トラブルシューティング

～WebOTX Standard/Enterprise Edition V5

の問題解決方法～

2006 年 9 月 4 日
第 1 版

第二システムソフトウェア事業部
AP サーバ開発グループ

目次

１．　初めに 4
２．　機能(障害関連) 5
2.1 アプリケーションプロセス異常終了自律復旧 5
2.1.1 異常終了の検出と自律復旧 5
2.1.2 リクエスト保証 6
2.1.3 障害情報記録(Java) 6
2.1.4 障害情報記録(C++) 6
2.2 無応答障害回避 7
2.2.1 スレッド起動停止の実行時間超過の検出 7
2.2.2 オペレーション実行時間超過の検出と自律復旧 7
2.2.3 リクエスト保証（AP 応答監視タイマ) 7
2.3 キュー滞留数の確認 8
３．　障害発生時の対処 10
3.1 起動時障害 10
3.1.1 タイムアウト障害 10
3.1.2 プロセス起動処理失敗 11
3.1.3 遅延 13
3.2 アプリケーション異常終了 14
3.2.1 タイムアウト障害 14
3.2.2 オペレーション失敗 16
3.2.3 正常の終了 21
3.3 停止時障害 22
3.3.1 タイムアウト障害 22
3.3.2 遅延 23
3.4 OutOfMemory 23
3.5 アプリケーションストール 25
４．　プロセス監視 29
５．　ログ管理 32
5.1 Object Broker 32
5.2 WebOTX JNDI 35
5.3 WebOTX JMS 36
5.4 WebOTX 37

5.5 WebOTX WebCont 39
5.6 WebOTX WebAP JSP 41
5.7 WebOTX WebServer 42

１．　初めに

本資料(トラブルシューティング～WebOTX Standard/Enterprise EditionV5 の問題解決方法～)では、WebOTX　Standard Edition V5 または Enterprise Edition V5 を使用している環境において、どのような機能を利用できるのかについて 2 章に記載しています。そして、システム運用中に障害が発生した場合にどのように障害原因を特定していくのか、また障害発生の予防について 3 章に記載しています。また、システムの異常を速やかに検出するために、WebOTX のどのプロセスを監視すればいいのかについて 4 章に記載しています。最後に WebOTX で出力されるログについての詳細を 5 章に記載しています。

２．　機能(障害関連)

この章では WebOTX のプロセス自律復旧機能や障害の局所化など、障害関連の機能について記載しています。

2.1 アプリケーションプロセス異常終了自律復旧

WebOTX Standard Edition/Enterprise Edition では TP モニタ機能により、アプリケーションプロセスの状態を常に監視しています。アプリケーションプロセスが異常終了した場合、そのアプリケーションプロセスは自律的に復旧されます。またその際に、詳細な障害情報を記録しますので、後からの障害解析を容易に行うことができます。

2.1.1 異常終了の検出と自律復旧

アプリケーションプロセスの異常終了を検知し、そのプロセスが使用していたリソース(コネクション、共有メモリ、データベース資源)を WebOTX が解放します。さらに指定に従ってプロセスの再起動を行うことができます。これにより、障害発生後もそれ以前と同じ処理能力を維持することができます。

・プロセス異常終了時の自動再起動設定
　運用管理ツールのシステムのプロパティの[上限設定]-[プロセス障害時の再起動]
　(既定値：50 回)

プロセスグループが複数プロセスで構成される場合、その中の一つのアプリケーションプロセスが異常終了しても、他のアプリケーションプロセスは通常通り処理を行うことができます。このため、プロセスグループとしては、異常終了したアプリケーションプロセスが再起動されるまでの間もサービスの継続性を確保できます。

C++で作成されたサーバアプリケーションの場合は、異常終了発生後にオペレーションが閉塞します。アプリケーション作成者は、閉塞している間にプログラムのチェックを実施することになります。オペレーション単位で閉塞状態にすることができるので、正常動作している他のオペレーションは引き続き動作させることは可能です。閉塞されたオペレーションを呼び出すと以下のメッセージがイベントログ・シスログに出力され、問題のオペレーションは実行されません。

OTXM:<システム名>:<プロセスグループ名>:<プロセス ID>: E:1:TPS10-03201

OPERATION BLOCKED.TX-GROUP プロセスグループ名 プロセス ID

2.1.2 リクエスト保証

オペレーション実行中に何らかの障害が発生してプロセス終了となった場合、TP モニタがそのプロセスに代わりエラー応答をクライアントに返します。イベントログ・シスログにも情報を残します。キューで待っていたリクエストは、別プロセスが引き継いで処理をすることができます。他に起動中のアプリケーションプロセスがなく、かつプロセス再起動回数を使い果たした場合は別プロセスに引き継げませんが、この場合はエラー応答をクライアントアプリケーションに返すため、サーバアプリケーションの異常終了に引きずられてクライアントアプリケーションが無応答になることはありません。

2.1.3 障害情報記録(Java)

オペレーション実行中に Java ランタイム例外が発生した場合、正確には WebOTX の基盤で Java ランタイム例外を捕捉した場合はアプリケーションプロセスを異常終了させますが、この際に捕捉した例外のスタックトレースがアプリケーションログに記録されます。この情報から、Java ソースのどの行が原因で異常終了したかを特定することができます。

2.1.4 障害情報記録(C++)

オペレーション実行中にネイティブ例外が発生した場合、正確には例外ハンドリングを行う設定(C++デフォルト)にしたアプリケーションの WebOTX の基盤でネイティブ例外を捕捉した場合は該当スレッドを閉塞させます。スレッドの残りがない場合はプロセスを終了させます。ネイティブ例外を捕捉した際はイベントログ・シスログにメッセージを出し、ネイティブ例外エラーコードも共に出力します。

アプリケーションログにはアプリケーションエラーが発生した旨のエラーメッセージを出力します。これによりどのオペレーション呼び出しでエラーが発生したか特定することは可能ですが、Java のようにスタックを出力することはできません。スタックイメージを確認するには例外ハンドルの設定を無効にし、ワトソンログ出力を有効にする必要があります。

・例外ハンドルの設定
運用管理ツールのシステムのプロパティの[例外ハンドル]-[例外ハンドルを行う]
(既定値：行う)

2.2 無応答障害回避

アプリケーションプロセスが無応答や極度の遅延に陥った場合も、それを検出し、情報を採取し、リカバリを行う機能を実装しています。

2.2.1 スレッド起動停止の実行時間超過の検出

スレッド起動処理(サーバオブジェクトや常駐オブジェクト作成)、スレッド終了処理(サーバオブジェクトや常駐オブジェクト削除)の際の無応答障害を検出してプロセスは強制終了します。起動処理の障害で且つ再起動設定があれば再起動されます。スレッド初期化タイムアウト時間はプロセスグループ単位に設定できます。

・スレッド初期化時間設定
運用管理ツールのプロセスグループのプロパティの[スレッド制御]-[スレッド初期化時間] (既定値：600 秒)

2.2.2 オペレーション実行時間超過の検出と自律復旧

サーバアプリケーションでオペレーションを実行した際の無応答障害を検出し、自律復旧させることができます。実行時間上限設定を超えてもサーバアプリケーションから応答がない場合、WebOTX はアプリケーションプロセスが無応答障害に陥っていると判断し、アプリケーションプロセスを強制終了し、再起動設定があれば再起動させます。これによりクライアントに応答が返らない事態やサーバの資源を無制限に消費する事態を回避できます。実行時間上限値はオペレーション毎に設定することができます。

・実行時間上限設定
運用管理ツールのオペレーションのプロパティの[オペレーションの自動活性]-[実行時間上限]

