WebSphere Application Server for z/OS V7.0

WAS for z/OSのカスタマイズ
実践編

WebSphere software



© 2009 IBM Corporation

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。

© 2009 IBM Corporation

目次

- WAS for z/OS特有のタイムアウト設定
  - ディスパッチ・タイムアウト
  - トランザクション・タイムアウト
  - CPUタイム使用制限
- ポート・オープン・タイミングの設定
- wsadminツールとMBean
  - wsadmin
  - MBean
- 補足
  - wsadmin(Jython) Tips
  - MBeanの作成
  - リソース定義の例

© 2009 IBM Corporation

WAS for z/OS特有の
タイムアウト設定

© 2009 IBM Corporation

タイムアウトとは？

- 「タイムアウト」は、ある状態について、適切に設定された制限時間を超過した状態を指します
  - ※タイムアウトが起きたときの振る舞いは、タイムアウトの種類と設定によって異なります
- タイムアウトの例
  - 処理時間に上限を設ける
    - ディスパッチ・タイムアウト
    - トランザクション・タイムアウト
  - リソースの浪費を防ぐ
    - HTTPセッション・タイムアウト
    - コネクション・タイムアウト（JDBCやJMS QCFのコネクション・プール）
  - 両方（CPU時間の観点で上限を設けリソースの消費を防ぐ）
    - CPUタイム使用制限
- 当資料では、「ディスパッチ・タイムアウト」、「トランザクション・タイムアウト」、「CPUタイム使用制限」について説明します

© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/rtrb_understandingtimers.html

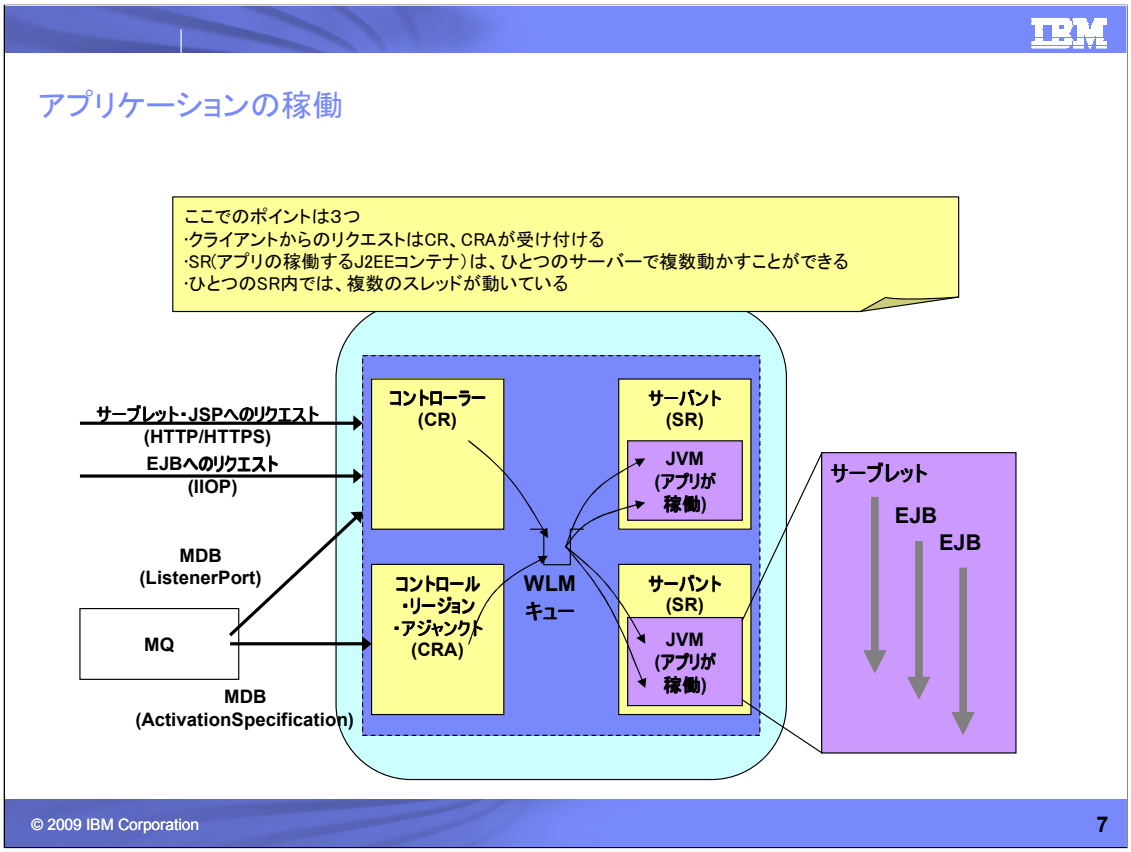
当資料で説明するタイムアウト

- ディスパッチ・タイムアウト
  - リクエストが受け付けられてから、処理し終わって応答を返すまでのタイムアウト
  - このタイムアウト時間を過ぎると、スレッドのリカバリーを試み（V7の新機能）、それでも応答が返らない場合は設定に応じてSRをABEND&再起動（リサイクル）します
- トランザクション・タイムアウト
  - アプリケーション中の、Unit Of Work(UOW)のタイムアウト
  - このタイムアウトを過ぎるとUOWはロールバックのためにマークされ、最終的にはロールバックされることになります
  - さらに特定の猶予期間内に処理が終わらなければ、SRをABEND&再起動（リサイクル）します
- CPUタイム使用制限 New V7
  - リクエストを処理するスレッドで消費したCPU時間の観点でのタイムアウト
  - この制限を超過するとCPUの割り当てのプライオリティーを下げて過剰なCPU消費を抑制し、設定に応じてSRをABEND&再起動（リサイクル）します

それぞれの設定は、個別に有効／無効にすることが可能です

© 2009 IBM Corporation

この資料ではWAS for z/OS特有の、SRのリサイクルを伴うタイムアウトである上記3つについて説明します。



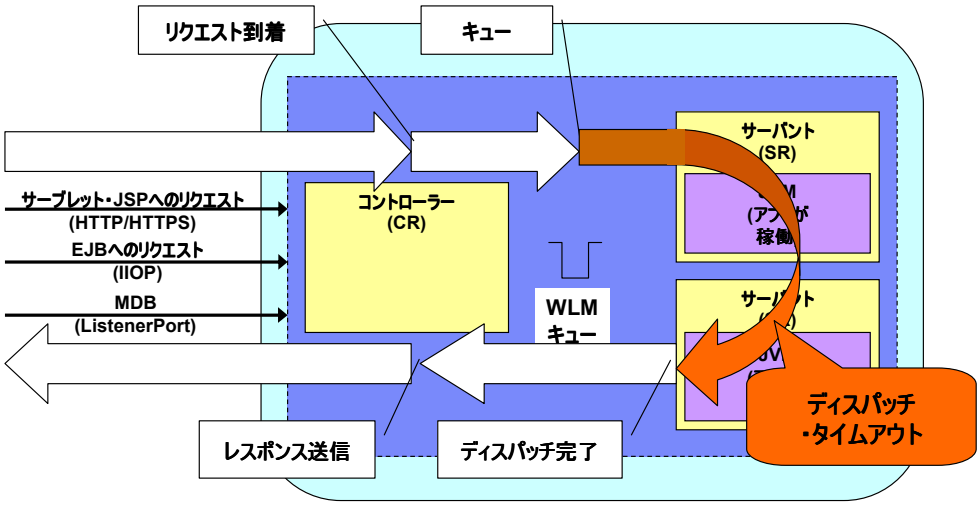
SRのリサイクルを伴うタイムアウトはz/OS版WASのみの特徴的な機能です。

この機能のメリットを理解し、有効に活用するには、

WAS for z/OSではひとつのサーバー内に複数のSRを構成することが可能で、ひとつのSRがABENDしてもサーバーとしての可用性を保つことが出来るそのトポロジーを理解しておくことが必要です。

ディスパッチ・タイムアウト

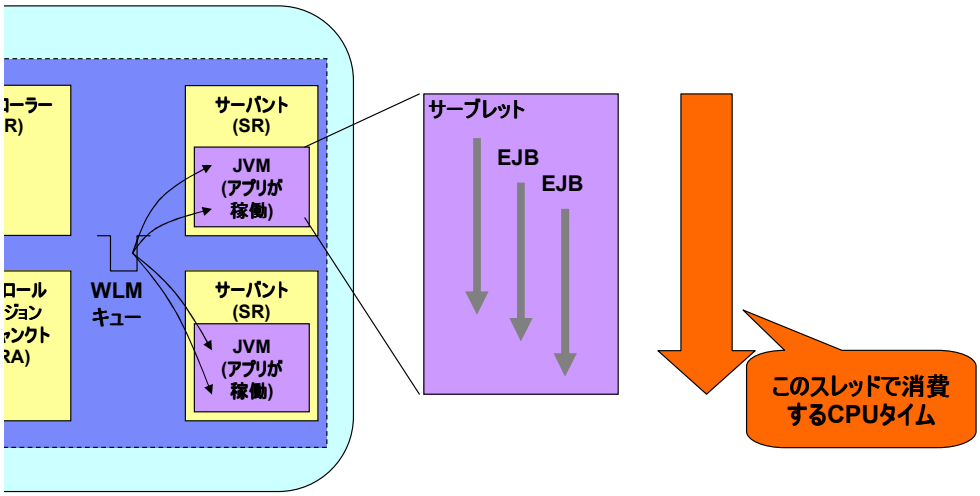
- ディスパッチ・タイムアウトは、CRがリクエストをWLMキューに入れてから、その処理が完了するまでの時間が監視されます



© 2009 IBM Corporation

CPUタイム使用制限

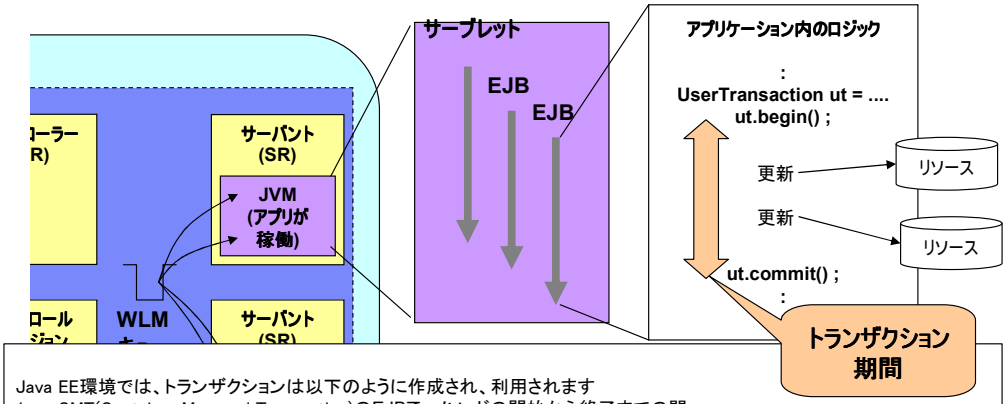
- CPUタイム使用制限は、リクエストを処理するためのスレッドで消費するCPU時間を監視します



© 2009 IBM Corporation

トランザクション・タイムアウト

- トランザクション・タイムアウトは、SR内でトランザクションを開始してから、COMMITまたはROLLBACKによってトランザクションが終了するまでの時間が監視されます



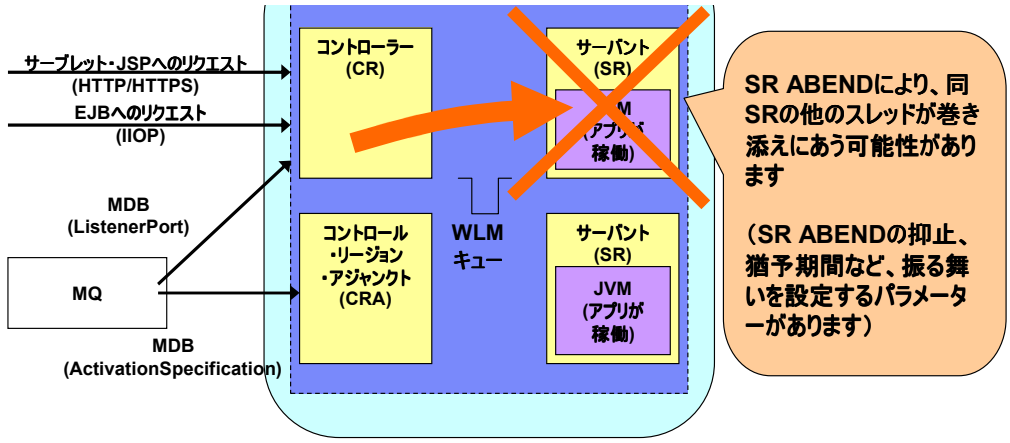
Java EE環境では、トランザクションは以下のように作成され、利用されます
1. CMT(Container Managed Transaction)のEJBで、メソッドの開始から終了までの間
2. サーブレットやBMT(Bean Managed Transaction)のEJBで、ユーザー・トランザクションを作成しbegin、commitを行う
（図は2.の場合の例です）

トランザクションは、複数のリソースの更新を同期させたい場合に使用します
一度のアプリケーション・フローの中で何度も使用される可能性があります

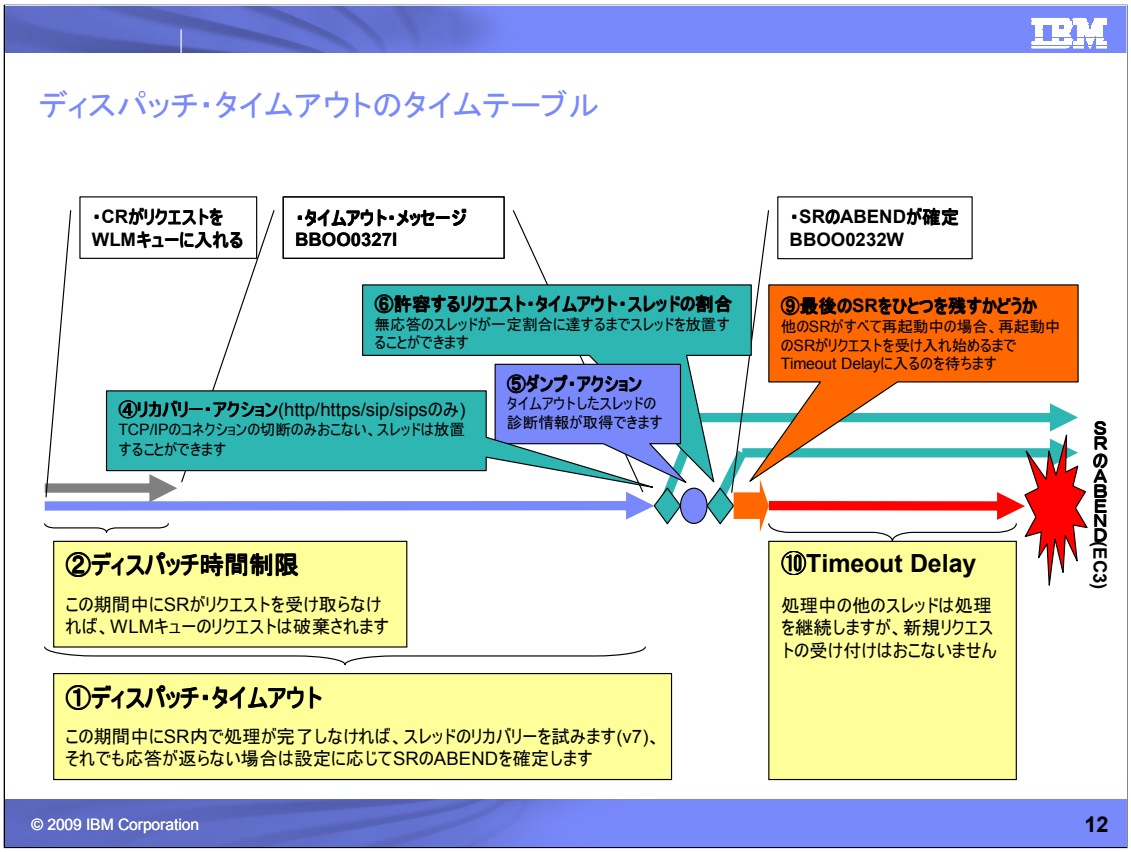
© 2009 IBM Corporation

タイムアウトに達すると、、、

- タイムアウトに達すると、稼働しているSRがABEND&再起動します
  - JVM内部では回復することができない暴走スレッドやハング・スレッドを、アドレススペースごと解除し、スレッドが握っていたリソースのロックなどを解放します
  - クライアントへのエンドポイントであるCR、CRAは残したままアプリケーションの稼働環境を再起動するため、クライアントへのエラー応答や、新規リクエストの稼働中のSRへの振り替えが行えます



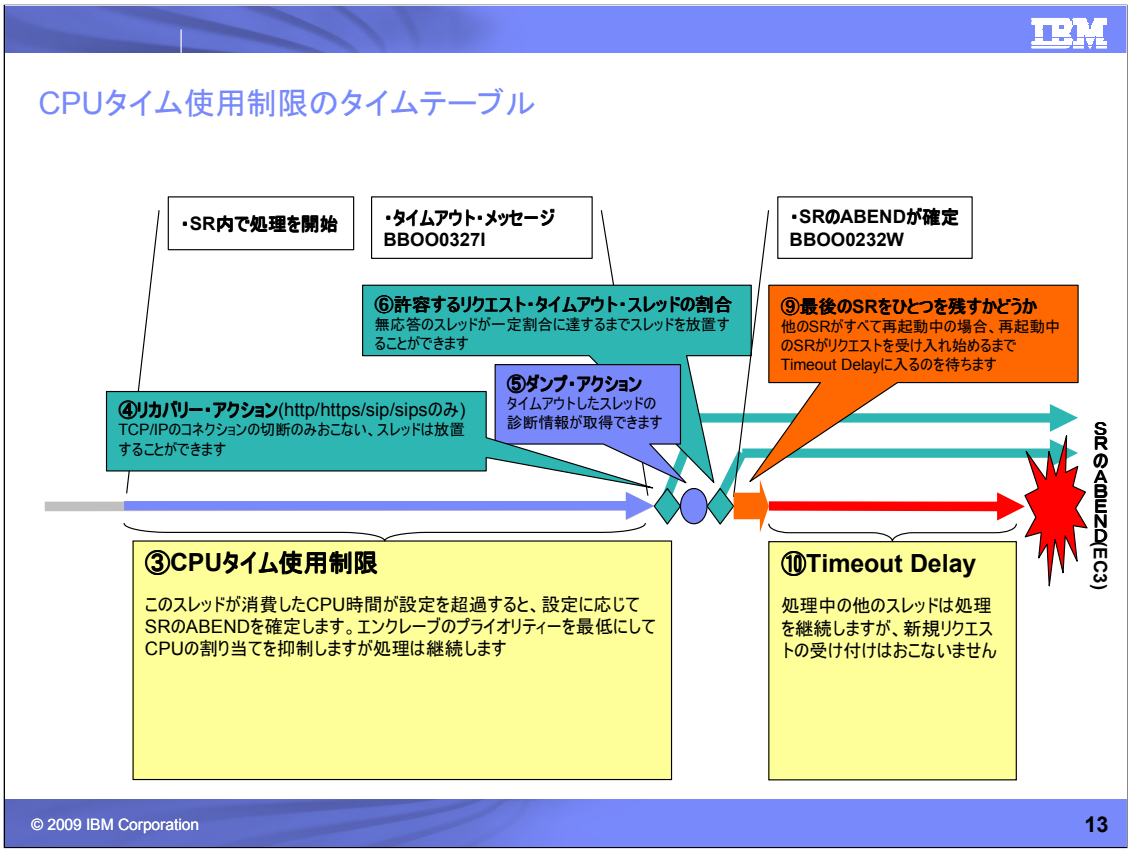
© 2009 IBM Corporation



ディスパッチ・タイムアウトについて、

CRがリクエストをWLMキューに入れてからSR ABENDに至るまでのタイムテーブルと、タイムアウト時間や分岐条件、発生するアクションを指定するパラメーターを記載しています。

