共通

監視すべき対象としては、システムログが挙げられます。システムログではログレベルにおける SEVERE、WARNING 以上のものを出力しています。それぞれのレベル設定は以下の通りです。

SEVERE	システムを継続運用するのに支障をきたす障害
WARNING	システムを継続運用することはできるレベルの障害

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

性能設計をする上で必要となる指針を記載します。

1） プロセス障害発生時の対処法共通

2） アライブチェック共通

3） 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、UNIX の場合、プロセスのコマンド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
apache.exe(Windows) httpd(UNIX)	-f /opt/WebOTX/domains/${domain.name}/config/WebServer/httpd.conf
oad.exe(Windows) oad(UNIX)	-WOdomain /opt/WebOTX/domains/${domain.name}
namesv.exe(Windows) namesv(UNIX)	-WOdomain /opt/WebOTX/domains/${domain.name}
java.exe(Windows) java (UNIX)	java -server -Dwebotx.funcid=jms -Ddomain.name=domain1
java.exe(Windows) java(UNIX)	funcid=agent domain.name=${domain.name}
tpmonitor.exe（Windows） tpmonitor（UNIX）	-n${domain.name}
iioplsn.exe（Windows） iioplsn（UNIX）	-sg_fname /opt/WebOTX/domains/${domain.name}/config/tpsystem/iiop.sg

2） アライブチェック共通
listimple コマンドを用いて、プログラムが表示されない場合、自動起動サーバ機能(oadj)は使用されていません。
下記手順でアライブチェック対象から外してください。

[管理ツール]→[ドメイン]→[アプリケーションサーバ]→[ObjectBroker サービス]→[oadj]
状態タブ→「アライブチェックモニタの自動登録を行う」のチェックを外す

3） ログ内で監視すべきメッセージ共通

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。また、配備されたアプリケーションが、アプリケーションで共通に利用するライブラリを利用する場合、WebOTX がそのライブラリを読み込めるように、ライブラリを正しく配置する必要があります。本節では、配備の仕組みと、アプリケーションが利用するライブラリを読み込む役割を持つクラスローダについて説明します。

2.9.1.概要説明

配備では、主に以下の項目に対し、説明を行います

1） 配備 共通 マルチ シングル
   - 配備とは
   - 配備先

2） 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） 配備 共通 マルチ シングル

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。作成したアプリケーションに対し、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、配備もその 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。
  - 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。
  - 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。
  - 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。

マルチ

- 配備先

マルチプロセスモードの場合、配備したアプリケーションは WebOTX のコアの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。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。このため、アプリケーションを配備する際には、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の配備なのかを指定する必要があります。

シングル

- 配備先

シングルプロセスモードの場合、Web コンテナは WebOTX のコアのプロセス上で動作し、EJB コンテナは、コアの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。そのため、シングルプロセスモードでは、EJB コンテナを利用しない Web アプリケーションを配備する際には、アプリケーショングループおよびプロセスグループの指定は必要ありません。EJB コンテナを利用する、EJB アプリケーションやエンタープライズアプリケーションの場合は、マルチプロセスモードの場合と同様、アプリケーショングループおよびプロセスグループの指定が必要です。

2） クラスローダ(ライブラリ配置) 共通

共通

配備された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

配備では、クラスローダの構造について知っておくことが重要となるため、設計指針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