2.2.3 リクエスト保証（AP 応答監視タイマ）

AP 応答監視タイマ機能によりサーバアプリケーションの無応答障害を検出し、別プロセスが異常状態のアプリケーションプロセスに成り代わってクライアントアプリケーションにエラー応答を返すことができます。これによりサーバアプリケーションの無応答障害が発生しても、クライアントアプリケーションは無応答状態とならずに応答を受け取ること

ができます。

・AP 応答監視タイマ設定
運用管理ツールのシステムのプロパティの[上限設定]-[AP 応答監視タイマ]

2.3 キュー滞留数の確認

クライアントからのリクエストは通信リスナが受付を行い、アプリケーション側のキューに一旦格納されます。キューはプロセスグループ単位のキューとプロセス単位のキューが作成されますが、ステートレスの場合はプロセスグループ単位のキューが、ステートフルの場合はプロセス単位のキューが使用されます。アプリケーション側では処理スレッドに空きがあればすぐにリクエストをキューから取りだして処理しますが、スレッドが塞がっている場合にはしばらくキューに滞留した状態が続くことになります。以下では、このキュー滞留数を確認する方法を説明します。

2.3.1 quewrt コマンドによるキュー滞留数の確認

このコマンドで確認できるのはプロセスグループ単位での滞留リクエスト数(ステートレスの場合)、プロセス単位での滞留リクエスト数(ステートフルの場合)です。コマンドは<インストールディレクトリ>/Trnsv/bin にあります。UNIX の場合、 /opt/WebOTX/Trnsv/lib を LD_LIBRARY_PATH (HP-UX PA_RISC は SHLIB_PATH) に設定し、WebOTX 運用ユーザでコマンドを実行して下さい。

> quewrt -n システム名 -M 収集間隔 -C 収集回数 [-f ファイル名]
システム名 : WebOTX システム名
収集間隔 : 情報の収集間隔を秒単位で指定（1～2^31-1。10 以上を推奨）
収集回数 : 情報の収集回数を指定（1～2^31-1）
ファイル名 : データ収集結果を格納するファイル名を 64 バイト以内で指定

結果表示例：

MODE	QUE-NAME	MSG-NUM	POOL-NAME	CONECT	DEQUE
Q	STPCTLQUE28752	0	ea60	ON	ON
Q	QUE_TIMER	0	746d6d70	ON	ON
Q	TPproc029983	0	ea60	ON	ON
Q	java	0	ea60	ON	ON
Q	_OTSLink	0	ea60	ON	ON
Q	JOURNALQUE0	0	13880	ON	ON
Q	SENDTPPQUE	0	ea60	ON	ON
Q	STPRCVQUE	0	ea60	ON	ON
Q	TPproc028754	0	ea60	ON	ON

```
  Q     IIOPLISTENER          0          ea60          ON        ON
  Q     TPproc028758          0          ea60          ON        ON
```

QUE-NAME

キュー名称です(WebOTX 内部で使用するキューも含まれます)。プロセスのキューは TPproc(プロセス ID) となっています。プロセス ID とプロセスグループの対応は、<WebOTX インストールディレクトリ>/Trnsv/catalog/<システム名>/history.act ファイルを参照して下さい。

MSG-NUM

キューに滞留しているリクエストの数が表示されます。

2.3.2 contps コマンドによるキュー滞留数の確認

このコマンドで確認できるのはオペレーション単位での滞留リクエスト数です。コマンドは<インストールディレクトリ>/Trnsv/bin にあります。UNIX の場合、/opt/WebOTX/Trnsv/lib を LD_LIBRARY_PATH (HP-UX PA_RISC は SHLIB_PATH) に設定し、WebOTX 運用ユーザでコマンドを実行して下さい。

```
> contps -n システム名 DI X TR
```

結果表示例：

```
********** TPBASE TR STATUS ****************************************************
TxID       :処理状態  :Tx 滞留数 :総 Tx 数    :最小応答:最大応答:平均応答:応答合計
LOX000     :IDLE      :         0:         0:       -1:       -1:       -1:         -1
UKN000     :IDLE      :         0:         0:       -1:       -1:       -1:         -1
APR000     :IDLE      :         0:         0:       -1:       -1:       -1:         -1
ABAAAB     :START     :         0:   1022991:        3:      503:        4:    4727079
ABAAAA     :IDLE      :         0:         0:       -1:       -1:       -1:         -1
ABaa00     :IDLE      :         0:         0:       -1:       -1:       -1:         -1
ABaa01     :IDLE      :         0:         0:       -1:       -1:       -1:         -1
ABaa20     :IDLE      :         0:         0:       -1:       -1:       -1:         -1
ABaa21     :IDLE      :         0:         0:       -1:       -1:       -1:         -1
********** END OF TR STATUS ******************************************************
```

Tx 滞留数

TxID に対応するオペレーションで処理待ちとなっているリクエストの数です。ここに表示される 6 桁の TxID と CORBA オペレーションの対応付けには、イベントジャーナルを実行した時に<WebOTX インストールディレクトリ>/Trnsv/catalog/<システム名>/tmp にできる otx.t2o ファイルを参照して下さい。

３．　障害発生時の対処

この章では障害発生時の対処方法について記載しています。　WebOTX では障害発生時に重要な情報をイベントログ・シスログに出力します。以下に、主な障害について説明します。

・TPS10-11101　→　「3.1.1 タイムアウト障害」を参照して下さい。
・TPS10-11001 またはTPS10-00402　→　「3.1.2 プロセス起動処理失敗」を参照してください。
・TPS10-13301　→　「3.2.1 タイムアウト障害」及び「3.3.1 タイムアウト障害」を参照してください。
・TPS10-04401　→　「3.2.2.1 ネイティブ例外への対応」を参照してください。
・TPS10-13201 またはTPS15-01107　→　「3.2.2.2 APアボートの対応」を参照してください。
・TPS15-01214 またはTPS15-01210　→　「3.2.3 正常の終了」を参照してください。

3.1 起動時障害

起動操作を行った際に起動完了に著しく時間がかかった、または失敗した場合を起動障害とします。

3.1.1 タイムアウト障害

起動時の処理では無応答状態を避けるためにタイマ監視しています。タイマ値まで待っても各処理が完了しない場合は起動失敗とみなして異常終了します。タイマはデフォルトで 10 分となっています。運用管理ツールでプロセスグループの状態を確認したときに、起動処理中の状態が長く続いている場合は、このタイムアウト障害が繰り返し発生している場合があります(但しプロセス障害時の再起動回数を 2 以上としている場合です)。イベントログ・シスログに TPS10-11101 が出力されている場合は、起動処理で遅延しています。

状況の確認方法

以下のメッセージがイベントログ・シスログにでます。

OTXM: <システム名>:<プロセスグループ名>:<プロセス ID>: E:1:TPS10-11101 THREAD INIT TIME OUT.　CODE: x　TX-GROUP <プロセスグループ名> <プロセ

ス ID>

また、以下のメッセージが AP ログにでます。

TPS10-11101 THREAD INIT TIME OUT CODE: x TX-GROUP <プロセスグループ名> <プロセス ID>

対象AP の特定

イベントログ・シスログを確認してください。

原因の特定

イベントログ・シスログ、AP ログを確認してください。イベントログ・シスログに TPS10-11101 が出力され、プロセス起動に失敗した場合、以下の処理で遅延している可能性がありますので確認してください。

・常駐オブジェクトのコンストラクタ
・常駐オブジェクトの生成時コールバック
・サーバオブジェクトのコンストラクタ(ステートレス・アパートメントの場合)
・サーバオブジェクトの生成時コールバック(Java, ステートレス・アパートメントの場合)
・名前サーバ登録時フック(ステートレス・アパートメントの場合)

