



WebSphere Application Server for z/OS V7.0

Overview

WebSphere software



© 2009 IBM Corporation

WAS for z/OS V7.0の新機能を中心とした、概要です。

免責事項

当資料は、2008年9月に発表されたWebSphere Application Server for z/OS Version 7.0 を前提として作成したものです。

当資料に含まれている情報は正式なIBMのテストを受けていません。また明記にしろ、暗黙的にしろ、何らの保証もなしに配布されるものです。

この情報の使用またはこれらの技術の実施は、いずれも使用先の責任において行われるべきものであり、それらを評価し実際に使用する環境に統合する使用先の判断に依存しています。

それぞれの項目は、ある特定の状態において正確であることがIBMによって調べられていますが、他のところで同じ、または同様の結果が得られる保証はありません。これらの技術を自身の環境に適用することを試みる使用先は、自己の責任において行う必要があります。

登録商標

1. AIX, CICS, Cloudscape, DB2, IBM, IMS, Language Environment, Lotus, MQSeries, MVS, OS/390, RACF, Redbooks, RMF, Tivoli, WebSphere, z/OS, zSeriesは IBM Corporation の米国およびその他の国における商標です。
2. Microsoft, Windows は Microsoft Corporation の米国およびその他の国における商標です。
3. Java, J2EE, JMX, JSP, EJB は Sun Microsystems, Inc. の米国およびその他の国における商標です。
4. UNIX はThe Open Groupの米国およびその他の国における登録商標です。
5. 他の会社名, 製品名およびサービス名等はそれぞれ各社の商標です。

目次

- ▶ WAS for z/OS 基礎知識
- ▶ WAS for z/OS V7.0 新機能
- ▶ 補足: マイグレーション関連、サポート関連

基礎知識: z/OS版のWASの特徴は、以前から受け継がれています。

新機能: WASの新機能のうち、z/OS版に特化した部分をフォーカスします。

補足: 既存のWASシステムからのバージョンアップについて記述します。

WAS for z/OS
基礎知識

WAS for z/OS によるオープン・メインフレーム

WASをz/OSで動かすメリットは、System zプラットフォームの長所を継承している点です

System zとz/OS

- ▶**信頼性**
 - ▶長年培った高い信頼性
- ▶**可用性**
 - ▶並列シスプレックスを核とする連続可用性の実績
- ▶**スケーラビリティ**
 - ▶並列シスプレックス構成による究極のスケール・アップ基盤
- ▶**運用管理**
 - ▶最高レベルのIT簡素化による、運用管理の容易性。
- ▶**セキュリティ**
 - ▶セキュリティ・サーバーと暗号エンジンによるハイ・セキュアな基盤

WAS for z/OS

オープンなアプリケーション基盤を、40年間培われたメインフレーム技術により堅牢に支える、比類のないIT基盤を実現



オープン・テクノロジー

- ▶**UNIX**
 - ▶UNIX95認証取得（ソース・レベルの互換性を提供）
- ▶**TCP/IP**
 - ▶N/Wの標準インターフェースの提供
- ▶**オープン基盤ミドルウェア**
 - ▶Web, LDAPなど
- ▶**Java & Java EE**
 - ▶バイナリー・レベルの互換性を提供
 - ▶Javaによるビジネス・トランザクション
- ▶**Webサービス**
 - ▶アプリケーション間の高度なインターオペラビリティの実現
- ▶**SOA**
 - ▶既存の資産を継承した、緩やかなアプリケーション統合

IBM System z および その主流オペレーティング・システムであるz/OSは、1964年に登場したSystem/360から40年間以上培われたメインフレーム技術により、高い信頼性、可用性を持ちます。

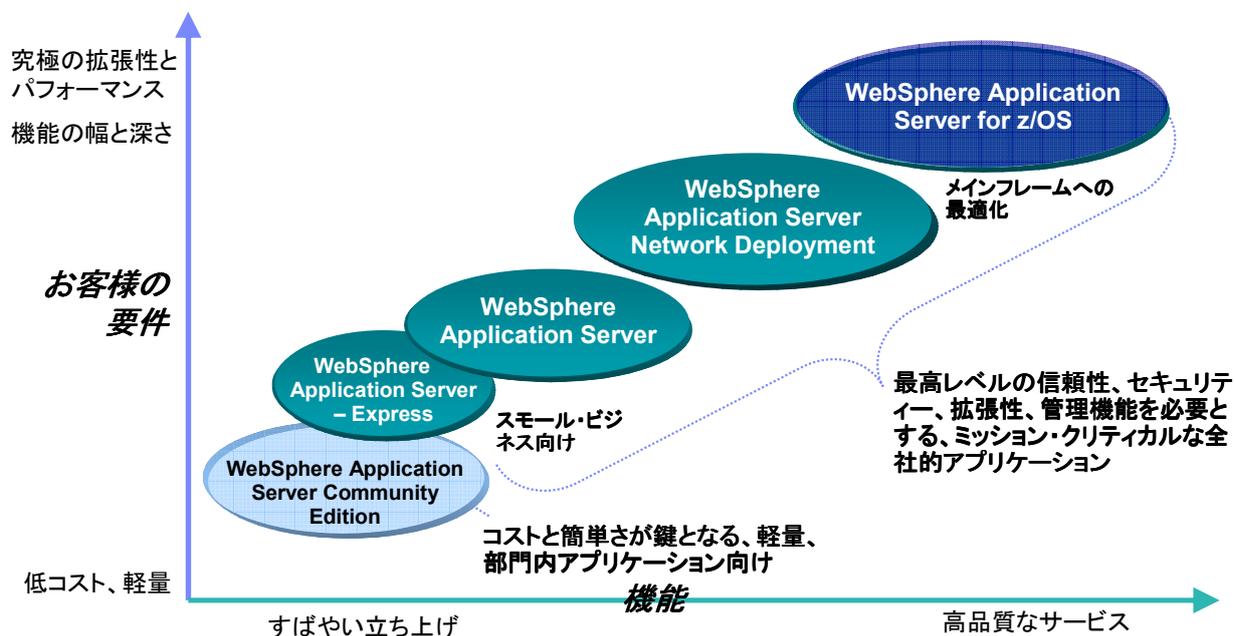
並列シスプレックス機能とは、複数システムを単一システムイメージ稼働することが出来る機能です。この機能により、高い可用性と究極のスケール・アップ基盤を兼ね備えています。