共通

- クラスローダの構造

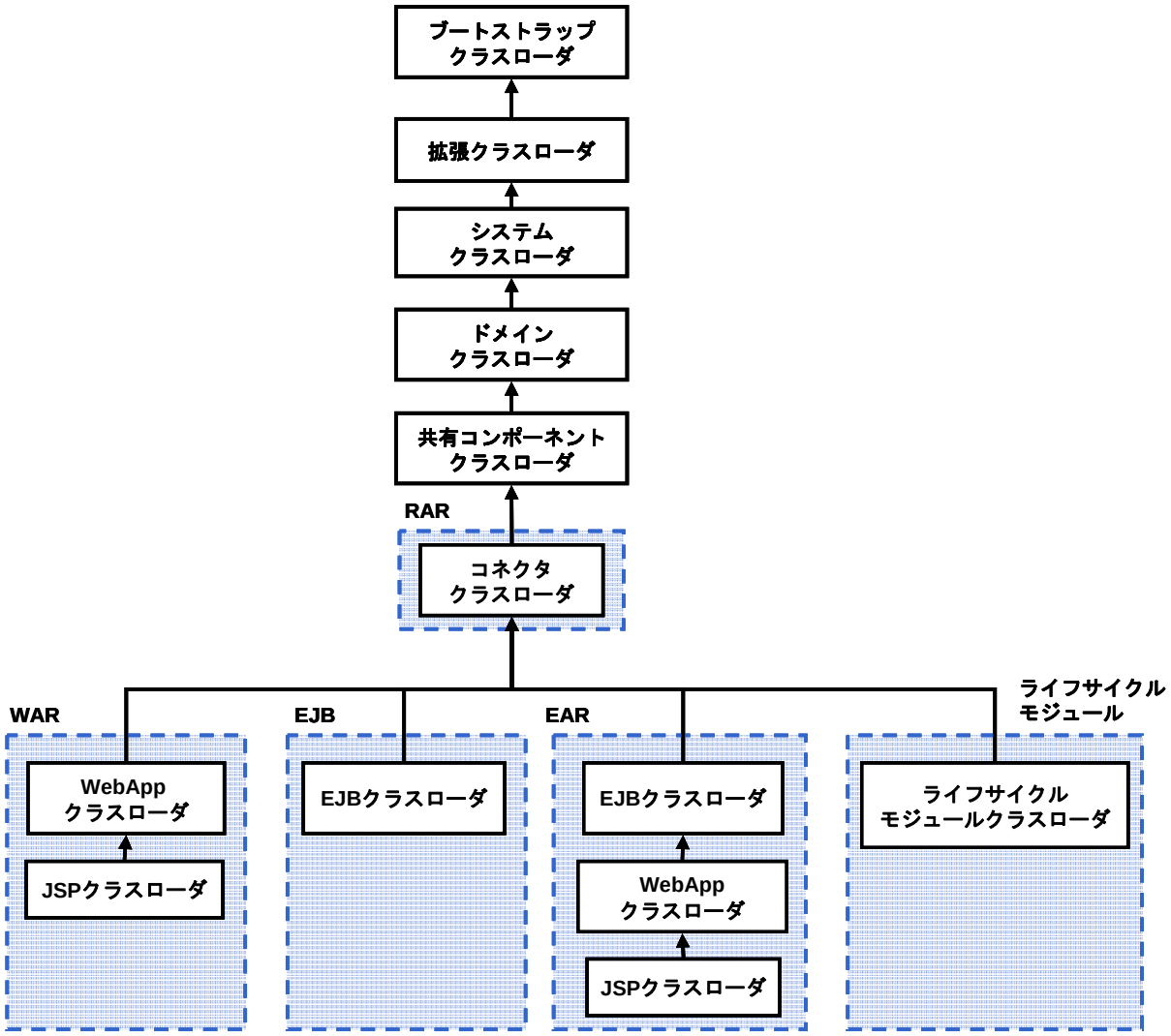


図 2.9.1.クラスローダ階層構造

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。JDK のクラスとドメインの lib/ext 下に置かれた jar または zip ファイルをロードします。
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。
システムクラスローダ	WebOTX のシステムライブラリ、classpath-prefix、server-classpath, classpath-suffix 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。
ドメインクラスローダ	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 または zip ファイルと lib/classes 下に置かれたクラスファイルをロードするクラスローダです。
共有コンポーネントクラスローダ	ドメインに配備された共有コンポーネントに含まれるクラスをロードするクラスローダです。EJB、CORBA Java コンポーネントでのみ使用可能です。インストール時に Web コンテナの動作モードをマルチプロセスモードで選択した場合は Web モジュールからも使用できます。共有コンポーネントを配備解除することにより動的にクラスをアンロードすることが可能です。
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 に含まれるかどうかを問わず、リソースアダプタは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。
WebApp クラスローダ	Java EE アプリケーションまたは Web モジュールに含まれる 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	Java 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。
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。

クラスローダの構造は、ツリー型の親子関係となっています。WebOTX では、上記のような階層構造となっています。
ブートストラップクラスローダからコネクタ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。

  - delegate=true 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)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。

