HITACHI
Inspire the Next

TPBroker Version 5
トランザクショナル分散オブジェクト基盤

TPBroker 運用ガイド

手引書

3021-3-J29

前書き

■ 対象製品

●適用 OS：Windows Server 2016，Windows Server 2019，Windows 10 x64

P-2964-AF64 Cosminexus TPBroker 05-24-01

●適用 OS：AIX V7.1，AIX V7.2

P-1M64-CF61 Cosminexus TPBroker 05-24-01

●適用 OS：Red Hat Enterprise Linux 7.1 (AMD/Intel 64)，Red Hat Enterprise Linux 8.1 (AMD/Intel 64)

P-9S64-AF61 Cosminexus TPBroker 05-25

これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

■ 発行

2020 年 3 月 3021-3-J29

■ 著作権

はじめに

このマニュアルは，トランザクショナル分散オブジェクト基盤 TPBroker の運用方法について説明したものです。

TPBroker では，ORB 機能，OTS 機能，および ADM 機能を使用してシステムの運用を行います。このマニュアルでは，これらの機能のうち，ORB 機能のトラブルシュート，ORB 機能の運用に必要な拡張機能，TPBroker とほかのプログラムプロダクトとの連携方法，およびバージョンアップ時の移行について説明しています。

■ 対象読者

システム管理者，システム設計者，またはオペレータで，TPBroker を使用して分散オブジェクトコンピューティング環境を運用する方を対象としています。

次の知識がある方を対象としています。

- C++または Java
- OTS
- CORBA
- TPBroker に関する基本的な知識
- Microsoft Cluster Service（Microsoft Cluster Service と連携させる場合）
- HA モニタ（HA モニタと連携させる場合）
- HACMP（HACMP と連携させる場合）
- JP1/ServerConductor/Deployment Manager（JP1/ServerConductor/Deployment Manager と連携させる場合）

■ 文法の記号

このマニュアルで使用する記号の意味を示します。

ただし，C++言語および Java 言語のインタフェースやコーディング例の説明は，それぞれの言語の文法規則に従います。これらの記号の意味は適用されません。

文法記述記号	意味
|	この記号で区切られた項目を選択できることを表します。 （例）admstop -f 1|2 これは，-f オプションに 1 か 2 のどちらかを指定できることを表します。

文法記述記号	意味
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを表します。 （例）{-t|-T トランザクショングローバル識別子} これは，-t と-T トランザクショングローバル識別子のうち，どちらかを指定することを表します。
…	この記号で示す直前の項目を繰り返し指定できることを表します。 （例）tslsrm -o ファイル名 [,ファイル名]… これは，-o オプションのファイル名を繰り返し指定できることを表します。
～	この記号のあとにユーザ指定値の属性を表します。
<< >>	ユーザが指定を省略したときの省略値を表します。
< >	ユーザ指定値の構文要素を表します。
(( ))	ユーザ指定値の指定範囲を表します。
<英数字>	英字と数字（0～9）で指定することを表します。
<文字列>	任意の文字の配列で指定することを表します。
<パス名>	記号名称，¥，/，および.（ピリオド）で指定することを表します。 ただし，パス名は使用する OS に依存します。

目次

1 TPBroker の運用の概要

この章では，このマニュアルで説明する TPBroker の運用の概要について説明します。

1.1　TPBroker の運用

ここでは，TPBroker を運用するときに使用する機能について説明します。

TPBroker を運用するときに使用する機能を次に示します。

- ORB のトラブルシュート機能
- ORB の拡張機能
- Microsoft Cluster Service との連携
- HA モニタとの連携
- HACMP との連携
- JP1/ServerConductor/Deployment Manager との連携

各機能の詳細は，2 章以降を参照してください。

なお，適用 OS やプログラムプロダクトによって，使用できる機能が異なります。プログラムプロダクトに関する詳細は，「1.2　Cosminexus TPBroker と TPBroker との相違点」を，適用 OS に関する詳細は，「1.3　サポートする機能」を参照してください。

1.2 Cosminexus TPBroker と TPBroker との相違点

TPBroker では，プログラムプロダクトによってサポートする機能が異なります。

Cosminexus TPBroker と TPBroker との，サポートする機能の相違点について次の表に示します。

表 1-1 Cosminexus TPBroker と TPBroker との相違点

機能		プログラムプロダクト	
		Cosminexus TPBroker	TPBroker
ORB	Java ORB	○	○
	C++ ORB	×	○
OTS	Java OTS	○	○
	C++ OTS	×	○
ADM		×	○

（凡例） ○：サポート　×：未サポート

1.3　サポートする機能

TPBroker では，適用 OS によって，サポートする機能が異なります。ここでは，機能ごとに，サポートする適用 OS について説明します。

1.3.1　ORB のトラブルシュート機能

ORB のトラブルシュート機能では，適用 OS によって，取得できるファイルが異なります。

サポートする機能と適用 OS の関係について次の表に示します。

表 1-2　サポートする機能と適用 OS の関係（ORB のトラブルシュート機能）

機能	適用 OS		
	AIX	Linux	Windows
モジュールトレースの取得	○	○	○
エラーログの取得	○	○	○
通信トレースの取得	○	○	○
メッセージログの取得	○	○	○
バーボースログの取得	○	○	×
スタックトレースの取得	×	×	○※
ネーミングサービス名前空間情報ログの取得	○	○	○

（凡例）　○：サポート　×：未サポート

注※

Windows（Visual Studio 2005）版では，未サポートです。

1.3.2　ORB の拡張機能

ORB の拡張機能を使用するには，環境変数やプロパティの設定が必要になります。ただし，適用 OS によって，使用できる環境変数やプロパティが異なります。詳細は，「3.2　環境変数とプロパティの一覧」を参照してください。

1.3.3　他プログラムプロダクトとの連携

他プログラムプロダクトとの連携では，適用 OS によって，連携できるシステムが異なります。

サポートする機能と適用 OS の関係について次の表に示します。

表 1-3　サポートする機能と適用 OS の関係（他プログラムプロダクトとの連携）

機能	適用 OS		
	AIX	Linux	Windows
Microsoft Cluster Service との連携	×	×	○
HA モニタとの連携	○	○	×
HACMP との連携	○	×	×
JP1/ServerConductor/Deployment Manager との連携	×	○	○

（凡例）　○：サポート　×：未サポート

2 ORB のトラブルシュート機能

この章では，ORB のトラブルシュート機能について説明します。

2.1　ORB のトラブルシュート機能で出力できるファイル

ここでは，ORB のトラブルシュート機能で出力できるファイルの概要について説明します。

次に示すファイルが出力できます。

- モジュールトレース
- エラーログ
- 通信トレース
- メッセージログ
- バーボースログ（UNIX）
- スタックトレース（32 ビット用 Windows（(Visual Studio 2005）版は除く））（C++ ORB）
- ネーミングサービス名前空間情報ログ

出力したファイルは，障害発生時にトラブルシュート情報として，保守員に送付する必要があります。定期的にバックアップを取得してください。なお，保守員とは，ご購入契約に基づくお問い合わせ窓口のことです。

参考事項

JavaVM のスレッドダンプを取得するためのコマンドがあります。詳細は，「2.11　hdumpns コマンドの使用方法（Cosminexus TPBroker）（Windows)」を参照してください。