一方、z/OSは、UNIX System Services (USS) 機能やTCP/IPを取り込みながらオープンな環境に対応しています。

WAS for z/OSは、オープンなアプリケーション基盤を、長年培われたメインフレーム技術により堅牢に支える、比類のないIT基盤を実現するミドルウェアです。

各WAS製品の位置づけ

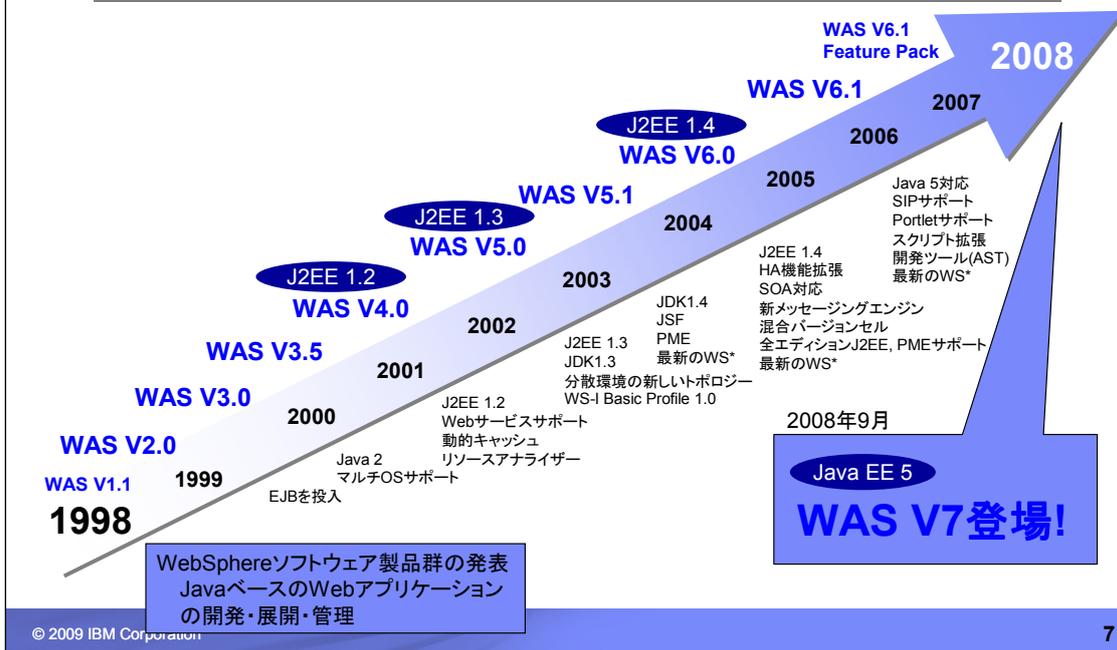
お客様の要件に合わせて選択できる幅広いアプリケーション基盤を提供しています



WASファミリーの中で、z/OS版は最上位の要件を満たすようなポジショニングがされています。

進化を続けるWAS for z/OS

バージョンアップにより、最新の標準仕様への対応、管理機能の強化、信頼性の向上



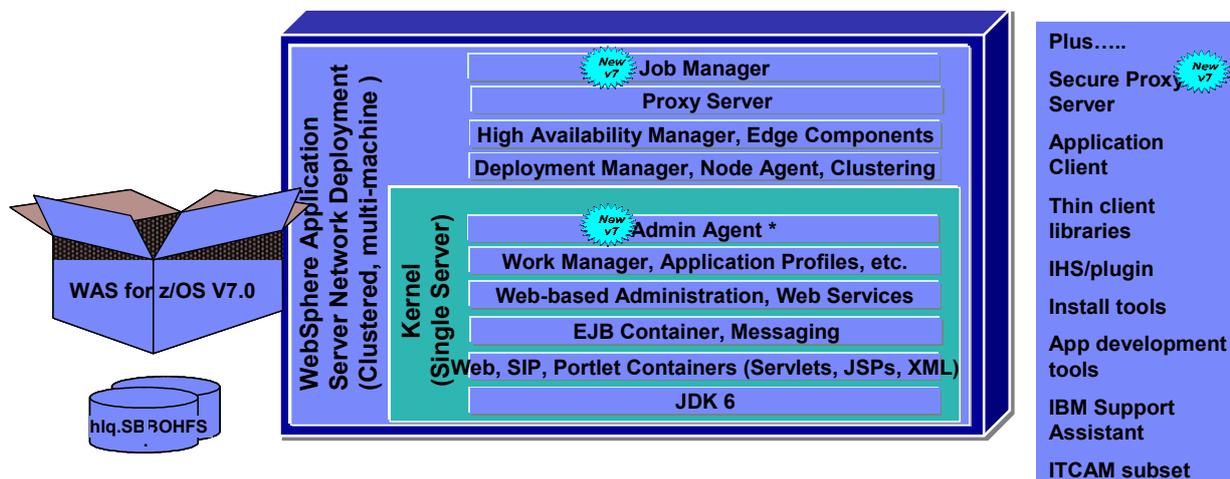
WebSphereソフトウェア製品群の発表
JavaベースのWebアプリケーション
の開発・展開・管理

V7はメジャー・バージョンですが、V5以降その基本的な設計は変わっていません。これは進化していないという意味ではなく、今後も新しいトレンドを取り込みながらお客様の資産を成長させていくことができることを意味しています。

WAS for z/OS 製品パッケージング

WAS V6のオフリングとほぼ同様のパッケージングを提供しています

- z/OS版のパッケージは分散系でのND版に相当
- z/OS版はSMP/Eによるテープからの導入
- V6.1までに存在したSBBLOAD等は撤廃され、実体はUSSのHFS/zFSのみとなった



このチャートでは、WAS V7で提供されるコンポーネントを示しています。

ほとんどのコンポーネントはWAS V6と同等ですが、追加された主なものとして次のものが挙げられます。

Admin Agent: WAS Express構成とBase構成のための集中管理エージェント

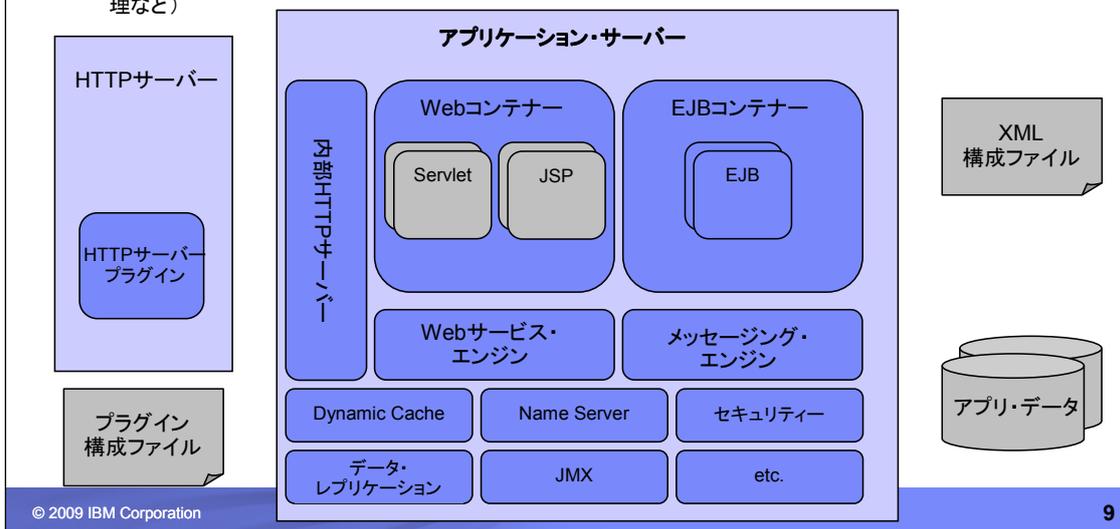
Job Manager: 運用、管理などの処理をジョブとして一箇所にキューイングし、複数Base/Express環境と複数NDセル環境に対して非同期で実行する管理コンポーネント

Secure Proxy Server: Controlledゾーン(非武装地帯)へ配置するためにセキュリティーを強化したProxy Server

WAS 基本アーキテクチャー

WASはJavaベースのトランザクション・サーバーとして総合的な機能を持ち合わせています

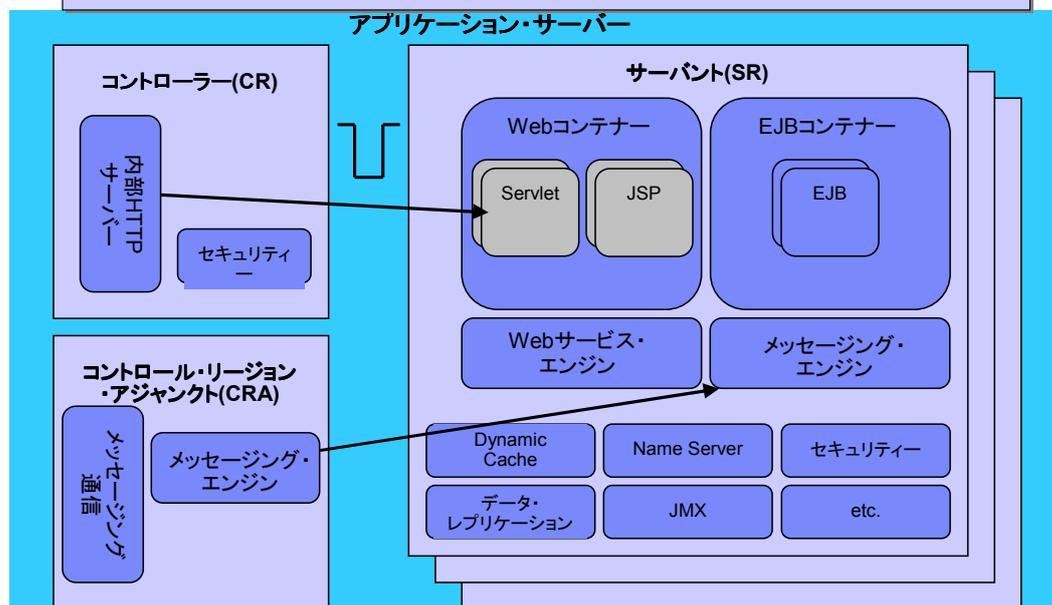
- WASはハイ・ボリュームでミッション・クリティカルな企業アプリケーションの稼働基盤を提供
 - J2EE アプリケーションの稼働環境
 - ビジネス・アプリケーションが快適に動作するためのサービス(データベース接続、スレッド、ワークロード管理など)