復旧方法

AP の処理を修正してください。

予防のための対策

サーバ AP の起動にかかる時間が理論上ありえる範囲なのかありえないのかによって対策は異なります。理論上ありえないのであれば AP の処理を見直す必要があります。理論上ありえる範囲の場合は運用管理ツールのプロセスグループのプロパティの[スレッド制御]-[スレッド初期化時間]を必要に応じてあらかじめ大きな値を設定してください。

3.1.2 プロセス起動処理失敗

運用管理ツールでプロセスグループの状態を確認したときに、起動処理中の状態が長く続いている場合は、この起動処理失敗が繰り返し発生している可能性があります(但しプロセス障害時の再起動回数を 2 以上としている場合です)。プロセスが起動してすぐに停止状態になるといった場合も、起動処理中に例外している可能性があります。イベントログ・

シスログに TPS10-11001 または TPS10-00402 が出力されている場合は、起動処理中に例外しています。

状況の確認方法

・ スレッド初期化前及びスレッド終了後の処理で例外した場合
以下のメッセージがイベントログ・シスログにでます。

> OTXM: <システム名>:<プロセスグループ名>:<プロセス ID>: I:1:TPS10-00402 ACCEPTED ABNORMAL TERMINATION REQUEST FROM TPP. TX-GROUP <プロセスグループ名> <プロセス ID>

Java アプリケーションの場合、スタックダンプが AP ログにでます。

・ スレッド起動処理で例外した場合
以下のメッセージがイベントログ・シスログにでます。

> OTXM: <システム名>:<プロセスグループ名>:<プロセス ID>: E:1:TPS10-11001 THREAD INFORMATION INIT ERROR. CODE:x TX-GROUP <プロセスグループ名> <プロセス ID>

Java アプリケーションの場合、スタックトレースが AP ログにでます。

対象AP の特定

イベントログ・シスログを確認してください。

対象オペレーションの特定

AP ログに出力されたスタックトレースを確認してください（Java アプリケーションの場合)。イベントログ・シスログに TPS10-00402 が出力されている場合は、以下の処理で例外している可能性がありますので確認してください。

- リソースアダプタの start の処理(EJB)
- ステートレス Bean,またはメッセージドリブン Bean のコンストラクタ、ejbCreate の処理(EJB)
- jp.co.nec.WebOTX.InitTerm.initialize(Java)
- WebOTX_Init() (C++)
- コンポーネント初期化関数
- サーバオブジェクトのコンストラクタ(ステートレス・フリーの場合)

・サーバオブジェクトの生成時コールバック(Java, ステートレス・フリーの場合)
・名前サーバ登録時フック(ステートレス・フリーの場合)

イベントログ・シスログに TPS10-11001 が出力されている場合は、以下の処理で例外している可能性がありますので確認してください。

・常駐オブジェクトのコンストラクタ
・常駐オブジェクトの生成時コールバック
・サーバオブジェクトのコンストラクタ(ステートレス・アパートメントの場合)
・サーバオブジェクトの生成時コールバック(Java, ステートレス・アパートメントの場合)
・名前サーバ登録時フック(ステートレス・アパートメントの場合)

また、AP トレースに以下のエラーログが出力されていないか確認して下さい。

・APProlog abnormally done. アプリケーションの初期化に失敗しました。
・Thread does not have initialized(ret=XX). スレッドの初期化に失敗しました。
・ERROR:ClassNotFound

この場合 AP トレースに出力されたエラーログより障害を解析してください。

復旧方法

イベントログ・シスログ、AP ログから例外の原因を特定してください。例外がタイミングによるものであれば、プロセス再起動回数を 2 以上に設定することにより自動的に復旧される可能性があります。

予防のための対策

イベントログ・シスログ、AP ログから例外の原因を特定し、AP の処理を修正してください。

3.1.3 遅延

AP 起動に時間がかかった場合、AP を起動後すぐの呼び出しが何故か失敗したが暫くしたら呼べるようになった、などの場合、起動処理に時間がかかっている可能性があります。AP の起動がなかなか終わらない(プロセスグループのアイコンが緑にならない)という場合も、起動処理が遅延している可能性があります。

状況の確認方法

運用管理ツールでプロセスグループの状態を確認してください。

復旧方法

アプリケーションプロセスの起動で遅延している場合は、遅延箇所を修正してください。

予防のための対策

サーバ AP の起動にかかる時間が、理論上ありえないのであれば AP の処理を見直す必要があります。タイムアウト時間を設定すると、起動処理を監視し、ストール状態を回避することができます。運用管理ツールのプロセスグループのプロパティの[スレッド制御]-[スレッド初期化時間]を必要に応じて変更してください。

3.2 アプリケーション異常終了

起動後しばらく動作していたものが異常終了したケースです。イベントログ・シスログ、AP ログまたは<WebOTX インストールディレクトリ>/Trnsv/catalog/<システム名>/history.act を調査してください。AP ログはデフォルトでは以下のファイルに出力されます。

<WebOTX インストールディレクトリ>/Trnsv/logs/<プロセスグループ名>.<プロセス ID>.log

3.2.1 タイムアウト障害

リクエストを受け付けて実装部分(オペレーション)を呼び出した時にそのオペレーションが無応答状態になることを避けるためにタイマ監視しています。タイマ値まで待ってもオペレーションが完了しない場合は失敗とみなしてイベントログ・シスログにエラーメッセージを出力し、プロセスは終了します。

状況の確認方法

実行時間超過が起きた場合、イベントログ・シスログには以下のメッセージが出力されます。

OTXM:<システム名>:<プロセスグループ名>:<プロセス ID>: E:1:TPS10-13301 Tx TIME OVER ERROR. CODE:1 TX-GROUP <プロセスグループ名> <プロセス ID>

また、クライアントアプリケーションには NO_RESPONSE(3927)のエラーを返します。

問題の検出方法

イベントログ・シスログの監視

TPS10-13301 で始まるメッセージを監視してください。

対象AP の特定

イベントログ・シスログに出力される、TPS10-13301 のメッセージから該当のプロセスグループを特定します。

対象オペレーションの特定

AP ログに以下のように出力されます。

TPS10-13301 Tx TIME OVER ERROR. CODE:x TX-GROUP <プロセスグループ名> <プロセス ID> elptime-over info：TXID=AAABAB,…

この AP ログの TXID から実行時間超過が発生したオペレーションを特定できます。TXID とオペレーション名の関連を調べるためにイベントジャーナルを採取してください。イベントジャーナルは運用管理ツールのシステムを右クリックし、「イベントジャーナルの編集」の実行で採取できます。実行すると、<WebOTX インストールディレクトリ>/Trnsv/catalog/<システム名>/tmp ディレクトリに otx.t2o ファイルが作成され、その中に TXID とオペレーションの対応関係が記載されています。

回避方法

原因の究明ができるまではプロセス自動再起動設定（運用管理ツールのシステムのプロパティの[上限設定]-[プロセス障害時の再起動]）を大きめにすることで、実行時間超過によってサービス停止とならないようにしてください。

復旧方法

実際にこの現象が発生してしまった場合の復旧方法について説明します。

プロセスグループの再起動）

再起動設定をしている場合は自動で再起動されているはずなので復旧は済んでいます。再起動回数を使い果たした場合や再起動設定がされていない場合に備え、念のため対象の AP の状態を確認し、停止状態の場合は再起動させてください。

システムの再起動）

システム全体に影響が及んでしまっている場合は局所的な対応では復旧しない可能性

があります。その場合はシステム再起動を行なってください。

予防のための対策

問題のあったオペレーションがそもそもストールなどによって応答の望みがまったくないものなのか(いつまで待っても応答が返らないものなのか)、やむを得ない理由により完了までに時間がかかっていたのかにより、対策は異なります。

問題の AP の修正)