なお，ORB のトラブルシュート機能で出力できるファイルのことを，トラブルシュートファイルと呼びます。

2.1.1 以降で，各ファイルの概要について説明します。

2.1.1　モジュールトレース（概要）

ここでは，モジュールトレースの概要について説明します。

モジュールトレースでは，TPBroker の内部メソッドの入り口および出口で，呼び出しメソッドや引数の情報などを取得しています。

モジュールトレースは TPBroker の保守用の資料であり，プロセスハングやプロセスダウン時に有効です。

2.1.2　エラーログ（概要）

ここでは，エラーログの概要について説明します。

エラーログでは，Java ORB の処理の中で，システム例外が発生した場合に，例外が発生することになった直接要因と，発生個所を特定できる情報を取得しています。また，それ以外に運用面で重要と思われる個所で情報を取得しています。エラーログは無条件に出力されます。

取得できるシステム例外を次に示します。

- COMM_FAILURE
- DATA_CONVERSION
- INTERNAL
- NO_IMPLEMENT
- NO_RESOURCES
- NO_RESPONSE
- OBJECT_NOT_EXIST
- TIMEOUT
- UNKNOWN

エラーログから，例外や発生事象の詳細要因を把握することができ，問題点を絞り込むことができます。ただし，TPBroker では正常に動作している場合でもさまざまな例外が発生しているため，エラーログに例外の発生情報が出力されていても，必ずしも障害が起きているとは限りません。

エラーログは TPBroker の保守用の資料であり，プロセスハングやプロセスダウン時に有効です 。

エラーログの動作は，定義ファイルを作成し，その定義ファイルに定義句を指定することによって変更できます。定義ファイルがない場合には，デフォルトの設定で動作します。定義ファイルは，環境変数 VBROKER_ADM に指定されたディレクトリに"HVIORB_DEF"の名称で作成してください。

2.1.3 通信トレース（概要）

ここでは，通信トレースの概要について説明します。

通信トレースでは，TPBroker から C++または Java の通信管理プログラムに制御を渡すインタフェース部分で，次に示す情報を取得しています。

- 通信管理プログラムのシステムコール（C++）または API（Java）の引数，および結果
- GIOP または TPBroker 独自プロトコルに従った電文

通信トレースは TPBroker の保守用の資料であり，プロセスハングやプロセスダウン時，および通信障害時に有効です 。

2.1.4　メッセージログ（概要）

ここでは，メッセージログの概要について説明します。

プロセスの起動および停止時や TIMEOUT などの例外発生時のメッセージは，次に示す方法で取得しています。

- メッセージログ（Java ORB 限定）
- イベントビューアのアプリケーションログ（Windows）
- syslog（UNIX）

Java ORB を使用している場合，イベントビューアのアプリケーションログや syslog などの OS の機能だけでなく，TPBroker が出力するメッセージログを使用して，メッセージを取得しています。

メッセージの内容については，「8.　メッセージ」を参照してください。

メッセージログの動作は，定義ファイルを作成し，その定義ファイルに定義句を指定することによって変更できます。定義ファイルがない場合には，デフォルトの設定で動作します。定義ファイルは，環境変数 VBROKER_ADM に指定されたディレクトリに"HVIORB_DEF"の名称で作成してください。

2.1.5　バーボースログ（概要）

ここでは，バーボースログの概要について説明します。

バーボースログとは，バーボースモードで出力されるログのことをいいます。バーボースモードとは，osagent の処理に関する情報を出力する状態のことです。

-v オプションで osagent を起動すると，バーボースモードになり，標準出力上にバーボースログが出力されます。

-g オプションで osagent を起動すると，自動的に hvmgtee コマンドが起動され，バーボースログをファイルに出力することができます。hvmgtee コマンドは，UNIX の FIFO を利用し，標準出力に出力される osagent のバーボースログ，およびエラー出力を受け取り，ファイルや標準出力への出力を制御する機能を持ちます。デフォルトでは，バーボースログをファイルに出力します。hvmgtee コマンドについては，「(1)　hvmgtee コマンドの起動」，および「(2)　hvmgtee コマンドの動作変更」を参照してください。

バーボースログは TPBroker の保守用の資料であり，プロセスハングやプロセスダウン時に有効です。UNIX だけで取得できます。

(1)　hvmgtee コマンドの起動

hvmgtee コマンドを起動するには，次のコマンドを実行します。

```
osagent -g
```

hvmgtee コマンドを起動することで，バーボースログがファイルに出力され，標準出力への出力は抑止されます。

ただし，バーボースログを標準出力に出力するか，ファイルに出力するかどうかの動作を定義ファイルで変更できます。詳細は「(2)　hvmgtee コマンドの動作変更」を参照してください。

また，hvmgtee コマンドでエラーが発生した場合には，エラーメッセージが表示されます。エラーメッセージについては，「8.5　hvmgtee コマンドのエラーメッセージ」を参照してください。

(2)　hvmgtee コマンドの動作変更

hvmgtee コマンドの動作は，定義ファイルに定義句を設定することで変更できます。定義句を設定しない場合，デフォルトで動作します。定義ファイルは，環境変数 VBROKER_ADM に指定されたディレクトリに，"HVMGTEE_DEF"のファイル名で作成します。

hvmgtee コマンドの起動時，および稼働中の一定インターバルごと（デフォルトは 180 秒）に定義ファイルが読み込まれるため，osagent を再起動しないで，設定を変更できます。ただし，定義句の中には，osagent の起動時にだけ有効となるものもあるので設定時には注意してください。定義ファイルが読み込まれる間隔も定義句で変更できます。

hvmgtee コマンドの動作変更の詳細は，「2.9.5　バーボースログ（定義句）」を参照してください。

2.1.6　スタックトレース（概要）

ここでは，スタックトレースの概要について説明します。

TPBroker の C++ ORB アプリケーションの処理で例外が発生した場合に，例外が発生することになった直接要因と，発生個所を特定できる情報を取得しています。

取得できる例外を次に示します。

- C言語構造化例外が発生した場合
- キャッチできない例外が発生した場合

スタックトレースによって，例外発生事象の詳細要因を把握することができ，問題点の絞り込みに役立ちます。

スタックトレースは TPBroker の保守用の資料であり，アプリケーション処理で例外が発生したときに有効です。Windows（(Visual Studio 2005）版は除く）だけで取得できます。

2.1.7　ネーミングサービス名前空間情報ログ（概要）

ここでは，ネーミングサービス名前空間情報ログの概要について説明します。

ネーミングサービスに登録される名前空間情報を取得しています。ログファイルは，ネーミングサービスのプロセス単位に生成されます。

ネーミングサービス名前空間情報ログは TPBroker の保守用の資料であり，ネーミングサービスに関連した障害が発生したときに有効です。

2.2　出力ディレクトリとファイル名

この節では，出力ディレクトリとファイル名について説明します。

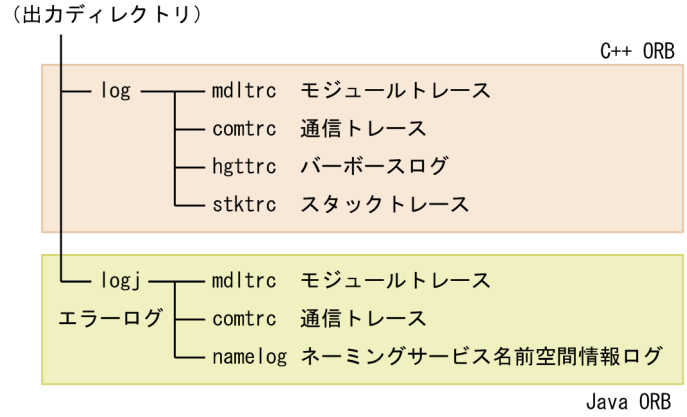
出力ディレクトリは次のとおりです。

- Java ORB の場合
  ${VBROKER_ADM}/../logj（%VBROKER_ADM%¥..¥logj）
- C++ ORB の場合
  ${VBROKER_ADM}/../log（%VBROKER_ADM%¥..¥log）