WASの基本知識として、サーバー内の論理的なコンポーネントの区分を示したものです。

WAS for z/OSの“サーバー”

WAS for z/OSは、サーバーごとに3つのリージョンに機能が分かれて動作します



z/OS版で最初に挙げられる違いは、この「CR/SR構成」と呼ばれるものです。ひとつのサーバーはCR (サーバーとしての窓口的な役割)と、SR (アプリケーションが走る場所)に分かれていて、SRは1つまたは複数のクローンを持つことができ、スケールアップに対応します。CRAはV6以降に追加されたもので、メッセージング機能であるSI BusやMDBリスナー (アクティベーション・スペック)が動作します。CRAはオプションで、これらの機能を使わない場合は不要です。

CRとSRの役割

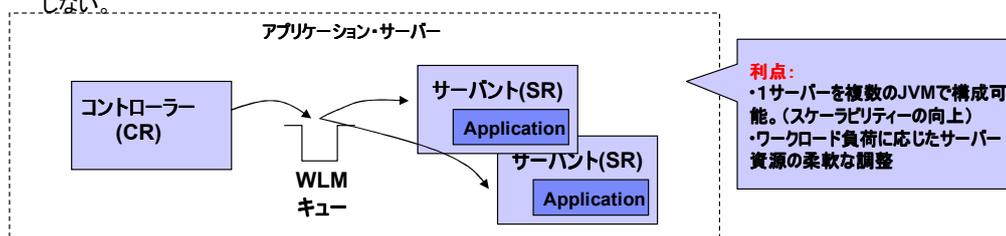
CRでトランザクションを受け付け、SRでアプリケーション処理が実行されます

CRの役割

- クライアントとの通信管理
- トランザクション管理
- セキュリティー管理

SRの役割

- J2EEアプリケーションの稼働環境 (JVM)
- CRとSR間はWLM Queue Managerサービスにより連携。ワークロードに応じて柔軟なSR数の調整が可能。
- リクエストのグループでサービス・クラスを分け、処理の優先順位づけを行うことも可能。(HTTP, JMS, IIOP)
- 個々の処理を行う「ワーカーレッドプール」の数は分散系と異なり、設定により固定値で拡張・縮小しない。



CR/SR構成のメリットは、

1サーバーだけの構成でも、スケーラビリティを持つことができる

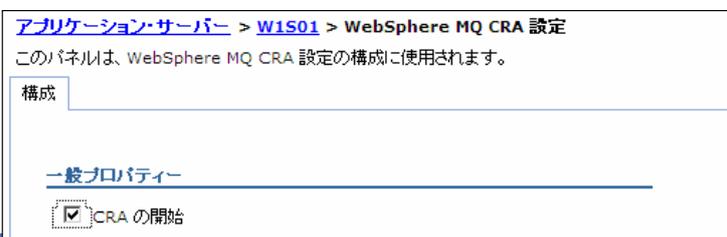
複数のSRを持つことで、可用性が高くなる(一つのSRがABENDしてもサーバーダウンに至らない)

ことにあります。

CRAの役割

CRAはWAS内蔵のメッセージング・エンジンが動作する場所です

- ▶ WAS内蔵のメッセージング・エンジン「SI Bus」が動作
 - サービス統合>バス でSI Busを構成
 - リソース>JMS>接続ファクトリー>Default messaging providerでJMS資源を定義
- ▶ MDB (Message-Driven Bean) の新しいリスナーである、「Activation Specification」が動作
 - Activation SpecがListenするキュー/トピック資源は、SI Bus及びWMQの両方が対象
 - SI Busを使わずWMQを使ってActivation Specを構成する場合、以下にチェック
 - サーバー>サーバー・タイプ>WebSphere Application Server>サーバー名>メッセージング>WebSphere MQ CRA 設定>CRAの開始
 - 従来の「リスナー・ポート」は存在(こちらはCRで動作)するが、V7で非推奨となった



CRAはその名のとおり、CRに近い役割ですが、SI BusやMDBリスナーが動作するため、CRよりも多めのメモリー消費となることがあります。

WAS for z/OS V7.0 新機能
-分散系との共通項目-

新機能（分散系との共通項目）

アプリケーションは最新の標準APIレベルに対応し、インフラ管理・運用が改善されました

アプリケーション開発

- Ease of Development
 - Java EE 5, EJB3.0, JPA
- 接続性が高く高機能なSOAアプリケーションの開発
 - JAX-WS2.1、JAX-B2.1、WS-I Reliable Secure Profile 1.0、WS-I Basic Profile 1.2/2.0

詳細はWAS V7アナウンスメント・ワークショップ資料をご参照ください（URLは最後のページに記載）

アプリケーションは分散系と同じAPIレベルで同じものがそのまま動作する

インフラ基盤構築

- サーバー構築
 - Windows PC上のWAS構成ツール（※1）によるサーバー構成情報（プロファイル）の入力・管理
 - プロパティファイルベース構成ツール
- サーバー管理
 - Job Manager、Admin Agent
- サーバー運用
 - WAS自体による監査機能
 - Fine-grained管理セキュリティ

※1 以下からダウンロードし、Windows PCにインストールして使用する
http://www-01.ibm.com/support/docview.wss?rs=404&context=SS7K4U&dc=D400&uid=swg24020368&loc=en_US&cs=UTF-8&lang=en

V7の新機能のうち、プラットフォーム間で共通の機能については、特にアプリ系についての詳細はここでは割愛しています。

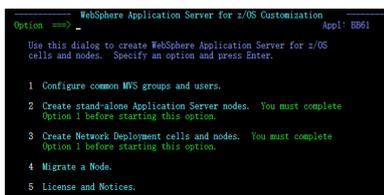
詳細につきましては、「アナウンスメント・ワークショップ資料」をご覧ください。

サーバー構成ツール

サーバー構成パラメータの入力がGUIベースで分かりやすくなりました。
複数のサーバー構成パラメータを一覧することができ、管理が容易になりました。

WAS V6.1までの手順

TSOにログオンして、構成パネルを実行し
構成パラメータを入力。JCLを生成する



構成JCLを順に実行し、サーバー構成
HFSを作成

サーバーを起動

WAS V7.0での手順

PCに「サーバー構成ツール」を導入。
構成パラメータを入力。JCLを生成し
FTPでz/OS上にアップロードする



構成JCLを順に実行し、サーバー構成
HFSを作成

サーバーを起動

インフラ系の新機能について、少し補足します。

V6.1まではTSOセッションにてセットアップJCLを作るためのパネルがありましたが、V7で廃止されました。

同等の機能を持つeclipseベースのGUIツールでJCLを作り、それをz/OS上にFTPでアップロードする必要があります。

プロパティ・ファイル・ベース構成ツール

多岐に渡るWASの構成パラメーターをバッチ処理風に変更管理する新しい方法です。

➤ 使用例

- 既存のサーバー構成からプロパティ・ファイル(config.props)を作成

```
wsadmin> AdminTask.extractConfigProperties(' -propertiesFileName  
config.props -configData Server=server1')
```