問題のあったオペレーションがストールなどにより応答の望みがない場合、問題を解決するためには AP を修正する必要があります。解決までの間は現状のままの設定として、実行時間監視により無応答状態を回避するしかありません。

実行時間上限の見直し)

AP に無応答を引き起こす問題がなく、実行時間上限設定が短すぎたために本障害になったと考えられる場合は、運用管理ツールのオペレーションのプロパティの[オペレーションの自動活性]-[実行時間上限]を見直してください。

3.2.2 オペレーション失敗

リクエストを受け付けて実装部分(オペレーション)を呼び出したものの、それが正常終了せずに Java ランタイム例外やネイティブ例外などを検出した場合、イベントログ・シスログにエラーメッセージ TPS10-04401(ネイティブ例外)、TPS10-13201(Java ランタイム例外)を出力してプロセスは終了します。

3.2.2.1 ネイティブ例外への対応

事象説明

例外ハンドリングをする設定(C++AP のデフォルト設定)で不正アドレス参照などが発生すると本エラーとなります。本エラーが起きるとクライアントには CORBA::UNKNOWN(3923)のエラーが返ります。該当のプロセスは例外の起きたスレッドを除く残りのスレッドで実行を継続します。残りのスレッドが 0 になった場合はプロセス終了します。

状況の確認方法

イベントログ・シスログの確認)

ネイティブ例外の発生を WebOTX で検出した場合、イベントログ・シスログに以下のメ

ッセージが出力されます。

OTXM:<システム名>:<プロセスグループ名>:<プロセス ID>: E:200:TPS10-04401 EXCEPTION OCCURED. CODE:<例外コード(Windows),・シグナル番号(UNIX)> <アプリケーショングループ名> <プロセスグループ名> <プロセス ID>

AP ログの確認)

ネイティブ例外の発生を WebOTX で検出した場合、AP ログに以下のメッセージが出力されます。

Error: Exception occurred. repositoryID=<リポジトリ ID> operation=<オペレーション名>

問題の検出方法

イベントログ・シスログの監視

TPS10-04401 のメッセージを監視することで検出可能です。

対象AP の特定

TPS10-04401 のメッセージから特定します。

対象オペレーションの特定

AP ログに出力されたメッセージから、例外が発生したオペレーションを特定します。

対象コードの特定

通常の方法では特定できません。もし再現可能なら特定する方法があります。下記「原因不明時の調査方法」を参照してください。

復旧方法

実際にこの現象が発生してしまった場合の復旧方法について説明します。

プロセスグループの再起動)

通常はプロセス再起動されるので特に対処する必要がないのですが、再起動回数を越えてしまうと、プロセスグループが停止状態になってしまいます。その場合はプロセスグループの起動が必要です。

オペレーションの閉塞解除)

オペレーション実行中にアプリケーション例外などが発生した場合(特に C++モジュール)設定によってはオペレーションが閉塞してしまいます。閉塞してしまうとオペレーションの状態が停止状態となりアプリケーショングループを再起動するまで該当オペレーションを呼出すことができなくなります。その場合はオペレーションを再起動する必要があります。

予防のための対策

この問題を予防するための対策については以下の方法があります。

問題の AP の修正)

完全に問題を解決するためには AP を修正する必要があります。例外コードを踏まえて、対象のオペレーションのコードを確認してください。

プロセス数の設定)

予防ではありませんが、ネイティブ例外が発生した場合の影響を軽減するためにプロセス数を増やすことが効果的な場合があります。ネイティブ例外は発生し、全スレッドが使用不能になると該当プロセスは再起動しますが、再起動が完了するまでの間の呼び出しはエラーになってしまいます。そこで複数のプロセスを設定することにより、再起動処理中も他のプロセスで処理が続行できるように設定できます。運用管理ツールのプロセスグループのプロパティの[プロセス制御]- [プロセス数]の設定を見直してください。なお、ステートフルの場合はこの対処はあまり効果がありません。

再起動回数の設定)

アボートした場合のプロセスの再起動回数について設定します。運用管理ツールのシステムのプロパティの[上限設定]-[プロセス障害時の再起動]の設定を見直してください。なお、24 時間無停止運用を行なっている場合、アプリケーショングループ停止が行なわれないため、プロセス再起動回数がリセットされません。少ない頻度でアボートが発生していてもいずれは再起動回数を越えて再起動が行なわれなくなります。対策として運用管理ツールのシステムのプロパティの[上限設定]-[プロセス正常と仮定]を設定して、一定時間経過した場合は再起動回数をリセットする必要があります。

原因不明時の調査方法

もし再現性がある場合は、運用管理ツールのシステムのプロパティの[例外ハンドル]-[例外ハンドルを行う]のチェックを OFF にして再現させてください。出力された core(UNIX)ないしワトソンログ(Windows)で発生箇所特定してください。core ファイルは、<WebOTX インストールディレクトリ>/Trnsv ディレクトリ直下に出力されます。

3.2.2.2 AP アボートの対応

事象説明

WebOTX ではJava のオペレーション実行中に起きた例外についてはそれを捕捉してプロセス終了します。捕捉した場合は内部的に TPSAbort(-1)というプロセス終了の API を発行してプロセス終了します。その際別のスレッドで実行中のオペレーションは完了を待ち合わせます。本エラーが起きるとクライアントに CORBA::UNKNOWN(3926)が返ります。またサーバ AP のトレースには例外捕捉時のスタックトレースを出力します。なお、サーバ AP がオペレーション中で自ら TPSAbort(-1)を呼び出した場合も本エラーとなりますが、その場合はスタックトレースを出力しません。

状況の確認方法

イベントログ・シスログの確認)

アボートによりプロセス終了した場合、イベントログ・シスログに以下のメッセージが出力されます。

> OTXM:<システム名>:<プロセスグループ名>:<プロセス ID>: E:1:TPS10-13201 ABORT IS PROCESS END. CODE:0
> OTXM:<システム名>::<プロセス ID>: W:13:TPS15-01107 Process abnormal end. PID=[<プロセス ID>],Class[<プロセスグループ名>], ped[<アプリケーショングループ名>.ped]

AP ログの確認)

アボートによりプロセス終了した場合、AP ログに以下のメッセージが出力されます。

> TPS10-13201 ABORT IS PROCESS END. CODE:0

また、Java ランタイム例外が発生した場合は AP ログに Java のスタックトレースが出力されます。

> 例:
> 2006/08/15 10:23:53.125|001: Error: Occur exception(catch java.lang.RuntimeException) at Invoking. message=null
> java.lang.RuntimeException
> at complex.basicApObj.exceptionEnd(basicApObj.java:21)

at complex.basicApPOA._invoke(basicApPOA.java:55)

・・・

history.act の確認）

アボートなどによりプロセスが異常終了した場合、history.act に以下のメッセージが出力されます。

TPS15-01107 Process abnormal end. PID=[<プロセス ID>], class[<プロセスグループ名>], ped[<アプリケーショングループ名>.ped]

問題の検出方法

イベントログ・シスログの監視

アボートが発生した場合、イベントログ・シスログが出力されますのでそのメッセージを監視します。

対象AP の特定

イベントログ・シスログに出力される、TPS15-01107 のメッセージから該当のプロセスグループを特定します。

対象オペレーションの特定

Java ランタイム例外の場合は、AP ログに出力されるスタックトレースから例外が発生したオペレーションを特定します。

対象コードの特定

Java ランタイム例外が発生した場合は AP ログにスタックトレースが出力されていますので、そこから該当の箇所を特定してください。サーバ AP がオペレーション中で自ら TPSAbort(-1)を呼び出した場合にはスタックトレースは出力されませんので、コードにて TPSAbort(-1)を呼び出している箇所を探して、原因を特定してください。

