HITACHI
Inspire the Next

Hitachi Application Server

開発・運用時のガイド [Windows]

導入母体に起因する留意点

2016.10
October

はじめに

本書は、開発・運用フェーズで使用するドキュメントとして、Hitachi Application Server の導入母体に起因する留意点について記述しています。

１．対象とする読者

本書は、Hitachi Application Server を使用し、システムを設計・構築・運用する立場にある方を対象としています。

２．対象とする製品

Hitachi Application Server 10-11

■商標類

■製品名と機能名の表記

本書では、製品名を次のように表記しています。

表記		製品名
Application Server		Hitachi Application Server
HAS		
DAS		Domain Administration Server
Developer’s Kit for Java		Hitachi Developer’s Kit for Java
JavaEE Server		Hitachi JavaEE Server
JavaVM ログファイル		日立 JavaVM ログファイル
Web Server		Hitachi Web Server
Windows	Windows Server 2008 R2	Microsoft® Windows Server® 2008 R2 Standard 日本語版
		Microsoft® Windows Server® 2008 R2 Enterprise 日本語版

表記		製品名
		Microsoft® Windows Server® 2008 R2 Datacenter 日本語版
	Windows Server 2012	Microsoft® Windows Server® 2012 Standard 日本語版
		Microsoft® Windows Server® 2012 Datacenter 日本語版
	Windows Server 2012 R2	Microsoft® Windows Server® 2012 R2 Standard 日本語版
		Microsoft® Windows Server® 2012 R2 Datacenter 日本語版

なお、32 ビット版の Windows を Windows x86 と表記することがあります。また、64 ビット版の Windows を Windows x64 と表記することがあります。

■英略語の表記

本書では、英略語を次のように表記しています。

英略語	英字での表記
EJB	Enterprise JavaBeans™
Java	Java™
Java EE	Java™ Platform, Enterprise Edition
JDBC	Java™ Database Connectivity
	JDBC™
JSP	Java ServerPages™
	JSP™
JTA	Java™ Transaction API
Servlet	Java™ Servlet

なお、32 ビット版の Windows を Windows x86 と表記することがあります。また、64 ビット版の Windows を Windows x64 と表記することがあります。

■発行元

株式会社日立製作所 ICT 事業統括本部　サービスプラットフォーム事業本部

目　次

1 Application Server に関する注意事項

Application Server に関する注意事項を説明します。

本章の構成

1.1 Developer's Kit for Java

(1)java.lang.Thread クラスの suspend()や resume()使用時の注意事項

【現象】

アプリケーションがハングアップする。

【詳細な現象】

java.lang.Thread クラスの suspend()や resume()を呼び出すプログラムを実行すると，アプリケーションがハングアップする場合があります。

【発生条件】

java.lang.Thread クラスの suspend()や resume()を呼び出すプログラムを実行した場合。

詳細は,

http://docs.oracle.com/javase/8/docs/technotes/guides/concurrency/threadPrimitiveDeprecation.html

をご覧ください。

【回避策】

なし。

(2)jdb 上での java.lang.Thread クラスの suspend()や resume()使用時の注意事項

【現象】

jdb がハングアップする。

【詳細な現象】

java.lang.Thread クラスの suspend()や resume()を呼び出すプログラムを，jdb 上で実行または jdb でアタッチすると，jdb がハングアップする場合があります。

【発生条件】

java.lang.Thread クラスの suspend()や resume()を呼び出すプログラムを，jdb 上で実行または jdb でアタッチした場合。

【回避策】

なし。

(3)java.util.concurrent.ScheduledThreadPoolExecutor クラス使用時の注意事項

【現象】

java.util.concurrent.ScheduledThreadPoolExecutor クラスが正常に動作しない。

【詳細な現象】

java.util.concurrent.ScheduledThreadPoolExecutor クラスのオブジェクトに登録したタスクが，指定された遅延時間や実行間隔が経過しても全く実行されない場合があります。

【発生条件】

以下の条件がすべて重なる場合。

(a) 以下に挙げる java.util.concurrent.ScheduledThreadPoolExecutor クラスのタスク登録用メソッドのどれかを呼び出してタスクを登録する

・<V> ScheduledFuture<V> schedule(Callable<V> callable, long delay, TimeUnit unit)

・ScheduledFuture<?> schedule(Runnable command, long delay, TimeUnit unit)

・ScheduledFuture<?> scheduleAtFixedRate(Runnable command, long initialDelay, long period, TimeUnit unit)

・ScheduledFuture<?> scheduleWithFixedDelay(Runnable command, long initialDelay, long delay, TimeUnit unit)

(b) (a)の呼び出し時に，タスク実行までの遅延時間や実行間隔を指定するための引数

・long delay

・long initialDelay

・long period

のどれかに対して，引数 TimeUnit unit で指定した単位から nano 秒に換算した際の値が 0 未満，または「java.lang.Long.MAX_VALUE - Java プロセス起動からの経過時間」以上の範囲内となるような時間を指定する。