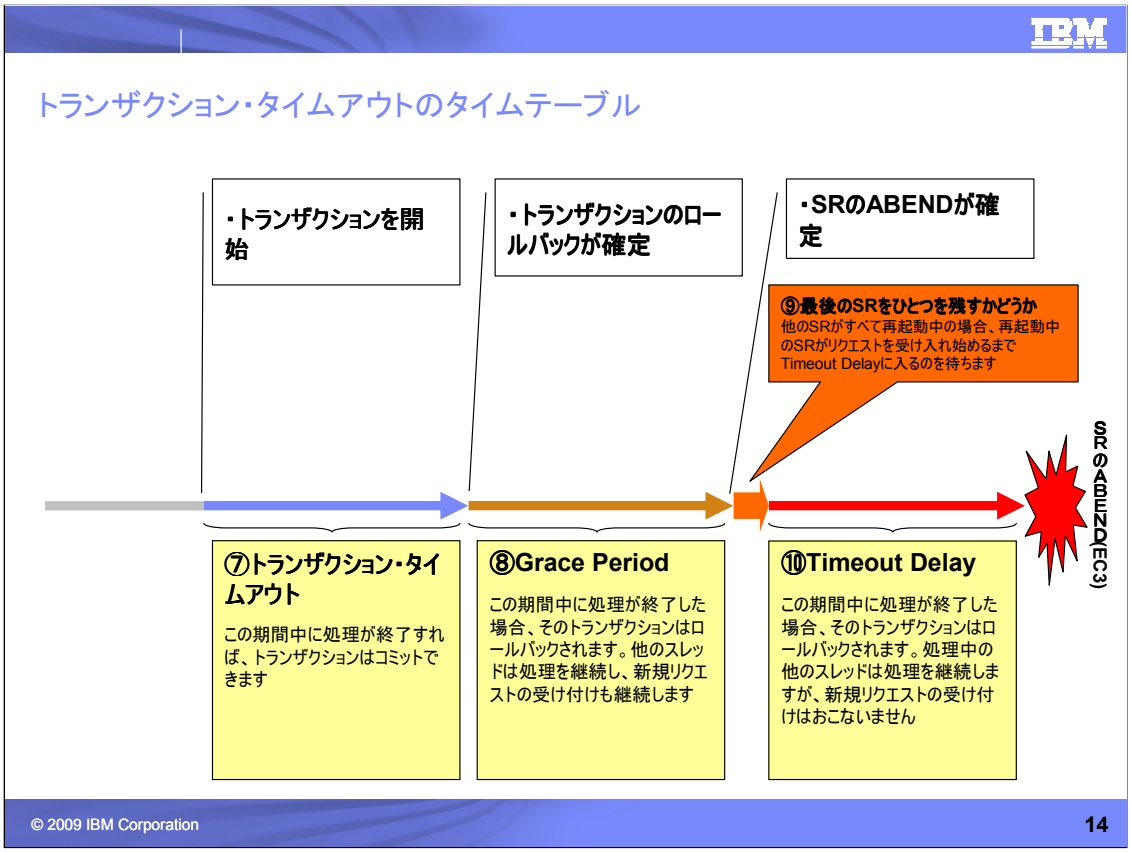
それぞれのパラメーターの番号は、以降のパラメーター説明ページの番号と対応しています。



CPUタイム使用制限について、

SR内で処理が開始されてからSR ABENDに至るまでのタイムテーブルと、タイムアウト時間や分岐条件、発生するアクションを指定するパラメーターを記載しています。

それぞれのパラメーターの番号は、以降のパラメーター説明ページの番号と対応しています。



トランザクション・タイムアウトについて、

トランザクションを開始してからSR ABENDに至るまでのタイムテーブルと、タイムアウト時間を指定するパラメーターを記載しています。

それぞれのパラメーターの番号は、以降のパラメーター説明ページの番号と対応しています。

スレッドのリカバリー

New V7

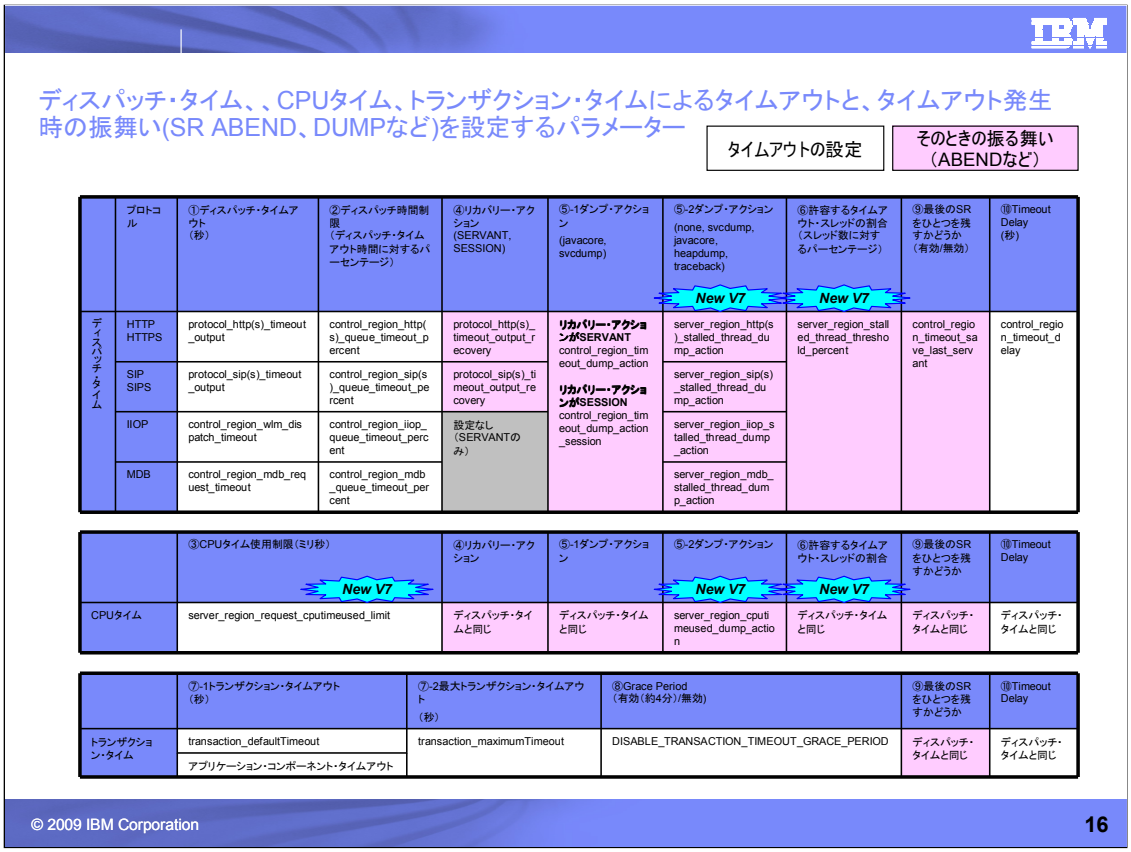
- WAS V7より、ディスパッチ・タイムアウトによりCRがSRをABENDする前に、SRがブロックされたスレッドの解除を試みるようになりました
- これまでは、Thread.sleep()や外部リソースの応答待ちなどによりスレッドがブロックされたままディスパッチ・タイムアウト時間を経過すると、即SR ABENDが確定しましたが、SRでスレッドのブロックを解除してCRに対して即座に応答を返すことが出来た場合には、ディスパッチ・タイムアウトとならずにSR ABENDを回避することが可能となります
- スレッドのブロックの解除は、ODIと呼ぶ割り込みオブジェクトを実装し登録したランタイムに対してSRのタイマーが割り込みをかけることによって行いますので、必ずブロックを解除できるわけではありません
- アプリケーションが例外をcatchして処理を継続している場合や無限ループからは抜けだすことが出来ません。この場合は従来と同じくスレッドは処理を継続し、設定にしたがってSRをABENDします

※Javaのランタイムでは、Object.wait(), Thread.sleep(), Thread.join(), Lock.lockInterruptibly(), Condition.await() などのマルチスレッド処理の制御に使用するAPIのブロックの解除が対応しています

© 2009 IBM Corporation

WAS for z/OS V7より、タイムアウト時間経過時にスレッドの解除を試みるようになりました。

スレッドの解除の試みにより、上記JavaランタイムのAPIではInterruptedExceptionが発生しますので、これらを使用しているアプリケーションでは例外処理の見直しが必要となる可能性があります。



ディスパッチ・タイム、CPUタイム、トランザクション・タイムによるタイムアウトにかかわるパラメーターを一覧してあります。

それぞれのパラメーターの番号は、先のタイムテーブルのパラメーターの番号、以降のパラメーター説明ページの番号と対応しています。

個々のパラメーターの説明は、以降のパラメーター説明のページ、および、下記Infocenterを参照してください。

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/rtrb_controllingtimeout.html

①ディスパッチ・タイムアウト

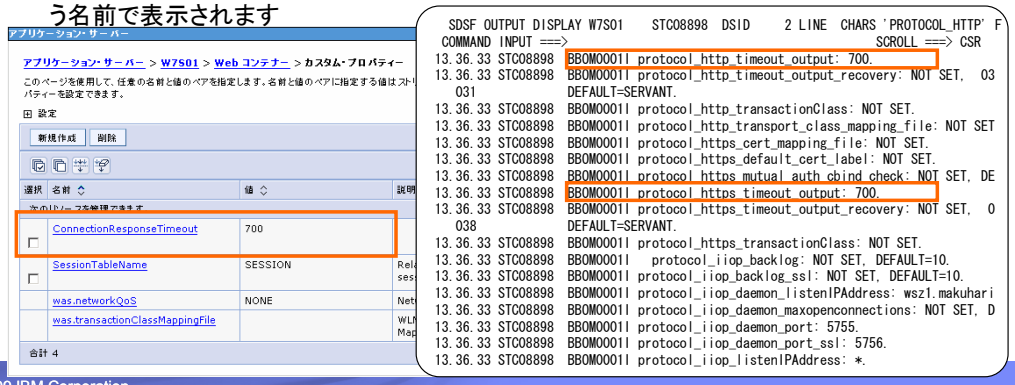
- CRがリクエストをWLMキューに入れてから、SRにディスパッチされて処理が完了するまでの時間が設定を超過すると、CRはリクエストをクリーンナップ（WLMキューからリクエストを破棄、HTTP(S)/SIP(S)/IIOPではリクエスト元のTCP/IPソケットを切断）し、設定に応じてSRをABENDさせます
- SR ABEND時には、そのSRで処理されていた他のリクエストも終了させられてしまいます
  - ディスパッチ・タイムアウトを、稼働する業務処理時間に合わせて適切に設定する必要があります
  - 処理時間の大きく異なるアプリケーションは別のサーバーで動かすようにする方が良いでしょう
- タイムアウト値を0に設定して無効にする、あるいは、リカバリー・アクションや許容するタイムアウト・スレッドの割合設定によってSRのABENDを回避することもできます
  - ビジーループ等が発生している場合、そのスレッドのみを停止させる手段がありませんので、最終的には手動でSRを停止させる必要があります
  - 何らかのリソースにロックをかけている場合、SRを停止させるまでそのロックが解除されない可能性があります
  - リクエストを処理するためのスレッドがひとつ使えなくなります

© 2009 IBM Corporation

ディスパッチ・タイムアウトの説明 その1/4

ディスパッチ・タイムアウトの設定方法 – HTTP/HTTPS

- 管理コンソールでの設定
  - 設定箇所:アプリケーション・サーバー > サーバー名 > Web コンテナー設定 > Webコンテナー >カスタム・プロパティーに、以下の変数を設定
    - ConnectionResponseTimeout=(秒数)
  - この変数を設定しない場合のデフォルト値は300秒です
  - 0を指定すると、タイムアウトが無効になります
- JOBLOGには、protocol_http_timeout_outputおよびprotocol_https_timeout_outputという名前で表示されます



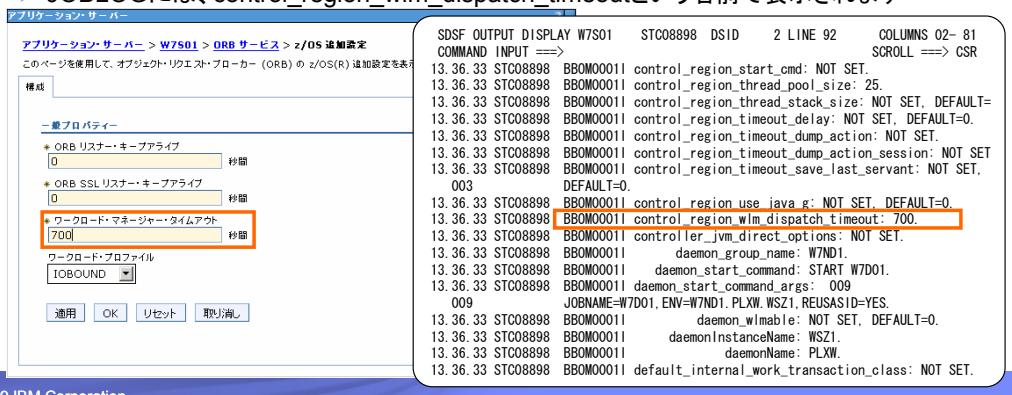

```
 SDSF OUTPUT DISPLAY W7S01    STC08898  DSID     2 LINE  CHARS 'PROTOCOL_HTTP' F
 COMMAND INPUT ===>                                    SCROLL ===> CSR
13.36.33 STC08898  BBOMO001I protocol_http_timeout_output: 700.
13.36.33 STC08898  BBOMO001I protocol_http_timeout_output_recovery: NOT SET,  03
   031             DEFAULT=SERVANT.
13.36.33 STC08898  BBOMO001I protocol_http_transactionClass: NOT SET.
13.36.33 STC08898  BBOMO001I protocol_http_transport_class_mapping_file: NOT SET
13.36.33 STC08898  BBOMO001I protocol_https_cert_mapping_file: NOT SET.
13.36.33 STC08898  BBOMO001I protocol_https_default_cert_label: NOT SET.
13.36.33 STC08898  BBOMO001I protocol_https_mutual_auth_cbind_check: NOT SET, DE
13.36.33 STC08898  BBOMO001I protocol_https_timeout_output: 700.
13.36.33 STC08898  BBOMO001I protocol_https_timeout_output_recovery: NOT SET,  0
   038             DEFAULT=SERVANT.
13.36.33 STC08898  BBOMO001I protocol_https_transactionClass: NOT SET.
13.36.33 STC08898  BBOMO001I   protocol_iiop_backlog: NOT SET, DEFAULT=10.
13.36.33 STC08898  BBOMO001I protocol_iiop_backlog_ssl: NOT SET, DEFAULT=10.
13.36.33 STC08898  BBOMO001I protocol_iiop_daemon_listenIPAddress: wsz1.makuhari
13.36.33 STC08898  BBOMO001I protocol_iiop_daemon_maxopenconnections: NOT SET, D
13.36.33 STC08898  BBOMO001I protocol_iiop_daemon_port: 5755.
13.36.33 STC08898  BBOMO001I protocol_iiop_daemon_port_ssl: 5756.
13.36.33 STC08898  BBOMO001I protocol_iiop_listenIPAddress: *.
```

© 2009 IBM Corporation

ディスパッチ・タイムアウトの説明 その2/4

ディスパッチ・タイムアウトの設定方法 - IIOP

- 管理コンソールでの設定
  - 設定箇所:アプリケーション・サーバー > サーバー名 > コンテナー・サービス > ORBサービス > z/OS追加設定の、下記のフィールドに設定
    - ワークロード・マネージャー・タイムアウト
  - 設定値:秒数 (デフォルトは1200秒)
  - 0を指定すると、タイムアウトが無効になります
- JOBLOGには、control_region_wlm_dispatch_timeoutという名前で表示されます

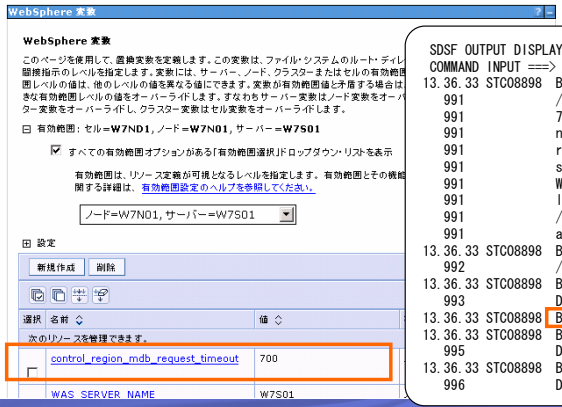


© 2009 IBM Corporation

ディスパッチ・タイムアウトの説明 その3/4

ディスパッチ・タイムアウトの設定方法 – MDB(ListenerPort)

- 管理コンソールでの設定
  - 設定箇所：環境 > WebSphere 変数で適切な有効範囲を選択し、以下の変数を設定
    - control_region_mdb_request_timeout=（秒数）
  - この変数を設定しない場合のデフォルト値は120秒です
  - 0を指定すると、タイムアウトが無効になります




```
SDSF OUTPUT DISPLAY W7S01    STC08898  DSID     2 LINE 73       COLUMNS 02- 81
COMMAND INPUT ===>                                            SCROLL ===> CSR
13.36.33 STC08898  BBOM0001I  control_region_libpath:  991
   991             /WebSphere/W7ND1/W7N00/DeploymentManager/java64/bin:/WebSpher
   991             7N00/DeploymentManager/java64/bin/j9vm:/WebSphere/W7ND1/W7N00
   991             ntManager/java64/lib/s390/j9vm:/WebSphere/W7ND1/W7N00/Deploym
   991             r/java64/lib/s390:/WebSphere/W7ND1/W7N00/DeploymentManager/ja
   991             s390x/j9vm:/WebSphere/W7ND1/W7N00/DeploymentManager/java64/li
   991             WebSphere/W7ND1/W7N00/DeploymentManager/lib:/WebSphere/W7ND1/
   991             loymentManager/java64/jre/bin:/WebSphere/W7ND1/W7N00/Deployme
   991             /java64/jre/lib/s390/j9vm:/WebSphere/W7ND1/W7N00/DeploymentMa
   991             a64/jre/lib/s390x/j9vm:/WebSphere/W7ND1/W7N00/DeploymentManag
13.36.33 STC08898  BB000156I CONTINUATION 0001 FOR BBOM0001I ON TCB=005E6E00  99
   992             /jre/lib/s390x.
13.36.33 STC08898  BBOM0001I control_region_mdb_queue_timeout_percent: NOT SET,
   993             DEFAULT=99.
13.36.33 STC08898  BBOM0001I control_region_mdb_request_timeout: 700.
13.36.33 STC08898  BBOM0001I control_region_sip_queue_timeout_percent: NOT SET,
   995             DEFAULT=99.
13.36.33 STC08898  BBOM0001I control_region_sips_queue_timeout_percent: NOT SET,
   996             DEFAULT=99.
```