- テキスト・エディターを使ってプロパティを変更

```
## Configuration properties file for cells/HF70C/nodes/HF70N/servers/HF70S|server.xml  
## Extracted on Wed Oct 29 06:53:04 GMT 2008  
## Section 1.0  
## Cell=!(cellName):Node=!(nodeName):Server=!(serverName)  
## SubSection 1.0  
# Server Section#ResourceType=ServerImplementing  
ResourceType=Server  
ResourceId=Cell=!(cellName):Node=!(nodeName):Server=!(serverName)
```

- 変更したプロパティをserver1の構成に適用

```
wsadmin> AdminTask.applyConfigProperties( '[-propertiesFileName  
config.props]' )
```

- 構成の保管

```
wsadmin> AdminConfig.save()
```

この機能も運用関連で、WASのパラメータ変更管理に使うことができる、バッチ処理型の構成ツールです。

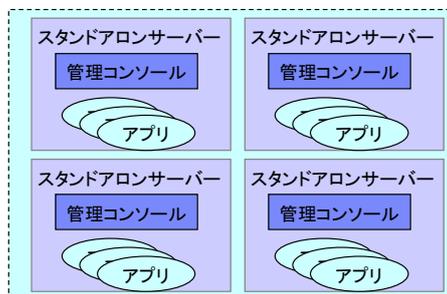
変更部分を整理しログできることから、z/OS技術者には馴染みやすいものといえます。

Admin Agent

スタンドアロン・サーバー構成がたくさんある場合に、管理コンソールを一元化できます。

- スタンドアロン・サーバーがたくさんある場合、管理コンソールも多くなってしまうという問題
- 管理コンソールは一箇所にしたい。- Admin Agentに一元化
 - 個々のスタンドアロンサーバーをAdmin Agentに登録する
 - 複数スタンドアロンサーバーの集中管理が可能
- 管理機能をAdmin Agentに集中させることで、各サーバーのフットプリントの縮小と起動時間の短縮を実現
 - 構成管理/アプリケーション管理/コマンド・マネージャー/ファイル転送/管理コンソールの機能を移行

現状



Admin Agent 導入後



スタンドアロン・サーバーはそれぞれに管理コンソール・アプリケーション (isclite.ear) が含まれるため、あまりスリムとはいえません。そこで、管理エージェント・サーバーを一箇所に置き、そのほかの多くのスタンドアロン・サーバーからisclite.earを不要とすることができます。

Job Manager

サーバーの構成管理をジョブとして実行するためのサーバー機能がJob Managerです。

- ▶ 運用、管理などの処理をジョブとして一箇所にキューイングし、非同期で実行する管理コンポーネント

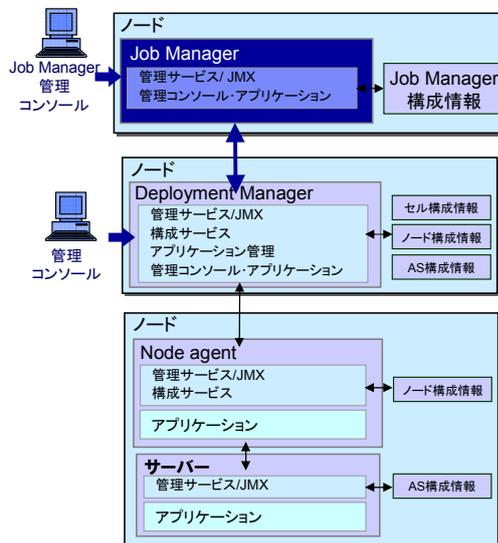
例：複数のセルにあるアプリケーション・サーバーを時間を指定して起動させる

- ▶ ジョブの非同期実行をサポート

- 指定した時刻に実行
- 特定の時間を越えたタイミングでの無効化
- 特定の時間間隔での繰り返し実行
- ジョブの完了通知の送信 (e-mail)

- ▶ 以下のジョブに関して柔軟な管理機能を提供

- アプリケーション・サーバーの始動・停止・作成・削除
- クラスター(メンバー)の始動・停止・作成・削除
- アプリケーションのインストール・アンインストール・更新・始動・停止
- ファイルの配布・収集・除去
- wsadminスクリプトの実行
- プロキシ・サーバーの作成・削除
- プロパティの構成
- インベントリ
- 状況



ジョブ・マネージャーはサーバー構成や管理を自動スケジュールするための新機能です。

監査機能

WAS内部でセキュリティーのログを取ることができるようになりました。

- ▶ 管理コンソール操作や wsadminの管理操作を内部的にログする機能
- ▶ アプリケーションのセキュリティーログを取る
- ▶ サードパーティのセキュリティー製品と連携できる
- ▶ 管理ロールを、管理者と分けて Auditorという名前で新設

Audit Records		
Hostname think . ReportTime Sep 1, 2008, 21:27:40		
Record Number	Event Type	Outcome
0	SECURITY_RESOURCE_ACCESS	SUCCESS
CreationTime=Mon Sep 01 21:20:04 CDT 2008	Action=preinvoke MBean	ProgName=Server (module)
RegistryType=null	Domain=global	Realm=defaultWIMFileBasedRealm
RemoteAddr=null	RemotePort=null	RemoteHost=null
ResourceName=getState	ResourceType=SM_MBEAN	ResourceUniqueId=0
1	SECURITY_RESOURCE_ACCESS	SUCCESS
CreationTime=Mon Sep 01 21:20:04 CDT 2008	Action=preinvoke MBean	ProgName=Server (module)
RegistryType=null	Domain=global	Realm=defaultWIMFileBasedRealm
RemoteAddr=null	RemotePort=null	RemoteHost=null
ResourceName=getState	ResourceType=SM_MBEAN	ResourceUniqueId=0
2	SECURITY_AUTHN	REDIRECT
CreationTime=Mon Sep 01 21:20:31 CDT 2008	Action=formlogin	ProgName=iscLite
RegistryType=WIMUserRegistry	Domain=global	Realm=defaultWIMFileBasedRealm
RemoteAddr=127.0.0.1	RemotePort=1064	RemoteHost=localhost
ResourceName=POST	ResourceType=web	ResourceUniqueId=0
3	SECURITY_RESOURCE_ACCESS	SUCCESS
CreationTime=Mon Sep 01 21:20:31 CDT 2008	Action=resourceAccess	ProgName=/loginError.jsp
RegistryType=WIMUserRegistry	Domain=global	Realm=defaultWIMFileBasedRealm
RemoteAddr=127.0.0.1	RemotePort=1064	RemoteHost=localhost

監査機能は、グローバル・セキュリティをオンにした場合に動作します。
V7の新機能ですが、現状での使い勝手はまだ途上段階といえます。

WAS for z/OS V7.0 新機能
- z/OS版固有部分 -

新機能（z/OS版の機能拡張）

WAS for z/OS独自の新機能は、より高い信頼性・高い管理の可視化が求められる環境で発揮されます。

- XCFによるHAマネージャー障害検知
 - コア・グループ・メンバー障害検知のプロトコルにXCFを使用可能
- FRCAサポート(Fast Response Cache Accelerator)
 - WAS HTTPトランスポートでFRCAが使用可能
 - 静的/動的コンテンツをTCP/IPでキャッシュ
- SMFレコード120 type9の新設
 - 従来のSMF120より詳細なパフォーマンスデータが取得できる
- WOLA(WebSphere Optimized Local Adapters)
 - WASと同じOS下にある外部アドレス空間(バッチ・CICSなど)とをクロスメモリー接続し、高速・相互に呼び出しができる機能拡張。
- その他の機能拡張

z/OS版の新機能については、詳細を記述します。
このページはその見出しです。

XCFによるHAマネージャー障害検知

WAS v6まではTCP/IPレベルのみでハートビートするHAマネージャーの機能改善です

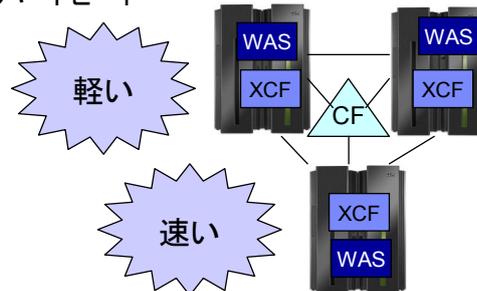
➤ XCFによるHAマネージャー障害検知

- コア・グループ・メンバー障害検知の protocols にXCFが使用可能
 - コア・グループの全てのサーバーがWAS for z/OS V7.0であること
 - コア・グループの全てのサーバーが、ひとつのSysplex上で稼働していること
 - XCFによるHAマネージャー障害検知の設定がされていること(デフォルトではない)
- サーバーは起動時にXCFグループのメンバーとしてJOINします