Long.MAX_VALUE 値を nano 秒に換算した時間は約 292 年に相当しますので，そのような大きな遅延時間を設定すると現象が発生します。

【回避策】

なし。

(4)総称クラス使用時の注意事項

【現象】

java.lang.VerifyError が発生する。

【詳細な現象】

総称クラスを使用すると，アプリケーション実行時に java.lang.VerifyError が発生することがあります。

【発生条件】

以下の条件がすべて重なる場合に発生することがあります。

(a) 任意の総称クラスを定義している。

(b) (a)の総称クラスを java.lang.Number の任意のサブクラスにバインドしている。

(c) (a)の型変数を，後置インクリメントまたは後置デクリメントしている。

(d) (b)でバインドした型を引数に取ることができるメソッドに対して，引数渡しで(c)を行っている。

(e) (a)～(d)の条件が重なる Java ソースをコンパイルしてクラスファイルを生成する。

(f) (e)のクラスファイルを実行する。

【回避策】

型変数のインクリメントまたはデクリメントを，引数渡しの後に実行してください。

(5)javax.imageio.ImageIO クラス使用時の注意事項

【現象】

クラスローダが最大でひとつリークする。

【詳細な現象】

Web アプリケーションクラスローダがリークし，J2EE アプリケーションをアンデプロイしても

Metaspace 領域の使用サイズが減少しないことがあります。リークする Web アプリケーションクラスローダの数は，J2EE サーバにつき最大でひとつです。

【発生条件】

J2EE アプリケーション内で，javax.imageio.ImageIO クラスを使用した場合に発生することがあります。

また，JAXB や JAX-WS 機能で画像データを取り扱った場合に発生することがあります。

【回避策】

J2EE アプリケーションのアンデプロイ後に J2EE サーバの停止・再起動をしてください。

(6)デバッグまたはプロファイリング用オプションについて

JavaVM オプションの-Xprof や-Xdebug，JVMTI エージェントの hprof や jdwp(-agentlib:<libname> などで指定)は，プログラムの開発用ユーティリティとして提供されているものですので，システムの運用では指定しないようにしてください。

(7)JVMPI, JVMDI について

JVMPI(Java Virtual Machine Profiler Interface)と JVMDI(Java Virtual Machine Debug Interface)を廃止しました。これらの代わりに JVM TI(JVM Tool Interface)を利用してください。

(8)AWT コンポネントでの JIS X0213:2004 の第三水準，第四水準文字の非サポートについて

AWT コンポネントは JIS X0213:2004 の第三水準，第四水準文字をサポートしていません。AWT コンポネントが出力する情報で該当する文字の部分は文字化けします。

該当する文字を扱う場合には，Swing コンポネントを使用してください。

(9) JavaVM ログファイル，スレッドダンプログファイル，明示管理ヒープ機能のイベントログファイルおよびエラーリポートファイルのマルチバイトの非サポートについて

JavaVM ログファイル，スレッドダンプログファイル，明示管理ヒープ機能のイベントログファイルおよびエラーリポートファイルは，マルチバイト文字をサポートしていません。日本語のクラス名称など，該当する文字の部分は文字化けします。

(10)URLClassLoader でクラスをローディングする場合について

URLClassLoader 経由で jar ファイルからクラスをローディングする場合，ローディング処理中に JavaVM の内部で作成されるオブジェクトが JavaVM 終了時まで削除されないことがあります。

適宜 Java アプリケーションを再起動してください。

(11)RMI 使用時に発生する GC について

RMI を使ってリモートオブジェクトの登録や参照をすると，定期的に GC が発生することがあります。GC 発生間隔は，sun.rmi.dgc.client.gcInterval プロパティまたは，sun.rmi.dgc.server.gcInterval プロパティにミリ秒単位で指定することができ，デフォルトは 3600000 (1 時間)です。GC 発生間隔を変更する場合は，任意の間隔をミリ秒単位で

sun.rmi.dgc.client.gcInterval プロパティまたは,
sun.rmi.dgc.server.gcInterval プロパティに指定してください。
sun.rmi.dgc.client.gcInterval プロパティまたは,
sun.rmi.dgc.server.gcInterval プロパティに指定できる値の範囲は 1～Long.MAX_VALUE-1 です。詳細は,
http://docs.oracle.com/javase/8/docs/technotes/guides/rmi/sunrmiproperties.html
をご覧ください。

(12)Java2D 使用時について

Java2D の DirectDraw および Direct3D の問題を回避するため，DirectDraw と Direct3D を無効にするプロパティ sun.java2d.noddraw に true を指定してください。

(13)JNI プログラム中で使用できないライブラリについて

JavaVM 内で管理している情報が不正に更新される場合があるため，JNI プログラム中で以下のライブラリは使用できません。

・setjmp()
・longjmp()

(14)メソッド長の上限について

JavaVM の仕様によって，1 メソッドのバイトコードは 64KB 以内にする必要があります。
64KB を超える場合は，クラスファイル生成時にエラーとなるか，クラスのロード時に java.lang.LinkageError 例外が発生します。
また，1 メソッドの大きさが 64KB 以内であっても，非常に複雑で行数が多い場合は，以下のような弊害が発生する場合があります。
・GC 処理の実行に非常に時間がかかる。
・JIT コンパイルに非常に時間がかかる。
・JIT コンパイルに非常に多くのメモリを消費する。
さらに，ローカル変数情報出力機能が有効の場合は，以下の弊害も発生する場合があります。
・拡張スレッドダンプの出力に時間がかかる。
・スレッドスタックトレースの取得に時間がかかる。
・例外発生時の例外オブジェクト生成処理に時間がかかる。
そのため，Java ソース上の 1 メソッドの大きさは，コメントや空行を除き，およそ 500 行以内を推奨します。

(15)java.io.tmpdir プロパティについて

java.io.tmpdir プロパティを指定する場合は，書き込み権限のある実在するフォルダを指定してください。
java.io.tmpdir プロパティの初期値は Windows API の GetTempPath()関数で得られるフォルダです。
また，Java RMI の動的クラスローディング機能や Java API の java.io.File.createTempFile()では,

java.io.tmpdir プロパティで指定されたフォルダに一時ファイルを作成します。
これらの機能を正常に動作させるため，JavaVM プロセス起動中は一時ファイルの作成先を削除しないでください。