復旧方法

通常はプロセス再起動されるので特に対処する必要がないのですが、再起動回数を越えてしまうと、プロセスグループが停止状態になってしまいます。その場合はプロセスグループの起動が必要です。

予防のための対策

この問題を予防するための対策については以下の方法があります。

問題の AP の修正）

完全に問題を解決するためには AP を修正する必要があります。Java ランタイム例外によりアボートしている場合は、該当 AP ログに出力されているスタックトレースからアボート箇所の特定を行なうことができます。発生箇所を中心に確認して原因を排除してください。ただし、該当 AP で例外を意図的に catch している場合等、スタックトレースが出力されない事もあります。TPSAbort(-1)によりアボートしている場合は呼び出し箇所を特定し、そこに至ってしまった要因を取り去るようにしてください。

プロセス数の設定）

予防ではありませんが、AP アボートが発生した場合の影響を軽減するためにプロセス数を増やすことが効果的な場合があります。AP アボートが発生すると該当プロセスは再起動しますが、アボートしてから再起動が完了するまでの間の呼び出しはエラーになってしまいます。そこで複数のプロセスを設定することにより、1 つがアボートしても他のプロセスで処理が続行できるように設定できます。運用管理ツールのプロセスグループのプロパティの[プロセス制御]- [プロセス数]の設定を見直してください。なお、ステートフルの場合はこの対処はあまり効果がありません。

再起動回数の設定）

アボートした場合のプロセスの再起動回数について設定します。運用管理ツールのシステムのプロパティの[上限設定]-[プロセス障害時の再起動]の設定を見直してください。なお、24 時間無停止運用を行なっている場合、アプリケーショングループ停止が行なわれないため、プロセス再起動回数がリセットされません。少ない頻度でアボートが発生していてもいずれは再起動回数を越えて再起動が行なわれなくなります。対策として運用管理ツールのシステムのプロパティの[上限設定]-[プロセス正常と仮定]を設定して、一定時間経過した場合は再起動回数をリセットする必要があります。

3.2.3 正常の終了

・TPS15-01214、TPS15-01210

これらは<WebOTX インストールディレクトリ>/Trnsv/catalog/<システム名>/history.act に出力されるプロセス及びプロセスグループの正常停止メッセージであり、運用停止時に出力されます。異常終了時は別のメッセージが出力されます。AP が異常終了したと思って調査したものの実際には運用停止操作が行われていた、といったことがまれにあります。history.act を確認し、終了が異常か正常かを見極めてください。

3.3 停止時障害

停止操作を行った際に停止完了に著しく時間がかかった、または失敗した場合を停止障害とします。アプリケーションプロセスの異常終了時の停止処理に著しく時間がかかった場合も停止障害とします。

3.3.1 タイムアウト障害

アプリケーショングループ、もしくはプロセスグループの停止処理がタイムアウトしたという場合は、アプリケーションプロセスの停止処理がタイムアウトした可能性があります。

停止時の処理では無応答状態を避けるためにタイマ監視しています。タイマ値まで待っても各処理が完了しない場合は停止失敗とみなしてその後の処理を行わずに異常終了します。タイマはデフォルトで 40 秒となっています。イベントログ・シスログに TPS10-13301 が出力されている場合は、停止処理中にタイムアウトしています。

状況の確認方法

スレッド停止処理でタイムアウトした場合、以下のメッセージがイベントログ・シスログにでます。

> OTXM: <システム名>:<プロセスグループ名>:<プロセス ID>: E:1:TPS10-13301 Tx TIME OVER ERROR. CODE:ｘｘｘｘ TX-GROUP <プロセスグループ名> <プロセス ID>

また、以下のようなメッセージが AP ログにでます。

> TPS10-13301 Tx TIME OVER ERROR. CODE:ｘｘｘｘ TX-GROUP <プロセスグループ名> <プロセス ID> elptime-over info : S_TIME=08-02/20:23:42.7030 LIMIT=60FUNCTION=ThreadEpilo g :00E87990 LIBRARY=<ライブラリ名>:00E80000

対象AP の特定

イベントログ・シスログを確認してください。

原因の特定

イベントログ・シスログ、AP ログを確認してください。イベントログ・シスログに TPS10-13301 が出力され、プロセス停止が強制終了となった場合、以下の処理で遅延している可能性がありますので確認してください。

- 名前サーバ削除時フック(ステートレス・アパートメントの場合)
- サーバオブジェクトの解放時コールバック(Java, ステートレス・アパートメントの場合)
- サーバオブジェクトのデストラクタ(C++, ステートレス・アパートメントの場合)
- 常駐オブジェクトの解放時コールバック
- 常駐オブジェクトのデストラクタ(C++)

3.3.2 遅延

アプリケーショングループ、もしくはプロセスグループの停止処理に時間がかかるという場合は、アプリケーションプロセスの停止処理が遅延している可能性があります。AP 停止に時間がかかった場合は前記タイムアウト障害の一歩手前の状況と考えられます。

状況の確認方法

運用管理ツールでプロセスグループの状態を確認してください。

復旧方法

アプリケーションプロセスの停止で遅延している場合は、プロセスグループを強制停止し、再度起動してください。

3.4 OutOfMemory

事象説明

該当のプロセスでメモリを使いすぎた場合に発生します。メモリリークや必要以上に大きなメモリ確保が原因かもしれませんが、単に設定値が不適切なことが原因のこともあります。

状況の確認方法

AP ログに java.lang.OutOfMemoryError のスタックトレースが出力されます。

復旧方法

通常はプロセス再起動されるので特に対処する必要がないのですが、プロセス自動再起

動回数を越えてしまうと、プロセスグループが停止状態になってしまいます。その場合はプロセスグループの起動が必要です。

予防のための対策

メモリリークの確認）

プロファイリングなどを採取して、メモリを多く使用している箇所、またはメモリリークしていないかなどを確認してください。確認できたら、該当箇所のソースを修正してメモリ使用量を減らしたりメモリリークを改善したりしてください。

Java ヒープサイズの確認）

大きなメモリ確保などして、Java の最大ヒープサイズまでメモリを使い果たしてしまった場合に
java.lang.OutOfMemoryError が発生する可能性があります。このような場合には、運用管理ツールのプロセスグループのプロパティの[Java VM オプション]-[ヒープの最大サイズ]を見直してください。

ヒープを使い果たしてしまったかどうかは[Java VM オプション]-[その他引数]に「-verbose:gc」を指定すると、GC のタイミングで AP ログに現在使用しているヒープサイズが出力されますので、その値が最大ヒープサイズに近づいているかどうかで判断できます。

Permanent 領域の確認）

JVM は、クラスやメソッドの情報を Permanent 領域という領域に格納します。ロードされるクラスの総数 がかなり多い場合など、Permanent 領域を使い果たしてしまった場合には java.lang.OutOfMemoryError が発生する可能性があります。このような場合には、運用管理ツールのプロセスグループのプロパティの[Java VM オプション]-[その他引数]に「-XX:PermSize=<初期値> -XX:MaxPermSize=<最大値>」を指定すると Permanent 領域を調整できるのでこの値を見直してください。

Permanent 領域を使い果たしてしまったかどうかは[Java VM オプション]-[その他引数]に「-verbose:gc」を指定すると、GC のタイミングで AP ログに現在使用しているヒープサイズが出力されますので、その値が最大ヒープサイズに近づいているかどうかで判断できます。ヒープサイズがまだ余裕があるのにもかかわらず java.lang.OutOfMemoryError が発生するような場合は、Permanent 領域が不足している可能性があります。

HP-UX の場合は、[Java VM オプション]-[その他引数]に「-Xverbosegc」を指定することで、AP ログに現在使用している Permanent 領域のサイズを出力することができますので、これで判断することもできます。