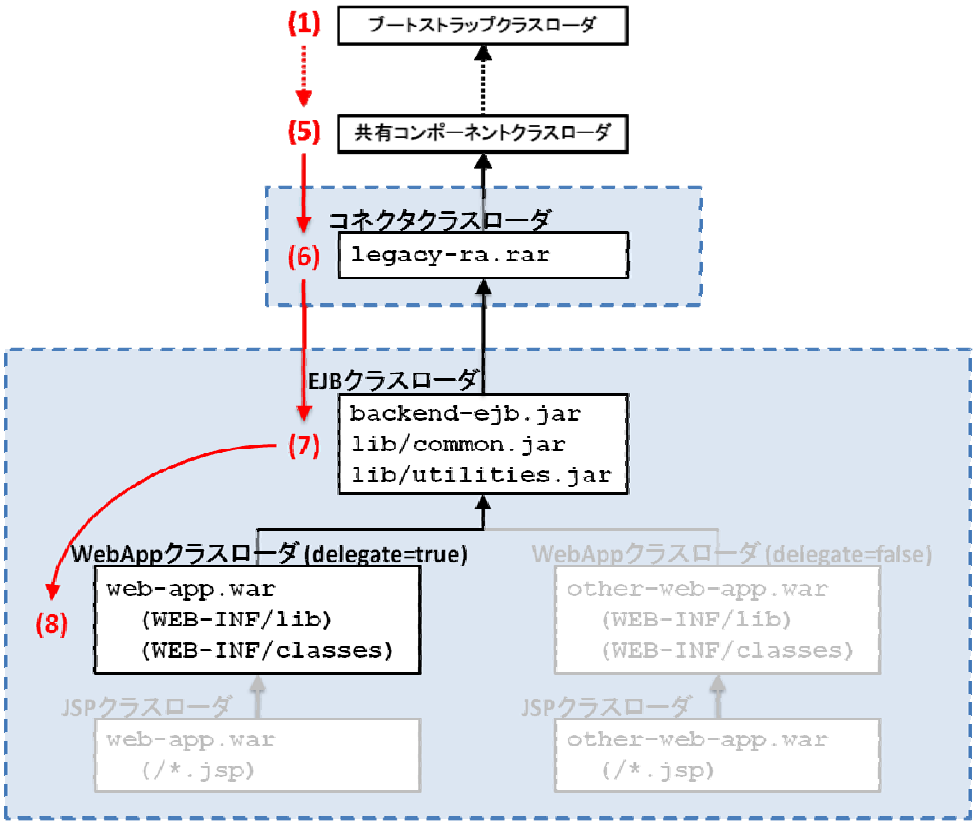


図 2.9.2.delegate=true の場合

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。

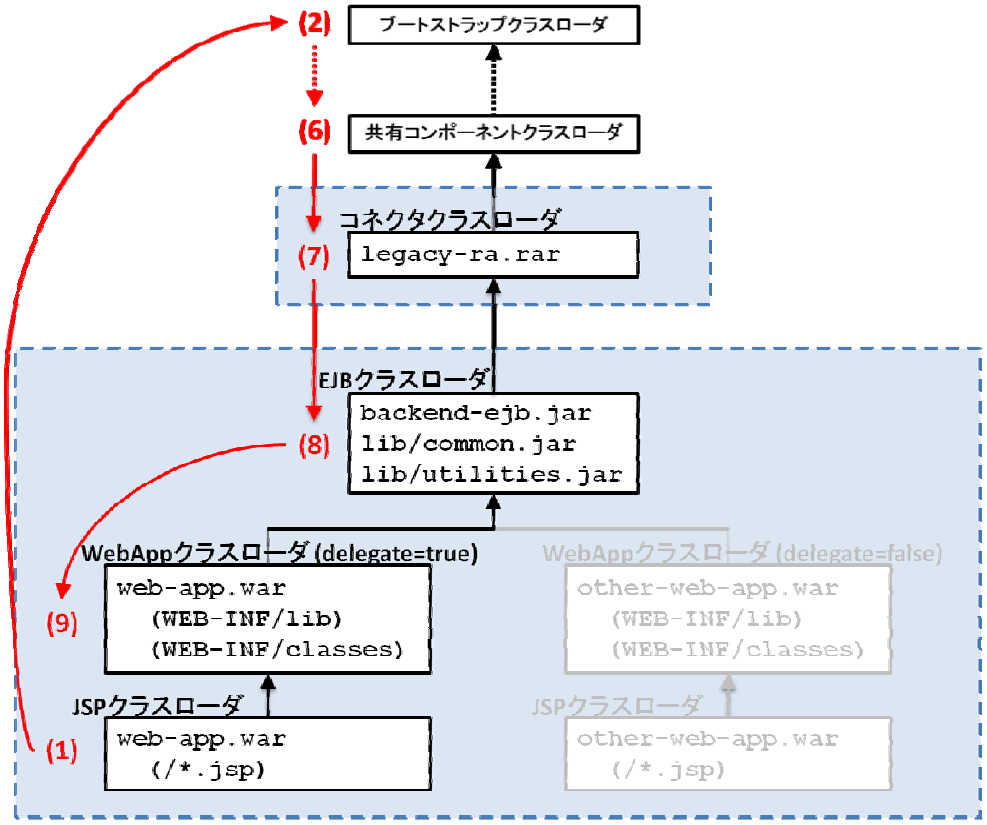


図 2.9.3　delegate=true の場合(JSP クラスローダ)