(16)管理者特権が必要な独立プロセスの起動について

管理者特権がないユーザで起動した java アプリケーションから，java.lang.Runtime.exec()または java.lang.ProcessBuilder.start()によって管理者特権を持つ独立プロセスを起動する場合は，次の手順で行います。
[1] 管理者特権として起動するコマンドにマニフェストを追加します。マニフェスト追加の方法については，マイクロソフト社のドキュメントを参照ください。
[2] Windows の cmd.exe を介して起動してください。例えば，sample.exe を起動する場合には次のようになります。
・Runtime.getRuntime().exec("cmd.exe", "/c", "sample.exe");
・new ProcessBuilder("cmd.exe", "/c", "sample.exe").start();
この java アプリケーションを起動すると，[ユーザー アカウント制御]ダイアログボックスが表示されるので，[続行]をクリックしてアプリケーションを続行してください。

(17)java.util.prefs.Preferences.systemNodeForPackage によるデータ共有について(Windows x86 版のみ)

管理者特権を持たないユーザが java.util.prefs.Preferences.systemNodeForPackage を使用した場合，異なるユーザ間でのデータ共有機能は使用できません。
具体的には，同一のキーを引数に指定した場合でも，異なるユーザ間では，異なったデータが返ります。

(18)java.util.prefs.Preferences.systemNodeForPackage によるデータ共有について(Windows x64 版のみ)

管理者特権を持たないユーザが java.util.prefs.Preferences.systemNodeForPackage を使用した場合，異なるユーザ間でのデータ共有機能は使用できません。
具体的には，java.util.prefs.BackingStoreException がスローされます。

(19)ファイルディスクリプタのクローズについて

java.lang.Runtime.exec()および java.lang.ProcessBuilder.start()で起動した子プロセスは，
Process.getInputStream/getErrorStream/getOutputStream の
おのおののメソッドで取り出したストリームを通じてプロセス間通信をします。
親プロセスでは，これらのメソッドを使わない場合でも３つのファイルディスクリプタ
を消費することに注意してください。
ファイルディスクリプタをクローズするのは以下の場合です。
・Process オブジェクトのファイナライズが完了した場合
・Process.destroy()を呼び出した場合
・これらストリームに対して明示的に close()メソッドを呼び出した場合

(20)システムプロパティ java.ext.dirs, java.library.path について

システムプロパティ java.ext.dirs は，通常のアプリケーションクラスよりも優先してロードするクラスを含む jar ファイルを配置するディレクトリを指定するシステムプロパティです。
このプロパティのデフォルト値は，"<Application Server インストールディレクトリ>¥jdk¥jre¥lib¥ext"です。
システムプロパティ java.library.path は，ユーザのネイティブライブラリの検索パスを指定するシステムプロパティです。このシステムプロパティのデフォルト値は，"<Application Server インストールディレクトリ>¥jdk¥bin，<システムディレクトリ>，<Windows ディレクトリ>，%PATH%，カレントディレクトリ"です。

(21)インストゥルメンテーション機能が出力するメッセージについて

インストゥルメンテーション機能の処理中にメモリ不足になった場合に，以下のメッセージが出力され，インストゥルメンテーション機能の処理が失敗することがあります。

*** java.lang.instrument ASSERTION FAILED ***

インストゥルメンテーション機能の詳細は，下記の URL を参照してください。

http://docs.oracle.com/javase/jp/8/api/java/lang/instrument/package-summary.html

(22)java.net.Socket.connect()のタイムアウトについて

java.net.Socket.connect()でのソケットの接続が，OS に設定されている TCP 通信のタイムアウト値によってタイムアウトすることがあります。TCP 通信のタイムアウト値になると，java.net.Socket.connect()に指定したタイムアウト値よりも前や，タイムアウト値を指定していない場合であっても，接続がタイムアウトします。
TCP 通信のタイムアウト値によってタイムアウトした場合，以下の詳細メッセージを含む java.net.ConnectException 例外がスローされます。

Connection timed out: connect　[errno=10060, syscall=select]

TCP 通信のタイムアウトの詳細については，OS のドキュメントを参照してください。

(23)クラスロードのタイミングについて

Java におけるクラスファイルのメモリへの読み込み(クラスロード処理)は，プログラムの実行中に，そのクラスが初めて必要となったタイミングで実行されます。そのため，ある処理の初回実行でクラスロードの回数が多くなると，2 回目以降と比較して，処理時間が長くなることがあります。このような場合は，処理の実行に必要なクラスのロードを事前にすることにより，処理時間が改善します。

(24)java.lang.Runtime.runFinalizersOnExit()，java.lang.System.runFinalizersOnExit()メソッドについて

java.lang.Runtime.runFinalizersOnExit()，java.lang.System.runFinalizersOnExit()メソッドは非推奨 API です。runFinalizersOnExit()メソッドを使用している時，意図していないオブジェクトのファイナライザが呼び出されることで，J2EE サーバの終了時に動作が異常になることがあります。サーバの終

了時に動作が異常になる時は，以下のどちらかを実施してください。

・jvm.api.runfinalizersonexit.Disable を指定する。

（runFinalizersOnExit()メソッドの処理を実行せず，メソッドをリターンさせるプロパティ）

・runFinalizersOnExit()メソッドを使用しない方式にプログラムを変更する。

なお，プログラム実行中に runFinalizersOnExit()メソッドが呼び出されたかどうかは，クラスライブラリトレース機能(-XX:+HitachiJavaClassLibTrace)を使用することで確認できます。