仮想メモリ使用量の確認）

仮想メモリが不足している場合にも java.lang.OutOfMemoryError が発生する可能性があります。AP 起動中の仮想メモリサイズ(Windows), スワップサイズ(UNIX)を確認して不足しているようでしたら、増やしてください。

カーネルパラメータの確認）

カーネルパラメータ maxdsiz(64bit 版は maxdsiz_64bit)が不足している可能性もあります。現在の使用量は簡単には確認できませんが、この値を増やして様子を見ることも試してください。

プロセス数の設定）

予防ではありませんが、AP アボートが発生した場合、該当プロセスは再起動しますが、アボートしてから再起動が完了するまでの間の呼び出しはエラーになってしまいます。そこで複数のプロセスを設定することにより、1 つがアボートしても他のプロセスで処理が続行できるように設定できます。運用管理ツールのプロセスグループのプロパティの[プロセス制御]-[プロセス数]を見直してください。

プロセス自動再起動回数の設定）

アボートした場合のプロセスの再起動回数について設定します。運用管理ツールのシステムのプロパティの[上限設定]-[プロセス障害時の再起動]を見直してください。

なお、24 時間無停止運用を行なっている場合、アプリケーショングループ停止が行なわれないため、プロセス再起動回数がリセットされません。少ない頻度でアボートが発生していてもいずれは再起動回数を越えて再起動が行なわれなくなります。対策として運用管理ツールのシステムのプロパティの[上限設定]-[プロセス正常と仮定]を設定して、一定時間経過した場合は再起動回数をリセットする必要があります。

その他）

応答データサイズが大きい時､その応答送信処理の延長で java.lang.OutOfMemoryError となってしまうことがあります。電文長とヒープサイズの関係も気をつけてください。

3.5 アプリケーションストール

事象説明

サーバ AP の処理がストールした場合、最初は特定の処理のみが応答を返さない現象となりますが、さらに多重度を使い切るなどキュー滞留に影響したりするとプロセスグルー

プ全体に影響が広がります。さらに応答を返さないリクエストが滞留すると Web サーバのスレッド枯渇などを招いてしまい、システム全体へ影響が広がります。

状況の確認方法

ストールしたプロセスではその後の要求に対して処理ができなくなるため、キュー滞留が発生する可能性があります。まずはキュー滞留しているプロセスグループがないか、運用管理ツールのプロセスグループの「キューイング数」をプロセス起動時にご確認ください。また、キュー滞留数はコマンドで確認する事もできます。「2.3 キュー滞留数の確認」を参照してください。

問題の検出方法

AP ストールの検出方法として、該当オペレーションに実行時間上限の設定を行なうことにより、ストール事象を検出することができます。実行時間上限で設定した時間経過しても応答がない場合は TP モニタにより検出され、強制終了させられます。運用管理ツールのオペレーションのプロパティの[オペレーションの自動活性]-[実行時間上限]で設定してください。

原因の特定

UNIX 環境かつ該当 AP が Java の場合、無応答障害発生中にスタックトレースを採取してください。これにより行レベルでストール箇所が判明します。

スタックトレース採取方法)

まずキュー滞留が発生したプロセスグループを運用管理ツールから特定し、<WebOTX インストールディレクトリ>/Trnsv/catalog/<システム名>/history.act からプロセスを特定してください。

例）以下の例ではプロセスグループ AAA は 1 プロセス構成であり、PID は 1111 です。

```
Mon Aug 8 11:54:43 2006 TPS15-01215 Process created. PID=[1111],
CLASS=[AAA], PED=[BBB.ped], [/opt/WebOTX/Trnsv/bi
```

特定したプロセスに対して SIGQUIT を送信してください。また、プロセスが複数あり、特定できない場合は複数のプロセス全てに対して SIGQUIT を送信してください。

例）kill -QUIT 1111

AP ログにスタックトレースが出力されます。

注）AP ログが作成されていない場合(ログが一行も出ていない場合)、事前に AP ログを作成する必要があります。運用管理ツールのプロセスグループのプロパティを右ク

リックし「トレースの設定」で、一旦トレースレベルを 6 に上げた後、トレースレベルを元に戻してください(既定値 5)。

復旧方法

プロセスグループを再起動してください。また、システム全体に影響が及んでしまっている場合は局所的な対応では復旧しない可能性があります。その場合はシステム再起動を行ってください。

予防のための対策

この問題を予防するための対策については以下の方法があります。

問題の AP の修正)

完全に問題を解決するためには AP を修正する必要があります。

実行時間上限の設定)

原因不明などで AP の修正が困難な場合は回避策として該当オペレーションに実行時間上限の設定を行なうことによりストール事象を回避することができます。実行時間上限で設定した時間経過しても応答がない場合は TP モニタにより強制終了させられます。運用管理ツールのオペレーションのプロパティの[オペレーションの自動活性]-[実行時間上限]で設定してください。

また、WebOTX 標準修正(2005 年 12 月版)では、実行時間超過発生時に自動でスタックトレースを採取するよう機能強化されています。この機能を使用すれば、ストールが発生した場合に実行時間上限の設定により自動でプロセス再起動されるだけでなく、ストール箇所の特定も同時に行う事ができます。

キューイング数上限の設定)

キューイング数上限を設定することにより、想定外のキューイングがあった場合は即クライアントにエラーメッセージを返却する事ができます。運用管理ツールのプロセスグループのプロパティの[キューの最大数]-[キューの最大数]で設定してください。また、キューイング数上限はシステム毎、アプリケーショングループ毎の設定も可能です。

プロセス数、スレッド数の設定)

プロセス、スレッドの多重度が小さい場合、クライアントからの要求数や処理時間によってはなかなか応答が返らず、ユーザはストールしているのではないかと感じることがあります。ただ単に応答時間が長くなっているだけの場合はプロセス数、スレッド数が適切であるか確認してください。

４．　プロセス監視

この章ではプロセス監視を行う場合、どのプロセスを監視対象とすべきかについて記載しています。

次に示すプロセス名のものを監視対象としてください(Windows の場合拡張子 exe がプロセス名につきます)。

プロセス名	説　明	業務への影響	
tpmMain	TP モニタ子プロセス	○	WebOTX に関するすべてのアクセスに支障あり。
tpmonitor	TP モニタ親プロセス	―	なし(tpmMain の監視を行う)。
woadmsv	運用管理サーバ	△	運用管理ツール、コマンドからの運用操作が不可(システムとしては動作可能)。
tpadmd	TPadm デーモン	△	運用管理ツール、コマンドからの運用操作が不可(システムとしては動作可能)。
tpssendtpp	tpsend プロセス	△	運用管理ツール、コマンドからの以下の運用操作が不可。 • トレース設定動的変更 • クライアントにメッセージ送信（クライアント管理ライブラリ使用時） • コンポーネントの動的変更 • サーバプロセスメッセージ通知（woadmcom NOTIFYMESSAGE）
wockacount	ログイン認証プロセス(Windows 版には存在しません)	△	運用管理ツールからの接続が不可。
worelay	Transaction サービス連携用プロセス	△	Transaction Service との連携が不可。
systpp	システム TPP	△	運用管理ツールからの一部操作(起動、停止など)