出力ディレクトリの下には，各トラブルシュートファイルを格納するためのディレクトリがあります。

出力ディレクトリの構成を次の図に示します。

図 2-1　出力ディレクトリの構成



注意事項

環境変数 HVI_TRACEPATH を設定すると，その環境変数に指定したディレクトリが出力ディレクトリになります。詳細は，「2.7.2(1)　HVI_TRACEPATH」を参照してください。

ご使用の OS が Windows の場合に，%ProgramFiles% 下の書き込み制限のあるディレクトリが出力ディレクトリであると，%LocalAppData%¥VirtualStore¥Program Files ディレクトリの下に出力されます。

トラブルシュートファイルが出力されない場合

トラブルシュートファイルが出力されない場合があります。C++ ORB のときは，標準エラー出力にエラーメッセージが出力されます。エラーメッセージについては，「8.4　トレース情報取得ができない場合に出力されるメッセージ」を参照してください。

TPBroker では，プロセス起動後，最初の ORB.init 実行時に出力ディレクトリがないと，ファイル作成，読み取り，および書き込み権限のあるディレクトリを自動的に作成します。

また，モジュールトレース，通信トレース，スタックトレース，およびバーボースログが出力されるときに，出力ディレクトリ下にそれぞれファイルを格納するディレクトリを作成します。

そのため，トラブルシュートファイルが出力されない場合，出力ディレクトリの作成に失敗しているケースが考えられます。

次のことを確認してください。

- 作成するディレクトリの親ディレクトリがあるか
- 出力ディレクトリの権限が制限されていないか
- 作成するディレクトリの絶対パスの長さがシステムの上限内か

  環境変数 HVI_TRACEPATH，または環境変数 VBROKER_ADM に設定するパスの長さを，210 - (<実行形式ファイル名称の長さ（Java ORB の場合 10）>)バイト以下にすることをお勧めします。

2.2.1 モジュールトレース（出力ディレクトリとファイル名）

ここでは，モジュールトレースの出力ディレクトリ，ファイル名，および出力タイミングについて説明します。

TPBroker では，稼働しているプロセス上にバッファを確保し，そこにモジュールトレース情報を記録します。バッファに記録されたモジュールトレース情報は，プロセスごとにあるタイミングでファイルに出力されます。

モジュールトレースは次の表のとおりに作成されます。

表 2-1 モジュールトレース（出力ディレクトリとファイル名の一覧）

項目名	ORB の種類	
	Java ORB	C++ ORB
出力ディレクトリ	${VBROKER_ADM}/../logj/mdltrc （%VBROKER_ADM%¥..¥logj¥mdltrc）	${VBROKER_ADM}/../log/mdltrc （%VBROKER_ADM%¥..¥log¥mdltrc）
ファイル名	<付加名称※1>.<プロセス ID>.<時刻情報※2>.mdl.dat • 付加名称 デーモンまたはコマンドの場合だけ，付加されます。 • 時刻情報 環境変数 HVI_TRACEFILENAME_TOD で付加するかどうかを設定できます。	osagent の場合 osagent.<プロセス ID>.<時刻情報※2>.mdl.dat アプリケーションの場合 <実行形式ファイル名称>.<プロセス ID>.<時刻情報※2>.mdl.dat • 時刻情報 環境変数 HVI_TRACEFILENAME_TOD で付加するかどうかを設定できます。 また，UNIX では，次に示すファイル名の場合があります。 unknown.<プロセス ID>.mdl.dat 次の場合だけです。 • ORB_init()の引数 argc に"0"を指定する • ORB_init()の引数 argv に"NULL"を指定する • argv[0]に"NULL"を指定する

項目名	ORB の種類	
	Java ORB	C++ ORB
		• argv[0]にハイフン（"-"）を指定する
作成タイミング	Windows • ネーミングサービスを含むデーモン，またはコマンドの場合 プロセス起動時 • アプリケーションの場合 プロセスで最初に ORB.init()を発行したとき UNIX プロセス終了時	Windows プロセス起動時 UNIX プロセス終了時

注※1

付加名称を次の表に示します。

デーモンまたはコマンド	付加名称
ネーミングサービス	nameserv
osfind	osfind
ゲートキーパー（Windows）	gatekeeper
irep	irep

出力ディレクトリの絶対パスの長さが，210 - (<実行形式ファイル名称の長さ（Java ORB の場合 10）>)を超えると，付加名称が付加されない場合やファイル出力に失敗する場合があります。

注※2

時刻情報は，ファイルが作成された時刻です。YYYYMMDDHHMMSS 形式で付加されます。

モジュールトレース情報は，バッファの上限サイズまで出力されたあと，ラップアラウンドします。上限サイズは，環境変数 HVI_MTRENTRYCOUNT で変更できます。

ファイルは，環境変数 HVI_MTRFILECOUNT で指定した値（例えば n 個とする）を最大数として生成され，更新時刻の古いものから順に削除します。ただし，以下の場合にファイルが n 個を超えることがあります。

- 削除しようとしたファイルが使用中の場合（使用者がいなくなるまでそのファイルが存在する）
- n 個を超えるプロセスで表 2-1 に示す「作成タイミング」が重なる場合

2.2.2 エラーログ（出力ディレクトリとファイル名）

ここでは，エラーログの出力ディレクトリとファイル名について説明します。

エラーログは次の表のとおりに作成されます。

表 2-2　エラーログ（出力ディレクトリとファイル名の一覧）

項目名	説明
出力ディレクトリ	${VBROKER_ADM}/../logj （%VBROKER_ADM%¥..¥logj）
ファイル名	• hvilogj1.txt • hvilogj2.txt

エラーログの出力ファイルは，出力ディレクトリごとに二つ作成し，スワップさせながら使用します。一つのファイルの上限サイズは定義句 HVI_ORBLOG_SIZE で指定できます。最後にエラーログを出力したあとのエラーログの出力ファイルのサイズが定義句 HVI_ORBLOG_SIZE の指定値を超えた場合にスワップします。