- 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。

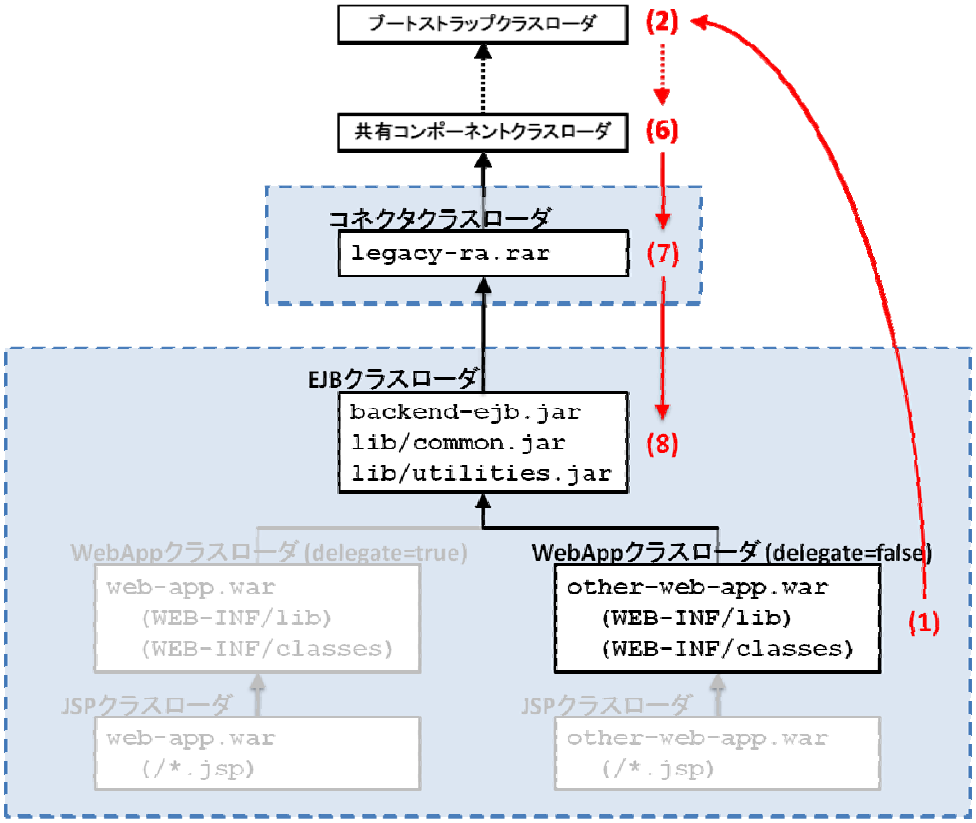


図 2.9.4.delegate=false の場合

また、JSP を読み込む場合は以下のような順番となります。

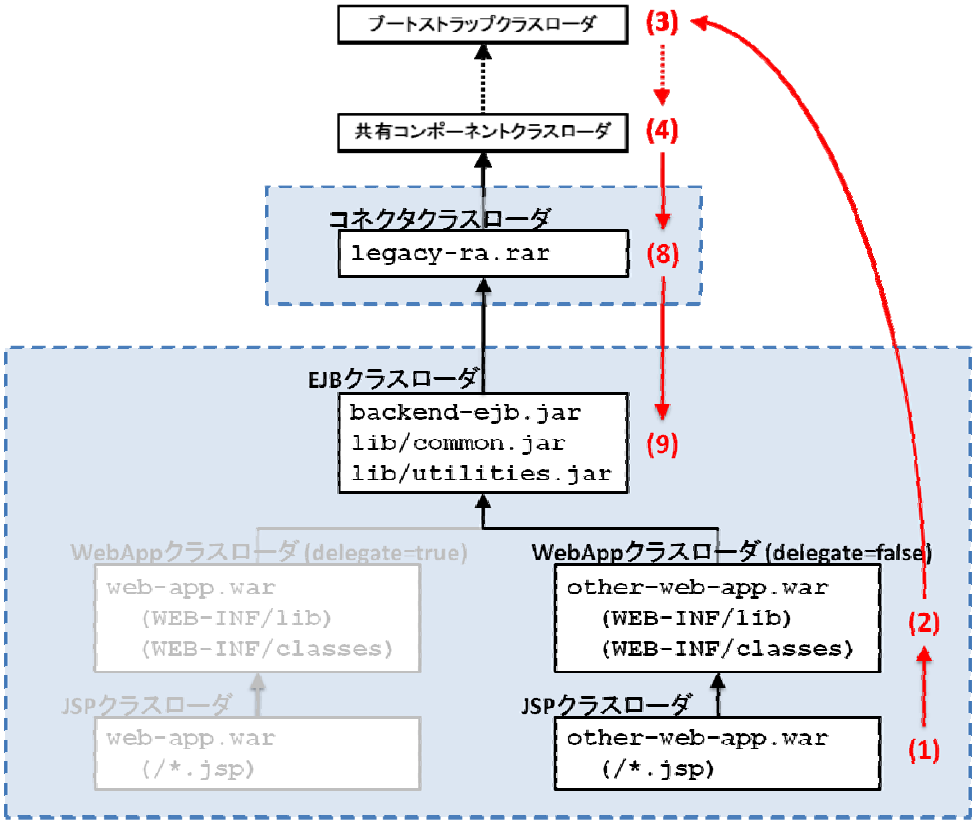


図 2.9.5.delegate=false の場合(JSP クラスローダ)

- 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 または zip 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>/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>/classes
JDBCドライバの配置	JDBCドライバは WebOTX 本体をロードしているシステムクラスローダが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システム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

配備の操作方法など、詳細については WebOTX マニュアル「運用編－アプリケーションの配備」を参照して下さい。

3.その他

3.1.使用上の条件の確認

■ 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。

■ 概要

- リソース

WebOTX を利用する際に必要なメモリ容量、固定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固定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、インストール時に既定値を選択して動作させた場合を表しています。
ハード ディスク容量には、C コンパイラや J2SE SDK などの関連ソフトウェアのディスク消費量は含まれていません。

・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 Edition

プラットフォーム	ハード ディスク	メモリ容量
Windows(x86)	300MB 以上	最小 384 MB、推奨 512 MB 以上
Windows(x64)	300MB 以上	最小 1 GB、推奨 1.5 GB 以上
HP-UX	400MB 以上	最小 1.5 GB、推奨 2 GB 以上
Linux(x86)	300MB 以上	最小 640 MB、推奨 1GB 以上
Linux(x64)	300MB 以上	最小 1 GB、推奨 1.5 GB 以上
Solaris	400MB 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(x86)	Windows Server® 2003 R2, Standard Edition Windows Server® 2003 R2, Enterprise Edition Windows Server® 2003 R2, Datacenter Edition Windows Server® 2003, Standard Edition Windows Server® 2003, Enterprise Edition Windows Server® 2003, Datacenter Edition

	Windows® 2000 Server SP4 Windows® 2000 Advanced Server SP4
Windows(x64)	Windows Server® 2003 R2, Standard x64 Edition Windows Server® 2003 R2, Enterprise x64 Edition Windows Server® 2003 R2, Datacenter x64 Edition Windows Server® 2003, Standard x64 Edition Windows Server® 2003, Enterprise x64 Edition Windows Server® 2003, Datacenter x64 Edition
HP-UX	HP-UX 11i v2 HP-UX 11i v3（注意） 標準修正 V7.11.02.00 の適用が必要
Linux(x86)	Red Hat Enterprise Linux AS 4.0 Red Hat Enterprise Linux ES 4.0 MIRACLE LINUX V4.0
Linux(x64)	Red Hat Enterprise Linux AS 4.0 Red Hat Enterprise Linux ES 4.0 MIRACLE LINUX V4.0
Solaris	Sun Solaris 9 Sun Solaris 10

・Java 2 SDK、Standard Edition

WebOTX システムは、実行時に Java™ 2 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
J2SE 1.4.2 J2SE 5.0 Java SE 6 (HP-UX 以外)