➤ XCFを使用しない障害検知は、TCP/IPでのハートビート

➤ メリット

- 従来のハートビートの仕組みに比べ
 - CPU使用率の軽減
 - 障害検知時間の短縮



- ※ XCF: Cross System Coupling Facility
- ※ XCFグループ: Sysplexの中の複数OS間のシグナリング・サービスを使用して連携を取る機能グループ

ひとつめは、“XCFによるHAマネージャー障害検知”です。

従来のHAマネージャーでの障害検知は、コア・グループ・メンバーがActiveであることを確認するために、メンバー間でTCP/IPプロトコルによるハートビートが行われていました。

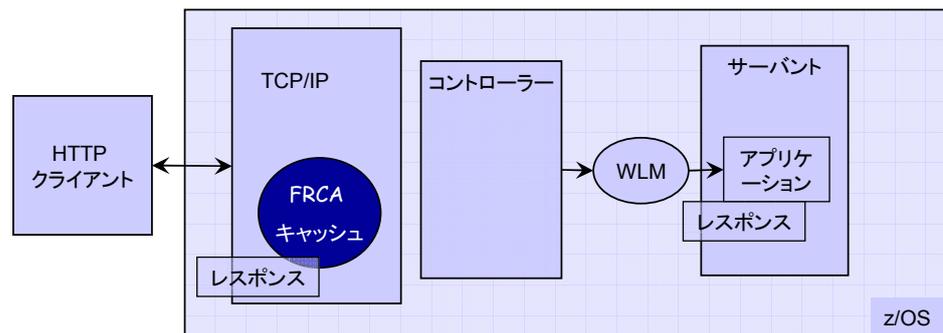
WAS for z/OS V7で新しく登場した“XCFによるHAマネージャー障害検知”では、WASがXCFグループのメンバーとして参加することで、障害検知にXCFを利用することができるようになりました。XCFグループとは、Sysplexの中の複数OS間のシグナリング・サービスを使用して連携を取る機能グループです。

“XCFによるHAマネージャー障害検知”では、従来のハートビートの仕組みに比べ、CPU使用率を軽減でき、障害検知時間を短縮することができます。

FRCAサポート

FRCAはz/OSで実績のあるキャッシュ高速化技術で、WASがこれを使用可能です。

- WAS HTTPトランスポートでFRCAが使用可能
 - z/OSではIBM HTTP Serverで利用されている技術
 - 実績のあるz/OS技術の活用
 - 静的/動的コンテンツをTCP/IPでキャッシュ
- メリット
 - レスポンス・タイムの向上
 - CPU使用率の軽減
- 前提
 - z/OS V1.9



FRCA(Fast Response Cache Accelerator)

AFPA(Adaptive Fast-Path Architecture)と呼ばれることがあります。

FRCAは様々なプラットフォームで使われている技術です。HTTPリクエストへのレスポンスをTCP/IP層にキャッシュしてTCP/IPからHTTPリクエストへのレスポンスを返すことができます。

z/OSでは、z/OSのベース・エレメントであるIBM HTTP Serverで多く利用されている技術です。(WAS付属のHTTP Server (Apacheベース) ではありません)

WAS for z/OS V7では、HTTPトランスポートでFRCAが可能になりました。FRCAキャッシュは動的キャッシュ(DynaCache)を通して構成されます。静的オブジェクトと同様にサーブレットとJSPなどの動的オブジェクトもキャッシュできます。

この機能はz/OS V1.9で利用可能です。

SMFレコード120-9

SMFにWASが出力するパフォーマンス情報が、より柔軟に取れるようになりました。

- ▶ SMFレコード120とは
 - WAS for z/OSが出力するパフォーマンス情報
- ▶ V6までのSMFレコード120(タイプ1,3,5,6,7,8)は
 - SMF Activity Recordが取得の場合はパフォーマンスへの影響が大きい
 - 設定の変更にはサーバーの停止・再始動が必要
- ▶ V7で取得可能になったSMFレコード120タイプ9は
 - パフォーマンス・データ取得のオーバー・ヘッドが軽減
 - データの取得、未取得をサーバーの再起動なしに、動的に設定可能

※SMF: System Management Facility
 ※SMFは、z/OS上のサブシステムが出力する情報をファイルに書き出すz/OSサービス

内容	スコープ	レコード・タイプ	主な出力データ
サーバー	サーバー毎	タイプ1(Activity) タイプ3(Interval)	・起動されているサーバー・リージョンの数 ・接続クライアント・セッション数 ・セッション毎のデータ転送量
EJBコンテナ	アプリケーション毎	タイプ5(Activity) タイプ6(Interval)	・メソッド毎のコール数 ・平均、最大応答時間
Webコンテナ	Webアプリケーション(warファイル)毎	タイプ7(Activity) タイプ8(Interval)	・HTTPセッション数 ・サーブレット毎のコール数 ・サーブレット毎の平均、最大応答時間
リクエスト	リクエスト毎	タイプ9	・タイプ1,5,7と同様のデータ

**New
V7**

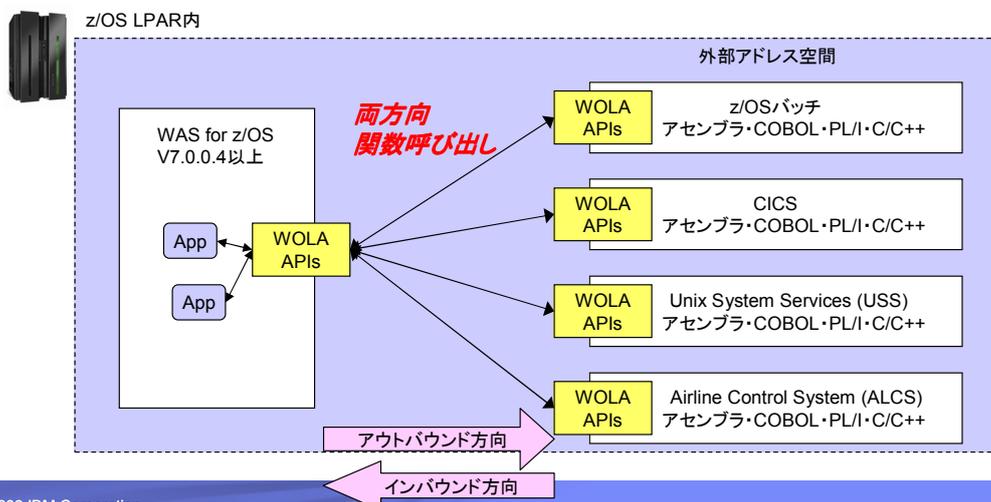
SMF(System Management Facility)は、z/OS上のサブシステムが出力する情報をファイルに書き出すz/OSサービスです。

V6までのSMFレコード120(タイプ1,3,5,6,7,8)は、SMF Activity Recordが取得の場合はパフォーマンスへの影響が大きいものでした。

V7で取得可能になったSMFレコード120タイプ9は、パフォーマンス・データ取得のオーバー・ヘッドが軽減され、動的設定が可能になりました。

WOLA (WebSphere Optimized Local Adapters)

- WOLA=WAS for z/OS V7と同一のz/OS LPAR(ローカル)にあるバッチジョブやサブシステムの外部アドレス空間と、相互に関数呼び出しを行うための機能拡張
- WASから外部アドレス空間(WASからみたアウトバウンド)と、外部アドレス空間からWAS(インバウンド)の両方向に呼び出せる点が新しい機能。