2.2.3　通信トレース（出力ディレクトリとファイル名）

ここでは，通信トレースの出力ディレクトリ，ファイル名，および出力タイミングについて説明します。

通信トレースのファイルはプロセスごとに次の表のとおりに作成されます。

表 2-3　通信トレース（出力ディレクトリとファイル名の一覧）

項目名	ORB の種類	
	Java ORB	C++ ORB
出力ディレクトリ	${VBROKER_ADM}/../logj/comtrc （%VBROKER_ADM%¥..¥logj¥comtrc）	${VBROKER_ADM}/../log/comtrc （%VBROKER_ADM%¥..¥log¥comtrc）
ファイル名	<付加名称※1>.<プロセス ID>.<時刻情報※2>.comt.dat • 付加名称 デーモンまたはコマンドの場合だけ，付加されます。 • 時刻情報 環境変数 HVI_TRACEFILENAME_TOD で付加するかどうかを設定できます。	osagent の場合 osagent.<プロセス ID>.<時刻情報※2>.comt.dat アプリケーションの場合 <実行形式ファイル名称>.<プロセス ID>.<時刻情報※2>.comt.dat • 時刻情報 環境変数 HVI_TRACEFILENAME_TOD で付加するかどうかを設定できます。 また，UNIX では，次に示すファイル名になる場合があります。 unknown.<プロセス ID>.comt.dat 次の場合だけです。 • ORB_init()の引数 argc に"0"を指定する • ORB_init()の引数 argv に"NULL"を指定する • argv[0]に"NULL"を指定する • argv[0]にハイフン（"-"）を指定する

項目名	ORB の種類	
	Java ORB	C++ ORB
作成タイミング	Windows 　プロセスで最初に通信を行った時 UNIX 　プロセスで最初に通信を行った時	Windows 　プロセス起動時 UNIX • osagent の場合 　プロセス起動時 • アプリケーションの場合 　プロセスで最初に CORBA::ORB_init()を発行したとき

注※1

付加名称を次の表に示します。

デーモンまたはコマンド	付加名称
ネーミングサービス	nameserv
osfind	osfind
ゲートキーパー（Windows）	gatekeeper
irep	irep

出力ディレクトリの絶対パスの長さが，210 - (<実行形式ファイル名称の長さ（Java ORB の場合 1）>)を超えると，上記名称が付加されない場合や出力に失敗する場合があります。

注※2

時刻情報は，ファイルが作成された時刻です。YYYYMMDDHHMMSS 形式で付加されます。

通信トレース情報はファイルの上限サイズまで出力されたあと，ラップアラウンドします。上限サイズは，環境変数 HVI_COMTENTRYCOUNT で変更できます。

ファイルは，環境変数 HVI_COMTFILECOUNT で指定した値（例えば n 個とする）を上限として生成され，更新時刻の古いものから順に削除します。ただし，以下の場合にファイルが n 個を超えることがあります。

- 削除しようとしたファイルが使用中の場合(使用者がいなくなるまでそのファイルが存在する)
- n 個を超えるプロセスで表 2-3 に示す「作成タイミング」が重なる場合

2.2.4　メッセージログ（出力ディレクトリとファイル名）

ここでは，メッセージログの出力ディレクトリとファイル名について説明します。

メッセージログは次の表のとおりに作成されます。

表 2-4　メッセージログ（出力ディレクトリとファイル名の一覧）

項目名	説明
出力ディレクトリ	${VBROKER_ADM}/../logj （%VBROKER_ADM%¥..¥logj）
ファイル名	• hvimsgj1.txt • hvimsgj2.txt

メッセージログの出力ファイルは，出力ディレクトリごとに二つ作成し，スワップさせながら使用します。一つのファイルの上限サイズは定義句 HVI_ORBMSGLOG_SIZE で指定できます。メッセージが出力された場合，<ファイルへの出力済みサイズ> ＋ <出力しようとするメッセージサイズ>が定義句 HVI_ORBMSGLOG_SIZE の指定値を超えるときにスワップします。

2.2.5　バーボースログ（出力ディレクトリとファイル名）

ここでは，バーボースログの出力ディレクトリとファイル名について説明します。

バーボースログの出力ファイルは，osagent のプロセスごとに hvmgtee コマンドを使用して作成されます。出力ディレクトリ，ファイル名を次の表に示します。

表 2-5　バーボースログ（出力ディレクトリとファイル名の一覧）

項目名	説明
出力ディレクトリ	${VBROKER_ADM}/../log/hgttrc
ファイル名	• osagent.<プロセス ID>.hgt1.log • osagent.<プロセス ID>.hgt2.log

バーボースログの出力ファイルはプロセスごとに二つ作成され，スワップさせながら使用します。ファイルの上限サイズは，定義句 HVI_GTEE_FILESIZE で指定できます。最後にバーボースログを出力したあとの出力ファイルのサイズが，定義句 HVI_GTEE_FILESIZE の指定値を超えた場合，または，osagent に対して limit などで制限したファイルサイズを超えた場合にスワップします。

osagent が正常終了した場合，または SIGINT，もしくは SIGTERM を受信した場合，hvmgtee コマンドは正常終了し，終了前に出力されていたログ（スワップしたファイルも含む）は，compress コマンドを使用して圧縮されます。圧縮されたファイルは，ファイル名が<hvmgtee コマンドが出力したファイル名>.Z に変更されます。また，圧縮後のファイル名と同じファイルが同一ディレクトリに存在した場合はそのファイルを上書きします。定義句 HVI_GTEE_LOGPRESS で，ファイルを圧縮するかどうかを指定できます。

バーボースログのファイル数は，定義句 HVI_GTEE_FILECOUNT で上限値を指定できます。osagent に-g オプションを指定して起動したときに，<既存のログファイル※1 数> ＋ 2 が上限値を超えていると，上限値を超えた数だけ，既存のログファイルを更新日付の古いものから削除します。

なお，以下の場合は，上限値を超えるログファイルが出力されることがあります。

- osagent に-g オプションを指定して複数同時に起動する場合
- osagent に-g オプションを指定して起動したときに，起動した osagent と同じプロセス ID のログファイル（圧縮されたファイルのみ）が既に存在している場合

注※1

起動する osagent と同じプロセス ID のログファイル（圧縮されたファイルも含む）は，既存のログファイルに含まれません。

定義句については，「2.9.5　バーボースログ（定義句）」を参照してください。

2.2.6　スタックトレース（出力ディレクトリとファイル名）

ここでは，スタックトレースの出力ディレクトリ，ファイル名，および出力タイミングについて説明します。

スタックトレースは次の表のとおりに作成されます。

表 2-6　スタックトレース（出力ディレクトリとファイル名の一覧）

項目名	説明
出力ディレクトリ	%VBROKER_ADM%¥..¥log¥stktrc
ファイル名	visstk1.log ・ ・ ・ visstk<XXX>.log （XXX：ファイルカウント）
作成タイミング	例外発生時

ファイルは，環境変数 HVI_STKFILECOUNT で指定した値（例えば n 個とする）を上限として生成され，スワップさせながら使用します。

2.2.7　ネーミングサービス名前空間情報ログ（出力ディレクトリとファイル名）

ここでは，ネーミングサービス名前空間情報ログの出力ディレクトリ，ファイル名，および作成タイミングについて説明します。

ネーミングサービス名前空間情報ログはプロセスごとに次の表のとおりに作成されます。

表 2-7　ネーミングサービス名前空間情報ログ（出力ディレクトリとファイル名の一覧）

項目名	説明
出力ディレクトリ	${VBROKER_ADM}/../logj/namelog （%VBROKER_ADM%¥..¥logj¥namelog）
ファイル名	• namelog1.<プロセス ID>.<時刻情報※>.txt • namelog2.<プロセス ID>.<時刻情報※>.txt 時刻情報は，環境変数 HVI_TRACEFILENAME_TOD で付加するかどうかを設定できます。
作成タイミング	ネーミングサービスに対して，最初にコンテキストやオブジェクトの登録処理を行ったとき