(25)異なる OS 間で発生する文字化けについて

文字コードの Unicode へのマッピングは OS によって異なる場合があるため，以下の場合に文字化けが発生する場合があります。文字化けが発生する代表的な文字に，― ～ ∥ － ¢ £ ¬があります。

・UTF-8 などの Unicode のエンコーディングを用いて，異なる OS 間で上記文字データを受け渡す場合。

・上記文字を含む文字列リテラルがあるソースプログラムから作成したクラスファイルを，異なる OS 上でリコンパイルせずにそのまま実行する場合。

(26)エンコーディング時に発生する文字化けについて

以下に挙げる文字を java プログラムで扱う際には，エンコーディングに MS932，windows-31j，cswindows31j のどれかを指定してください。それ以外のエンコーディングを指定した場合，文字化けすることがあります。

・NEC 拡張文字　①②...⑳，ⅠⅡ...Ⅹ，㈱，㍾㍽㍼㍻，ﾐｯｷｰ など。

・NEC 選定 IBM 拡張文字　ⅰⅱ...ⅹ など。

・IBM 拡張文字　ⅠⅡ...Ⅹ，ⅰⅱ...ⅹ，№，℡ など。

[例]

```
import java.io.*;
class encode_eucjis {
   public static void main( String arg[] ) {
      try {
         String string_data = "ⅠⅡⅢ";
         byte[] data = string_data.getBytes();
         InputStreamReader isr =
            new InputStreamReader(
                              new ByteArrayInputStream( data ), "eucjis");
         char[] read_data = new char[4];
         isr.read( read_data, 0, 4 );
         System.out.println(new String(read_data));
      }
      catch ( Exception e ) {
         e.printStackTrace();
      }
```

```
        }
    }
```

上記のプログラムは，IBM 拡張文字を eucjis でエンコーディングしているため，実行すると以下のように文字化けします。

```
java encode_eucjis
???
```

文字化けを回避するためには，MS932，windows-31j，cswindows31j のどれかを指定してエンコーディングしてください。

(27)Java API で使用する各種ファイルについて

Java API で使用するポリシーファイルやログイン構成ファイルなどの各種コンフィグレーションファイルは，UTF-8 エンコーディング方式でエンコーディングする必要があります。

(28)Java がサポートする UTF-8 エンコーディングについて

Java がサポートする UTF-8 エンコーディングは，BOM(Byte Order Mark)なしの UTF-8 です。Windows のテキストエディタ「メモ帳」で「文字コード」に"UTF-8"を指定して保存したデータは BOM あり UTF-8 となるため，Java では扱うことができません。

(29)java.net.URL が扱う URL 文字列のエンコーディングについて

java.net.URL が扱う URL 文字列のエンコーディングは標準 UTF-8 ではなく，Modified UTF-8 です。Modified UTF-8 ではない文字列を与えた場合には java.lang.IllegalArgumentException が発生します。

(30)JNI 関数で文字列操作をする場合の UTF-8 エンコーディングについて

JNI では, 以下の文字列操作をする関数で使用するエンコーディングは, 標準 UTF-8 ではなく, Modified UTF-8 です。

NewStringUTF
GetStringUTFLength
GetStringUTFChars
ReleaseStringUTFChars
GetStringUTFRegion

(31)if 文の判定でジャンプできる長さの上限について

JavaVM の仕様では, if 文の判定でジャンプできる長さに制約があります(バイトコードで前後 32KB 以内)。これを超える場合は，クラスファイルの生成時にエラーとなるか，クラスのロード時に java.lang.LinkageError 例外が発生します。

(32)java.nio.channels.FileChannel クラスの map/transferFrom/transferTo メソッドについて

map メソッドでは，2GB を超えるファイルはサポートしていません。
また，transferFrom/transferTo メソッドでの count 指定で 2GB 以上の値はサポートしていません。

(33)スレッドのスタックサイズについて

JNI関数AttachCurrentThread()などを使用してネイティブスレッドをJavaVMに接続する場合は-Xss値よりも大きなスタックサイズのスレッドで接続してください。

1.2 Java EE Server

1.2.1 asadmin コマンドを使用する場合の注意事項

(1)asadmin change-master-password コマンドのメッセージについて

asadmin change-master-password コマンドの実行でエラーが発生した場合，エラーメッセージの一部が文字化けする場合があります。文字化けしていない部分のメッセージに従って対処してください。

(2) asadmin コマンドの同時実行について

asadmin start-domain, stop-domain サブコマンドと list-domains サブコマンドは同時に実行しないでください。同時に実行した場合，実行したどちらかのサブコマンドでタイムアウトが発生し，実行に失敗することがあります。
実行に失敗した場合は，失敗したサブコマンドを再度実行してください。

(3)asadmin restore-domain コマンドについて

asadmin restore-domain コマンドの domain-name オプションにはドメイン名を指定してください。

(4)asadmin deploy コマンドについて

asadmin deploy コマンドの実行中に以下のいずれかの条件に該当すると，asadmin undeploy コマンドまたは，asadmin disable コマンドで以下のエラーが発生します。

Application Not Registered

[発生条件]
(a)DAS または サーバインスタンスがダウンした場合
(b)DAS と サーバインスタンスの間で通信できない場合
(c)DAS と サーバインスタンスの間で通信タイムアウトが発生した場合(*1)
(d)asadmin deploy コマンドを実行時にディスクフルやパーミッションがないなどのファイル I/O エラーが発生した場合