© 2009 IBM Corporation

ディスパッチ・タイムアウトの説明 その4/4

②ディスパッチ時間制限

- CRがWLMキューにPUTしたリクエストをSRが受け取らないまま一定時間が経過した場合に、CRはリクエストをクリーンナップ（WLMキューからリクエストを破棄、HTTP(S)/SIP(S)/IIOPではリクエスト元のTCP/IPソケットを切断）します
- この場合SRで処理は開始していないので、SRのABENDは行いません
- SRがハングしているときやすべてのスレッドが使用中でリクエストが滞留した場合に、SRにディスパッチされた直後にタイムアウトしてSR ABEND確定となるのを回避することができます
- 管理コンソールでの設定
  - 設定箇所：環境 > WebSphere 変数で適切な有効範囲を選択し、以下の変数を設定
    - control_region_http_queue_timeout_percent ... HTTP
    - control_region_https_queue_timeout_percent ... HTTPS
    - control_region_sip_queue_timeout_percent ... SIP
    - control_region_sips_queue_timeout_percent ... SIPS
    - control_region_iiop_queue_timeout_percent ... IIOP
    - control_region_mdb_queue_timeout_percent ... MDB
  - 設定値：ディスパッチ・タイムアウト時間に対するパーセンテージ（0-99で指定）
  - この変数を設定しない場合のデフォルト値は99パーセントです。SRがハングしていてディスパッチ出来ない場合は、リクエストはディスパッチ・タイムアウト時間が経過するまでWLMキューに滞留し続けます

© 2009 IBM Corporation

(ABEND頻発時要検討)

③CPUタイム使用制限

New V7

- WAS V7より、ひとつのリクエストでスレッドが消費できるCPU時間を抑制するパラメーターが追加されました
- スレッドが消費したCPU時間が設定を超過すると、設定により診断情報を取得し、エンクレーブのプライオリティーを下げてCPUの割り当てを抑制します
  - 常に制限が有効なわけではありません。たとえば、JNIからProgram Callで他のアドレス空間にあるプログラムにブランチしているような場合は、PCルーチンから戻るまでCPU時間の超過を検知するCPUタイマーからのシグナルを受け取ることができません
  - プライオリティーを下げるだけなので、CPUの利用度が低ければ、CPUタイム使用制限に達しても処理は実行され続けます
- CRではリクエストをクリーンナップし、スレッドを無応答とみなして、設定にしたがってSRをABENDさせます
- 管理コンソールでの設定
  - 設定箇所：環境 > WebSphere 変数で適切な有効範囲を選択し、以下の変数を設定
    - server_region_request_cputimeused_limit
  - 設定値：アプリケーション要求が消費できる CPU 時間を、ミリ秒単位で指定

© 2009 IBM Corporation

エンクレーブのプライオリティーを下げるために、WLMサービスのIWMERESマクロでエンクレーブをQUIESCEします。

④リカバリー・アクション

- HTTP(S)/SIP(S)の場合は、ディスパッチ・タイムアウト、CPUタイム使用制限に達してもSRをABENDさせずにリクエストのクリーンナップ（ WLMキューからリクエストを破棄、リクエスト元のTCP/IPソケットを切断）のみ行い、処理しているスレッドは放置する、という設定が可能です
- 管理コンソールでの設定
  - 設定箇所：環境 > WebSphere 変数で適切な有効範囲を選択し、以下の変数を設定
    - protocol_http_timeout_output_recovery ... HTTP
    - protocol_https_timeout_output_recovery ... HTTPS
    - protocol_sip_timeout_output_recovery ... SIP
    - protocol_sips_timeout_output_recovery ... SIPS
  - 設定値：次の2つのどちらかを指定する
    - SERVANT ... タイムアウトに達すると、スレッドのリカバリーを試み、ブロックされたスレッドを解除できなければCRはリクエストをクリーンナップし、設定にしたがってSRをABEND、再起動させる（デフォルト）
    - SESSION ... タイムアウトに達すると、CRはリクエストのクリーンナップのみ行う。処理を中断するための試行は何も行わず、スレッドは稼働し続ける
- 設定値SESSIONは、スレッドのリカバリーなどのWAS V7で追加された機能を活用できませんので、旧来よりタイムアウトによるABENDを避けるためにこのパラメーターを設定している場合は、許容するタイムアウト・スレッドの割合などの設定に置き換えて、当パラメータにはデフォルトのSERVANTを使用することをおすすめします

© 2009 IBM Corporation

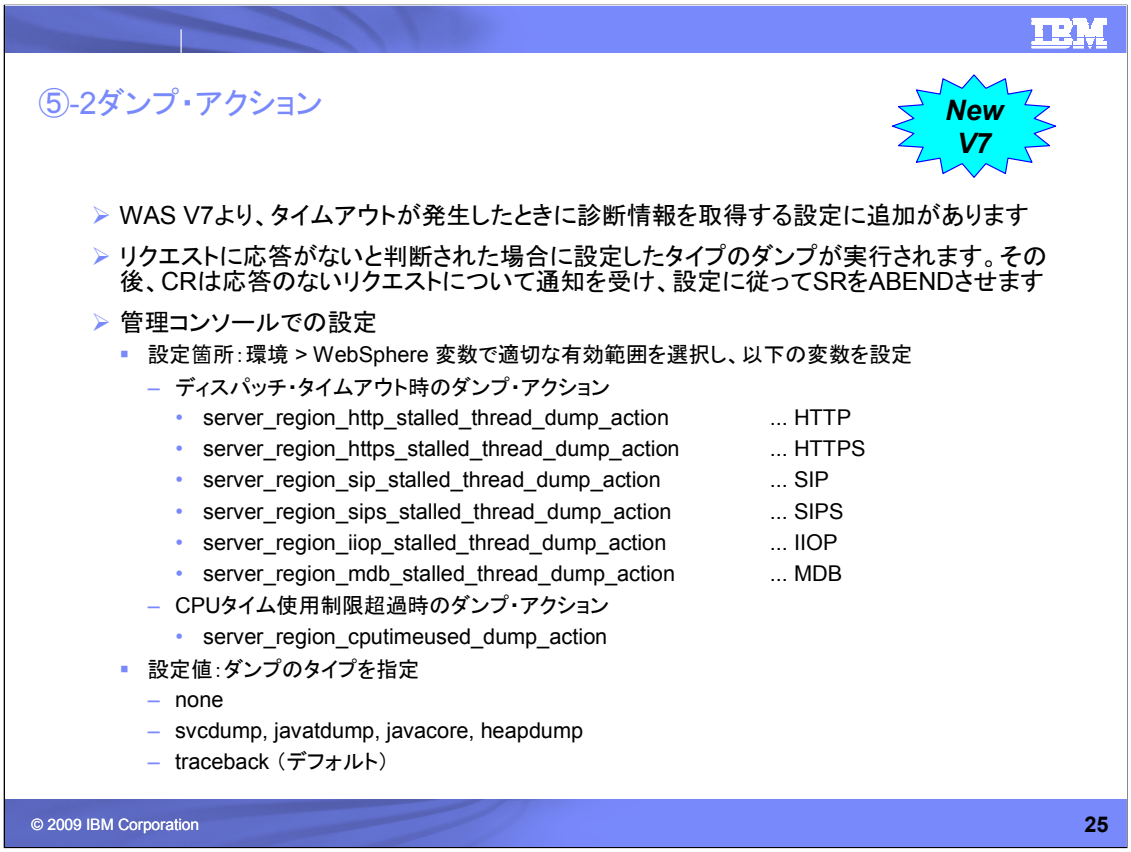
⑤-1ダンプ・アクション

- タイムアウトが発生したときに、設定したタイプのダンプを実行します
- 管理コンソールでの設定
  - 設定箇所：環境 > WebSphere 変数で適切な有効範囲を選択し、以下の変数を設定
    - リカバリー・アクションがSESSIONでなく、Timeout Delayが0以外でタイムアウトとなったとき
      - control_region_timeout_dump_action
    - リカバリー・アクションがSESSIONで、タイムアウトとなったとき
      - control_region_timeout_dump_action_session
  - 設定値：JAVACORE, SVCDUMP, JAVATDUMP
  - この変数を設定しない場合はどちらも取得しない設定となります
- MODIFYコマンドでの設定
  - コマンド：動的に上記設定を変更できます
    - リカバリー・アクションがSESSIONでなく、Timeout Delayが0以外でタイムアウトとなったとき
      - F <server>,TIMEOUTDUMPACTION=n
    - リカバリー・アクションがSESSIONで、タイムアウトとなったとき
      - F <server>, TIMEOUTDUMPACTIONSESSION=n
  - 設定値(n)：NONE, JAVACORE, SVCDUMP, JAVATDUMP

© 2009 IBM Corporation

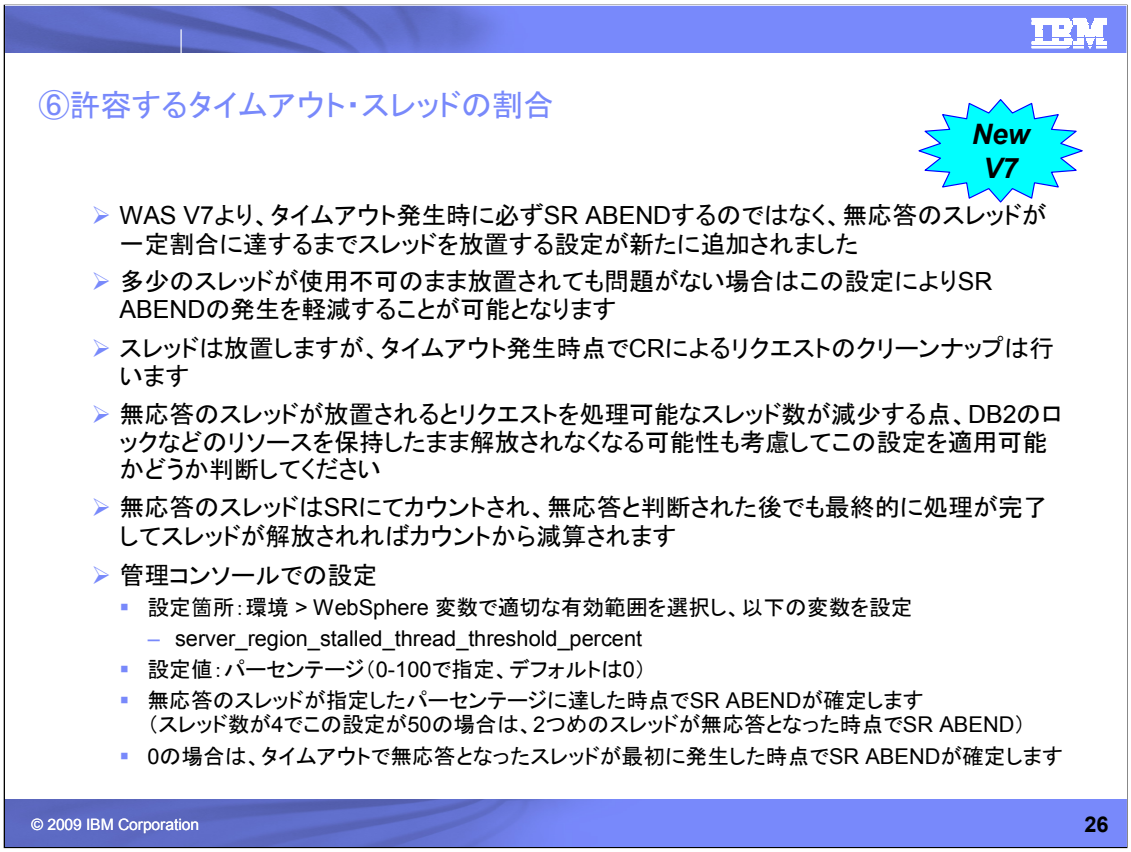
次ページの⑤-2と独立して設定できます。

サーバー稼働中に一時的にDUMP取得させたい場合にMODIFYコマンドにて設定変更が可能です。



タイムアウト発生時の状況を確認するためにtraceback（ディスパッチ・スレッドのコールスタックを出力します）は有用です。

通常はデフォルトで運用し、必要に応じてdumpを設定、もしくは、⑤-1にて動的にDUMP出力を設定することをお奨めします。



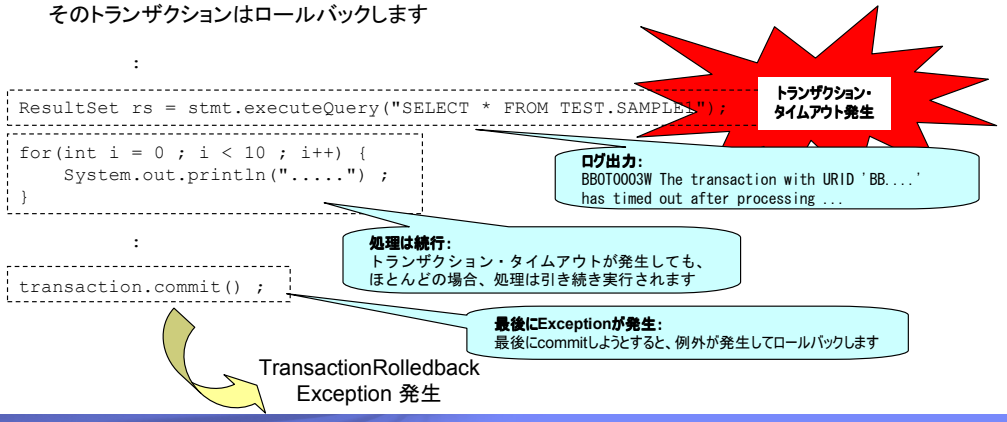
(ABEND頻発時要検討)

スレッド数が4でこの設定が51の場合は、3つめのスレッドが無応答となった時点でSR ABENDが確定します。

この設定が100の場合は、すべてのスレッドが使用不可になった時点でSR ABENDが確定します

⑦トランザクション・タイムアウト

- トランザクション・タイムアウトが発生すると、メッセージBBOT0003Wが出力されます
- 即座にアプリケーションに制御が戻るわけではなく、処理は継続されます
  - たとえば、DB2の処理中にトランザクション・タイムアウトが発生しても、処理は中断されません
- 最終的には、ロールバックします
  - UserTransaction.commit()メソッドを呼ぶ、CMTのEJBから抜ける、等によってコミットを発行しても、そのトランザクションはロールバックします



© 2009 IBM Corporation

トランザクション・タイムアウトの説明 その1/3

トランザクション・タイムアウトの設定方法 - アプリケーション・コンポーネント・タイムアウト

- アプリケーション・コンポーネント・タイムアウト
  - アプリケーションの中で、トランザクション・タイムアウト値を明示指定することが可能です
  - 設定しない場合は合計トランザクション存続時間タイムアウトの指定が有効になります

サーブレット、またはBean管理トランザクション(BMT) のEJBでユーザー・トランザクションを使用する場合は、プログラム中で指定可能

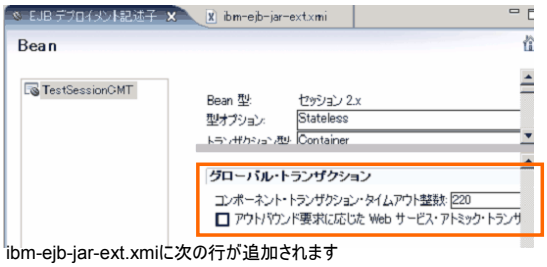
```
InitialContext ic =
    new InitialContext() ;
UserTransaction ut = (UserTransaction)
    ic.lookup("java:comp/UserTransaction");

ut.setTransactionTimeout(120) ;

ut.begin() ;
    :
//ここでトランザクション処理
    :
ut.commit() ;
```

コンテナー管理トランザクション(CMT)のEJBの場合はWASの拡張機能で指定可能

RAD等



ibm-ejb-jar-ext.xmiに次の行が追加されます

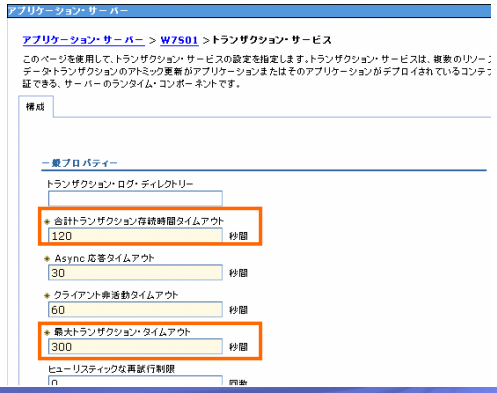
```
<GlobalTransaction
xmi:id="GlobalTransaction_114...."
componentTransactionTimeout="220" />
```

© 2009 IBM Corporation

トランザクション・タイムアウトの説明 その2/3

トランザクション・タイムアウトの設定方法 – 合計トランザクション存続時間タイムアウト、最大トランザクション・タイムアウト

- 合計トランザクション存続時間タイムアウト
  - アプリケーション・コンポーネントタイムアウトを指定しない場合、デフォルト値として適用されます
  - このタイムアウトを 0 に設定すると、最大トランザクション・タイムアウトの値が代わりに使用されます
- 最大トランザクション・タイムアウト
  - むやみに大きな値が指定されることを防ぐために、アプリケーション・コンポーネントタイムアウト、合計トランザクション存続時間タイムアウトに対する上限値を設定することが可能です



- 管理コンソールでの設定
  - 設定箇所：アプリケーション・サーバー > サーバー名 > コンテナー・サービス > トランザクション・サービス の、下記のフィールドに設定
    - 合計トランザクション存続タイムアウト
    - 最大トランザクション・タイムアウト
  - 設定値：秒数

© 2009 IBM Corporation

トランザクション・タイムアウトの説明 その3/3

⑧Grace Period

- トランザクション・タイムアウトが発生した段階で、その処理はロールバックされることが決定されます
  - 実際には、そのトランザクションが終了するときにロールバックが実行されます
  - このことは、該当のトランザクションが継続して実行され続けた場合、例えばDB2のロックは保持され続けるということを意味します
- WAS for z/OSでは、トランザクション・タイムアウト発生後、該当のトランザクションがGrace Periodを過ぎても実行され続けていた場合、そのトランザクションの稼働するSRをABENDさせることを決定します
- Grace Period中は、他のスレッドは通常どおりリクエストを受け付けて処理します
- Grace Periodは、約4分または無効のどちらかを指定できます
- 管理コンソールでの設定
  - 設定箇所：アプリケーション・サーバー > サーバー名 > コンテナー・サービス > トランザクション・サービス > カスタム・プロパティーに、以下の変数を設定
    - DISABLE_TRANSACTION_TIMEOUT_GRACE_PERIOD
  - 設定値：次の2つのどちらかを指定する
    - true ... Grace Periodが無効
    - false ... Grace Periodが有効（デフォルト）
  - 無効とした場合、トランザクション・タイムアウトが発生した段階で、そのSRはABENDすることが決定されます