WOLAはWAS V7のPTFとして提供されています(7.0.0.4)。

ちょっといい話 その1

WASを制御するMVSコマンドがより便利になりました。

▶ サーバント数を動的に変更するMVSコマンド

- サーバント数変更でサーバーの再起動が不要になった
- 注: 複数SRモードで稼働している必要がある
 - 設定: 管理コンソール: アプリケーション・サーバー > server_name > サーバー・インフラストラクチャー > サーバー・インスタンス → 複数インスタンス使用可能 にチェック
- コマンド例: F WAS7A,WLM_MIN_MAX=2,2
- 実行例: SDSF DAでのアドレス空間

```
JOBNAME StepName ProcStep JobID Owner
WAS7AD WAS7AD BBODAEMN STC00906 WSCRUI
WAS7AS WAS7AS BBOPACR STC00918 WSCRUI
WAS7ASS WAS7ASS BBOPASR STC00922 WSSRU1
WAS7ASS WAS7ASS BBOPASR STC00923 WSSRU1 ← コマンドで動的追加されたサーバント
```

▶ STACKTRACE出力のMVSコマンド

- ハングしているような場合での問題初期切り分けに有効(Javacoreの簡易版)
- コマンド例: F WAS7A,STACKTRACE
- JOBLIST STDERRに全Javaスレッドのスタック・トレースが出力される
- 出力例: JOBLIST STDERR STACKTRACE

```
BBOJ01171: JAVA THREAD STACK TRACEBACK FOR THREAD WebSphere:ORB.thread.pool t=005cb108:
Traceback for thread WebSphere:ORB.thread.pool t=005cb108:
com.ibm.ws390.orb.CommonBridge.nativeRunApplicationThread(Native Method)
com.ibm.ws390.orb.CommonBridge.runApplicationThread(CommonBridge.java:459)
```

- ・サーバント数が動的に変更できるようになりました
- ・コマンド起動で、任意のタイミングで、全Javaスレッドのスタック・トレースを出力できるようになりました

ちょっといい話 その2

SRの同時起動やスレッド数の調整がより柔軟になりました。

▶ サーバント・パラレル・スタート

- wlm_servant_start_parallelカスタム・プロパティ
- 2つめ以降のサーバントの平行起動
- 設定:管理コンソール:環境 > WebSphere変数 > 新規作成
 - プロパティ名: wlm_servant_start_parallel
 - 値:
 - 1: 2つめ以降のサーバントの平行起動
 - 0: 全てのサーバントを順次起動
- 前提: z/OS V1.9

▶ サーバント・ワーカー・スレッド数の任意指定

- servant_region_custom_thread_countカスタム・プロパティ
- 設定:管理コンソール:環境 > WebSphere変数 > 新規作成
 - プロパティ名: servant_region_custom_thread_count
 - 値: サーバント・ワーカー・スレッド数指定 (1 - 100)
- 注:ワークロード・プロファイル=CUSTOMで稼働している必要があります(デフォルト=CUSTOM)
 - ワークロード・プロファイル設定:管理コンソール:アプリケーション・サーバー > server_name > コンテナ・サービス > ORBサービス > z/OS追加設定 →ワークロード・プロファイル(プルダウン選択)
- 参照:V6.1までのサーバント・ワーカー・スレッド数指定: Workload Profile

ワークロード・プロファイル	スレッド数	CPU (zAAP) が2つの場合
ISOLATE	1	1
IOBOUND (デフォルト)	MIN(30, MAX(5, CPU数*3))	6
CPUBOUND	MAX((CPU数-1), 3)	3
LONGWAIT	40	40

- サーバントの平行・スタートができるようになりました

z/OS V1.9のWLM機能を利用します。

z/OS V1.8以前の環境ではwlm_servant_start_parallelカスタム・プロパティ設定は、意味を持ちません。

- サーバント・ワーカー・スレッド数の任意指定ができるようになりました

ちょっといい話 その3

EJBクライアントを非J2EEコンテナのスタンドアロンで実行する方法が簡単になりました。

➤ EJBシンクライアントの提供

- EJBクライアントをJ2SE環境で実行するためのJARを提供
- launchClientはEJBクライアントとして必要なくなった

z/OS client – z/OS serverの場合のクライアント設定

- ・JavaはWASv7内蔵のJava6を使用し、LIBPATHを追加設定する必要あり

```
IBMUSER#home/hide/work>export LIBPATH=/usr/lpp/zWebSphere/V7R0/lib/modules:$LIBPATH
IBMUSER#home/hide/work>export PATH=/usr/lpp/zWebSphere/V7R0/java/bin:$PATH
IBMUSER#home/hide/work>java -classpath /usr/lpp/zWebSphere/V7R0/runtimes/com.ibm.ws.ejb.thinclient.zos_7.0.0.jar:InteropEJB.jar:InteropClient.jar -
Djava.naming.provider.url=iiop://localhost:2809 -Djava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory
com.ibm.samples.interop.client.InteropClient ejb/InteropEJBHome
```

Windows client – z/OS serverの場合のクライアント設定

- ・JavaはIBM Java 5/6が使用可能。Sun JDKの場合はcom.ibm.ws.orb_7.0.0.jarもCLASSPATHに必要
- ・com.ibm.ws.ejb.thinclient_7.0.0.jarはWindows版WASやRAD7.5のruntimesディレクトリにあるものを使用する

```
D:\DOC\SAMPLES>java -classpath com.ibm.ws.ejb.thinclient_7.0.0.jar:InteropEJB.jar:InteropClient.jar -Djava.naming.provider.url=iiop://wsz2:2809 -
Djava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory com.ibm.samples.interop.client.InteropClient ejb/InteropEJBHome
CLIENT MESSAGE - Client is running from hostname AA2893302 at ipaddress 9.170.247.110
WAS Initial Context created now looking up ejb/InteropEJBHome
Looking up and narrowed ejb/InteropEJBHome complete
Creating remote object
Calling remote method getMessage()
SESSION BEAN Message - If you see this message you made the remote call to session bean at hostname = WSZ2 at ipaddress = 192.168.43.132
```

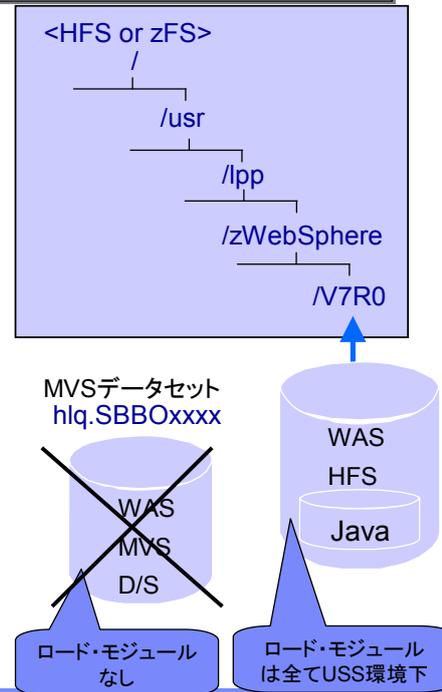
今までは、J2SE (スタンドアロンJava) でEJBクライアントを動かす場合に、かなり複雑な手順が必要でした。

V7ではクライアント・コンテナと呼ばれるlaunchClient.sh/batは不要となり、PATH指定のみで実行できます。

WAS for z/OSの実体について

製品ライブラリーはHFS/zFSに一元化され、PDS/Eロードモジュールは撤廃されました。

- 製品ライブラリーのロード・モジュールを全てUSS環境下のファイルへ
 - SBBLOAD, SBBOLPA, SBBOLD2は、USS環境下のファイルへ移動(HFS or zFS)
 - 導入や運用の複雑さを軽減