注※

時刻情報は，ファイルが作成された時刻です。YYYYMMDDHHMMSS 形式で付加されます。

出力ディレクトリの絶対パスの長さが 210 を超えると，出力に失敗する場合がありますので注意してください。

ファイルはネーミングサービスのプロセスごとに最大で二つ作成し，スワップさせながら使用します。一つのファイルの上限サイズは環境変数 HVI_NAMELOGFILESIZE によって指定できます。ログを出力する場合にログファイルのサイズがこの設定値を超えているときには，もう一つのファイルにスワップします。

ファイルは，環境変数 HVI_NAMELOGFILECOUNT で指定した値（例えば n 個とする）のプロセス数分，出力ディレクトリ下に保持されます。ネーミングサービスでは一つのプロセス当たりに，最大で二つのログファイルを作成できますので，最大で n×2 個のファイルを保持します。新規にログファイルを作成する場合に指定値を超えているときには，更新時刻の古いものから順に削除します。ただし，削除しようとしたファイルが使用中の場合は，使用者がいなくなるまでファイルは存在します。そのため，n×2 個以上のファイルが存在することもあります。

2.3　システム例外のマイナーコード

この節では，システム例外のマイナーコードについて説明します。

TPBroker で発生するシステム例外に設定されるマイナーコードを次に示します。

- 一意な値のマイナーコード
  発生個所，要因ごとに一意な値が割り当てられています。そのため，このマイナーコードの値を基にして，例外が発生することになった要因と発生個所を特定することができ，問題の絞り込みに有効です。
- 標準のマイナーコード
  OMG が規定しています。

環境変数で一意な値のマイナーコードだけか，または両方のマイナーコードを使用するかどうかを設定できます。詳細は，「2.8.7　システム例外のマイナーコード（環境変数）」を参照してください。

2.4　環境設定

この節では，ORB トラブルシュート機能の環境設定について説明します。

環境設定の流れを次の図で示します。詳細は 2.5 以降を参照してください。

図 2-2　環境設定の流れ

設定できる値

TPBroker の ORB トラブルシュート機能では，環境変数，定義句，およびプロパティで詳細な設定ができますが，項目によって設定できる値が異なります。事前に次の表で確認してください。

表 2-8　項目ごとの設定方法

項目	設定できる値		
	環境変数	定義句	プロパティ
モジュールトレース	○	×	○
エラーログ	×	○	×
通信トレース	○	×	○
メッセージログ	○※	○	○※
バーボースログ	×	○	×
スタックトレース	○	×	○
ネーミングサービス名前空間情報ログ	○	×	○
マイナーコード	○	×	○

（凡例）　○：設定できる　×：設定できない

注※　イベントビューアのアプリケーションログ（Windows），および syslog（UNIX）に関することだけが設定できます。

設定した値の優先順位

環境変数とプロパティは同時に設定できます。同時に設定した場合の優先順位について次に示します。

トレース共通環境変数 ＜ 環境変数 ＜ プロパティ

定義句を設定する場合の優先順位を次に示します。

トレース共通環境変数 ＜ 定義句

2.5　TPBroker の基本的な設定

この節では，TPBroker の基本的な設定について説明します。

ORB のトラブルシュート機能を使用する前に，TPBroker の基本的な設定については完了しておいてください。

なお，環境変数 VBROKER_ADM に設定する値が，デフォルトのトラブルシュートファイルの出力ディレクトリになります。環境変数 VBROKER_ADM については，マニュアル「TPBroker ユーザーズガイド」を参照してください。

2.6　出力ディレクトリの作成

この節では，出力ディレクトリの作成について説明します。

ユーザが出力ディレクトリを指定する場合，ファイルを出力するディレクトリを作成します。ディレクトリを作成するときの注意については，「2.7.2(1)　HVI_TRACEPATH」を参照してください。作成したディレクトリのパスを環境変数 HVI_TRACEPATH に設定することで，出力ディレクトリを指定できます。

デフォルトの出力ディレクトリを使用する場合には，この手順は不要です。

2.7　トレース共通環境変数の設定

この節では，トレース共通環境変数の設定について説明します。

トレース共通環境変数では，すべてのトラブルシュートファイルに共通の設定をします。必要に応じて設定してください。

2.7.1　トレース共通環境変数の一覧

トレース共通環境変数を次の表に示します。

表 2-9　トレース共通環境変数の一覧

項番	環境変数	説明	備考
1	HVI_TRACEPATH	次に示すトラブルシュートファイルの出力ディレクトリを設定します。 • モジュールトレース • 通信トレース • エラーログ • メッセージログ • スタックトレース • バーボースログ • ネーミングサービス名前空間情報ログ	－
2	HVI_TRACEFILENAME_TOD	モジュールトレース，通信トレース，およびネーミングサービス名前空間情報ログの出力ファイル名に時刻情報を付加するかどうかを指定します。	－
3	HVI_MAPFILEINIT	出力ディレクトリの領域の初期化処理を実施するかどうかを指定します。	UNIX 限定

（凡例）　－：説明なし

2.7.2　トレース共通環境変数

トレース共通環境変数の詳細について説明します。

(1)　HVI_TRACEPATH

(Windows)　～<1 バイト～210 バイト以下のパス名>

(UNIX)　～<1 バイト～980 バイト以下のパス名>

次に示すトラブルシュートファイルの出力先を絶対パスで設定します。

- モジュールトレース
- 通信トレース
- エラーログ
- メッセージログ
- スタックトレース
- バーボースログ
- ネーミングサービス名前空間情報ログ

各トラブルシュートファイルは，Java ORB では${VBROKER_ADM}/../logj（%VBROKER_ADM%¥..¥logj），C++ ORB では${VBROKER_ADM}/../log（%VBROKER_ADM%¥..¥log）に出力されますが，この環境変数を指定することで，環境変数 VBROKER_ADM で指定されるディレクトリと異なるディレクトリに出力先を変更することができます。

なお，設定するディレクトリは，次の条件を満たしている必要があります。

- 指定したディレクトリのパスの長さが，Windows では 1～210 バイト，UNIX では 1～980 バイトの範囲内である。
- 指定されたディレクトリが存在する（C++ ORB だけ）。
  ディレクトリの作成例を示します。
  - Windows
    mkdir %HVI_TRACEPATH%
  - UNIX
    mkdir $HVI_TRACEPATH
- 次に示すディレクトリにファイル作成，読み取り，および書き込み権限がある。
  - Windows
    %HVI_TRACEPATH%
    %HVI_TRACEPATH%¥comtrc
    %HVI_TRACEPATH%¥stktrc
    %HVI_TRACEPATH%¥mdltrc
    %HVI_TRACEPATH%¥namelog
  - UNIX
    $HVI_TRACEPATH
    $HVI_TRACEPATH/comtrc
    $HVI_TRACEPATH/mdltrc
    $HVI_TRACEPATH/hgttrc
    $HVI_TRACEPATH/namelog