*1: asadmin ユーティリティコマンドの実行ホストが DAS と異なり，asadmin ユーティリティリードタイムアウトのみが発生したときは除く

この現象が発生した場合は，以下の手順でアンデプロイを実施することができます。
(a)DAS を起動する(起動している場合は再起動する)
(b)asadmin delete-application-ref コマンドを使い，サーバインスタンスからアプリ

ケーションへのアプリケーション参照の削除を実施する。(*2)
(c)asadmin undeploy コマンドを使い，アプリケーションをドメインからアンデプロイする。
(d)サーバインスタンスを起動する(起動している場合は再起動する)

*2: サーバインスタンスが起動していない，または DAS とサーバインスタンスの通信ができない場合 警告メッセージが表示されますが，delete-application-ref サブコマンドの実行に問題はありません。

(5)asadmin undeploy コマンドについて

asadmin undeploy コマンドの実行中に以下のいずれかの条件に該当すると，エラーが発生し，再度 asadmin undeploy コマンドが実行できなくなります。

[発生条件]
(a)DAS または サーバインスタンスがダウンする
(b)DAS と サーバインスタンスの間で通信できない場合
(c)DAS と サーバインスタンスの間で通信タイムアウトが発生した場合(*1)
*1: asadmin ユーティリティコマンドの実行ホストが DAS と異なり，asadmin ユーティリティリードタイムアウトのみが発生したときは除く

この現象が発生した場合は，以下の手順を実施してください。
(a)DAS と サーバインスタンスを再起動する
(b)asadmin undeploy コマンドを実行する

(6)asadmin list-batch-jobs コマンドについて

HAS では JavaBatch のバッチジョブを実行する度に INSTANCECOUNT が増加します。
INSTANCECOUNT が増加するにつれて "asadmin list-batch-jobs" のレスポンスが遅くなります。
また， "asadmin list-batch-jobs" を実行すると多くの性能解析トレース情報を出力しますので，
トレース情報がラップする可能性があります。
そのため, "asadmin list-batch-jobs" を実行する際には性能解析トレースの情報の退避を検討ください。

(7)asadmin stop-instance コマンド, stop-cluster コマンドについて

stop-instance サブコマンドおよび stop-cluster サブコマンドの--kill オプションの指定に応じて，以下のように挙動が変化します。

・--kill=false（または--kill 指定なし）の場合, サブコマンドはサーバインスタンスの停止処理を実行し，サーバインスタンスが正常に停止したことを確認したら，サブコマンドは正常終了します。

・--kill=true の場合，サブコマンドはサーバインスタンスの停止処理を実行し，その停止処理が失敗した場合は，OS の機能によるサーバインスタンスのプロセス停止を実行します。その後，サーバインスタンスが停止したどうかに関わらず，サブコマンドは正常終了します。

(8)DAS やサーバインスタンスの起動時に asadmin_launch.log に出力される警告メッセージについて

【現象】

DAS やサーバインスタンスの起動時に asadmin_launch.log に警告メッセージが出力されます。

【詳細な現象】

DAS やサーバインスタンスの起動時，asadmin_launch.log にクラス名「com.sun.enterprise.glassfish.bootstrap.osgi.BundleProvisioner$DefaultCustomizer getLocations」と「Skipping entry　because it is not an absolute URI.」を含む警告メッセージが出力されます。

DAS やサーバインスタンスの起動が正常に完了した場合，このメッセージは無視してください。

【発生条件】

DAS やサーバインスタンスを起動した場合。

【回避策】

なし。

(9) domain.xml が参照できない状態でのサブコマンドの実行について

domain.xml が参照できない状態で，以下のサブコマンドを実行した場合，コンソールに”NoAdminPortEx”が出力されてコマンドが失敗します。

・stop-domain
・list-domains
・delete-domain
・restart-domain
・change-master-password

(10) domain.xml の更新に失敗した場合の注意事項

asadmin コマンドを実行し domain.xml の更新に失敗した場合，次のディレクトリに"domain<数字>.xml"という名前のファイルが残る場合があります。このファイルは削除してください。

<ドメインルートディレクトリ※><ドメイン名>¥config

※create-domain 実行時に--domaindir オプションで指定したディレクトリパス。デフォルトは<Application Server インストールディレクトリ>¥javaee¥glassfish¥domains

(11) asadmin setup-local-dcom コマンドについて

asadmin setup-local-dcom コマンドを一度も実行したことのない環境で asadmin setup-local-dcom コマンドを実行した場合，コンソールに以下のメッセージが表示される場合があります。

コマンドが正常終了していれば，これらのメッセージは無視してください。

・key: CLASSES_ROOT¥CLSID¥{72C24DD5-D70A-438B-8A42-98424B88AFB8}, error num: 5, error msg: アクセスが拒否されました。
・key: CLASSES_ROOT¥CLSID¥{76A64158-CB41-11D1-8B02-00600806D9B6}, error num: 5, error msg: アクセスが拒否されました。

(12) asadmin start-domain コマンドの--upgrade オプションについて

asadmin start-domain コマンドに--upgrade オプションを指定してアップグレードを実行した場合，次の 2 つのメッセージが出力される場合があります。

・次の文字列を含むログレベルが SEVERE のメッセージ

Exception in thread "<スレッド名>"

・次の文字列を含むログレベルが SEVERE のメッセージ

java.lang.NullPointerException

at

org.glassfish.osgijavaeebase.OSGiContainer$DeployerAddedThread.run(OSGiContainer.java:300)