製品ライブラリーのロード・モジュールが全てUSS環境下のファイルとして提供されるようになりました。

WAS for z/OSの実体について

PDS/Eロードモジュールがなくなり、USS内のロードモジュールがプロセスの実体です。

- Q. WAS for z/OS V7.0はUnixプロセスとして、どのように見えているか？
- A. USSランチャー(BPXBATSL/BPXBATA2)とその子プロセス
 - 以前のバージョンでは、独立したモジュール(hlq.SBBOLOAD内のBBOCTL/BBOSR/BBODAEMN)が実体で、その中にJVMがロードされていた

ps -efコマンドの結果

UID	PID	PPID	C	STIME	TTY	TIME	CMD	
ASCR1	50462746	1	-	01:24:05	?	0:14	BPXBATA2	
ASCR1	131104	50462746	-	01:24:05	?	0:14	/WebSphere/W1S01/AppServer/lib/bboctlm 00 REC=N	CR
ASCR1	16908321	1	-	01:24:07	?	0:00	BPXBATA2	
ASCR1	131106	16908321	-	01:24:07	?	0:00	/WebSphere/W1S01/AppServer/lib/bbodmnm	Daemon
ASSR1	16908323	1	-	01:24:26	?	0:24	BPXBATSL	
ASSR1	131108	16908323	-	01:24:26	?	0:24	/WebSphere/W1S01/AppServer/lib/bboosrnr 64	SR

今までのバージョンのWASをお使いの方は、起動JCLにあったEXECステートメントのBBOCTL/BBOSRがなくなってV7はどうなっているのかとお考えでしょう。答えは、USSのバッチシェルである、BPXBATCHでbboctlm,bboosrnrという、これはCで書かれたロードモジュールを呼び出し、その中でJavaがロードされています。従ってネイティブであることには変わりありません。

zAAPに関する改善点

z/OSの機能改善によって、zAAPの設定方法がよりわかりやすく柔軟になりました。

- ▶ zAAPはJavaの処理をするための専用CPUで、コストメリットがある
- ▶ z/OS 1.9以降で、IEAOPTxxの指定方法とzAAPの動作が変更されている
 - IFACROSSOVER=YES/NO は廃止
 - IFAHONORPRIORITY=YES/NO に一本化
 - YES(デフォルト)の場合は、zAAPが溢れた場合にGCPを使用する
 - ZAAPAWMT=n (n=1-499000 microseconds, デフォルトは12000)
 - AWM=Alternate Wait Management
 - 減らした場合、レスポンスタイムの向上が期待できるがGCPへ溢れる量が増える
 - レスポンスタイムが十分で、GCPへの溢れを減らしたい場合、この値を増やす
 - NOを指定すると、zAAPにオフロードできるものは、GCPIには溢れない
- ▶ z/OS 1.9以降であればzAAPに対応するWASで有効

WASそのものとは少し離れますが、Java専用プロセッサのzAAPの扱いがOSレベルによって変更されています。

IEAOPTxxの指定の仕方が、以前はわかりにくかったものが、シンプルになりました。

前提ハードウェア・ソフトウェア

前提となるハードウェア、ソフトウェア情報です。

▶ 前提ハードウェア

- プロセッサ： 前提のz/OSをサポートするハードウェア
- メモリー： 最小 512MB、1GB以上推奨
- ディスク： 約7GB (詳細はProgram Directoryを参照)

▶ 必須ソフトウェア

- ▶ z/OS V1.7, V1.8, V1.9, V1.10

▶ オプションのソフトウェア

- ▶ CICS TS V2.3以降
- ▶ CICS TG V6.0以降
- ▶ IMS V9以降
- ▶ DB2 V8.1以降
- ▶ WebSphere MQ V6以降

▶ 導入方法

- SMP/E導入の前提
 - z/OS V1.7, V1.8, V1.8, V1.10の何れか(2009年4月現在)
 - Java 2 Technology Edition SDK 1.4以降
 - IBM SMP/E for z/OS V3.3以降

WAS for z/OS V7の前提ハードウェア、前提ソフトウェアです。

詳細は、Program Directoryおよび関連ドキュメントをご参照ください。

<http://www.ibm.com/software/webservers/appserv/was/library/v70/was-zos/index.html#Product%20documentation>

まとめ

WAS for z/OS V7.0の新機能を概説しました。

- ▶ WAS for z/OSは、オープンなアプリケーション基盤を、40年間培われたメインフレーム技術により堅牢に支える比類のないIT基盤を実現します
- ▶ WAS for z/OS V7は、プラットフォーム共通の機能拡張に加え、高可用性、高パフォーマンスの追加機能を提供します
 - XCFによるHAマネージャー障害検知 → 障害検知
 - スレッド・ハング・リカバリー → 障害排除
 - FRCAサポート → パフォーマンス向上
 - SMFレコード120-9 → 詳細なパフォーマンス・データ取得

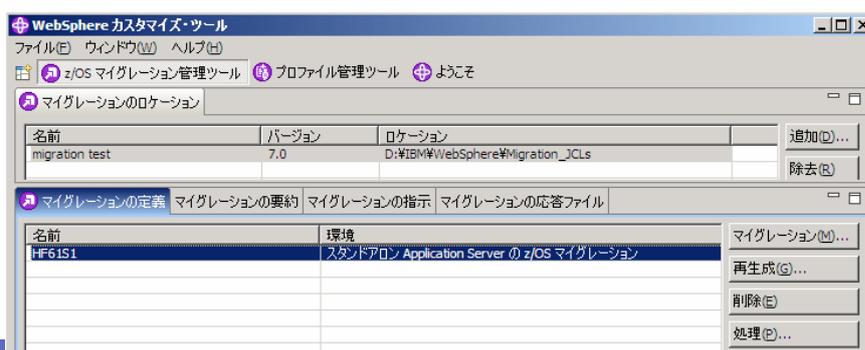
WAS for z/OS V7は、プラットフォーム共通の機能拡張に加え、高可用性、高パフォーマンスの追加機能を提供します。

補足
マイグレーション関連
製品サポート関連

サーバー構成ファイルの移行

WASのバージョンアップによって、サーバー構成ファイル群も移行が必要です。

- V5.0以前
 - マイグレーションの自動化ツールは提供されていないため、手作業でV7の新規サーバー構成を作り、移行済みアプリケーションをデプロイする
- V5.1以降
 - WebSphereカスタマイズ・ツールの「マイグレーション管理ツール」にてマイグレーションJCLを生成でき、このJCLを実行することで自動マイグレーションが可能。
 - ND構成の場合、旧バージョンのノードとの混合セルをサポート。この場合、まずデプロイメント・マネージャーをV7にする



© 2009 IBM Corporation

35

既存のサーバー構成を新バージョンに移行するには、カスタマイズされ稼働中の既存のサーバー構成ファイル（構成HFS）の移行が必要です。

新バージョンの元で新規セットアップを行い、管理コンソールで構成パラメータを一つずつ設定していく方法でもよいのですが、ツールとJCLにて構成HFSを一挙に変換する方法があります。

この場合、既存の構成HFSとは別に新バージョンの構成HFSが作られますが、念のため既存の構成HFSはバックアップを取得してから作業してください。

ユーザー・アプリケーションの移行 (WASバージョンview)

WASのバージョンアップによって、サポートAPIレベルが上がるため、既存のアプリケーションは下位互換性を確認してください。

- アプリケーションがPure Javaである場合、分散系の情報がz/OSでも適用されます
- WAS V6.1からの移行
 - 特に変更なくV7に移行できる
- WAS V6.0以前からの移行
 - J2EE 1.4にRADなどのツールでマイグレーションする
 - JDK1.4とJava5/6の間のAPI変更が少しあるため注意。分散系のマイグレーション資料がz/OS版でも適用可能。
 - 詳細はsunのドキュメントで確認
- 一般的にV7はJ2EE1.4以前のレベルとの互換性は高い
 - 多くのV4アプリケーションは、変更無しで稼働可能
 - ほとんどのV5/V6 のアプリケーションは、変更無しで稼働可能
 - (詳細は次ページ)