・Web ブラウザ・プラグイン

Web 版統合運用管理コンソールでは、実行環境に対する操作機能が Adobe Flash で提供されます。 Adobe Flash では、Adobe Flash Player プラグインが必要です。 Adobe Flash Player プラグインがシステムにインストールされていない場合、管理コンソールで操作をするには、Adobe Flash Player プラグインをインストールする必要があります。

・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 1.3.39 以降、2.0.61 以降 Apache HTTP Server 1.3.39 以降、2.0.61 以降 Internet Information Services (IIS) 5.0、6.0 Sun Java System Web Server 6.1 Sun ONE Web Server 6.0 以降

ただし、マルチプロセスモードの場合は WebOTX Web サーバ以外は対応できません。

・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 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	サポートするデータベースサーバ
Oracle	Oracle8i R8.1.7 以降
IBM	DB2 Universal Database 8.1.4 DB2 V9.1
Microsoft	Microsoft SQL Server 2000 Microsoft SQL Server 2005

3.2.パラメーター一覧

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

統合運用管理ツール

設定パラメータ(属性)	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 -Xmx	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]-[JavaVM オプション]を変更
エージェントプロセスの最大パーマネントサイズ -XX:MaxPermSize	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パーマネント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]-[JavaVM オプション] を変更
エージェントプロセスの GC ログ	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]-[JavaVM オプション] 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
プロセスグループの GC ログ	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

設定パラメータ(属性)	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 -Xmx	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> create-jvm-options -Xmx
エージェントプロセスの最大パーマネントサイズ -XX:MaxPermSize	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パーマネントサイズを指定します。	otxadmin> create-jvm-options –XX:MaxPermSize
エージェントプロセスの GC ログ	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> create-jvm-options "-verbose¥:gc"
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
プロセスグループの GC ログ	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=-verbose:gc

2） 使用していないサービスを抑止する

共通

統合運用管理ツール

設定パラメータ(属性)	説明	設定方法
JMS サービス起動抑止 enabled	サーバ起動時におけるライフサイクルモジュール(サービス)の起動可否を表します。	[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
RCS の起動抑止 rcs-cpp-startup	C++ AP を用いてトランザクション管理を行う場合(true)は Transaction サービス起動時に C++ AP 用 RCS プロセスを自動起動します。	[アプリケーションサーバ]-[Transaction サービス]の「C++ AP 用の RCS を自動起動」のチェックをはずす

運用管理コマンド

設定パラメータ(属性)	説明	設定方法
JMS サービス起動抑止 enabled	サーバ起動時におけるライフサイクルモジュール(サービス)の起動可否を表します。	otxadmin> set server.internal-lifecycle-module.JMSProvider.enabled=false
RCS の起動抑止 rcs-cpp-startup	C++ AP を用いてトランザクション管理を行う場合(true)は Transaction サービス起動時に C++ AP 用 RCS プロセスを自動起動します。	otxadmin> set server.transactionservice.rcs-cpp-startup=false

3.2.2.多重度

1） Web サーバの多重度設定

共通

統合運用管理ツール

設定パラメータ(属性)	説明	設定方法
最大同時接続数 Windows:ThreadsPerChild Unix: 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。UNIX 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

設定パラメータ(属性)	説明	設定方法
最大同時接続数 Windows:ThreadsPerChild Unix: 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。UNIX 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)	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。15000 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

マルチ

ファイルを直接編集する

設定パラメータ(属性)	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	IIOP プラグインの同時実行数を指定します。	${INSTANCE_ROOT}/config/WebCont/ior_workers.properties に指定

worker.otxiiop.cachesize		

シングル

ファイルを直接編集する

設定パラメータ(属性)	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 worker.ajp13.cachesize	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

共通

統合運用管理ツール

設定パラメータ(属性)	説明	設定方法
同時接続クライアント数 simultaneousConnectionClients	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 WebOTX Application Server 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]-[IIOP リスナ]-[利用可能な同時接続クライアント数] を変更
1 プロセス当たりの多重度 multiplexprocess	1 プロセス当たりのリクエスト処理同時実行多重度を指定します	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [TP システム]-[IIOP リスナ]-[1 プロセス当たりの多重度] を変更
スレッド数 threadCount	スレッド数を指定します	[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>]を変更
プロセス数 processCount	プロセス数を指定します	[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>]を変更

運用管理コマンド

設定パラメータ(属性)	説明	設定方法
同時接続クライアント数 simultaneousConnectionClients	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 WebOTX Application Server に接続する場合に設定してください。	otxadmin> set tpsystem.IIOPListener.simultaneousConnectionClients=0
1 プロセス当たりの多重度 multiplexprocess	1 プロセス当たりのリクエスト処理同時実行多重度を指定します	otxadmin> set tpsystem.IIOPListener.multiplexprocess=128
スレッド数 threadCount	スレッド数を指定します	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>. threadCount=3
プロセス数 processCount	プロセス数を指定します	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>. processCount=1

シングル

運用管理コマンド

設定パラメータ(属性)	説明	設定方法
最小プロセッサ数 min-processors	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.http-service.http-listener.<リスナ ID>.min-processors=<最小数>
最大プロセッサ数 max-processors	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.http-service.http-listener.<リスナ ID>.max-processors=<最大数>
限界プロセッサ数 limit-processors	Web コンテナは、アクティブなリクエスト受け付けプロセッサの数が最大数に近づいた時に運用管理コンソールに警告を通知します。そ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.http-service.http-listener.<リスナ ID>.limit-processors=<閾値>