© 2009 IBM Corporation

⑨最後のSRをひとつ残すかどうか

- タイムアウトとなるリクエストが連続して発生した場合に、すべてのSRがタイムアウトによって同時にABENDし、サービスが中断するのを軽減することが可能です
- 複数SR構成にて使用可能なSRが最後の1枚となった場合、タイムアウトの発生によってSRのABENDが確定してもすぐにはSRをABENDさせず、再起動中のSRがリクエストを受け付け始めてから、Timeout Delay、ABENDさせます
- この設定により、タイムアウトにてリカバリー出来なかったスレッドは中断されないで処理を継続しますので、DB2のロックなどのリソースの解放がおくれる可能性があります
- 管理コンソールでの設定
  - 設定箇所：環境 > WebSphere 変数で適切な有効範囲を選択し、以下の変数を設定
    - control_region_timeout_save_last_servant
  - 設定値：次の2つのどちらかを指定する
    - 0 ... 無効（デフォルト）
    - 1 ... 有効
  - 1に設定すると、使用可能な最後のSRは、新規のSRがリクエストを受け付け始めるまでTimeout Delayの開始、および、SR ABENDが抑止されます
  - 起動時のSRが2枚以上の構成（サーバー・インスタンス設定にて、複数インスタンス使用可能で、かつ、最小インスタンス数が1より大きい）の場合のみ有効です

© 2009 IBM Corporation

使用可能なSRが最後の1枚となったときにタイムアウトが発生しても、再起動中のSRがリクエストを受け付け始めるまではTimeout Delay期間に入らず新規リクエストを受け付け続けます。

再起動中のSRがリクエストを受け付け始めた時点で最後のSRはTimeout Delay期間に突入し、ABENDします。

⑩Timeout Delay

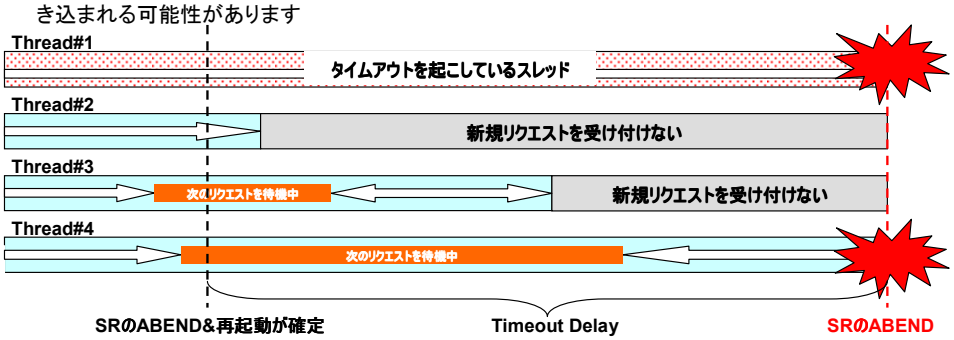
- タイムアウトによってABENDすることが決定してから、実際にABENDするまでのTimeout Delayを設定することが可能です
- Timeout Delay中は、新規のリクエストの受け付けを停止します
  - SR中の各スレッドは、それぞれの処理が終了した時点で、次のリクエストを受け付けるための待ち状態に入るか、それとも処理を停止するかを決定します
  - ABENDが確定し、Timeout Delay中に終了したスレッドは、次のリクエストを受けつけることはできないと判断し、処理を停止します
  - リクエストの振り分けをWLMによって行う場合はTimeout Delay中であってもリクエストを受け付けてしまう場合があります（次ページ参照）
- リクエストの処理時間に対して、十分長いTimeout Delayを設定することをお勧めします
  - Timeout Delayが十分に長ければ、リクエスト数の多い環境ではABEND前に全てのスレッドがリクエストを受け付けない状態になっていることが期待できます
- 管理コンソールでの設定
  - 設定箇所：環境 > WebSphere 変数で適切な有効範囲を選択し、以下の変数を設定
    - control_region_timeout_delay=（秒数）
  - この変数を設定しない場合のデフォルト値は0です
  - 0の場合はTimeout Delayは無効になり、ABENDの確定後すぐにSRはCRによってABENDされます

© 2009 IBM Corporation

Timeout Delayの説明 その1/2

Timeout Delay時のリクエスト受け付け（WLM使用時）

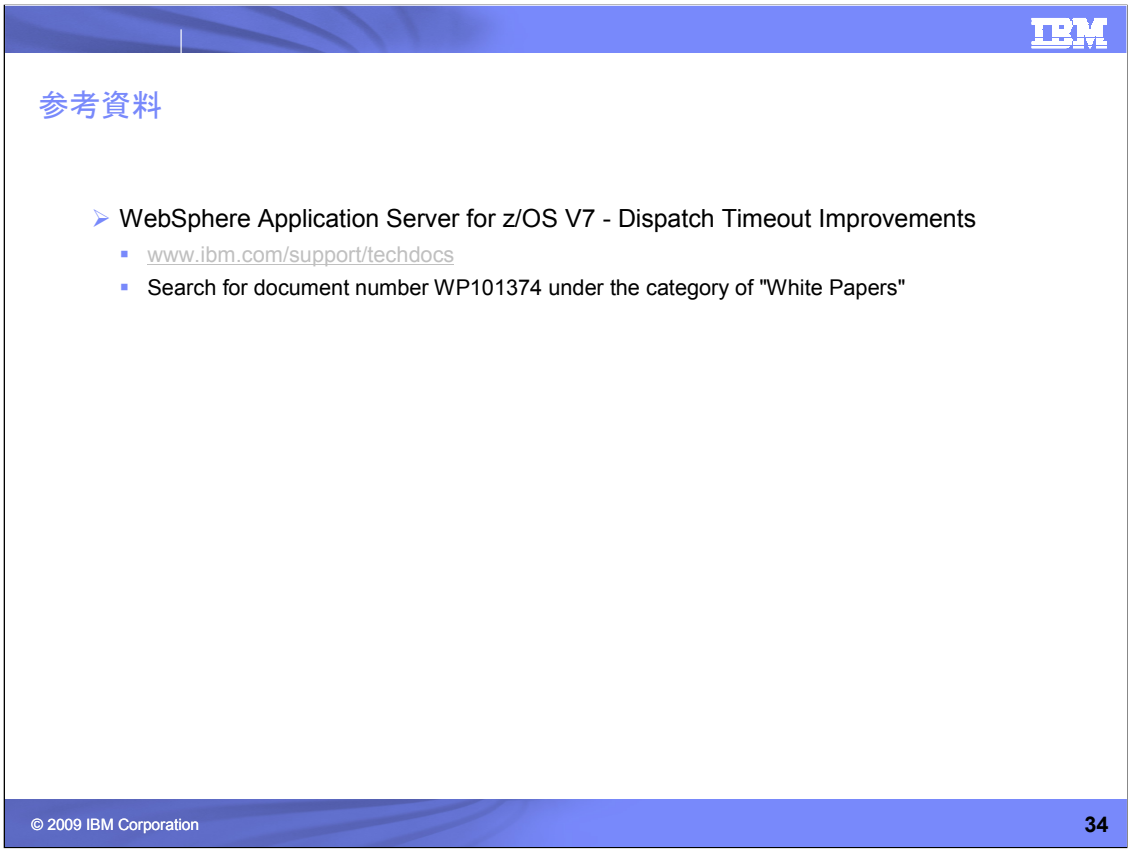
- リクエストの振り分けをWLMによって行う場合（デフォルト）、Timeout Delay中であってもリクエストを受け付けてしまう場合があります
  - ABENDが確定する前に終了したスレッドは、次のリクエストを受け付けるための待ち状態に入るため、Timeout Delay中であってもリクエストを受け付けます
  - したがって、Timeout Delayの終わりの方で受け付けられたリクエストは、処理の途中でABENDに巻き込まれる可能性があります



- ※リクエストの振り分けポリシー設定については、カスタマイズ基礎編「SRが複数枚の場合の振分け」をご参照ください

© 2009 IBM Corporation

Timeout Delayの説明 その2/2



WAS V7で拡張されたタイムアウトについて解説されています。

ポート・オープン・タイミングの設定

© 2009 IBM Corporation

ポート・オープン・タイミングの設定

- WAS for z/OSは、リクエストを受け付けるCRと、リクエストを実行するSRに構成が分かれています
- CRが起動している状態で、SRがまだ処理を実行できない場合でもリクエストを受け付けてSRが処理可能となるまでリクエストを保留するか、SRが処理可能でない場合はリクエストの受け付け自体を行わないか、を設定することが可能です
- 管理コンソールでの設定
  - 設定箇所：環境 > WebSphere 変数で適切な有効範囲を選択し、以下の変数を設定

サーバー起動後、すべてのSR使用不可となった時

- control_region_dreg_on_no_srs
  - 設定値
    0(default) ... HTTP/IIOP/MDB(ListenerPort)リクエストの受け付けを停止しない
    1 ... リクエストの受け付けを停止する
- control_region_confirm_recovery_on_no_srs（control_region_dreg_on_no_srs=1のとき有効）
  - 設定値
    0(default) … 受け付けを停止後、最小数として設定した数のSRが使用可能になれば受け付け再開する
    1 … WTORメッセージBBOO0297Aに対する応答があるまでリクエストの受け付け再開を保留する

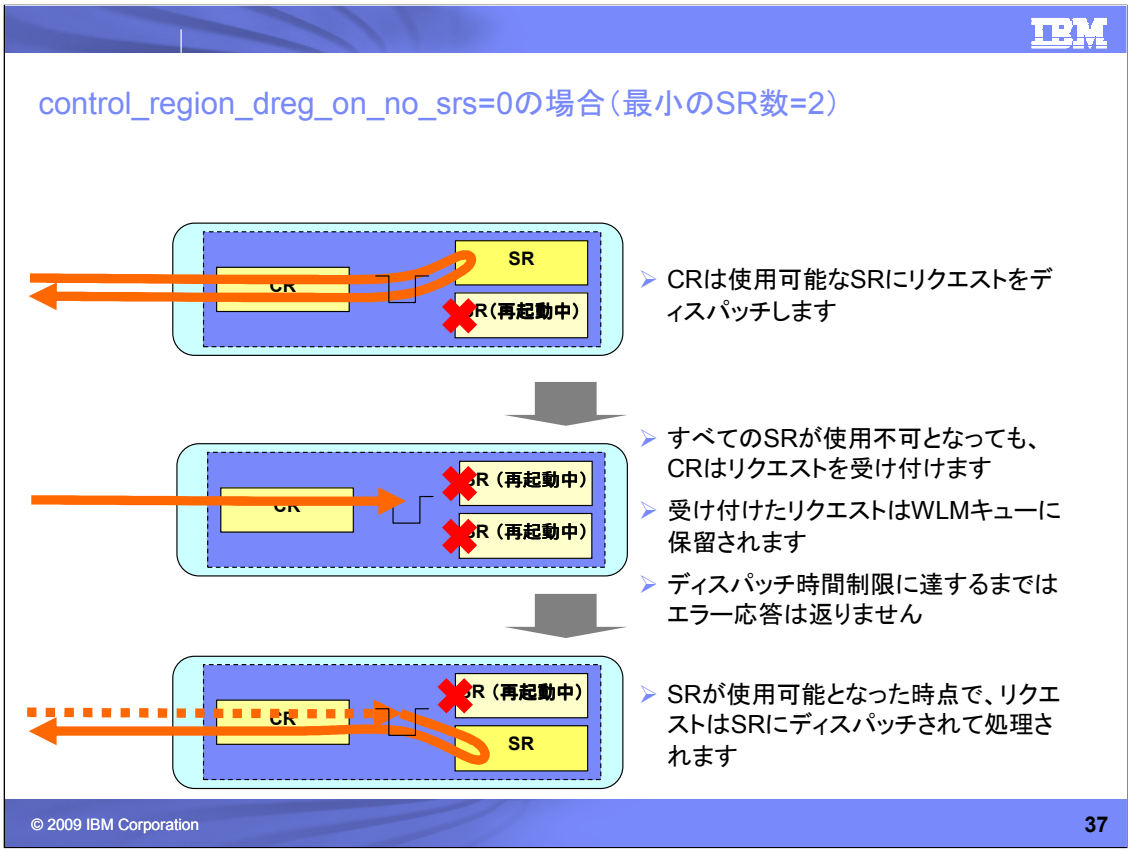
サーバー起動時

- protocol_accept_http_work_after_min_srs
  - 設定値
    true(default) … 最小数として設定した数のSRが起動するまでHTTPリクエストの受け付けを開始しない
    false … CRが起動した時点でHTTPリクエストの受け付けを開始する
- protocol_accept_iiop_work_after_min_srs
  - 設定値
    true … 最小数として設定した数のSRが起動するまでIIOPリクエストの受け付けを開始しない
    false(default) … CRが起動した時点でIIOPリクエストの受け付けを開始する

© 2009 IBM Corporation

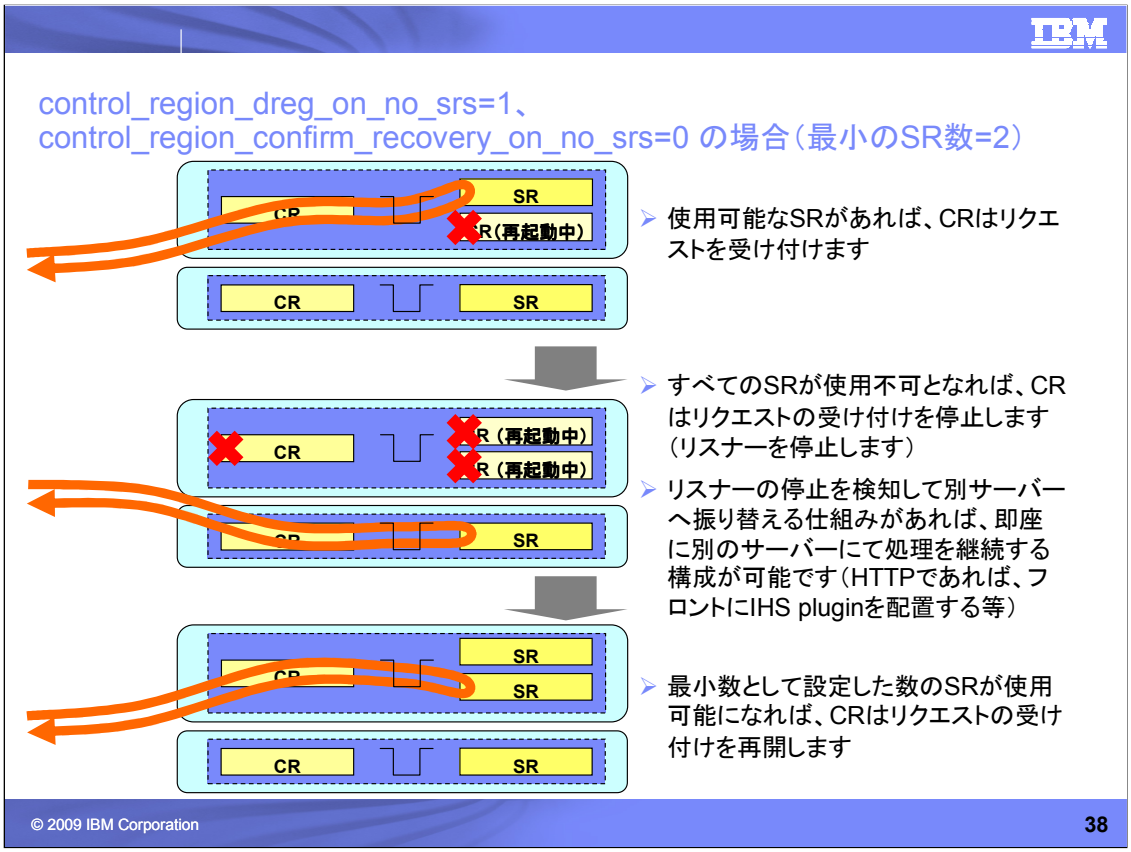
WAS for z/OS特有のCRとSRの複数アドレス・スペース構成での振舞いを規定する設定です。