			が不可。
jnlwrt	ジャーナルライタ	△	WebOTX システムのジャーナル採取が不可。
olftplsn	OLF リスナ	△	OLF 通信が不可、WebOTX 印刷キットを利用したシステムに影響。
iioplsn	IIOP リスナ	○	IIOP 通信が不可、クライアントからの全ての呼び出しに支障。
TIMMSGSND	タイマデーモン	△	運用管理ツール、コマンドからの以下の運用操作が不可。 • トレース設定動的変更 • クライアントにメッセージ送信（クライアント管理ライブラリ使用時） • コンポーネントの動的変更 • サーバプロセスメッセージ通知（woadmcom NOTIFYMESSAGE）
wosystpp	WebOTX システム TPP	△	運用管理ツールから端末情報取得不可、端末一覧の表示に支障。
wotsrcd	Transaction サービス管理用プロセス	△	Transaction サービスが使用不可。
THTPPJAVA	サーバ AP プロセス	△	該当プロセスグループが行なう業務処理が不可。
namesv	名前サーバプロセス	○	オブジェクトリファレンスの取得が不可、クライアントからの全ての呼び出しに支障。
java(*)	JNDI サーバプロセス	○	JNDI 呼び出しが不可、EJB 呼び出しや、データソース取得に支障。
WatchServer	WatchServer プロセス	△	WatchServer を利用したクラスタ負荷分散時の縮退運転が不可。

○: 監視することを推奨します。

△: 異常終了が業務運用に致命的な影響を与えないため、監視するかどうかはシステム要件で判断してください。

(*): JNDI サーバはプロセス名は java であり他のプロセス名だけでは他の java プロセスと判別ができません。特有の引数を持つので引数も含めて監視してください。

```
java -Xss512k  -Dwebotx.jndi.installpath=..
```

５．　ログ管理

この章では WebOTX で出力するログについて記載しています。

5.1 Object Broker

ログディレクトリ：(ObjectBroker インストールディレクトリ)/log
ファイル名：
1-1:ObLog.log
1-2:InterfaceRepository.log
1-3:corbaloc.log
1-4:namesv.log
1-5:message.log
1-6:oadjinit.log
1-7:oadj.log

1-1:ObLog.log
(1)概要
・IR サーバおよび corbaloc サーバ以外の AP が出力します。
・出力内容は、LogingLevel の設定値により異なります。

LoggingLevel=ERROR のときは、エラーの情報。
LoggingLevel=WARNING のときは、エラー、警告の情報。
LoggingLevel=INFORMATION のときは、エラー、警告の情報に加え以下が出力されます。
- コードセットの情報
- コネクションが相手からクローズされたことの情報
- フックを使用している場合はその情報

LogingLevel 未設定の場合は、出力されません。
(2)ファイルサイズ
・ログファイルのサイズは、デフォルトでは無制限です。
設定の LogLimit にログファイルのサイズの上限を指定(単位は KB)することにより、
指定したサイズを越えた場合、_bak という拡張子を付け退避します。
(3)世代管理

・1 世代のみの管理となります。(_bak ファイルは無条件に上書きされる)

1-2:InterfaceRepository.log

(1)概要
・IR サーバ(irsv)が出力します。
・出力内容は ObLog.log と同じです。
(2)ファイルサイズ
・ログファイルのサイズは、デフォルトでは無制限です。
設定の LogLimit にログファイルのサイズの上限を指定(単位は KB)することにより、
指定したサイズを越えた場合、_bak という拡張子を付け退避します。
(3)世代管理
・1 世代のみの管理となります。(_bak ファイルは無条件に上書きされます)

1-3:corbaloc.log
(1)概要
・corbaloc サーバ(corbaloc)が出力します。
・出力内容は ObLog.log と同じです。
(2)ファイルサイズ
・ログファイルのサイズは、デフォルトでは無制限です。
設定の LogLimit にログファイルのサイズの上限を指定(単位は KB)することにより、
指定したサイズを越えた場合、_bak という拡張子を付け退避します。
(3)世代管理
・1 世代のみの管理となります。(_bak ファイルは無条件に上書きされる)

1-4:namesv.log

(1)概要
・名前サーバ(namesv)が出力します。
・起動時のエラー内容を出力します。
(2)ファイルサイズ
・ログファイルのサイズは、デフォルトでは無制限です。
設定の LogLimit にログファイルのサイズの上限を指定(単位は KB)することにより、
指定したサイズを越えた場合、_bak という拡張子を付け退避します。
(3)世代管理
・1 世代のみの管理となります。(_bak ファイルは無条件に上書きされる)

1-5:message.log

(1)概要

・デフォルトでは出力されません。

・設定で MessagingLog=on、MessageBodyLog=on を指定することにより、
通信毎に通信内容、通信相手、通信量を出力します。

(2)ファイルサイズ

・ログファイルのサイズは、デフォルトでは無制限です。
設定の LogLimit にログファイルのサイズの上限を指定(単位は KB)することにより、
指定したサイズを越えた場合、_bak という拡張子を付け退避します。

(3)世代管理

・1 世代のみの管理となります。(_bak ファイルは無条件に上書きされる)

1-6:oadjinit.log

(1)概要

・oadj 起動日時、オプションの設定値などを出力します。

(2)ファイルサイズ

・ファイルサイズに制限はありませんが、出力量は数 KB 程度です。

(3)世代管理

・oadj を起動する時点で、既存の内容を削除します。世代管理はおこなっていません。

1-7:oadj.log

(環境設定でファイル名を変更可能)

(1)概要

・oadj 稼働中のログを出力します。
デフォルトではエラーレベルのログを出力します。
環境設定でログ出力レベルを変更できます。
ログが出力されない場合は、ファイルを作成しません。

(2)ファイルサイズ

・100KB 以上になるとバックアップを作成します(環境設定でサイズを変更可能)。
バックアップファイル名は <filename>.bak です(デフォルトでは、oadj.log.bak です)。

(3)世代管理

・1 世代のみ残します。
(2) で新たにバックアップファイルを作成するときに古いファイルを削除します。

※oadj の環境設定の方法については、
オンラインマニュアルの
[運用ガイド]
- [Object Broker]
  - [1. 環境設定について]
    - [1.3.7. oadj に関する設定項目]
を参照願います。

5.2 WebOTX JNDI

ログディレクトリ：(WebOTX インストールディレクトリ)/jndisp/logs
ファイル名：
2-1.jndiserver.log
2-2.startup.err

2-1:jndiserver.log
(1)概要
・JNDI サービス処理のログです。
(2)ファイルサイズ
・ファイルサイズが一定以上になるとそれまでの内容をバックアップして新規 jndiserver.log に出力します。
バックアップファイル名は"jndiserver.log.<番号>"です。
(3)世代管理
・一世代だけ残します。タイミングは起動時です。
(2)でバックアップしたファイルが複数あっても"jndiserver.log"しか
保存されないので注意してください。
バックアップファイル名は"jndiserver.log.1"です。

2-2:startup.err
(1)概要
・JNDI サービスを起動するまでに出力された、シェル標準（エラー）出力です。
※起動してからのログが jndiserver.log になります)
(2)ファイルサイズ

・制限ありません。
(3)世代管理
・世代管理は行われません。常に更新されます。

(2)についてですが、JNDI サービスが起動するまでの内容が出力されるため、
JNDI サービスが起動した後は一切出力がされません。

5.3 WebOTX JMS

ログディレクトリ：(WebOTX インストールディレクトリ)/JMS/log
ファイル名：
3-1:jmsservice0.log　(0～4)
3-2:ospi.log

3-1:jmsservice0.log　(0～4)
(1)概要
・JMS サーバが出力するログです。
　現状、INFO レベルで運用されていると認識しています。
　クライアントの接続状況、異常発生時の情報が出力されます。
(2)ファイルサイズ
・ファイルサイズを指定することができます。
(3)世代管理
・指定したファイル名、数、サイズのファイルをローテーションして
　出力します。サイズ超過時、起動時に切り替えします。

　※設定
　　Windows:(WebOTX インストールディレクトリ)¥JMS¥jms_service.properties
　　UNIX:/etc/WebOTX/JMS/conf/jms_service.properties
　　に以下の設定を行ないます。
　　webotx.jms.service.logfile=
　　webotx.jms.service.logfile_size=
　　webotx.jms.service.logfile_number=
　　webotx.jms.service.loglevel=
　　webotx.jms.rmi.loglevel=
　　webotx.jms.selector.loglevel=