3.2.3.通信/タイムアウト

共通

運用管理ツール

設定パラメータ(属性)	説明	設定方法
実行時間タイムアウト exetimeMax	オペレーションの実行時間に上限を設定する場合に設定します。実行時間が設定時間(秒)を過ぎてもレスポンスが返却されない場合オペレーション処理を中断します。	[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]- [アプリケーション]-[WebOTX J2EE アプリケーション]-[<アプリケーション名>]-*-[<オペレーション名>]-[オペレーションの自動活性化]-[実行時間上限] を変更
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。	[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]-[Object Broker 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
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	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	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

運用管理コマンド

設定パラメータ(属性)	説明	設定方法
実行時間タイムアウト	オペレーションの実行時間に上限を設定する場合に設定します。実行時間が設定時間(秒)を過ぎてもレスポンスが返却されない場合オペレーション処理を中断します。	otxadmin> set applications.j2ee-applications.CartApp.CartApp¥.ear.cart.POST.exetimeMax=-1
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.objectbrokerconfig.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

ファイルを直接編集する

設定パラメータ(属性)	説明	設定方法
Web ブラウザ設定タイムアウト ReceiveTimeout	Web ブラウザ設定タイムアウト	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
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

マルチ

ファイルを直接編集する

設定パラメータ(属性)	説明	設定方法

IIOP メソッド呼び出しタイムアウト worker.otxiiop.iiop_timeout	IIOP プラグインのメソッド呼び出しを行う際のタイムアウト値を設定します	<WebOTX インストールディレクトリ>/domains/<ドメイン名>/config/WebCont/ior_workers.properties で設定

シングル

ファイルを直接編集する

設定パラメータ(属性)	説明	設定方法
コネクションプールタイムアウト worker.ajp13.connection_pool_timeout	Web サーバから Web コンテ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
worker.ajp13.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

3.2.4.セッション

1） セッション管理について

共通

統合運用管理ツール

設定パラメータ(属性)	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

運用管理コマンド

設定パラメータ(属性)	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)

ファイルを直接編集する

設定パラメータ(属性)	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

統合運用管理ツール

設定パラメータ(属性)	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]-[jmx-connector]-[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 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]-[HTTP サービス]-[HTTP リスナ]-[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
IIOP リスナアライブチェックポート番号	IIOP リスナのアライブチェックを行うためのポート番号を指定します。 「5220+TP モニタのシステム ID」の値となります。	指定不可
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

設定パラメータ(属性)	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.jmx_connector.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 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.http-service.http-listener.admin-listener.port =4848
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 =5751
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 =5751

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
IIOP リスナアライブチェックポート番号 /etc/services ファイル webotx-mess	IIOP リスナのアライブチェックを行うためのポート番号を指定します。 「5220+TP モニタのシステム ID」の値となります。	指定不可
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)	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

シングル

統合運用管理ツール

設定パラメータ(属性)	説明	設定方法
AJP リスナ通信ポート番号 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]-[ajp リスナ]- [リスナ]-[平文ポート番号] を変更

運用管理コマンド

設定パラメータ(属性)	説明	設定方法
AJP リスナ通信ポート番号 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.http-service.http-listener.ajp-listener-1.port=8009

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、以下のパラメータの変更を行うことが可能です。
運用管理コマンド

設定パラメータ(属性)	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)

3.2.6.DB 連携

1) データソース

共通

データソースに関する設定方法を記述します

統合運用管理ツール

設定パラメータ(属性)	説明	設定方法
データソース種別 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 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
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。	[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
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。	[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
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。	[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
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。	[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 連携の有無 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)	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 EJP 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

設定パラメータ(属性)	説明	設定方法
データソース種別 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 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
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。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.serverName = <DB サーバ名>
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。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.portNumber = <DB ポート番号>
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。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.userName = <ユーザ名>
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。	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 連携の有無 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)	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

統合運用管理ツール

設定パラメータ(属性)	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廃棄可否 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
初期接続リトライ 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]-[コネクションプール]-[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]-[コネクションプール]-[接続リトライ間隔] を変更

運用管理コマンド

設定パラメータ(属性)	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廃棄可否 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>
初期接続リトライ 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 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。	otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>. connectRetryMax = 0
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

3) 状態監視

共通

統合運用管理ツール

設定パラメータ(属性)	説明	設定方法
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
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
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

設定パラメータ(属性)	説明	設定方法
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”
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
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

設定すべきパラメータはありません。

3.2.8.監視

設定すべきパラメータはありません。

3.2.9.配備

設定すべきパラメータはありません。

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。
tpadmd.exe (Windows) tpadmd (UNIX)	Standard/Enterprise Edition における TP モニタ運用管理用のサービスプロセスです。 「WebOTX TPBASEadm」を起動すると動作し、停止すると終了します。 UNIX におけるカレントディレクトリは${AS_INSTALL}/Trnsv です。