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/trun_auto_noservant.html



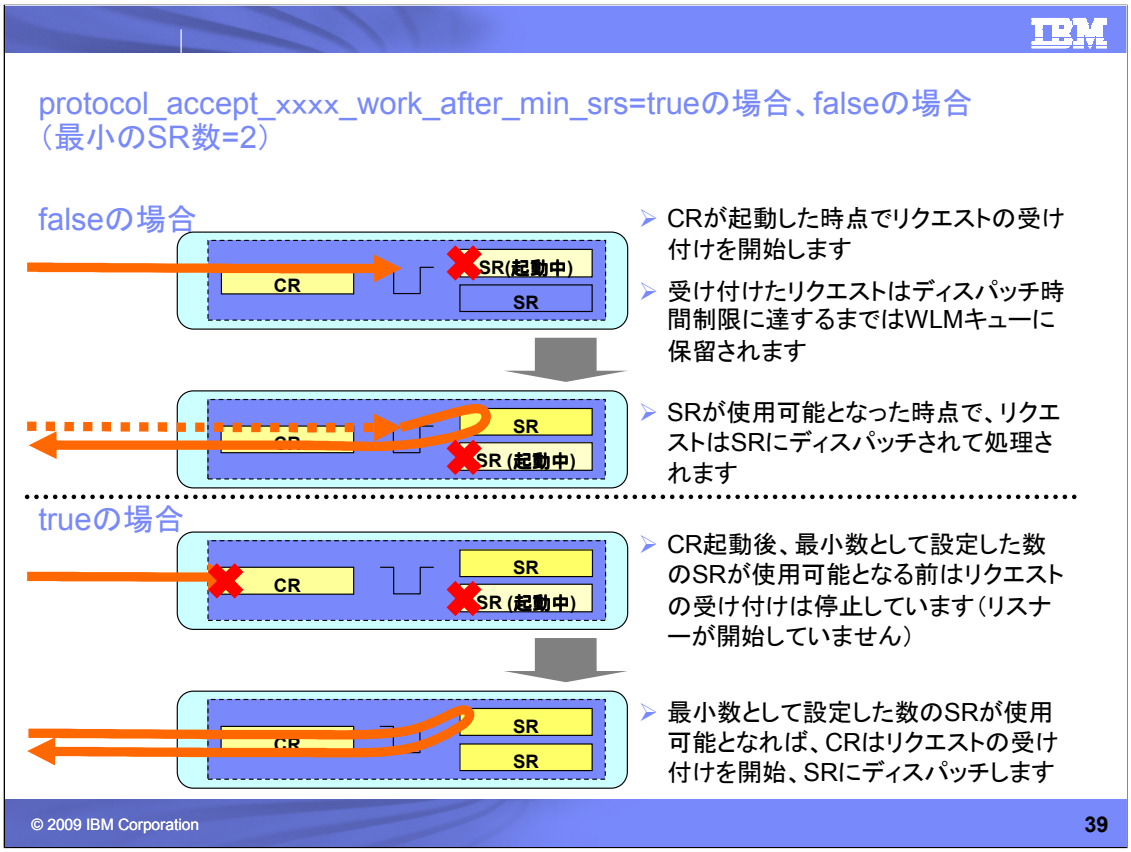
すべてのSRが使用不可になってもリクエストの受け付けを停止しません。

リクエストはWLMキューに保留され、SRの再起動が完了すればディスパッチされて処理されます。



すべてのSRが使用不可になったらリクエストの受け付けを停止します。

クラスター構成で、別のサーバーで処理が可能なときはそちらに処理を振り分けたい場合などにこの設定にします。



サーバー起動時に、SRが起動して処理可能となってからリクエストを受け付けたい場合にtrueに設定します。

（クラスター構成で、別のサーバーが先に処理可能となっているときはそちらに処理を振り分けたい場合などです）

IBM

wsadminツールとMBean

© 2009 IBM Corporation

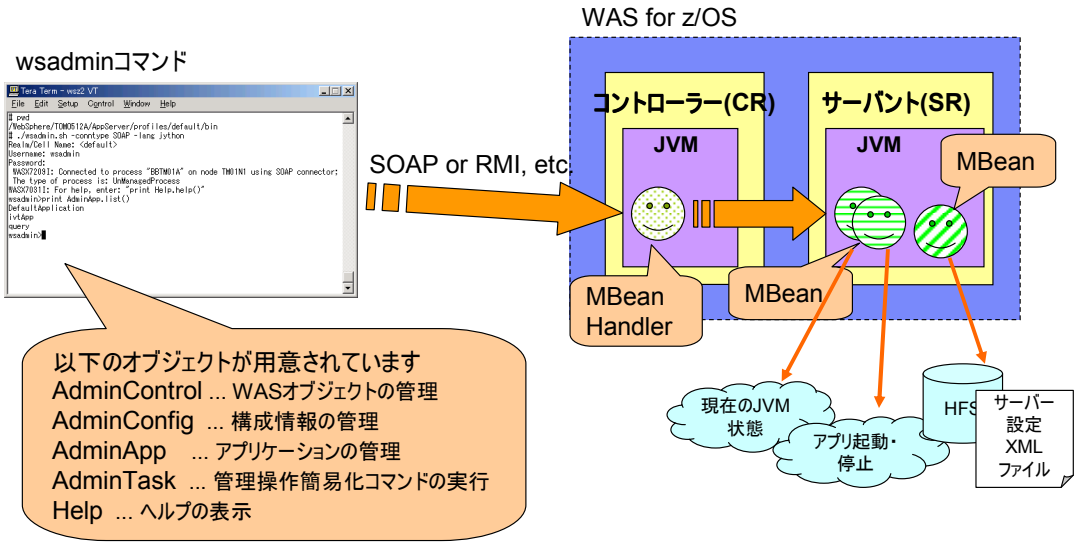
wsadminとは？

- 管理コンソールと同等の機能を、コマンドライン入力もしくはスクリプトで実現する仕組みです
- USSのシェル上、もしくはPCのコマンド・プロンプトから起動し、管理対象のWASと通信します
- スクリプト言語は、JythonまたはJaclの2つの言語を使用可能です
  - WAS V7.0ではJythonの使用が推奨されます。Jaclからの移行は必須ではありませんが、新規アプリケーションではJythonを使用することを検討してください
- 接続タイプは以下の5つから選択できます
  - SOAP ... SOAPで接続
  - RMI ... RMI / IIOPで接続 （WAS V7.0より非推奨）
  - JSR160RMI ... JMXリモートAPI（JSR 160）RMIで接続 （WAS V7.0より追加）
  - IPC ... プロセス間通信(IPC)で接続 （WAS V7.0より追加）
  - NONE ... ローカルHFSアクセス
    （スタンドアロン構成で、WASが起動していない場合のみ）

© 2009 IBM Corporation

wsadminイメージ

- PCやUSS上から、コマンドライン・ベースでWASを管理可能です



© 2009 IBM Corporation

wsadminの起動とコマンド入力の例

- WAS for z/OSを構成した後の<WAS_PROFILE_ROOT>/binに、wsadmin.sh がありますので、これを実行します

```
# ./wsadmin.sh -lang jython
Realm/Cell Name: <default>
Username: wsadmin
Password:
WASX7209I: Connected to process "dmgr" on node W7N00 using SOAP connector;  The
type of process is: DeploymentManager
WASX7031I: For help, enter: "print
wsadmin>
wsadmin>print Help.help()
WASX7028I: The Help object has two
        First, provide general help information for the the objects
                : (ヘルプが表示される)
wsadmin>print AdminApp.list()
DefaultApplication
wsadmin>
```

デフォルトはjaclです

Admin Securityを使用する場合にはユーザー、パスワードの入力が必要です。–user , -passwordオプションで指定することも可能です。

接続タイプや接続先にはwsadmin.properties ファイルの中にある省略時値が使用されます。オプションで指定することも可能です。指定方法は接続タイプにより異なりますので、-h オプションを指定して起動したときに表示されるヘルプをご確認ください。

© 2009 IBM Corporation

wsadminスクリプトを作成するには

- WAS V7.0インフォセンターに、Jythonの文法の簡単な説明があります
  Network Deployment (z/OS) バージョン 7.0 > アプリケーション・サービス提供環境のスクリプト記述 (wsadmin) > スクリプトの入門 > Jython
- インフォセンターには、スクリプトを使用したアプリケーションのデプロイや各種設定の変更についての例が多数あります
  Network Deployment (z/OS) バージョン 7.0 >アプリケーション・サービス提供環境のスクリプト記述 (wsadmin) 以下
- wsadminを直接起動し、インタラクティブに操作して動作を確認することができますので、後で入力したコマンドをテキストファイルに保存してスクリプトとすることができます
- wsadminを起動した後、プロンプトでヘルプを表示させることができます
- 設定により、管理コンソールでの操作に対応するJythonスクリプトを出力させることが可能です

© 2009 IBM Corporation

例1：アプリケーションのデプロイ

```
# ./wsadmin.sh -lang jython
 :
wsadmin>print AdminApp.list()
DefaultApplication
wsadmin>AdminApp.install('/tmp/SampleAppl.ear','[-node W7N01 -server W7S01]')
ADMA5016I: Installation of SampleAppl started.
 :
ADMA5013I: Application SampleAppl installed successfully.
''
wsadmin>AdminConfig.save()
''
wsadmin>print AdminApp.list()
DefaultApplication
SampleAppl
wsadmin>Sync1 = AdminControl.completeObjectName('type=NodeSync,node=W7N01,*')
wsadmin>AdminControl.invoke(Sync1, 'sync')
'true'
wsadmin>print AdminApp.isAppReady('SampleAppl')
ADMA5071I: Distribution status check started for application
WebSphere:cell=W7ND1,node=W7N01,distribution=true,expansion=no
ADMA5011I: The cleanup of the temp directory for application S
ADMA5072I: Distribution status check completed for application SampleAppl.
true
wsadmin>
```

デプロイされているアプリケーションを表示します

SampleAppl.earをデプロイします

変更を保存します

SampleApplが増えているのを確認できます

ノードを同期します

SampleApplが正常にインストールされたことを確認します

※この状態では、まだアプリケーションは始動していません

© 2009 IBM Corporation

例2:アプリケーションの始動

```
wsadmin>appManager = AdminControl.queryNames('type=ApplicationManager,process=W7S01,*')
```

AdminControlオブジェクトのqueryNamesメソッドで、アプリケーション・マネージャーMBeanを取得します

```
wsadmin>print appManager
WebSphere:name=ApplicationManager,process=W7S01,platform=dynamicproxy,node=W7N01,version=7
.0.0.3,type=ApplicationManager,mbeanIdentifier=ApplicationManager,cell=W7ND1,spec=1.0


wsadmin>AdminControl.invoke(appManager,'startApplication','SampleAppl')
''
wsadmin>
```

ApplicationManagerのstartApplicationメソッドを使って、アプリケーション SampleApplを始動します

ApplicationManager MBeanは、アプリの始動および停止機能を持つMBeanで、startApplication、stopApplicationという2つのメソッドを持っています

© 2009 IBM Corporation

例3:アプリケーションの稼働状況の確認

```
wsadmin>print AdminControl.completeObjectName('type=Application,name=SampleAppl,*')
WebSphere:name=SampleAppl,process=W7S01,platform=dynamicproxy,node=W7N01,J2EEName=SampleAppl,Ser
ver=W7S01,version=7.0.0.3,type=Application,mbeanIdentifier=cells/W7ND1/applications/SampleAppl.e
ar/deployments/SampleAppl/deployment.xml#ApplicationDeployment_1239778552701,cell=W7ND1,spec=1.0
```

アプリケーションが稼働しているときは、SampleApplを管理しているMBeanを取得することができます

```
wsadmin>AdminControl.invoke(appManager, 'stopApplication', 'SampleAppl')
''
```

SampleApplを停止します

```
wsadmin>print AdminControl.completeObjectName('type=Application,name=SampleAppl,*')

wsadmin>
```

停止しているときは、そのSampleApplを管理しているMBeanも削除されます

Application MBeanは、個々のアプリが稼働している間のみ活動化されているMBeanで、そのアプリの持つモジュール名等を取得することができます

© 2009 IBM Corporation

スクリプト・ファイルの実行 (1)

- 例1で入力したコマンドをファイルに記述し、実行させることができます

script.py

```
AdminApp.install('/tmp/SampleAppl.ear','[-node W7N01 -server W7S01]')
AdminConfig.save()
Sync1 = AdminControl.completeObjectName('type=NodeSync,node=W7N01,*')
AdminControl.invoke(Sync1, 'sync')
```

- スクリプト・ファイルは、-f オプションで指定します

```
# ./wsadmin.sh -lang jython -f /tmp/script.py -user wsadmin -password xxxxxxx
WASX7209I: Connected to process "dmgr" on node W7N00 using SOAP connector;
The type of process is: DeploymentManager
ADMA5016I: Installation of SampleAppl started.
:
```

※スクリプト・ファイルのデフォルトのエンコーディングはASCIIです
EBCDICのファイルを使用したい場合は、wsadmin.shに以下のオプションを付けてください
-javaoption -Dscript.encoding=Cp1047

© 2009 IBM Corporation

スクリプト・ファイルの実行 (2)

- 例3のコマンドを利用して、アプリケーションが稼働しているかどうかを判定するロジックをスクリプト言語で記述している例です
- ここでは、SampleAppl管理MBeanを変数appに代入し、appが空であればSampleApplが稼働していない、と判定しています

script2.py

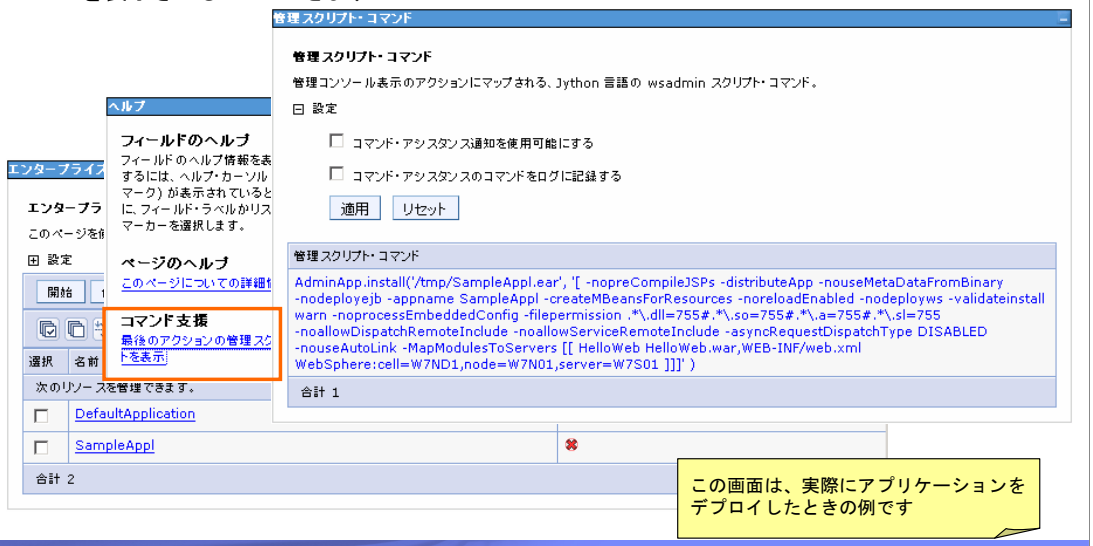
```
app=AdminControl.completeObjectName('type=Application,name=SampleAppl,*')
if app == '' :
        print 'SampleAppl is not started'
else :
        print 'SampleAppl is started'
```

```
# ./wsadmin.sh -lang jython -f /tmp/script2.py -user wsadmin -password xxxxxxx
WASX7209I: Connected to process "dmgr" on node W7N00 using SOAP connector;
The type of process is: DeploymentManager
SampleAppl is started
```

© 2009 IBM Corporation

wsadminスクリプト表示

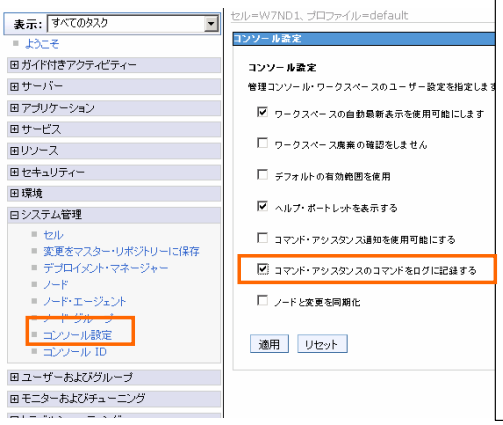
- 管理コンソール右側ヘルプの「コマンド支援」を使用すると、操作に対応したJythonコマンドを表示させることができます



© 2009 IBM Corporation

wsadminスクリプト出力

- 「コマンド・アシスタンスのコマンドをログに記録する」にチェックすると、
  commandAssistanceJythonCommands_（ユーザー名）.log
  というファイルにJythonコマンドが出力されます




```
# [4/15/09 8:39:18:322 GMT] ApplicationDeployment
AdminApp.list()

# [4/15/09 8:39:50:315 GMT] ??????????????????
> ??????????????????
AdminApp.install('/tmp/SampleAppl.ear', '[  -nopreCompileJSPs -
distributeApp -nouseMetaDataFromBinary -nodeployejb -appname
SampleAppl -createMBeansForResources -noreloadEnabled -
nodeployws -validateinstall warn -noprocessEmbeddedConfig -
filepermission .*¥.dll=755#.*¥.so=755#.*¥.a=755#.*¥.sl=755 -
noallowDispatchRemoteInclude -noallowServiceRemoteInclude -
asyncRequestDispatchType DISABLED -nouseAutoLink -
MapModulesToServers [[ HelloWeb HelloWeb.war,WEB-INF/web.xml
WebSphere:cell=W7ND1,node=W7N01,server=W7S01 ]]]' )

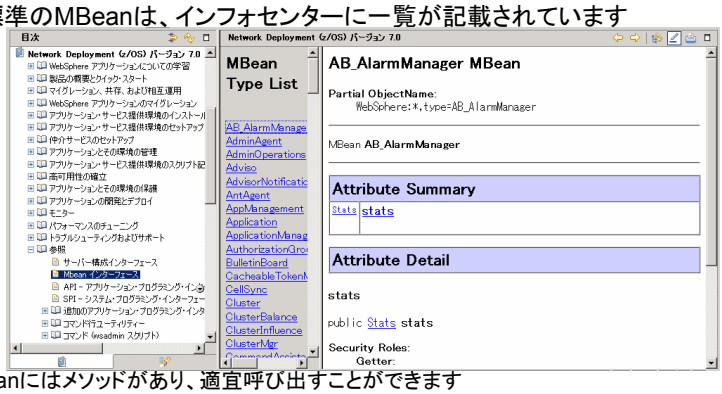
# [4/15/09 8:39:55:855 GMT] ??????????????????
> ??????????????????
AdminConfig.save()
```

※ブラウザの言語設定が「日本語」にした場合、コメントは?になりますが、動作上問題はありません。

© 2009 IBM Corporation

MBeanとは？

- Managed Beanの略です
- WASの実行環境で、大量に動いています
- それぞれのMBeanは、ある特定のリソースを管理するために存在します
- WAS標準のMBeanは、インフォセンターに一覧が記載されています



  - MBeanにはメソッドがあり、適宜呼び出すことができます
  - MBeanを作成し、登録することもできます

© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.javadoc.doc/web/mbeanDocs/index.html

MBeanの使い方

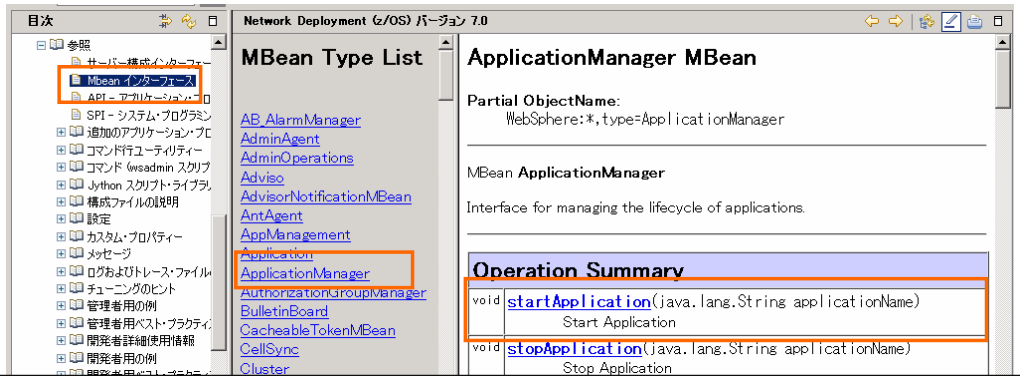
- MBeanは、必ず以下の6つの属性を持ちます
  - type ... 型を表します。インフォセンターに記載されています
  - name ... インスタンスを表します
  - cell, node, process ... 稼働環境を表します
  - mbeanInstance ... インスタンスと構成データの関連付けを表します
- AdminControlを使用して、MBeanを見つけ出し、そのMBeanのメソッドを呼び出すか、あるいはMBeanの属性のsetter/getterを呼び出します
- なお、type=Applicationのように、同じ型で複数のインスタンスが存在する場合は、nameも指定して探す必要があります
- 指定しない部分は、アスタリスク('*') で省略します

© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tjmx_programming.html

MBeanのメソッド呼び出し

- ApplicationManager MBeanのstartApplicationメソッドを呼び出して、アプリケーションを始動します（例2と同じです）

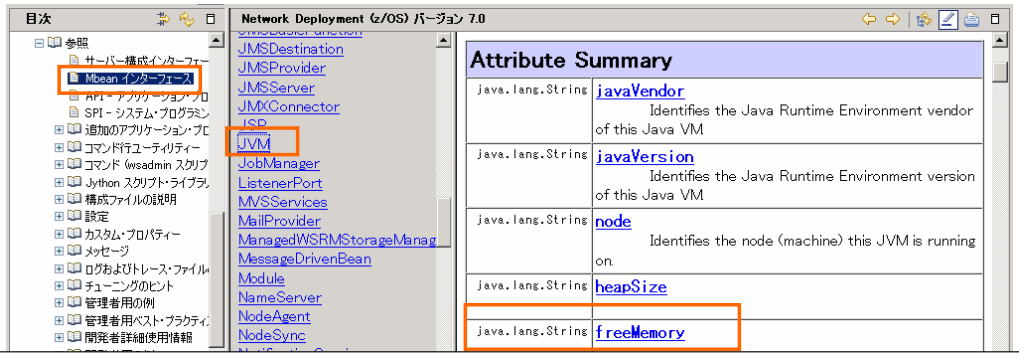



```
wsadmin>appManager = AdminControl.queryNames('type=ApplicationManager,process=W7S01,*')
wsadmin>print appManager
WebSphere:name=ApplicationManager,process=W7S01,platform=dynamicproxy,node=W7N01,version=7.0.0.3
,type=ApplicationManager,mbeanIdentifier=ApplicationManager,cell=W7ND1,spec=1.0
wsadmin>AdminControl.invoke(appManager, 'startApplication', 'SampleAppl')
''
```