・gogo: InterruptedException: sleep interrupted

上記のメッセージが出力されている場合，これらのメッセージは無視してください。

1.2.2 EJB を使用する場合の注意事項

(1)EJB 呼び出し中のアプリケーション停止について

【現象】

EJB を呼び出し中にアプリケーションを停止すると，NullPointerException が呼び出し元にスローされることがあります。

【詳細な現象】

EJB(Stateless Session Bean または Entity Bean)を呼び出し中にアプリケーションを停止すると，NullPointerException が呼び出し元にスローされることがあります。

【発生条件】

以下の条件がすべて重なる場合に発生します。

(a)EJB(Stateless Session Bean または Entity Bean)を呼び出し中である。

(b)アプリケーションを停止する。

【回避策】

なし。

(2) Remote インターフェースを使用するアプリケーション構成について

同一 Remote インタフェースを実装した SessionBean が同一サーバインスタンス上に複数存在する場合，各 SessionBean を含むアプリケーションを同時に開始できません。また，各 SessionBean が同一アプリケーション内に存在する場合，このアプリケーションを開始できません。

同一 Remote インタフェースを実装する SessionBean は，サーバインスタンス上で 1 つだけになるようなアプリケーション構成にしてください。

1.2.3 JSP を使用する場合の注意事項

(1)存在しない JSP にアクセスした場合の注意事項

リクエスト URI に対応する JSP ファイルが存在しない場合，メッセージ「PWC6117: File "null" not found」を je_message.log に出力します。

このメッセージが出力された時は，Web Server のアクセスログと対応付けて確認してください。

1.2.4 JTA を使用する場合の注意事項

(1)javax.transaction.TransactionManager について

アプリケーションから javax.transaction.TransactionManager インタフェースを使用することはできません。

(2)javax.transaction.UserTransaction について

Servlet の service メソッドで javax.transaction.UserTransaction を使用して開始したトランザクションは，そのメソッド内で決着させてください。

1.2.5 JAX-RS を使用する場合の注意事項

(1)JAX-RS 機能で非同期処理を使用する場合について

以下のアノテーションを使用する場合，サーブレット側に非同期処理を有効にする必要があります。

・JAX-RS 2.x (標準) の@javax.ws.rs.container.Suspend

・Jersey 2.x (実装依存) の@org.glassfish.jersey.server.ManagedAsync

サーブレット側の非同期処理を有効にするには web.xml の servlet 要素に
<async-supported>true</async-supported>
を設定してください。

(2)Java EE Application Client 機能を使用して JAX-RS クライアントを動作させる場合について

Java EE Application Client 機能を使用して JAX-RS クライアントを動作させる場合，

次の JAR ファイルをクラスパスに設定する必要があります。

<Application Server インストールディレクトリ>¥javaee¥glassfish¥modules¥jersey-client.jar

<Application Server インストールディレクトリ>¥javaee¥glassfish¥modules¥jersey-common.jar

<Application Server インストールディレクトリ>¥javaee¥glassfish¥modules¥jersey-guava.jar

<Application Server インストールディレクトリ>¥javaee¥glassfish¥

modules¥osgi-resource-locator.jar
<Application Server インストールディレクトリ>¥javaee¥glassfish¥
modules¥javax.ws.rs-api.jar
<Application Server インストールディレクトリ>¥javaee¥glassfish¥
modules¥aopalliance-repackaged.jar

1.2.6 JNI 利用時の注意事項

(1)ネイティブライブラリをロードするクラスについて

System#loadLibrary()を実行しネイティブライブラリをロードするクラスは
jar ファイルとしてアーカイブし，asadmin add-library コマンドでサーバインスタンスに
ロードしてください。

1.2.7 WebSocket を使用する場合の注意事項

(1)javax.websocket.PongMessage インタフェースについて

Microsoft IIS を介して WebSocket 通信を行う場合，クライアントから，
javax.websocket.PongMessage インタフェースを使用してリクエストを送信すると，
Java EE サーバにリクエストが到達することなく，リクエストが完了することがあります。

1.2.8 アプリケーションのデプロイに関する注意事項

(1)glassfish-resources.xml の /resources/connector-connection-pool/security-map 要素の使用について

glassfish-resources.xml に /resources/connector-connection-pool/security-map 要素が定義された，Connector Connection Pool のリソースを使用するアプリケーションをデプロイした場合，NullPointerException 例外が発生してデプロイに失敗します。

(2)アプリケーションデプロイ時のメッセージについて

EJB を含むアプリケーション，または beans.xml を含むアプリケーションをデプロイしたとき，以下のメッセージがサーバインスタンス起動後，メッセージログに一度出力される場合がありますが，動作上の影響はありません。

Class 'javax.ejb.PostActivate' not found, interception based on it is not enabled
Class 'javax.ejb.PrePassivate' not found, interception based on it is not enabled

(3)EJB タイマーサービスを使用したアプリケーションアンデプロイ時のメッセージについて

【現象】

アプリケーションをアンデプロイすると NullPointerException がログに出力されます。
【詳細な現象】
EJB タイマーサービスを使用するアプリケーションをアンデプロイすると
NullPointerException がスタックトレースログに出力されます。
アプリケーションのアンデプロイが正常終了した場合，この NullPointerException は無視して
てください。

1.2.9 エラーページに関する注意事項

(1) Java EE Server がデフォルトで返すエラーステータスコードのレスポンスボディに関する注意事項

Java EE Server が HTTP レスポンスがエラーステータスコードを返すとき，レスポンス
ボディは空白となります。エラーステータスコードを返すとき，レスポンスボディを設定する
場合は，以下の方法で実現して下さい。
(a)Servlet でエラーステータスコードの場合のレスポンスボディを設定する。
(b) Web Server の ProxyErrorOverride を有効にして，
エラーステータスコードの場合のレスポンスボディを Web Server で上書きする。