管理ドメインで動作するプロセス

プロセス名	説明
java.exe(Windows) java(UNIX)	エージェント(管理ドメイン)を起動するための JavaVM です。エージェントが停止すると終了します。 異常終了した場合、Windows 系の場合はエージェント(管理ドメイン)の JavaVM に直接影響はありませんが、エージェントの JavaVM から直接出力されるエラーメッセージが server.log に出力されなくなります。UNIX 系の場合はエージェントの JavaVM(後述)、および一般ドメインにおける全てのプロセスが終了します。 本プロセスを再起動するにはドメインの再起動が必要です。 引数に「funcid=startserv domain.name=WebOTXAdmin」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) java(UNIX)	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=WebOTXAdmin」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。 UNIX 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(UNIX)	エージェント(ドメイン)を起動するための JavaVM です。エージェントが停止すると終了します。 異常終了した場合、Windows 系の場合はエージェント(ドメイン)の JavaVM に直接影響はありませんが、エージェントの JavaVM から直接出力されるエラーメッセージが server.log に出力されなくなります。UNIX 系の場合はエージェントの JavaVM(後述)も終了します。本プロセスを再起動するにはドメインの再起動が必要です。 引数に「funcid=startserv domain.name=${DOMAIN_NAME}」という文字列が指定されますのでそれが目印になります。
java.exe(Windows) java(UNIX)	エージェントの JavaVM です。 異常終了した場合、該当ドメインへのアクセスができなくなります。速やかにドメイン再起動を行う必要があります。 引数に「funcid=agent domain.name=${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。UNIX におけるカレントディレクトリは${INSTANCE_ROOT}/config です。
apache.exe(Windows) httpd(UNIX)	HTTP サーバのデーモンプロセス。インストール時に「WebOTX Web サーバ」を選択した場合に有効です。Web サーバを起動すると動作します。 常時親プロセス(監視プロセス)と子プロセス(HTTP サービスデーモン)で構成されており、子プロセスが異常終了した場合、親プロセスは子プロセスを再起動します。 異常終了時は HTTP サーバの再起動が必要です。 UNIX におけるカレントディレクトリは${INSTANCE_ROOT}/です。
wojmsbrokersvc.exe (Windows) wojmsbrokerd (UNIX)	JMS 管理プロセス。 JMS を起動すると動作します。

	異常終了時は JMS の再起動が必要です。なお再起動方法については「※3」を参照ください。
java.exe (Windows) java (UNIX)	JMS デーモンプロセス。 JMS を起動すると動作します。 異常終了時は JMS の再起動が必要です。なお再起動方法については「※3」を参照ください。 引数に「funcid=jms domain.name=${DOMAIN_NAME}」という文字列が指定されますのでそれが目印になります。 UNIX におけるカレントディレクトリは{INSTANCE_ROOT}/config です。
oad.exe (Windows) oad (UNIX)	CORBA オブジェクト活性化デーモンプロセス。 Object Broker を起動すると動作します。異常終了時は新たな IIOP 通信(RMI/IIOP)が行えなくなります。 異常終了時は Object Broker の再起動が必要です。なお再起動方法については「※1」を参照ください。 UNIX におけるカレントディレクトリは{INSTANCE_ROOT}/config です。 Standard-J/Standard/Enterprise Edition で動作します
namesv.exe (Windows) namesv (UNIX)	CORBA 名前サーバデーモンプロセス。 Object Broker を起動すると動作します。異常終了時はオブジェクトの取得が行えなくなり IIOP 通信ができなくなります。 異常終了時は Object Broker の再起動が必要です。なお再起動方法については「※1」を参照ください。 UNIX におけるカレントディレクトリは{INSTANCE_ROOT}/config です。 Standard-J/Standard/Enterprise Edition で動作します
irsv.exe (Windows) irsv (UNIX)	CORBA インターフェースリポジトリデーモンプロセス。 Object Broker を起動すると動作します。異常終了時はオブジェクトのインタフェース情報の取得が行えなくなりますが WebOTX では irsv を利用していないため影響はありません。 異常終了時は Object Broker の再起動が必要です。なお再起動方法については「※1」を参照ください。 Standard-J/Standard/Enterprise Edition で動作します
corbaloc.exe (Windows) corbaloc (UNIX)	CORBALOC サーバデーモンプロセス。 Object Broker を起動すると動作します。異常終了時は CORBALOC サーバとして利用している場合、新たな IIOP 通信(RMI/IIOP)が行えなくなります。 異常終了時は Object Broker の再起動が必要です。なお再起動方法については「※1」を参照ください。 Standard-J/Standard/Enterprise Edition で動作します
java.exe (Windows) java (UNIX)	CORBA Java 自動起動デーモンプロセス。 Object Broker を起動すると動作します。異常終了時は Java アプリケーションの自動起動ができなくなります。また、ライセンスの取得もできなくなるため、IIOP 通信のコネクション数に制限が発生します。 異常終了時は Object Broker の再起動が必要です。なお再起動方法については「※1」を参照ください。 引数に「funcid=oadj domain.name=${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。 UNIX におけるカレントディレクトリは{INSTANCE_ROOT}/config です。 Standard-J/Standard/Enterprise Edition で動作します
rcssv.exe(Windows) rcssv(UNIX)	Transaction サービスで提供する C++アプリケーション用のリカバリプロセス(RCS) (Web Edition、および Standard-J Edition では使用しません) Transaction サービスを起動すると動作します。異常終了時はトランザクションの開始が行えなくなります。 異常終了時は Transaction 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「※2」を参照ください。 引数に「rcsid=XXX」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。 UNIX におけるカレントディレクトリは{INSTANCE_ROOT}/logs/TS です。
tpmMain.exe(Windows) tpmMain(UNIX)	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/Enterprise Edition 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りますが、親プロセスが監視を行なっているのでプロセスの監視は親プロセス(tpminitor)の方を監視してください。 UNIX におけるカレントディレクトリは${AS_INSTALL}/Trnsv です。