© 2009 IBM Corporation

MBeanの属性値の取得

- JVM MBeanのfreeMemory属性の値を取得する例です




```
wsadmin>jvm = AdminControl.queryNames('type=JVM,process=W7S01,*')
wsadmin>print jvm
WebSphere:name=JVM,process=W7S01,platform=proxy,node=W7N01,j2eeType=JVM,J2EEServer=W7S01,version
=7.0.0.3,type=JVM,mbeanIdentifier=JVM,cell=W7ND1,spec=1.0
wsadmin>AdminControl.invoke(jvm, 'getFreeMemory')
'Controller: 12283168;¥nServant 00000150000000001: 36119200;'
```

CRとSR、それぞれの属性が表示されています

© 2009 IBM Corporation

JavaプログラムからのMBeanの呼び出し

- Javaプログラムから直接MBeanを呼び出すこともできます
- このソースをコンパイルするためには、以下のimport文が必要です
  import com.ibm.websphere.management.*;

```
java.util.Properties connectProps = new java.util.Properties();
connectProps.setProperty(AdminClient.CONNECTOR_TYPE, AdminClient.CONNECTOR_TYPE_SOAP);
connectProps.setProperty(AdminClient.CONNECTOR_HOST, "WAS稼働ホスト名");
connectProps.setProperty(AdminClient.CONNECTOR_PORT, "SOAPポート番号");

AdminClient adminClient = AdminClientFactory.createAdminClient(connectProps);

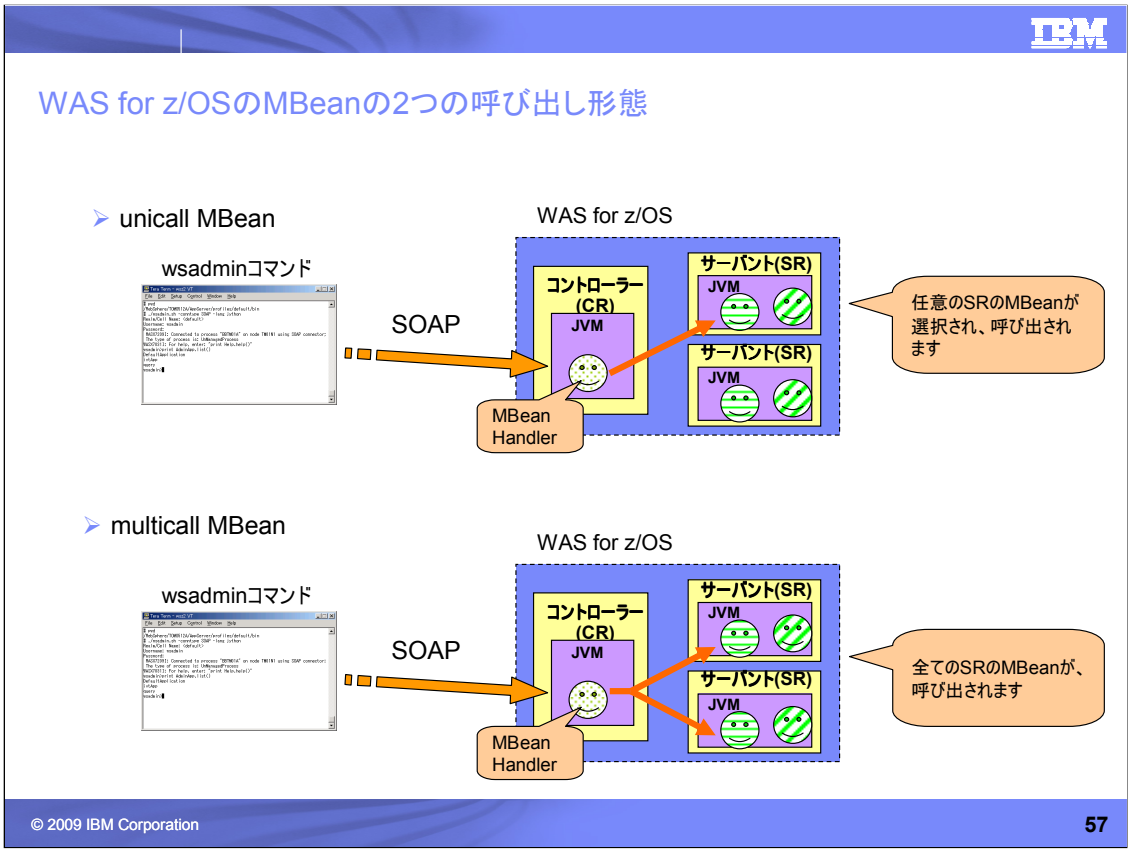
String query = "WebSphere:type=JVM,*" ;
javax.management.ObjectName queryName = new ObjectName(query) ;
java.util.Set s = adminClient.queryNames(queryName, null) ;

javax.management.ObjectName mbean = (ObjectName)s.iterator().next() ;
String ret = (String)adminClient.invoke(mbean, "getFreeMemory", new Object[0], new String[0]) ;
System.out.println("result = " + ret) ;
```

ホスト名、SOAPポートを指定します

このプログラムは、前ページのwsadminコマンドと同様の動作をする例です

© 2009 IBM Corporation



multicallは、ひとつのサーバーで複数のSRを稼動させることの出来るz/OS版特有の機能です。

Unicallの場合、Mbeanが実行されるSRの指定は出来ません。

ホット・サーバントで実行するか、ランダムに選択したサーバントで実行するかを指定するパラメーターは存在しますが、これに依存した設計は避けてください。

（ランダムにして、各SRで均等に実行されることを期待するなど）

com.ibm.websphere.management.jmx.random

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/xrun_jvm.html

unicall / multicall呼び出しについて

- MBeanのメソッドは、unicall または multicallのどちらかの呼び出し方法で定義されています
  （MBeanを作成するときに、MBeanディスクリプターXMLに定義します）
- 分散系WAS用に作成されたMBeanで、unicall / multicallの定義の無いMBeanメソッドは、unicall呼び出しになります

- unicallの場合は、任意のSRのMBeanがひとつ選択され、呼び出されます
- unicall メソッドの戻り値は、そのままクライアントに返されます

- multicallの場合は、全てのSRのMBeanのメソッドが呼び出されます
- multicall メソッドの戻り値をクライアントに返すためには、全てのMBeanから返される値を集約するためのクラスを、コントローラーで動かす必要があります（AggregationHandlerクラスの作成）

© 2009 IBM Corporation

（参考）MBeanディスクリプターXMLの例

- MBeanは、ロジックの実装クラスと、そのインターフェースを定義するMBeanディスクリプターXMLから構成されます
- このMBeanディスクリプターでは、SampleMBeanというtypeのMBeanについて、getter/setterを持つ属性 "sampleId" と、メソッド "hello" が定義されています

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MBean SYSTEM "MbeanDescriptor.dtd">

<MBean type="SampleMBean" version="5.0">
```

proxyInvokeType="multicall"と指定するとmulticall呼び出しになります（デフォルトはunicall）

属性の定義

```
<attribute name="sampleId" type="java.lang.String" proxyInvokeType="multicall"
          getMethod="getSampleId" setMethod="setSampleId"/>
```

メソッドの定義

```
<operation name="hello" role="operation" type="java.lang.String" impact="Action"
          targetObjectType="objectReference" proxyInvokeType="multicall">
          <signature>
          <parameter name="name" description="name string" type="java.lang.String"/>
          </signature>
</operation>

</MBean>
```

このメソッドは、引数として一つのStringをとり、戻り値もStringとして定義されています

© 2009 IBM Corporation

補足1．wsadmin(Jython) Tips

© 2009 IBM Corporation

Jython Tips 1

- コマンドの出力が大量にあって、見えにくい！場合
  - コマンドの出力をファイルに出力することができます
  - 例えば、AdminConfig.help() を実行すると、数十行の出力がありますが、これを /tmp/output.txt に出力する例は以下のようになります

```
temp_file = open('/tmp/output.txt", "w")

temp_file.write( AdminConfig.help() )

temp_file.close() ;
```

/tmp/output.txtを開く

writeメソッドで AdminConfig.help()の内容をファイルに出力

© 2009 IBM Corporation

Jython Tips 2

- 別のスクリプト・ファイルに関数を定義し、使いたい！場合
  - /usr/home/jython/lib/lib_file.py に、関数 func1 があるとします
  - プロパティ・ファイル sample_wsadmin.properties に次の一行を定義します
    python.path=/usr/home/jython/lib
  - そして、wsadminを起動するときに、-p オプションでプロパティ・ファイルを指定します
    wsadmin.sh –conntype SOAP –lang jython –p sample_wsadmin.properties
  - Jythonの中で、次のようにimport 文を実行すると、
    import lib_file
    インポートしたファイル名(lib_file)をパッケージ名として、関数func1を呼び出すことが可能です
    lib_file.func1()

```
#wsadmin.sh -conntype SOAP -lang jython
          -p sample_wsadmin.properties
wsadmin>import lib_file
wsadmin>lib_file.func1()
Hello, World!
```

sample_wsadmin.properties

```
python.path=/usr/home/jython/lib
```

/usr/home/jython/lib/lib_file.py

```
def func1() :
    print "Hello, World!"
```

© 2009 IBM Corporation

補足2. MBeanの作成

© 2009 IBM Corporation

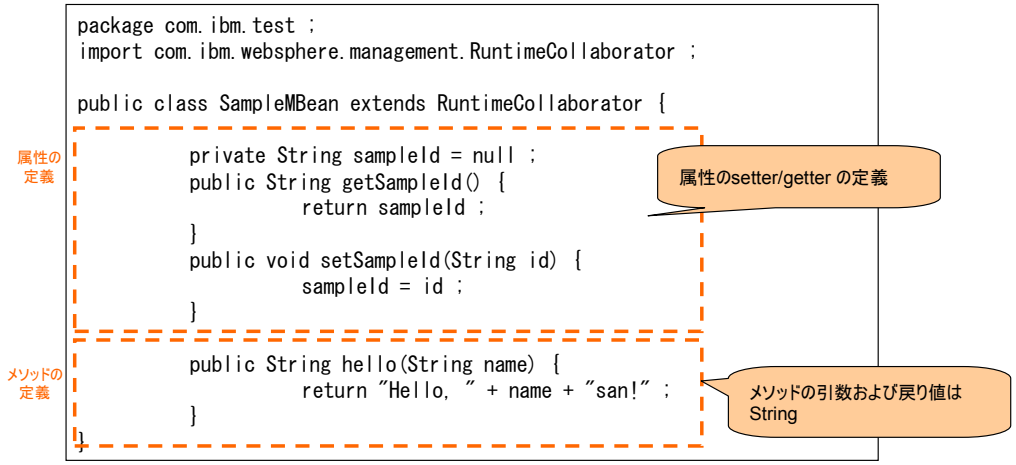
MBeanの作り方

1. MBeanを作るには、ロジックを実装したクラスと、インターフェースを定義したMBeanディスクリプターXMLを作成する必要があります
   - 実装クラスは、com.ibm.websphere.management.RuntimeCollaboratorを継承して作成します
   - MBeanディスクリプターXMLのDTDは MbeanDescriptor.dtd で、propertiesディレクトリ下にあります
2. 作成したクラス・XMLをまとめてJarファイルとし、管理コンソールの アプリケーション・サーバー > サーバー名 > 管理 > 管理サービス > 拡張MBeanプロバイダー でJarファイルへのパス登録します
3. 登録したMBeanプロバイダーの下の ExtensionMBean で、MBeanディスクリプターXMLへの相対パスと、MBeanのtypeを定義します
4. プログラム（サーブレットやEJB)で、登録したMBeanをactivateします

© 2009 IBM Corporation

MBeanの実装クラスの例

- このMBeanは、sampleId という属性と、helloというメソッドを持ちます



© 2009 IBM Corporation

MBeanディスクリプターXMLの例 (unicall)

- MBeanの実装に対応して、属性 sampleId と、メソッドhello を定義します
- このXMLでは、unicall型のメソッドを定義しています。つまり、任意の1つのSRが選択され、MBeanの処理が実行されます

SampleMBean.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MBean SYSTEM "MbeanDescriptor.dtd">

<MBean type="SampleMBean" version="5.0">

属性の定義
<attribute name="sampleId" type="java.lang.String"
           getMethod="getSampleId" setMethod="setSampleId"/>

メソッドの定義
<operation name="hello" role="operation" type="java.lang.String" impact="Action"
           targetObjectType="objectReference">
           <signature>
           <parameter name="name" description="name string" type="java.lang.String"/>
           </signature>
</operation>

</MBean>
```

属性の定義
setterが不要な場合はsetMethodは必要ありません

メソッドの定義
引数、戻り値の型を定義します

© 2009 IBM Corporation

MBeanの登録の例

- MBean実装クラスとMBeanディスクリプターXMLをJarファイルとしてまとめて、そのパスを拡張MBeanプロバイダーに登録します
- 拡張MBeanプロバイダー下のExtensionMBeanで、MBeanディスクリプターXMLへの相対パスと、MBeanのtypeを登録します



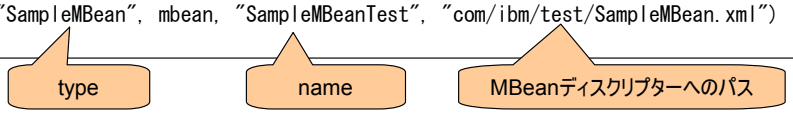
© 2009 IBM Corporation

MBeanのactivateの例

- サーブレットまたはEJBでそのMBeanをactivateします
- MBeanFactoryのactivateMBeanメソッドを使用してMBeanを使用可能にします
- activateMBeanの引数は、左から順に
  - MBean type
  - MBeanインスタンス
  - MBean name
  - MBeanディスクリプターXML名 (クラスパスからの相対パス)

```
import com.ibm.websphere.management.MBeanFactory ;
```

```
MBeanFactory mbf = AdminServiceFactory.getMBeanFactory() ;
SampleMBean mbean = new SampleMBean() ;
mbf.activateMBean("SampleMBean", mbean, "SampleMBeanTest", "com/ibm/test/SampleMBean.xml") ;
```



© 2009 IBM Corporation

MBeanディスクリプターXMLの例 (multicall)

- 属性やメソッドの定義に proxyInvokeType="multicall" と指定すると、multicall型の呼び出しになります
- multicall型の場合、AggregationHandlerを定義しなければクライアントに結果を戻せません

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MBean SYSTEM "MbeanDescriptor.dtd">

<MBean type="SampleMBean" version="5.0"
          aggregationHandlerClass="com.ibm.test.SampleAggregationHandler">

<attribute name="sampleId" type="java.lang.String" proxyInvokeType="multicall"
          getMethod="getSampleId" setMethod="setSampleId"/>

<operation name="hello" role="operation" type="java.lang.String" impact="Action"
          targetObjectType="objectReference" proxyInvokeType="multicall">
          <signature>
          <parameter name="name" description="name s      g" type="java.lang.String"/>
          </signature>
</operation>

</MBean>
```

AggregationHandlerの指定

指定しない場合のデフォルトはunicall

© 2009 IBM Corporation

AggregationHandlerの例

- multicall呼び出しのメソッド名で判断して、適切な処理を行う必要があります
- MBeanディスクリプターで定義された戻り値の型に合わせる必要があります

```
package com.ibm.test ;
import com.ibm.websphere.management.dynamicproxy.* ;