ただし，JAX-WS 仕様， JAX-RS 仕様に準拠した Web サービスが動作する環境では以下の理由の
ため，(b)の方法は推奨しません。
(b-1)JAX-WS 仕様に準拠した Web サービスでは，デフォルトでメッセージの処理中に発生する例外
を SOAP フォルトを使用して処理します。SOAP フォルトを使用して処理する場合，SOAP
Fault 電文を作成し，エラーステータスコード 500 でクライアントに返信します。
そのため， Web Server によって，レスポンスボディ上書きすると，仕様に違反し
た電文となり，クライアント側に不正な電文が送信されます。
(b-2)JAX-RS 仕様では，エラーコードに対する内容が規定されているため， Web Server
によって, レスポンスボディを上書きするとクライアント側に仕様に違反した電文が送信されます。
なお，仕様に違反した電文をクライアントが受け付けた場合の動作については，クライアント
開発元に確認してください。

1.2.10 セキュリティマネージャーを使用する場合の注意事項

(1)JavaScript エンジンを使用した場合について

セキュリティマネージャーを有効にして，JavaScript エンジンを使用したとき，
java.security.AccessControlException 発生時のスタックトレースが，
スタックトレースログに出力されることがありますが，動作上の影響はありません。

(2) server.policy ファイルを使用したアクセス権の設定について

セキュリティーマネージャーが有効なときは，add-library サブコマンドで追加
したライブラリや，<Application Server インストールディレクトリ>¥javaee

¥glassfish¥domains¥ドメイン名¥lib ディレクトリ に配置した JAR ファイル(JDBC ドライバなど)に，適切なアクセス権を設定してください。
<Application Server インストールディレクトリ>¥javaee¥glassfish¥domains ¥ドメイン名¥config¥server.policy ファイルを編集することでアクセス権の設定が可能です。

1.2.11 セッションレプリケーションを使用する場合の注意事項

(1)業務情報の引継ぎが可能となる障害の種類

セッションレプリケーションは，サーバダウンやプロセスダウンの単一障害が発生したケースにおいて，HTTP セッションを他のサーバに引き継ぐことができます。
多重障害や，クラスターを構成するホスト間の通信障害が起こったときは，業務情報の引継ぎができないことがあります。

(2)負荷分散装置の設定

負荷分散装置と Java EE サーバの間で，HTTP セッションの Sticky を有効にしてください。

(3)ホストの構成および設定

クラスターを構成するホストは以下の構成および設定にしてください。
・すべて同一サブネット上に配置する。
・互いにマルチキャストメッセージを送受信できる。
・各マシンのシステム時刻を可能な限り合わせる。

(4)Java EE アプリケーションの構成

非同期サーブレットを使用しない Java EE アプリケーションである場合に，セッションレプリケーションを使用することができます。

(5)HTTP セッションに登録可能なオブジェクト

セッションレプリケーションを使用するときには，直列化可能なオブジェクトだけを HTTP セッションに登録してください。以下に示すオブジェクトは直列化できません。
・java.io.Serializable インタフェースを実装していないオブジェクト
・直列化不可能なオブジェクトの参照を持つオブジェクト
また，writeObject()を実装し，直列化時に例外が発生するオブジェクトは，HTTP セッションに登録しないでください。

(6)HTTP セッションのサイズ上限

直列化後の HTTP セッションのサイズが合計 4MB 以下になるようにしてください。

(7)オンライン性能への影響

セッションレプリケーション使用時は，未使用時に比べてオンライン性能が劣化します。
事前に評価した上でご使用ください。

(8)メモリ設計への影響

HTTP セッションの複製データを Java EE サーバのメモリ領域に配置するため，
事前にメモリ見積もりを行ってください。

(9)明示管理ヒープ機能の制限

セッションレプリケーション有効時は，HTTP セッションに対して明示管理ヒープ機能は適用されません。

1.2.12 セッション永続化機能を使用する場合の注意事項

(1)セッション永続化機能使用時の HTTP レスポンスについて

【現象】

セッション永続化機能使用時に HTTP レスポンスを返信しないことがあります。

【詳細な現象】

HTTP セッションに登録するオブジェクトが Serializable インタフェースを実装していないオブジェクトをフィールドに持つ場合，セッション永続化機能使用時に HTTP レスポンスを返信しないことがあります。

【発生条件】

以下の条件がすべて重なる場合に発生します。

(a)HTTP セッションに登録するオブジェクトが Serializable インタフェースを実装していないオブジェクトをフィールドに持つ。

(b)HTTP セッションが復元される。

【回避策】

なし。

1.2.13 データベースを使用する場合の注意事項

(1)デフォルトデータソースの設定について

デフォルトデータソースを利用する場合には下記のとおり JDBC リソースを作成し，
データソースの設定を行ってください。正しく設定しない場合は，デフォルトデータソースで
エラーが発生する場合がありますのでご注意ください。

デフォルトデータソースの場合は以下のコマンドを実行してください。

asadmin create-jdbc-connection-pool

--datasourceclassname=<利用する DB の JDBC ドライバのクラス名>
--restype=<DataSource インタフェース名>
--property databaseName=<必要なデータベース用の設定>
<作成する JDBC コネクションプール名>
asadmin ping-connection-pool <作成する JDBC コネクションプール名>
asadmin create-jdbc-resource
--connectionpoolid <作成する JDBC コネクションプール名>
jdbc/__default
asadmin create-resource-ref
--target <アプリケーションを実行するサーバインスタンス名>
jdbc/__default