一方、アプリケーションの移行は個別にチェックが必要です。WASの固有APIを使っているアプリケーションでは、数は少ないものの、**deprecated** (非推奨) や **removed** (使えなくなる) 場合があり、アプリケーションの修正が必要となります。

ユーザー・アプリケーションの移行 (API spec view)

古いAPIレベルのユーザーJavaアプリケーションは、一部修正を要する場合があります。

➤ Webアプリケーション

- Servlet 2.2以降、JSP1.2以降のアプリケーションは基本的にWASv7のWebコンテナで稼働します。ただし、以下のものは考慮点があります
 - PageListServlet(.servletファイル)で画面遷移する作りのもの
 - encodeRedirectURLやコンテキストルートを含まない形でresponse.sendRedirectメソッドを呼んでいるもの
 - デフォルトのContent-TypeレスポンスヘッダーやgetWriterの後にsetContentTypeメソッドで文字コードを指定しているもの。こういったアプリケーションはWASのバージョンによって挙動が変わりますが、Webコンテナのカスタムプロパティに以下を設定することでWASの特定のバージョンと同じ挙動にすることができます。
 - 名前 com.ibm.ws.webcontainer.contenttypecompatibility
 - 値 V4, V5, V6, V7のいずれか
 - WAS V5.xから移行する場合、JSPの再コンパイル (precompile) が必要。
 - 詳細は以下のURLを参照のこと
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/tweb_migrate2.html

➤ EJBアプリケーション

- EJB 1.1, 2.0, 2.1のSession Beanは、そのままWASv7のEJB3.0コンテナ上で動作します。
- Entity Bean 1.1形式のものはEJB3.0で非推奨となり、動作が保証されません。Entity Bean2.x形式に変換してください。

JavaやJ2EEといった標準においても同様に、API仕様が追加・変更されているものがありますので、注意してください。

製品サポート情報

WAS6.0以降は、サポート期間が長くなりました。

- WAS for z/OS V7.0のサービス(集積PTF)は、基本的に4半期に1回もしくは2回更新される
- サービスはWASプラットフォーム間で同期をとって出荷される

〔サポート期間マトリックス〕

2009/09現在

WASバージョン	4.0.1	5.0	5.1	6.0	6.1	7.0
発表日	2001/09/26	2003/04/09	2004/05/26	2005/03/25	2006/04/12	2008/09/10
出荷開始日	2001/11/12	2003/05/02	2004/05/28	2005/03/25	2006/06/30	2008/09/26
営業活動終了	2003/12/31	2005/09/07	2007/02/28	2009/02/23	n/a	n/a
サポート終了	2005/04/30	2006/09/30	2008/09/30	2010/09/30	n/a	n/a
サポート・ポリシー (基本+延長)	3年+2年	3年+2年	3年+2年	5年+3年	5年+3年	5年+3年

WASにPTFを適用する場合、**fixpack**というサービスレベル単位(7.0.0.4など)で適用してください。これは、実体としてはUKxxxxのような個別PTFの組み合わせです。

機能サポート・マトリックス

前提OSレベルと接続するミドルウェアのドライバーレベルに注意してください。

機能	4.0.1	5.0.2	5.1	6.0.2	6.1	7.0
Javaバージョン	1.3.1	1.3.1	1.4.2	1.4.2	5	6
zAAP	X	X	※1	※1	O	O
コモン・コード	X	X	X	O	O	O
シェアード・クラス	X	X	X	X	O	O
DB2 JDBC ユニバーサル・ドライバー	X	O	O	O	O	O
DB2 JDBC レガシー・ドライバー	O	O	O	O	X	X
JMS接続の前提WMQ	~V5	~V6	~V6	~V7.0.1	~V7.0.1	V6, V7.0.1 WAS内蔵RAR ※2
サポートJ2EEレベル	1.2	1.2-1.3	1.2-1.3	1.2-1.4	1.3-1.4	1.3-5.0
IHS (IBM HTTP Server) plugin	Domino GO	Domino GO	Domino GO	Domino GO	Domino GO / Apache	Domino GO / Apache
64ビット・アドレッシング	X	X	X	X	O	O
サポート最小OSレベル	OS/390 2.8	OS/390 2.10	z/OS 1.2	z/OS 1.4	z/OS 1.6	z/OS 1.7

※1 z/OS 1.6以上でサポート

※2 詳細は <http://www-06.ibm.com/jp/domino01/mkt/cnpages1.nsf/page/default-0004CCDF>

旧バージョンからの移行の場合、WASのサポートするミドルウェア(DB2、CICSなど)接続においては、その接続相手のバージョンアップが必要となる場合があります。

非推奨機能、除去された機能

以前のバージョンからの移行の場合、非推奨 (Deprecated) 機能と除去 (Removed) された機能には注意が必要です。

▶ WAS for z/OS V7で新たに非推奨になった機能

カテゴリー	WAS for z/OS V7非推奨機能	推奨されるマイグレーション・アクション
サーバー構成	31ビット・アドレッシング・モード	64ビット・アドレッシング・モードへ移行 ・V7新規作成サーバーはデフォルトで64ビットで稼働 ・以前のリリースからのマイグレーション・サーバーは31ビットで稼働可能

▶ 以前のバージョンで非推奨になり、WAS for z/OS V7で除去された機能

カテゴリー	除去された機能	推奨されるマイグレーション・アクション
システム管理	ISPFカスタマイゼーション・ダイアログ (サーバー作成JOBやマイグレーションJOBの生成)	サーバー作成JOB ・WebSphere 構成ツール あるいは ・zpmmtコマンドを使用 マイグレーションJOB ・WebSphere 構成ツール あるいは ・zmmtコマンドを使用

WAS for z/OS V7非推奨機能、除去された機能

- ・サーバーの31ビット・アドレッシング・モードでの稼働は非推奨になりました。
- ・ISPFカスタマイゼーション・ダイアログは、V7では除去されました。

除去される予定はないが、今後改良もされない機能

V7から、Stabilizedという機能カテゴリーができました。

- Stabilized (安定した・枯れた) 機能として、今後は積極的な改善はしない
 - 同様の機能を引き継ぐ別のAPIが出てきたもの
 - JACLによるシステム管理 (wsadmin) 機能は、非推奨ではなくなったため、移行が必須ではありません。

カテゴリー	Stabilized機能	今後積極的に支持する代替の機能
プログラミングモデル	EJBに含まれる、Entity Bean 1.xおよび2.x、コンテナー管理パーシステンス(CMP)およびBean管理パーシステンス(BMP)の両方が含まれる。	JPA (Java Persistence API) を使用してEJBのデータベースアクセス機能を実装する。 JPAはEJB Entity Beanと比べると大幅にシンプルなプログラミングモデルとなっている。
プログラミングモデル	JAX-RPC (Java API for XML-based RPC) JavaによるWebサービスAPIの一つ。かつては主流であった。	JAX-WS (Java API for XML Web Services) 今後、JAX-RPCとJAX-WSのブリッジ機能は残るが、新しい機能サポートはJAX-WSのみに集中させ、JAX-RPCの機能拡張は行われない。
システム管理	JACL言語によるシステム管理 (wsadmin) 機能	Jython言語によるシステム管理 (wsadmin) 機能

標準として採用したのに、その後WASのサポートが別の標準に移ったために当初の標準が非推奨になってしまう。というジレンマは、過去に指摘されていたことです。これに対し、**Stabilized**機能として扱うことで、今後も非推奨とならないように改善されました。

参考資料

- WebSphere Application ServerのLibraryサイト(インフォセンターへのリンクなど)
 - <http://www.ibm.com/software/webservers/appserv/was/library/>
- WAS for z/OSバージョン別 APAR/PTF 早見表
 - <http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006970>
- WAS for z/OS 製品サポートサイト
 - http://www.ibm.com/software/webservers/appserv/zos_os390/support/
- WAS V7.0 アナウンスメント・ワークショップ資料
 - http://www.ibm.com/developerworks/jp/websphere/library/was/was7_ws/index.html

マニュアル類などで、詳細情報を検索することができます。