public class SampleAggregationHandler implements AggregationHandler {
    public Object aggregateResults(String methodName, Object[] params, String[] signatures,
        ServantMBeanResult[] results, StateObject stateObject) {
        if ("hello".equals(methodName)) {
            StringBuffer sbuf = new StringBuffer() ;
            for(int i = 0 ; i < results.length ; i++) {
                if (results[i] != null) {
                    if (i > 0) sbuf.append(", ") ;
                    sbuf.append(i).append(":").append(results[i].getStoken())
                      .append(":").append(results[i].getResult()) ;
                }
            }
            return sbuf.toString() ;
        }
        return "unknown method:" + methodName ;
    }
}
```

メソッド hello の場合のロジック

© 2009 IBM Corporation

補足3. リソース定義の例

© 2009 IBM Corporation

JDBC

- JavaからRDBにアクセスするためのAPIです
- DB2の提供するドライバーを使用することによって、DB2に接続します
- JDBCドライバーには、実装形態によってタイプ1からタイプ4まであります

- 管理コンソールで、「JDBCプロバイダー」を定義することによって、JDBCドライバーを指定します
- 管理コンソールで、「JDBCプロバイダー」に対して「データ・ソース」を定義します。アプリケーションに対するデータベースへの接続は、データソース・オブジェクトを介して行います

© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/welc6tech_dat.html

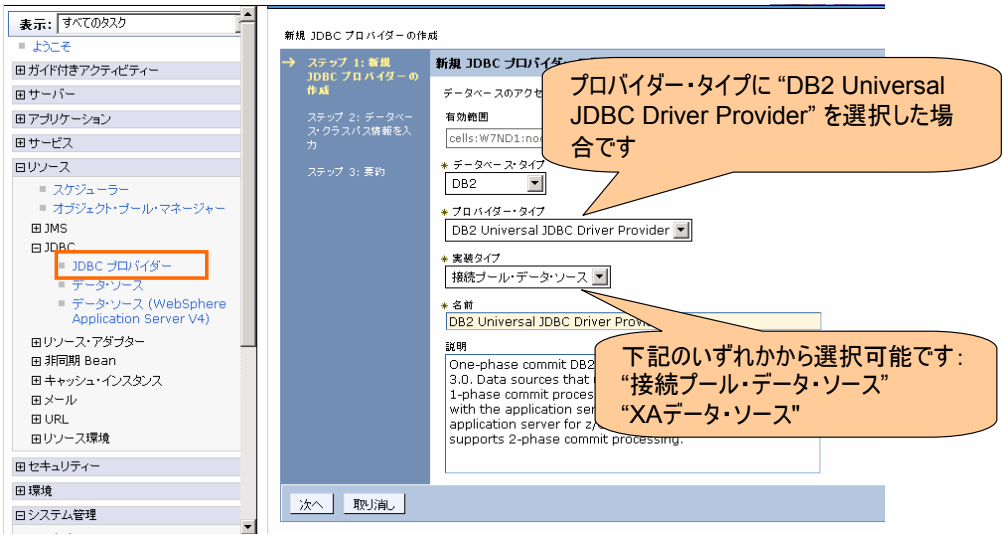
JDBCプロバイダーの種類

- DB2 for z/OSへの接続には、”DB2 Using IBM JCC Driver”、”DB2 Universal JDBC Driver Provider”を使用します
  - DB2 Using IBM JCC Driver … JDBC4.0対応のドライバー(db2jcc4.jar)を使用します
  - DB2 Universal JDBC Driver Provider … JDBC3.0対応のドライバー(db2jcc.jar)を使用します
- “接続プール・データ・ソース”は、Type2およびType4をサポートします
  - ローカルのDB2に対してはType2を使用可能で、この場合はトランザクションをRRSで管理します
- “XAデータ・ソース”は、Type4のみサポートします
  - トランザクションをXAで管理します
- DB2 for z/OS Local JDBC Provider (RRS) はWAS V6.1以降サポートされなくなりました

© 2009 IBM Corporation

DB2プロバイダー定義 ステップ1

- 管理コンソールで、リソース > JDBCプロバイダーの画面で定義します

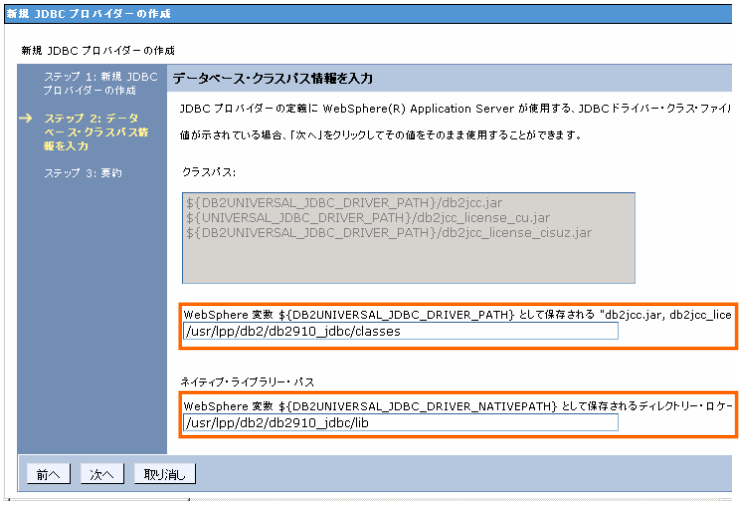


© 2009 IBM Corporation

DB2 Using IBM JCC Driverの場合も手順は同じです。

DB2プロバイダー定義 ステップ2

- HFS上の、DB2のインストール・ディレクトリを指定します



© 2009 IBM Corporation

データ・ソース定義 ステップ1

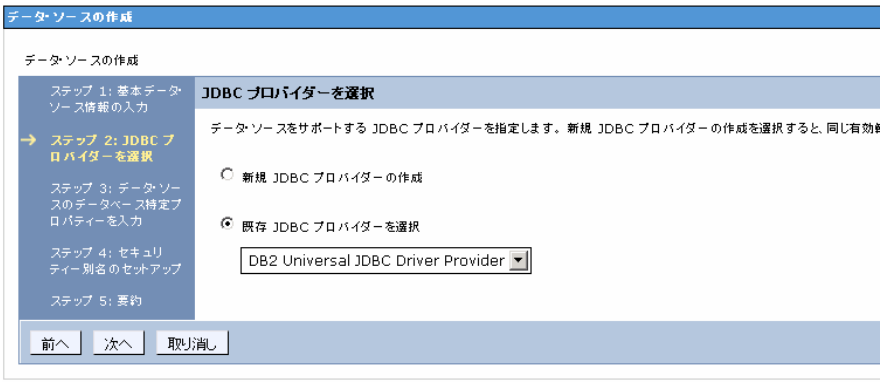
- 管理コンソールで、リソース > データ・ソースの画面で定義します



© 2009 IBM Corporation

データ・ソース定義 ステップ2

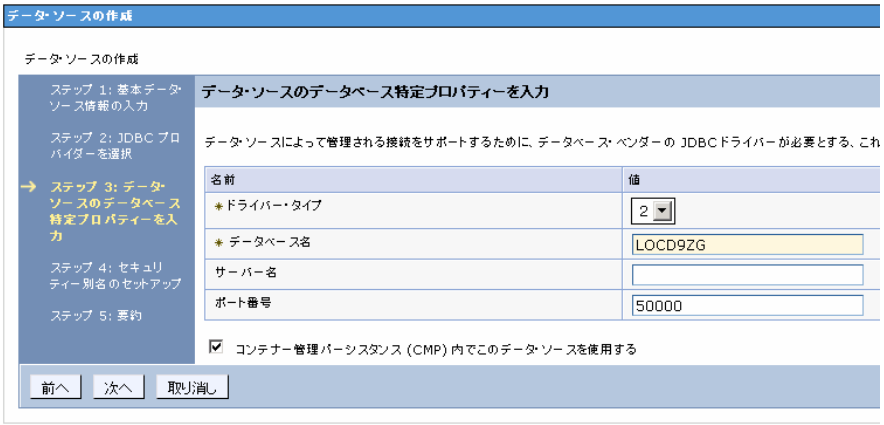
- すでに定義されているJDBCプロバイダーに対してデータ・ソースを定義する場合は、「既存JDBCプロバイダーを選択」を選択します



© 2009 IBM Corporation

データ・ソース定義 ステップ3

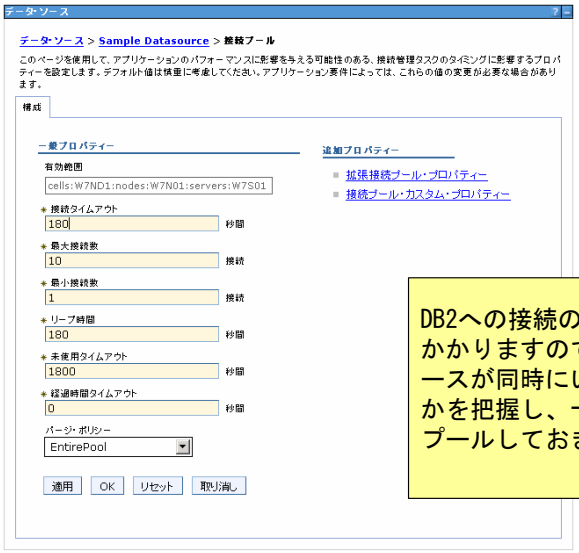
- 接続するDB2の情報を入力します。
  - Type2を選択する場合は、「データベース名」にロケーション名を入力します
  - Type4を選択する場合は、「データベース名」「サーバー名」「ポート番号」を入力します



© 2009 IBM Corporation

データ・ソース定義 ステップ4

- 定義したデータ・ソースの接続プール・プロパティーを適切に設定します



DB2への接続の確立にはコストがかかりますので、このデータソースが同時にいくつ利用されるかを把握し、十分な数の接続をプールしておきましょう

© 2009 IBM Corporation

データ・ソース定義 Type2接続の場合の追加定義

- Type2で接続する場合は、構成プロパティーのdb2.jcc.ssidにDB2サブシステム名またはグループ接続名の設定が必要です
- DB2 Accounting reportの取得を考えている場合は、データソース／コネクション・プロパティーのaccountingInterval、または、構成プロパティーのdb2.jcc.accountingIntervalにCOMMITを設定します
  - これを設定しない場合、アカウンティング・レコードはデータベースへの物理接続の終了時点で作成されます。データ・ソースではコネクションがプールされて再利用されますので、同一の接続で実行された複数のSQLがひとつに集計されることになります。また、プールから接続がパージされるか、SRが停止するまでアカウンティング・レコードが出力されません
  - これを設定すると、アカウンティング・レコードはコミット単位で作成されます。ロールバックされたSQLは別のCOMMITされたSQLにまとめて集計される点、CLASS1のELAPSED TIMEは同一の接続で前にCOMMITされてから次にCOMMITするまでの経過時間となる点に注意してください
- 構成プロパティーの設定方法
  - 管理コンソールのアプリケーション・サーバー > （サーバー名） > プロセス定義 > Servant > Java 仮想マシン > カスタム・プロパティー（SRのJVMカスタムプロパティ）に設定する
  - SRのJVMカスタムプロパティに設定したdb2.jcc.propertiesFile構成プロパティーで指定のファイル、または、DB2JccConfiguration.propertiesファイルの中で定義する
- データソース・プロパティーの設定方法
  - 管理コンソールのデータ・ソース > （データソース名） > カスタム・プロパティーで設定する
- コネクション・プロパティーの設定方法
  - DB2固有インプリメンテーションのデータソースのset（プロパティー名）メソッドを使用する（取得したデータソース・インスタンスをDB2 専用クラスにキャストして使用する）

© 2009 IBM Corporation

構成プロパティーやデータソース・プロパティーに設定できるパラメーターはほかにもいろいろとありますので、次ページに記載のマニュアルをご参照ください。

データ・ソース定義 その他の作業

- 「コンポーネント管理の認証別名」を使用する場合は、セキュリティー > 管理、アプリケーション、およびインフラストラクチャーの保護 > Java認証・承認サービス > J2C認証データ で、ユーザーIDおよびパスワードを定義する必要があります
- JDBCを初めて使用する場合には、JDBC用のパッケージをDB2にバインドしておく必要があります
- その他に実際に必要となるプロパティーや、DB2バインダーの使用方法等は、DB2マニュアル "Application Programming Guide and Reference for Java" (SC18-9842) を参照してください

© 2009 IBM Corporation

データ・ソースとアプリケーションのマッピング

- アプリケーション中で定義されているデータ・ソース参照と、WASで定義したデータソース定義とをマッピングします
- 管理コンソールでの設定
  - 設定箇所：エンタープライズ・アプリケーション > アプリケーション名 > リソース参照



© 2009 IBM Corporation

アプリケーション中で定義されているデータ・ソース参照

1.デプロイメント・ディスクリプターにリソース参照を定義し、データソースをlookupする際にはその参照をJNDI ENC名で指定した場合

2. JavaEE 5.0で@Resourceアノテーションでデータソースを定義した場合

SIBus

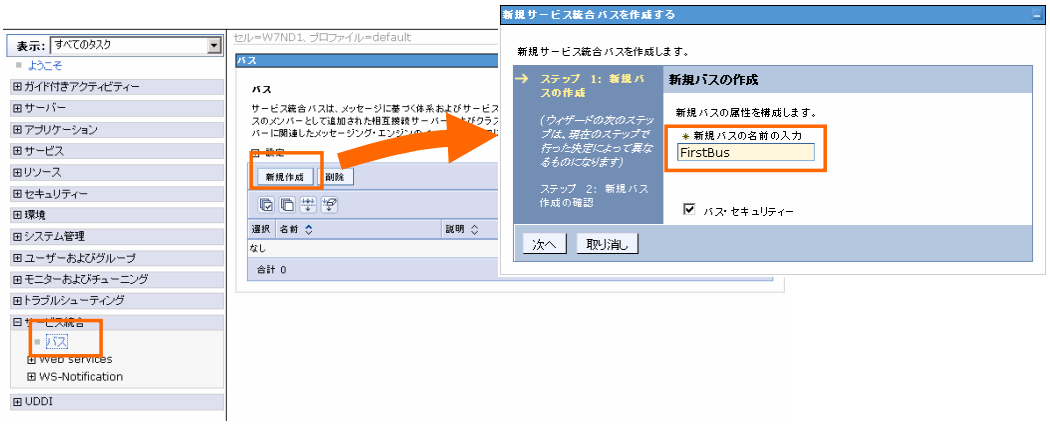
- WASに標準で備わっているメッセージング・エンジンです
- アプリケーション間の非同期通信や、Pub/Subによる通信をサポートします
- アプリケーションから、JMSでアクセスすることが可能です
- WAS for z/OSでSIBusを構成すると、CRA(Control Region Adjunct)と呼ばれるアドレススペースが立ち上がります

- ここでは、JMSアプリケーションからアクセス可能な、SIBusを構成する例を示します

© 2009 IBM Corporation

SIBus：バスの作成