注意事項

- 環境変数 HVI_TRACEPATH が設定されていない，または指定したパスの長さが Windows では 1～210 バイト，UNIX では 1～980 バイトの範囲にない場合，Java ORB では${VBROKER_ADM}/../logj（%VBROKER_ADM%¥..¥logj），C++ ORB では${VBROKER_ADM}/../log（%VBROKER_ADM%¥..¥log）を出力ディレクトリに設定します。

  ただし，設定した出力ディレクトリのパスの長さが範囲外の場合は，トラブルシュートファイルは出力されません。
- 環境変数 HVI_TRACEPATH で指定されたディレクトリが同じ環境で，Java ORB および C++ ORB のアプリケーションを混在して使用すると，トラブルシュートファイルがすべて同じディレクトリに出力されます。Java ORB のファイル数の上限はディレクトリ単位であることから，Java ORB および C++ ORB で出力されるファイルを合わせた数が上限となります。

  そのため，Java ORB のアプリケーション実行時にファイル数の上限を超えると，C++ ORB のトラブルシュートファイルが古いファイルとして削除されるおそれがあります。

(2) HVI_TRACEFILENAME_TOD

(Java ORB) ～<文字列> { true|false|Y|N } <<true>>

(C++ ORB) ～<文字列> { true|false } <<false>>

通信トレース，モジュールトレース，およびネーミングサービス名前空間情報ログの出力ファイル名に時刻情報を付加するかどうかを設定します。付加する時刻情報は，トラブルシュートファイルを作成した時刻（YYYYMMDDHHMMSS 形式）です。

- true|Y

  ファイル名に時刻情報を付加します。ファイル名は次のとおりになります。
  - 「<付加名称>.<プロセス ID>.<時刻情報>.<xxx>.dat」(Java ORB)
  - 「<実行形式ファイル名称>.<プロセス ID>.<時刻情報>.<xxx>.dat」(C++ ORB)
  - 「namelog1.<プロセス ID>.<時刻情報>.txt」または「namelog2.<プロセス ID>.<時刻情報>.txt」(ネーミングサービス名前空間情報ログ)
- false|N

  ファイル名に時刻情報を付加しません。ファイル名は次のとおりになります。
  - 「<付加名称>.<プロセス ID>.<xxx>.dat」(Java ORB)
  - 「<実行形式ファイル名称>.<プロセス ID>.<xxx>.dat」(C++ ORB)
  - 「namelog1.<プロセス ID>.txt」または「namelog2.<プロセス ID>.txt」(ネーミングサービス名前空間情報ログ)

上記のフォーマットで，xxx は，モジュールトレースの場合は mdl，通信トレースの場合は comt です。

また，トラブルシュートファイルの削除管理対象は，設定されたフォーマットの形式が"true"の場合は時刻情報を付加した形式のファイル，"false"の場合は時刻情報を付加していない形式のファイルに対して有効になります。

Java ORB では，"Y"または"N"で設定することもできます。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

(3) HVI_MAPFILEINIT（UNIX だけ）

(UNIX) ～<文字列> { true|false } <<false>>

出力ディレクトリの領域の初期化処理を実施するかどうかを指定します。

- true
  ファイル作成時の初期化処理を実施します。
- false
  ファイル作成時の初期化処理を実施しません。

ファイル作成時の初期化処理を実施しない（"false"を設定した）場合，トラブルシュートファイルへの情報出力時にディスク不足などが発生すると，アプリケーション実行時の予期しないタイミングでプロセスが強制終了されます。

"true"を指定することで，アプリケーションが該当プロセスで初回に発行する CORBA::ORB_init()または org.omg.CORBA.ORB.init()で，ファイル作成時の初期処理を実施し，この時点でディスク不足などが発生するとプロセスが強制終了されるようになります。そのため，予期しないタイミングでプロセスが強制終了することを防止できます。

注意事項

- トレース取得エントリ数によっては，初回に発行する CORBA::ORB_init()または org.omg.CORBA.ORB.init()のレスポンスが悪くなる場合があります。
- この機能によって，アプリケーションが強制終了した場合は，トラブルシュートファイルを取得するために必要なディスクの空き容量を確保してください。
  ディスクの空き容量については，「2.12.2　ディスク占有量」を参照してください。
- 指定範囲外の値を設定した場合は，デフォルト値が設定されます。

2.8　環境変数の設定

この節では，環境変数の設定について説明します。

環境変数では，トラブルシュートファイルごとに個別に設定します。必要に応じて設定してください。

2.8.1　環境変数の一覧

環境変数の一覧を次の表に示します。環境変数の詳細は，2.8.2 以降で説明します。

表 2-10　環境変数の一覧（ORB のトラブルシュート機能）

項番	環境変数	説明	サポート言語		備考
			Java ORB	C++ ORB	
1	HVI_MDLTRACE	モジュールトレースを取得するかどうかを設定します。	○	○	−
2	HVI_MTRENTRYCOUNT	モジュールトレースの出力ファイルのエントリ数の上限値を設定します。	○	○	−
3	HVI_MTRFILECOUNT	モジュールトレースの出力ディレクトリに作成するファイル数の上限値を設定します。	○	○	−
4	HVI_COMTRACE	通信トレースを取得するかどうかを設定します。	○	○	−
5	HVI_COMTENTRYCOUNT	通信トレースの出力ファイルのエントリ数の上限値を設定します。	○	○	−
6	HVI_COMTFILECOUNT	通信トレースの出力ディレクトリに作成するファイル数の上限値を設定します。	○	○	−
7	HVI_COMTTELEGSIZE	取得する通信電文のサイズを設定します。	○	○	−
8	HVI_MSGLOG_OUTPUT	メッセージログの出力先を設定します。	○	×	osagent にも設定可能
9	HVI_MSGLOG_CONSOLE	syslog への出力に失敗した場合に，メッセージを強制的にコンソールへ出力するかどうかを設定します。	○	×	UNIX 限定
10	HVI_MSGLOG_LEVEL	出力するメッセージのメッセージレベルを設定します。	○	○	−
11	HVI_MSGLOG_SECURITY	セキュリティにかかわる情報のメッセージ出力を抑止するかどうかを設定します。	○	○	−
12	HVI_MSGLOG_NO_PERMISSION	出力を抑止したいメッセージのメッセージ ID を設定します。	×	×	osagent 限定

項番	環境変数	説明	サポート言語		備考
			Java ORB	C++ ORB	
13	HVI_STKTRACE	スタックトレースを取得するかどうかを設定します。	×	○	Windows(C++ORB)限定
14	HVI_STKFILECOUNT	スタックトレースの出力ファイル数の最大値を設定します。	×	○	Windows(C++ORB)限定
15	HVI_NAMELOGOUTPUT	ネーミングサービス名前空間情報ログを取得するかどうかを設定します。	○	○	ネーミングサービス限定
16	HVI_NAMELOGFILESIZE	ネーミングサービス名前空間情報ログの出力ファイルのサイズを設定します。	○	○	ネーミングサービス限定
17	HVI_NAMELOGFILECOUNT	ネーミングサービス名前空間情報ログの出力ディレクトリ下に保持するログのプロセス数を設定します。	○	×	ネーミングサービス限定
18	HVI_NAMELOGSNAPSHOTINTERVAL	出力ファイルにネーミングサービス名前空間情報ログのスナップショットを出力する間隔を設定します。	○	○	ネーミングサービス限定
19	HVI_OMGVMCID	システム例外発生時に OMG が規定するマイナーコードを使用するかどうかを設定します。	×	○	－