<table>
<tr><td>tpmonitor.exe(Windows)
tpmonitor(UNIX)</td><td>tpmMain プロセスの親プロセスで主な役割は tpmMain プロセスの監視です。
Standard/Enterprise Edition 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。
UNIX におけるカレントディレクトリは${AS_INSTALL}/Trnsv です。</td></tr>
<tr><td>tpadmdMain.exe(Windows)
tpadmd(UNIX)</td><td>WebOTX エージェントプロセス(javaw)からの要求を受けるなど、TP モニタの運用操作に必要なプロセスです。
Standard/Enterprise Edition で動作します。
異常終了した場合、一時的に Standard/Enterprise Edition の一部運用操作がエラーとなりますが、自動的にプロセスが再起動され復旧します。
UNIX におけるカレントディレクトリは${AS_INSTALL}/Trnsv です。</td></tr>
<tr><td>tpssendtpp.exe(Windows)
tpssendtpp(UNIX)</td><td>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/Enterprise Edition 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。</td></tr>
<tr><td>systpp.exe(Windows)
systpp(UNIX)</td><td>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/Enterprise Edition 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。</td></tr>
<tr><td>jnlwrt.exe(Windows)
jnlwrt(UNIX)</td><td>ジャーナルを収集しているプロセスです。
Standard/Enterprise Edition で動作します。
異常終了した場合、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 システムの再起動です。</td></tr>
<tr><td>olftplsn.exe(Windows)
olftplsn(UNIX)</td><td>WebOTX 印刷キットを利用する場合に必要なプロセスです。
Standard/Enterprise Edition 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、WebOTX 印刷キット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>iioplsn.exe(Windows)
iioplsn(UNIX)</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プロセスです。
Standard/Enterprise Edition 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の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。
UNIX におけるカレントディレクトリは${AS_INSTALL}/Trnsv /logs です。</td></tr>
<tr><td>TIMMSGSND.exe(Windows)
TIMMSGSND(UNIX)</td><td>メッセージ送信を行う時間の管理を行っています。送信時間になるとメッセージを送信するように tpssendtpp プロセスに指示します。
Standard/Enterprise Edition で動作します。
異常終了時は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。</td></tr>
<tr><td>wosystpp.exe(Windows)
wosystpp(UNIX)</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/Enterprise Edition 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.exe
THTPPCTL_N.exe
THTPPCTL_N2003.exe
THTPPCTL_2005.exe
(Windows)
THTPPCTL6
THTPPCTL7
THTPPCTL8
THTPPJAVA2
THTPPOTS6
THTPPOTS7
THTPPOTS8
THTPPDB6
THTPPDB7
THTPPDB8
(UNIX)</td><td>サーバアプリケーションのプロセスです。
Standard/Enterprise Edition で動作します。
言語や利用機能によりモジュール名が変わります。

(Windows)
<table><tr><th>モジュール名</th><th>言語・利用機能</th><th>バージョン</th></tr><tr><td rowspan="2">THTPPCTL.exe</td><td>Java</td><td>全バージョン</td></tr><tr><td>C++(VC6.0)</td><td>V6 以下</td></tr><tr><td>THTPPCTL_N.exe</td><td>C++(VC .net 2002)</td><td>V6</td></tr><tr><td>THTPPCTL_N2003.exe</td><td>C++(VC .net 2003)</td><td>V6 以上</td></tr><tr><td>THTPPCTL_2005.exe</td><td>C++(VC 2005) V6</td><td>V7 以上</td></tr></table>
(UNIX)</td></tr>
</table>

	<table><tr><th>モジュール名</th><th>言語・利用機能</th><th>バージョン</th></tr><tr><td>THTPPCTL7</td><td>C++</td><td>V5</td></tr><tr><td>THTPPCTL8</td><td>C++</td><td>V6</td></tr><tr><td>THTPPCTL9</td><td>C++</td><td>V7</td></tr><tr><td>THTPPJAVA2</td><td>J2SE</td><td>全バージョン</td></tr><tr><td>THTPPOTS7</td><td>C++(TS 連携あり)</td><td>V5</td></tr><tr><td>THTPPOTS8</td><td>C++(TS 連携あり)</td><td>V6</td></tr><tr><td>THTPPOTS9</td><td>C++(TS 連携あり)</td><td>V7</td></tr><tr><td>THTPPDB7</td><td>C++(DB 連携あり)</td><td>V5</td></tr><tr><td>THTPPDB8</td><td>C++(DB 連携あり)</td><td>V6</td></tr><tr><td>THTPPDB9</td><td>C++(DB 連携あり)</td><td>V7</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により、自動的に復旧されます。 UNIX におけるカレントディレクトリは${AS_INSTALL}/Trnsv です。

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(UNIX)	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) javaw(UNIX)	統合運用管理ツール(otxadmingui)のプロセスです。 引数に「funcid=otxadmingui」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(UNIX)	配備ツール(deploytool)のプロセスです。 引数に「funcid=deploytool」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(UNIX)	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.objectbrokerservice.start
停止
otxadmin> invoke server.objectbrokerservice.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

※3 JMS サービスの起動・停止方法
起動
otxadmin> invoke server.jms-service.start
または
otxadmin> start-jms
停止

otxadmin> invoke server.jms-service.stop
または
otxadmin> stop-jms

NEC

WebOTX Application Server V7.1 構築ガイド

2009 年 12 月　第 4 版

日本電気株式会社