3-2:ospi.log

(1)概要

・JMS サーバの通信基盤となる Object Broker Java のログファイルです。
現状、WARN レベルで運用されていると認識しています。
通信上の異常発生時の情報が出力されます。

(2)ファイルサイズ

・ファイルサイズを指定することができます。

(3)世代管理

・指定したファイル名、サイズのファイルを 2 世代でローテーションして
出力します。サイズ超過時に切り替えます。

※設定

Windows:(WebOTX インストールディレクトリ)¥JMS¥conf¥jms.conf
UNIX:/etc/WebOTX/JMS/jms.conf
の javaargs
-Djp.co.nec.orb.LogLevel=
-Djp.co.nec.orb.MessageDumpLevel=
-Djp.co.nec.orb.LogFileName=
-Djp.co.nec.orb.LogFileSize=

5.4 WebOTX

ログディレクトリ：(WebOTX インストールディレクトリ)/Trnsv/logs
ファイル名：
4-1:webotx.log
4-2:wooperation.log
4-3:woadmsv.log
4-4:wodsbinderr.log

4-1:webotx.log

(1)概要

・運用管理サーバがイベントログに出力したメッセージを出力します。

(2)ファイルサイズ

・1MB を超える出力があると、.bak という拡張子をつけて退避します。

すでに.bak がある場合は上書きするため、最大サイズは 1MB(2 つ合わせて 2MB)です。

(3)世代管理

・一世代だけ残します(.bak)。
タイミングは 1MB を超える出力があったときです。

4-2:wooperation.log

(1)概要

・運用管理ツールから起動停止や更新処理を行ったときの履歴を出力します。

(2)ファイルサイズ

・1MB を超える出力があると、.bak という拡張子をつけて退避します。
すでに.bak がある場合は上書きするため、最大サイズは 1MB(2 つ合わせて 2MB)です。

(3)世代管理

・一世代だけ残します(.bak)。
タイミングは 1MB を超える出力があったとき、あるいはサービスを再起動したときです。

4-3:woadmsv.log

(1)概要

・運用管理サーバの内部トレースです。

(2)ファイルサイズ

・1MB を超える出力があると、.bak という拡張子をつけて退避します。
すでに.bak がある場合は上書きするため、最大サイズは 1MB(2 つ合わせて 2MB)です。

(3)世代管理

・一世代だけ残します(.bak)。
タイミングは 1MB を超える出力があったとき、あるいはサービスを再起動したときです。

4-4:wodsbinderr.log

(1)概要

・運用管理ツールから JNDI サーバへ登録、あるいは削除を行ったとき
にエラーになった場合に出力します。
(2)ファイルサイズ
・ファイルサイズは制限ありません。再起動で上書きします。
(3)世代管理
・上書きするため、世代管理は行っていません。

5.5 WebOTX WebCont

ログディレクトリ：(WebOTX インストールディレクトリ)/WebCont/logs
5-1:catalina.out
5-2:catalina_log.txt
5-3:localhost_log.txt
5-4:localhost_OTXJSP_log.txt
5-5:localhost_admin_log.txt
5-6:localhost_ejb_stubs_log.txt
5-7:localhost_ejbadmin_log.txt
5-8:localhost_examples_log.txt
5-9:localhost_manager_log.txt
5-10:localhost_access_log.txt
5-11:mod_jk.log

5-1:catalina.out

(1)概要
・標準出力、および標準エラー出力の内容
(2)ファイルサイズ
・ファイルサイズは制限ありません。ファイルは追記していきます。
(3)世代管理
・行いません。

5-2:catalina_log.txt
5-3:localhost_log.txt

(1)概要

・Web コンテナの動作内容
⑵ファイルサイズ
・ファイルサイズは制限ありません。
⑶世代管理
・運用管理コンソールの「ログデバッグ設定」のログ最大サイズ”に達したときバックアップを行います。
　世代数は“ログのバックアップ個数”で指定できます。

5-3:localhost_OTXJSP_log.txt
5-5:localhost_admin_log.txt
5-6:localhost_ejb_stubs_log.txt
5-7:localhost_ejbadmin_log.txt
5-8:localhost_examples_log.txt
5-9:localhost_manager_log.txt

⑴概要
・WebAP の処理内容
⑵ファイルサイズ
・ファイルサイズは制限ありません。
⑶世代管理
・運用管理コンソールの「ログデバッグ設定」のログ最大サイズ”に達したときバックアップを行います。
　世代数は“ログのバックアップ個数”で指定できます。

5-10:localhost_access_log.txt

⑴概要
・アクセスログ
⑵ファイルサイズ
・ファイルサイズは制限ありません。
⑶世代管理
・運用管理コンソールの「ログデバッグ設定」の“ログ最大サイズ”に達したときバックアップを行います。
　世代数は“ログのバックアップ個数”で指定できます。

5-11:mod_jk.log

(1)概要
・Web サーバのリダイレクタ（Web コンテナと通信を行う）部分のログ
(2)ファイルサイズ
・ファイルサイズは制限ありません。ファイルは追記していきます。
(3)世代管理
・行いません。

5.6 WebOTX WebAP JSP

ログディレクトリ：(WebAP JSP インストールディレクトリ)/Runtime/logs
6-1:error.log
6-2:trace.log
6-3:<IP アドレス>.log

6-1:error.log

(1)概要
・WebAP JSP 実行環境で発生したエラーの情報を出力します。
(2)ファイルサイズ
・運用管理／環境設定用 JSP ファイル（admin.jsp）の「環境管理」ページで
指定したエラーログ最大サイズ（既定値 1024KB）を超えると、ファイルの
先頭から上書きします。
エラーログ最大サイズに 0 を指定した場合は最大サイズは無制限になります。
(3)世代管理
・世代管理は行いません。

6-2:trace.log

(1)概要
・WebAP JSP 実行環境の処理のトレースを出力します。
(2)ファイルサイズ
・運用管理／環境設定用 JSP ファイル（admin.jsp）の「環境管理」ページで
指定したトレース最大サイズ（既定値 1024KB）を超えると、ファイルの

先頭から上書きします。
トレース最大サイズに 0 を指定した場合は最大サイズは無制限になります。
(3)世代管理
・世代管理は行いません。

6-3:<IP アドレス>.log

(1)概要
・サーバ AP のメソッドを実行した結果（アクセスログ）を出力します。
(2)ファイルサイズ
・運用管理／環境設定用 JSP ファイル（admin.jsp）の「環境管理」ページで
指定したアクセスログ最大サイズ（既定値 1024KB）を超えると、ファイル
の先頭から上書きします。
アクセスログ最大サイズに 0 を指定した場合は最大サイズは無制限になります。
(3)世代管理
・世代管理は行いません。

5.7 WebOTX WebServer

ログディレクトリ： <Web サーバインストールディレクトリ>¥logs 配下
7-1:access_log
7-2:error_log

7-1:access_log
(1)概要
・ブラウザから要求されたサーバへのリクエストを記録。
(2)ファイルサイズ
・ファイルサイズは制限ありません。ファイルは追記していきます。
(3)世代管理
・既定値では世代管理を行いません。ただし httpd.conf において、
TransferLog 設定で rotatelogs を設定することにより、世代管理が可能となります。
設定方法については、マニュアルの「運用ガイド」「Web サーバ」
「5.障害解析について」を参照してください。

7-2:error_log
(1)概要

・Web サーバのエラーログを記録。サーバに対して送信された
　リクエストのうち、失敗したリクエストも記録。

⑵ファイルサイズ

・ファイルサイズは制限ありません。ファイルは追記していきます。

⑶世代管理

・世代管理は行いません。