なお，一度使用されたデフォルトデータソースの定義を変更(削除して再作成)する場合は，
削除後から再作成の間にサーバインスタンスの再起動が必要です。

(2)EJB タイマーサービス利用時のコネクション数について

EJB タイマーサービス利用時の必要コネクション数は以下になります。
最小値：2 本
最大値：EJB タイマーサービスアプリケーションの同時実行数 + 1
ただし，アプリケーション実行期間中コネクション使用するのではないため，コネクションの
使い回しが可能です。同時実行数と性能を考慮し，コネクション数を見積もってください。

(3)EJB タイマーサービスを利用する場合のデータソースの設定について

EJB タイマーサービスを利用する場合には下記のとおり JDBC リソースを作成し，
データソースの設定を行ってください。正しく設定しない場合は，EJB タイマーサービスで
エラーが発生する場合がありますのでご注意ください。

EJB タイマーサービスの場合は以下のコマンドを実行してください。
asadmin create-jdbc-connection-pool
--datasourceclassname=<利用する DB の JDBC ドライバのクラス名>
--restype=<DataSource インタフェース名>
--property databaseName=<必要なデータベース用の設定>
<作成する JDBC コネクションプール名>
asadmin ping-connection-pool <作成する JDBC コネクションプール名>
asadmin create-jdbc-resource
--connectionpoolid <作成する JDBC コネクションプール名>
jdbc/__TimerPool
asadmin create-resource-ref
--target <アプリケーションを実行するサーバインスタンス名>

jdbc/__TimerPool

(4)コネクション障害検知機能について

glassfish-resources.xml を使用して JDBC コネクションプールを作成する場合，コネクション障害検知機能は使用できません。コネクション障害検知機能を使用する場合は，asadmin ユーティリティの create-jdbc-connection-pool サブコマンドを使用して JDBC コネクションプールを作成してください。

(5)EJB タイマーサービス利用時の注意事項

HAS は EJB タイマーサービスを利用するアプリケーションをデプロイしたとき，
EJB タイマーサービスで利用するデータベースにタイマー情報を格納するテーブルを
作成します。データベースに格納したタイマー情報は，アプリケーションのアンデプロイの
タイミングでのみ削除されます。
当該アプリケーションをアンデプロイすることなく
サーバインスタンスを削除するとデータベースのタイマー情報が削除されず残ります。
そのため，サーバインスタンスを削除する前に必ずアプリケーションをアンデプロイ
してください。

また，EJB タイマーサービスを利用しなくなった場合は
データベースから"EJB__TIMER__TBL"という名前のテーブルを削除してください。

1.2.14 ポートの設定に関する注意事項

(1) IIOP リスナーが使用するポートに関する注意事項

IIOP リスナーが使用するポートが競合した場合，JavaEE サーバは起動処理を中止することはありません。
IIOP リスナーのポートの競合を検知するには，スタックトレースログを確認してください。
IIOP リスナーのポートが競合した場合には"java.net.BindException: Address already in use: bind"を含むスタックトレースが出力されます。

1.2.15 ログレベル変更時の注意事項

(1)ログレベルを level3 に設定した場合について

ログレベルを level3 に設定すると，母体が出力するデバッグログの一部がログファイルに記録されることがあります。
このようなデバッグログについては無視してください。
また，本番環境ではログレベル level3 を使用しないでください。

例:

javax.management.InstanceAlreadyExistsException:
"amx:pp=/domain/resources,type=managed-executor-service,name=concurrent/__defaultManagedExecutorService"

1.2.16 使用できる文字範囲についての注意事項

(1)入力値として使用できる文字範囲について

次の項目の入力値には，ASCII 文字(0x20～0x7E の範囲)だけを使用してください。

・コマンド引数（set サブコマンドおよび get サブコマンドで使用する属性名および値を含む）
・システムプロパティ
・JavaVM オプション
・定義ファイル
・DD

なお，ディレクトリパスおよびファイルパスを指定する場合は，英数字(0x30～0x39, 0x41～0x5A, 0x61～0x7A)，ハイフン(0x2D)，アンダースコア(0x5F)，半角空白(0x20)，コロン(0x3A)，スラッシュ(0x2F)，バックスラッシュ(0x5C)，丸かっこ(0x28, 0x29)だけを使用してください。

ただし，スラッシュ(0x2F)，バックスラッシュ(0x5C)はディレクトリパスおよびファイルパスの区切り文字，コロン(0x3A)はドライブ文字にだけ使用してください。また，丸かっこ(0x28, 0x29)は，64 ビット OS に 32 ビットのプログラムをインストールした場合に使用されるフォルダ「Program Files (x86)」での使用に限ります。

1.2.17 非同期サーブレットを実行する場合の注意事項

(1)サーブレットの同期処理と非同期処理が同時に完了する場合について

【現象】

je_stacktrace.log に，java.lang.IllegalStateException: Not Suspended が出力される場合があります。

【詳細な現象】

サーブレットの同期処理と非同期処理が同時に完了した場合，タイミングによっては本来一度しか実行されないリクエスト完了処理が複数回実行される場合があり，その際にすでにリクエスト完了処理済みであることを示す IllegalStateException が発生します。

ただし，リクエスト処理自体は最初の完了処理によって正常に完了されますので，リクエスト処理への影響はありません。

【発生条件】

非同期サーブレットを実行した場合(タイミングに依存)。

－以上－