（凡例） ○：サポート　×：未サポート　－：説明なし

2.8.2　モジュールトレース（環境変数）

モジュールトレースに関する環境変数の詳細について説明します。

(1)　HVI_MDLTRACE

～<文字列> { true|false|Y|N } <<true>>

モジュールトレースを取得するかどうかを設定します。

- true|Y
  モジュールトレースを取得します。
- false|N
  モジュールトレースを取得しません。

設定値"Y"および"N"は，下位バージョンの ORB との共通環境を実現するために使用できますが，設定値"true"および"false"の使用を推奨します。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，C++ ORB

(2) HVI_MTRENTRYCOUNT

(32 ビット用 OS) ～<符号なし整数> ((10～30000000)) <<5000>>

(64 ビット用 OS) ～<符号なし整数> ((10～30000000)) <<10000>>

モジュールトレースの出力ファイルのエントリ数の上限値を設定します。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，C++ ORB

(3) HVI_MTRFILECOUNT

(Java ORB) ～<符号なし整数> ((1～256)) <<15>>

モジュールトレースの出力ディレクトリ（${VBROKER_ADM}/../logj/mdltrc（%VBROKER_ADM%¥..¥logj¥mdltrc)）に作成するファイル数の上限値を設定します。同時に実行する ORB アプリケーション数以上の値にしてください。

(C++ ORB) ～<符号なし整数> ((1～256)) <<10>>

モジュールトレースの出力ディレクトリ（${VBROKER_ADM}/../log/mdltrc（%VBROKER_ADM%¥..¥log¥mdltrc)）に作成するファイル数の上限値を設定します。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，C++ ORB

2.8.3 通信トレース（環境変数）

通信トレースに関する環境変数の詳細について説明します。

(1) HVI_COMTRACE

～<文字列> { true|false|Y|N } <<true>>

通信トレースを取得するかどうかを設定します。

- true|Y
  通信トレースを取得します。
- false|N
  通信トレースを取得しません。

設定値"Y"および"N"は，下位バージョンの ORB との共通環境を実現するために使用できますが，設定値"true"および"false"の使用を推奨します。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，C++ ORB

(2) HVI_COMTENTRYCOUNT

～<符号なし整数> ((100～30000000)) <<20000>>

通信トレースの出力ファイルのエントリ数の上限値を設定します。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，C++ ORB

(3) HVI_COMTFILECOUNT

(Java ORB) ～<符号なし整数> ((1～256)) <<15>>

通信トレースの出力ディレクトリ（${VBROKER_ADM}/../logj/comtrc（%VBROKER_ADM%¥..¥logj¥comtrc)）に，実行形式ファイルごとに作成するファイルの上限値を設定します。新規にファイルを作成する場合にこの設定値を超えているときには，更新時刻の古いものから削除します。同時に実行する ORB アプリケーション数以上の値にしてください。

(C++ ORB) ～<符号なし整数> ((1～256)) <<10>>

通信トレースの出力ディレクトリ（${VBROKER_ADM}/../log/comtrc（%VBROKER_ADM%¥..¥log¥comtrc)）に，実行形式ファイルごとに作成するファイルの上限値を設定します。新規にファイルを作成する場合にこの設定値を超えているときには，更新時刻の古いものから削除します。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，C++ ORB

(4) HVI_COMTTELEGSIZE

(Java ORB) ～<符号なし整数> ((0～1000000)) <<512>>（単位：バイト）

取得する通信電文のサイズを設定します。"0"を設定した場合は通信電文を取得しません。

注意事項

指定範囲外の値を設定した場合，または次の条件を満たす値を設定した場合は，デフォルト値が設定されます。

- 32 ビット用 OS の場合

  <環境変数 HVI_COMTTELEGSIZE の設定値> > {(<環境変数 HVI_COMTENTRYCOUNT の設定値> - 10)×32}

- 64 ビット用 OS の場合

  <環境変数 HVI_COMTTELEGSIZE の設定値> > {(<環境変数 HVI_COMTENTRYCOUNT の設定値> - 10)×40}

(C++ ORB) ～<符号なし整数> ((0～1000000)) <<528>>（単位：バイト）

取得する通信電文のサイズを設定します。"0"を設定した場合は通信電文を取得しません。

注意事項

指定範囲外の値を設定した場合，または次の条件を満たす値を設定した場合は，デフォルト値が設定されます。

<環境変数 HVI_COMTTELEGSIZE の設定値> > {(<環境変数 HVI_COMTENTRYCOUNT の設定値> - 10)×24}

使用言語

Java ORB，C++ ORB

2.8.4 メッセージログ（環境変数）

メッセージログに関する環境変数の詳細について説明します。

(1) HVI_MSGLOG_OUTPUT

(osagent) ～<文字列> { ALL|NONE } <<ALL>>

(Java ORB) ～<文字列> { ALL|MSGLOG|NONE } <<ALL>>

メッセージログの出力先を設定します。

- ALL
  イベントビューアのアプリケーションログ（Windows），syslog（UNIX），およびメッセージログへメッセージを出力します。
- MSGLOG
  メッセージログにだけメッセージを出力し，イベントビューアのアプリケーションログ（Windows）または syslog（UNIX）にはメッセージを出力しません。この設定文字列は Java ORB にだけ有効です。
- NONE
  メッセージを出力しません。

注意事項

設定文字列以外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，osagent

(2) HVI_MSGLOG_CONSOLE（UNIX 版 Java ORB だけ）

～<文字列> { true|false } <<true>>

syslog への出力に失敗した場合に，メッセージを強制的にコンソールへ出力するかどうかを設定します。

- true
  syslog への出力に失敗した場合，メッセージを強制的にコンソールへ出力します。
- false
  syslog への出力に失敗した場合，メッセージを出力しません。

注意事項

設定文字列以外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB（UNIX）限定

(3) HVI_MSGLOG_LEVEL

～<符号なし整数> { ERR|WARNING|INFO } <<INFO>>

出力するメッセージのメッセージレベルを設定します。各メッセージのレベルについては，「8.2 KFCB91000～KFCB91999 のメッセージ」を参照してください。

- ERR
  メッセージレベル E（Error）のメッセージだけを出力します。
- WARNING

メッセージレベル E（Error）およびメッセージレベル W（Warning）のメッセージを出力します。

- INFO
  すべてのメッセージを出力します。

注意事項

設定値以外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，C++ ORB

(4) HVI_MSGLOG_SECURITY

～<文字列> { true|false } <<false>>

セキュリティにかかわる情報のメッセージ出力を抑止するかどうかを設定します。セキュリティにかかわる情報とは，ポート番号などを指します。

- true
  セキュリティにかかわる情報のメッセージ出力を抑止します。
- false
  セキュリティにかかわる情報のメッセージを出力します。

注意事項

設定文字列以外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，C++ ORB

(5) HVI_MSGLOG_NO_PERMISSION（osagent だけ）

～<文字列または-1> { メッセージ ID|-1 } <<KFCB92006>>

出力を抑止したいメッセージのメッセージ ID を設定します。

メッセージプリフィックス ID（KFCB）およびメッセージ番号で構成されているメッセージ ID を設定することで，設定したメッセージ ID のメッセージ出力を抑止できます。