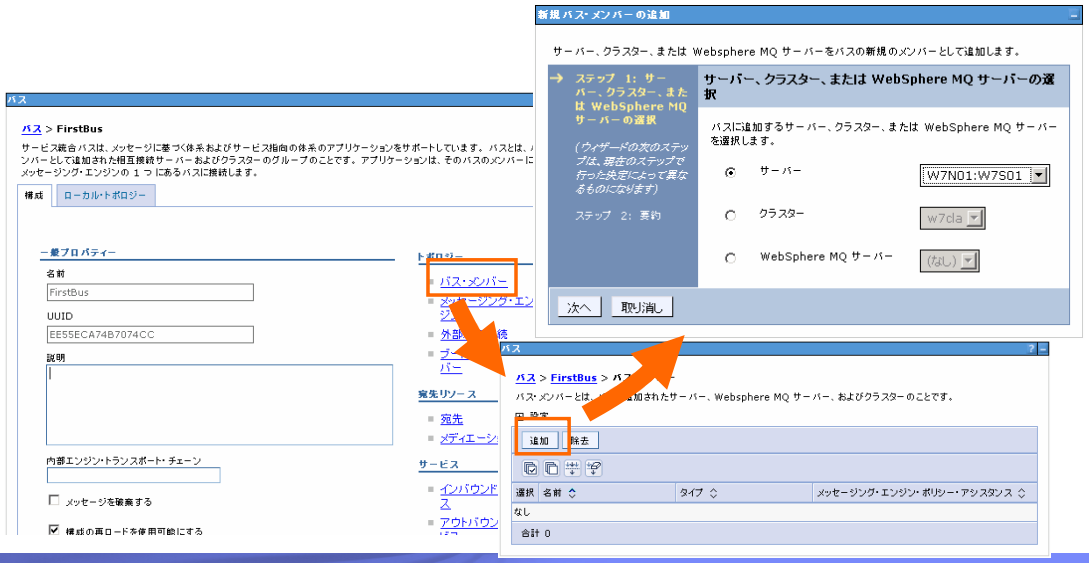
- 管理コンソールで、サービス統合 > バス と進み、[新規作成]を押します
- 新たに作成するSIBusの名前を入力します



© 2009 IBM Corporation

SIBus：バス・メンバーの追加 ステップ1

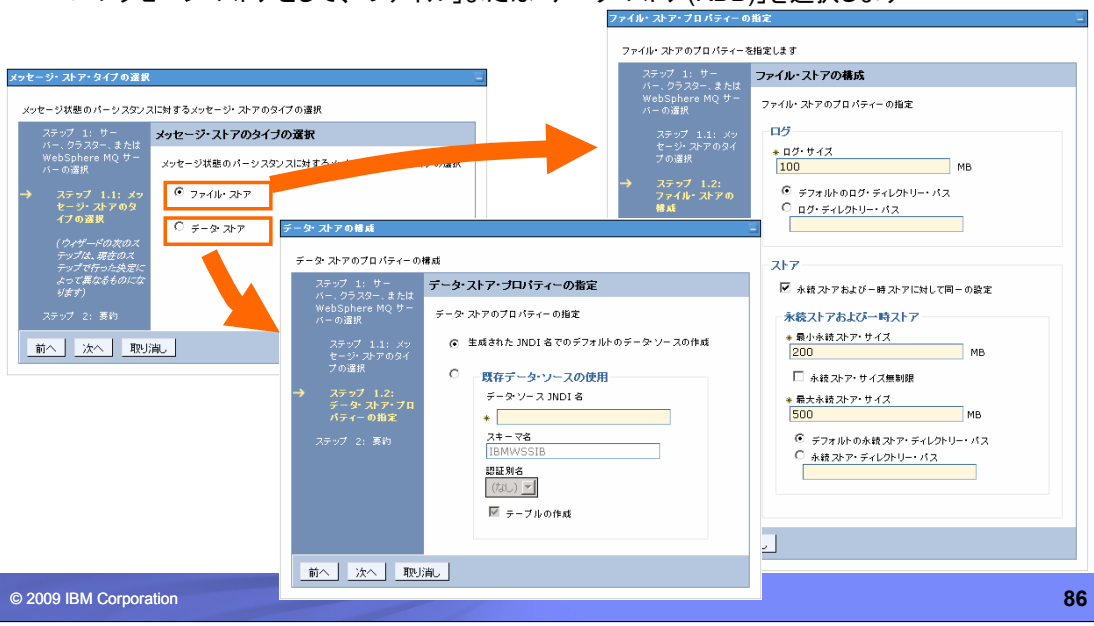
- 作成したSIBusを開き、バス・メンバーを追加します



© 2009 IBM Corporation

SIBus：バス・メンバーの追加 ステップ2

- メッセージ・ストアとして、「ファイル」または「データ・ストア(RDB)」を選択します



© 2009 IBM Corporation

バス・メンバーの追加が完了すると、アプリケーション・サーバー > （サーバー名） > SIB サービスの「サーバー始動時にサービスを使用可能にする」にチェックが入り、CRAが起動するようになります。

SIBus：宛先の追加

- SIBusに宛先（キュー）を追加します
- キューの名前は任意です



© 2009 IBM Corporation

SIBus：キュー接続ファクトリーの定義

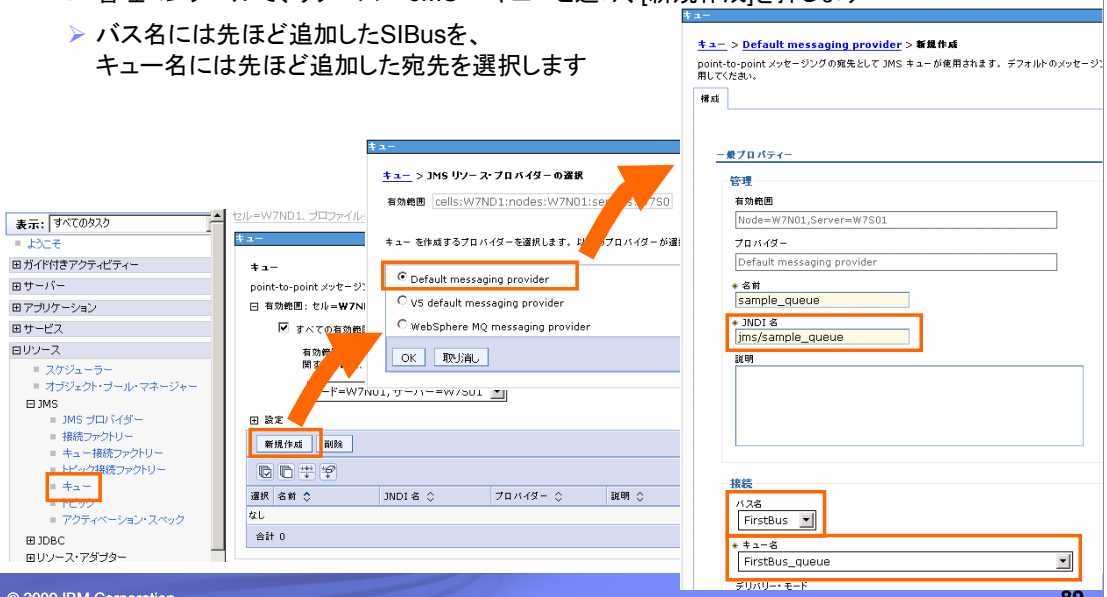
- 管理コンソールで、リソース > JMS > キュー接続ファクトリーと進み、[新規作成]を押します
- バス名には、先ほど追加したSIBusを選択します



© 2009 IBM Corporation

SIBus：キューの定義

- 管理コンソールで、リソース > JMS > キューと進み、[新規作成]を押します
- バス名には先ほど追加したSIBusを、
  キュー名には先ほど追加した宛先を選択します



© 2009 IBM Corporation

MQ

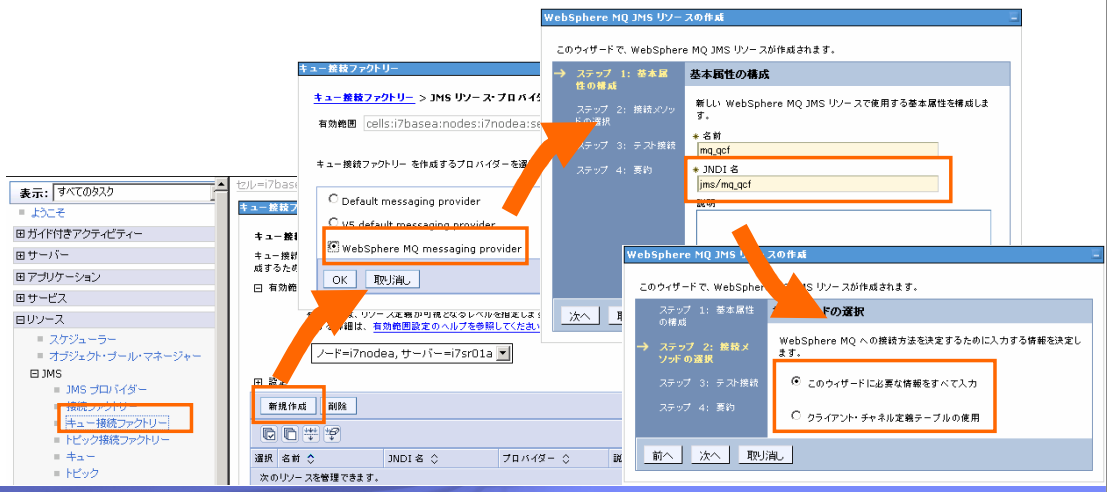
- WAS V7では、WebSphere MQとの接続に使用するWebSphere MQ messaging providerがWASの一部として提供されます
  - これには、WebSphere MQ V7 Resource Adapterを含みます
  - ネイティブ・ライブラリーは含みませんので、バインディング・モードでローカルのキュー・マネージャーと接続する場合は、同じマシン上に導入されたWebSphere MQのネイティブ・ライブラリーを設定する必要があります

© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/welc6tech_msg.html

MQ：キュー接続ファクトリーの定義　ステップ1

- 管理コンソールで、リソース > JMS > キュー接続ファクトリーと進み、[新規作成]を押します
- キュー・マネージャーとの接続情報は、クライアント・チャネル定義テーブルを使用するか、ウィザードで必要な情報をすべて入力するかを選択します

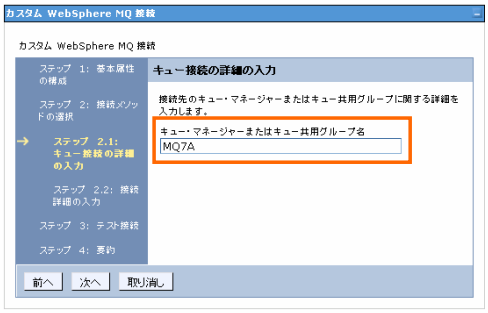


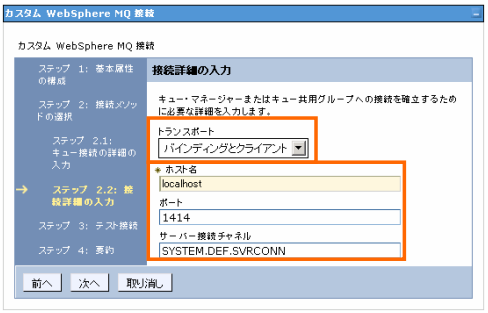
© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tmj_adm30.html

MQ：キュー接続ファクトリーの定義　ステップ2

- 必要な情報をすべて入力を選択した場合は、
  - キュー・マネージャー名（またはキュー共用グループ名）の入力
  - トランスポートの選択
  - トランスポート・タイプにクライアントを使用する場合はホスト名、ポート、サーバー接続チャネルを入力します

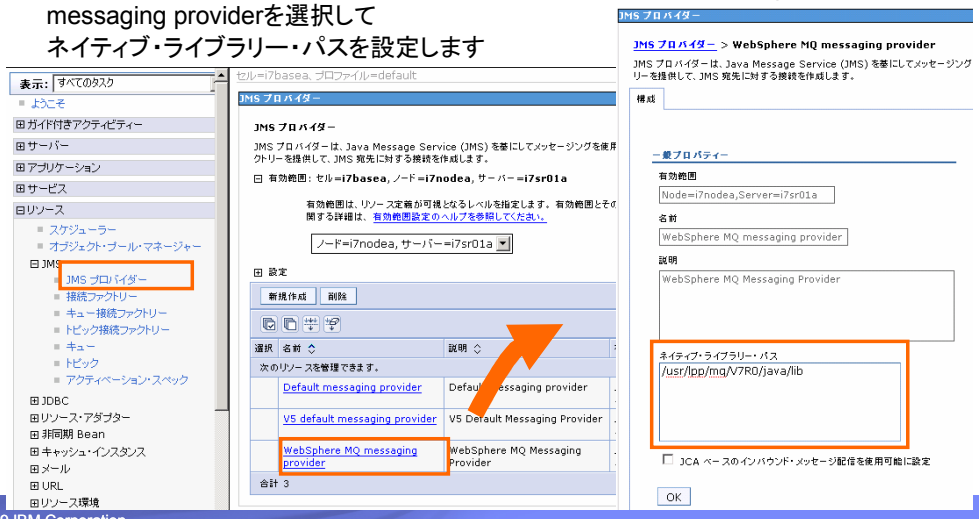




© 2009 IBM Corporation

MQ：JMSプロバイダーにネイティブ・ライブラリ・パスを設定

- トランスポート・タイプにバインディングを使用する場合は、JMSプロバイダーにネイティブ・パス情報を設定します
- リソース > JMS > JMSプロバイダーと進み、適切な有効範囲のWebSphere MQ messaging providerを選択して
  ネイティブ・ライブラリー・パスを設定します

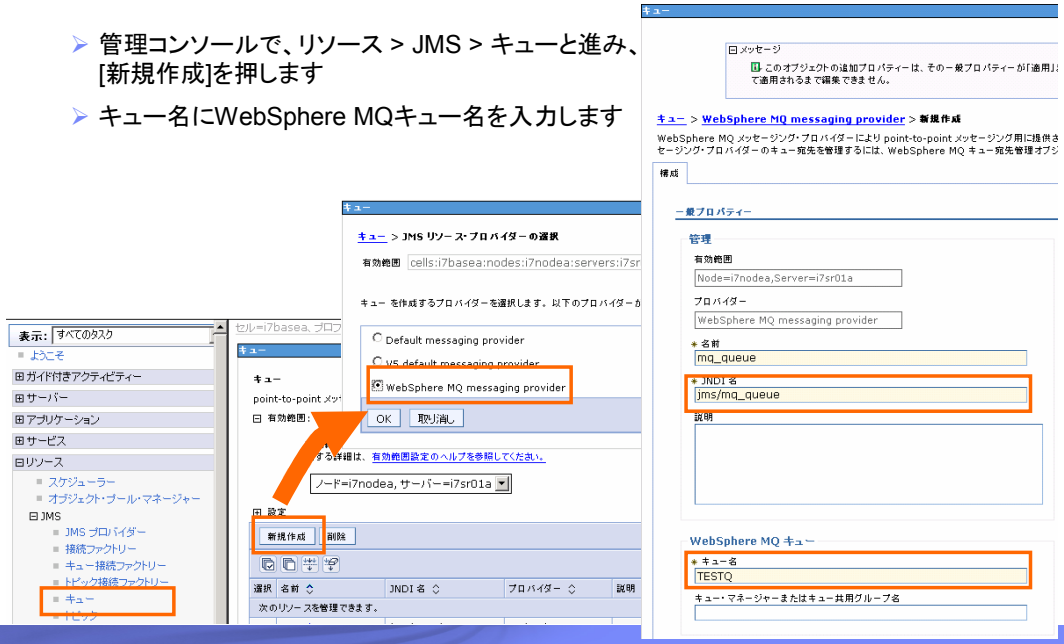


© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tmj_adm33.html

MQ:キューの定義

- 管理コンソールで、リソース > JMS > キューと進み、[新規作成]を押します
- キュー名にWebSphere MQキュー名を入力します

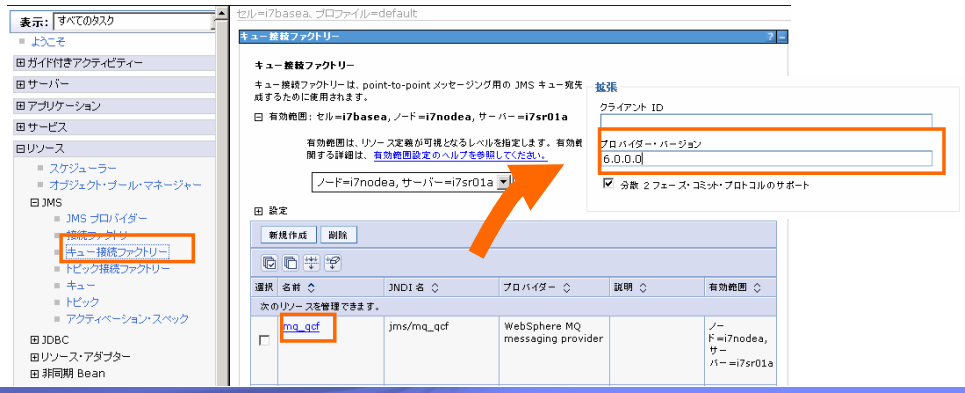


© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tmj_adm27.html

MQ：プロバイダー・バージョンの設定

- MQ V6 for z/OS、およびMQ V7 for z/OSと接続する場合は、WebSphere MQメッセージング・プロバイダー移行モードを使用します
- 管理コンソールで、リソース > JMS > キュー接続ファクトリと進み、WebSphere MQ messaging providerを使用するキュー接続ファクトリーを選択して、プロバイダー・バージョンに”6.0.0.0”を設定します

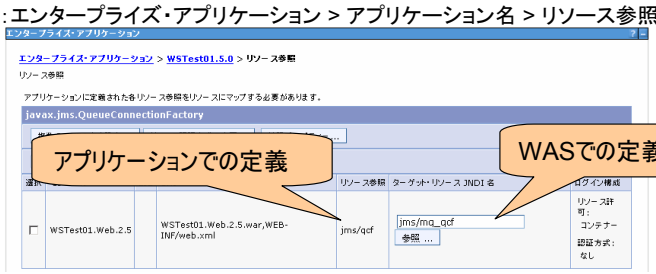


© 2009 IBM Corporation

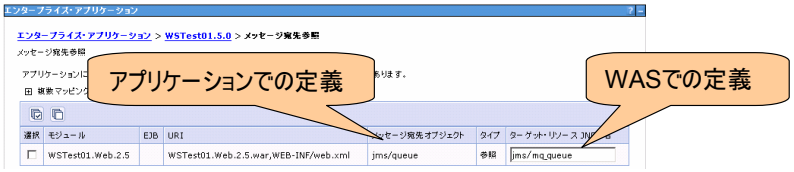
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tmj_adm630.html

JMSリソースとアプリケーションのマッピング

- アプリケーション中で宣言されているJMSリソースと、WASで定義したJMSのリソース定義とをマッピングします
- 管理コンソールでの設定
  - 設定箇所：エンタープライズ・アプリケーション > アプリケーション名 > リソース参照



  - 設定箇所：エンタープライズ・アプリケーション > アプリケーション名 > メッセージ宛先参照



© 2009 IBM Corporation

アプリケーション中で宣言されているJMSリソース

1.デプロイメント・ディスクリプターにリソース参照を定義し、JMSリソースをlookupする際にはその参照をJNDI ENC名で指定した場合

2. JavaEE 5.0で@Resourceアノテーションで JMSリソースを定義した場合

アクティベーション・スペック

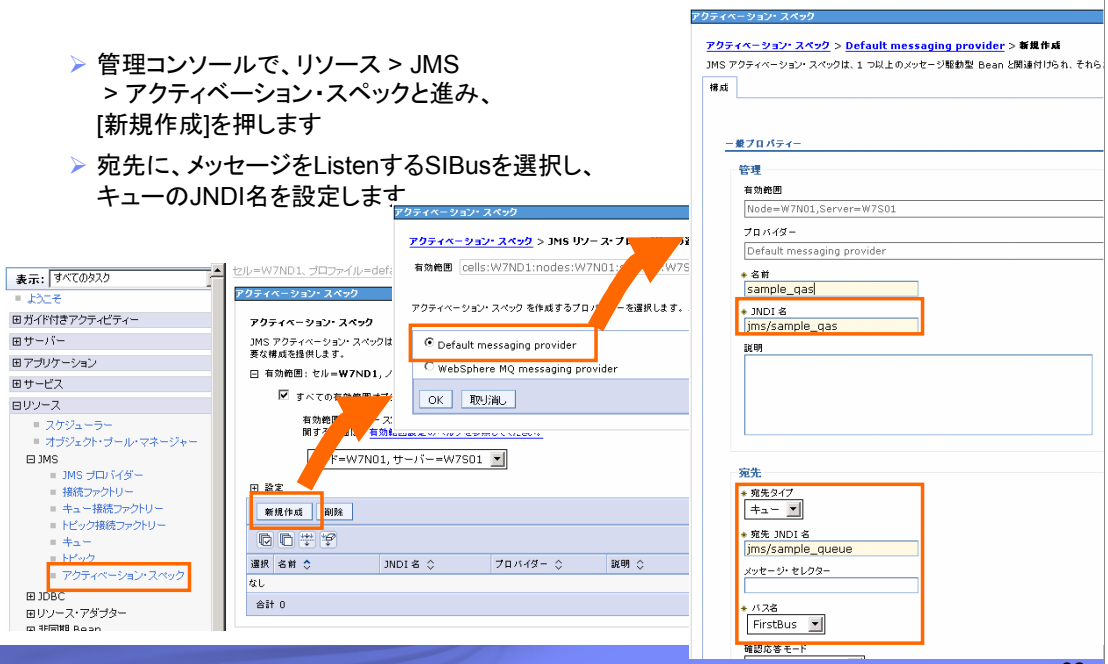
- WAS V7.0より、JMSリスナー・ポートを使用したMDBの駆動は非推奨となりました
  今後はリスナー・ポートの変わりに、アクティベーション・スペックを使用するようにMDBのバインディングを構成してください

© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tm_admin.html

アクティベーション・スペックの作成(SIBusの場合)

- 管理コンソールで、リソース > JMS > アクティベーション・スペックと進み、[新規作成]を押します
- 宛先に、メッセージをListenするSIBusを選択し、キューのJNDI名を設定します



© 2009 IBM Corporation

アクティベーション・スペックの作成（MQの場合） ステップ1

- 管理コンソールで、リソース > JMS > アクティベーション・スペックと進み、[新規作成]を押します
- 宛先JNDI名に、メッセージをListenするキューのJNDI名を設定します
- キューマネージャーとの接続情報の設定は、「 MQ：キュー接続ファクトリーの定義　ステップ2 」と同様に設定してください

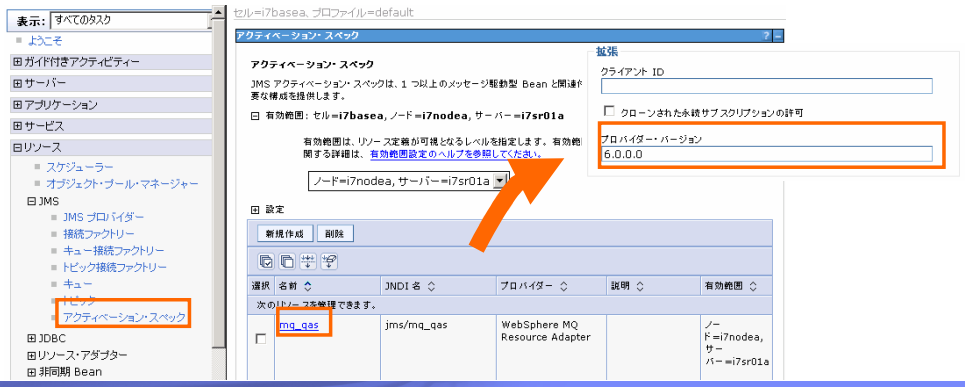


© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tmj_adm20.html

アクティベーション・スペックの作成（MQの場合） ステップ2

- MQ V6 for z/OS、およびMQ V7 for z/OSと接続する場合は、WebSphere MQメッセージング・プロバイダー移行モードを使用します
- 管理コンソールで、リソース > JMS > アクティベーション・スペックと進み、WebSphere MQ Resource Adapterを使用するアクティベーション・スペックを選択して、プロバイダー・バージョンに”6.0.0.0”を設定します

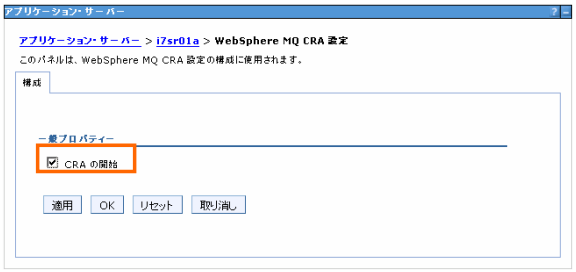


© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/tmj_adm20.html

アクティベーション・スペックの作成（MQの場合） ステップ3

- WAS V7 for z/OSでは、WebSphere MQアクティベーション・スペックは一部がCRAで、一部がSRで実行されます。WebSphere MQ アクティベーション・スペックを使用する場合は、CRAプロセスを使用可能に設定します
- 管理コンソールで、アプリケーション・サーバー > サーバー名 > 通信 > メッセージング > WebSphere MQ CRA 設定と進み、CRAの開始にチェックします

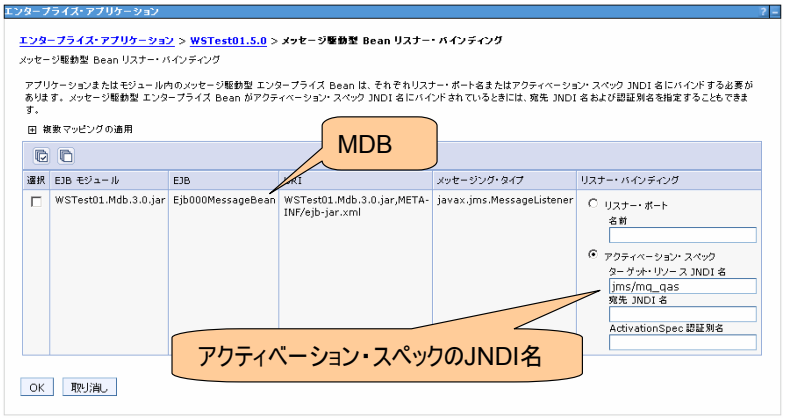


© 2009 IBM Corporation

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/urun_rapp_cra.html

アクティベーション・スペックとアプリケーションのマッピング

- アプリケーション中のMDBと、WASで定義したアクティベーション・スペックとをマッピングします
- 管理コンソールでの設定
  - 設定箇所：エンタープライズ・アプリケーション > アプリケーション名 > メッセージ駆動型Beanリスナー・バインディング



© 2009 IBM Corporation