- メッセージ ID
  設定したメッセージ ID のメッセージ出力を抑止します。
  複数のメッセージ ID を設定する場合は","で区切って設定してください。
- -1
  すべてのメッセージが出力されます。

注事事項

KFCB 以外のメッセージプリフィックス ID など，不正な値を設定した場合は，デフォルト値が設定されます。

使用言語

osagent 限定

2.8.5　スタックトレース（環境変数）

スタックトレースに関する環境変数の詳細について説明します。

(1)　HVI_STKTRACE

～<文字列> { true|false } <<true>>

スタックトレースを取得するかどうかを設定します。

- true
  スタックトレースを取得します。
- false
  スタックトレースを取得しません。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

C++ORB(Windows)限定

(2)　HVI_STKFILECOUNT

～<符号なし整数>((1～50000)) <<2000>>

スタックトレースの出力ファイル数の最大値を設定します。

使用言語

C++ORB(Windows)限定

2.8.6　ネーミングサービス名前空間情報ログ（環境変数）

ネーミングサービス名前空間情報ログに関する環境変数の詳細について説明します。

(1) HVI_NAMELOGOUTPUT

～<文字列> { ALL|REQUEST|NONE } <<ALL>>

ネーミングサービス名前空間情報ログを取得するかどうかを設定します。

- ALL
  ネーミングサービス名前空間情報ログを取得します。ネーミングサービスへの要求時のログと定期出力のログを取得します。
- REQUEST
  ネーミングサービス名前空間情報ログを取得します。ネーミングサービスへの要求時のログだけ取得します。
- NONE
  ネーミングサービス名前空間情報ログを取得しません。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB（ネーミングサービス限定），C++ ORB（ネーミングサービス限定）

(2) HVI_NAMELOGFILESIZE

～<符号なし整数>((8388608～1073741824)) <<8388608>>（単位：バイト）

ネーミングサービス名前空間情報ログの出力ファイルのサイズを設定します。ログを出力する場合にログファイルのサイズがこの設定値を超えているときには，もう一つのファイルへスワップします。

注意事項

指定範囲外の値，または無効な値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB（ネーミングサービス限定），C++ ORB（ネーミングサービス限定）

(3) HVI_NAMELOGFILECOUNT

～<符号なし整数>((1～256)) <<5>>

ネーミングサービス名前空間情報ログの出力ディレクトリ（${VBROKER_ADM}/../logj/namelog（%VBROKER_ADM%¥..¥logj¥namelog））下に，何プロセス数分のネーミングサービス名前空間情報ログファイルを保持するかを設定します。新規にログファイルを作成する場合にこの設定値を超えているときには，更新時刻の古いプロセスのログファイルから削除します。

ログファイルはプロセス ID ごとに二つ生成され，これらを二つで一組として処理するため，設定値の 2 倍のファイル数が生成されることがあります。

また，削除しようとしたファイルが使用中の場合，使用者がいなくなるまでファイルは存在するため，設定値の 2 倍より多くのファイルが存在することもあります。

注意事項

指定範囲外の値，または無効な値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB（ネーミングサービス限定）

(4) HVI_NAMELOGSNAPSHOTINTERVAL

～<符号なし整数>((30～86400)) <<600>>（単位：秒）

出力ファイルにネーミングサービス名前空間情報ログのスナップショットを出力する間隔を設定します。

スナップショットは，前回スナップショットを出力してからネーミングサービス名前空間情報ログに更新があった場合に出力されます。ネーミングサービス名前空間情報ログに更新がない場合，スナップショットは出力されません。

注意事項

指定範囲外の値，または無効な値を設定された場合は，デフォルト値が設定されます。

使用言語

Java ORB（ネーミングサービス限定），C++ ORB（ネーミングサービス限定）

2.8.7 システム例外のマイナーコード（環境変数）

システム例外のマイナーコードに関する環境変数の詳細について説明します。

(1) HVI_OMGVMCID

～<文字列> { true|false } <<false>>

システム例外発生時に OMG が規定するマイナーコードを使用するかどうかを設定します。

- true
  OMG が規定するマイナーコードを使用します。
- false
  OMG が規定するマイナーコードを使用しません。

環境変数の値とシステム例外に設定されるマイナーコードの対応を次の表に示します。

表 2-11　環境変数とマイナーコードの対応

HVI_OMGVMCID の設定値	OMG がマイナーコードを規定している例外	OMG がマイナーコードを規定していない例外
true	OMG が規定するマイナーコードを使用します。	一意な値のマイナーコードを使用します。
false（デフォルト）	一意な値のマイナーコードを使用します。	一意な値のマイナーコードを使用します。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

C++ ORB 限定

2.9　定義句の設定

この節では，定義句の設定について説明します。

定義句では，トラブルシュートファイルごとに個別に設定します。必要に応じて設定してください。

2.9.1　定義句の設定方法

定義句の設定方法について説明します。

定義ファイルに記述することで，定義句を設定します。次の手順で設定してください。

1. 定義ファイルの作成

   定義ファイルに定義句を記述します。記述形式を次に示します。

   ```
   set定義句=この定義句に対する定義値
   ```

   上記の記述形式以外で記述している行については，コメント行として扱います。また，一つの定義は 1 行で記述してください。改行文字を入れるなどして，2 行以上にわたって記述しているものは無効となります。

2. 定義ファイルの格納

   環境変数 VBROKER_ADM に設定されているディレクトリに，"HVIORB_DEF"または"HVMGTEE_DEF"の名称で定義ファイルを格納してください。

2.9.2　定義句の一覧

定義句の一覧を次の表に示します。定義句の詳細は，2.9.3 以降で説明します。

表 2-12　定義ファイル HVIORB_DEF に記述する定義句の一覧

項番	定義句	説明	サポート言語		備考
			Java ORB	C++ ORB	
1	HVI_ORBLOG_SIZE	エラーログの出力ファイルの上限サイズを設定します。	○	×	osagent 以外のコマンドにも使用
2	HVI_ORBMSGLOG_SIZE	メッセージログの出力ファイルの上限サイズを設定します。	○	×	－

（凡例）　○：サポート　×：未サポート　－：説明なし

表 2-13　定義ファイル HVMGTEE_DEF に記述する定義句の一覧

項番	定義句	説明	サポート言語		備考
			Java ORB	C++ ORB	
1	HVI_GTEE_INTERVAL	hvmgtee コマンドの稼働中に定義ファイルを読み取る間隔を設定します。	×	×	UNIX 版 osagent 限定
2	HVI_GTEE_OUTPUT	hvmgtee コマンドを経由したデータの画面出力，およびファイル出力をするかどうかを設定します。	×	×	UNIX 版 osagent 限定
3	HVI_GTEE_FILESIZE	バーボースログの出力ファイルの上限サイズを設定します。	×	×	UNIX 版 osagent 限定
4	HVI_GTEE_LOGPRESS	hvmgtee コマンドが正常終了したとき，または SIGINT，もしくは SIGTERM を受信して終了したときまでに，出力していたログファイルを圧縮するかどうかを設定します。	×	×	UNIX 版 osagent 限定
5	HVI_GTEE_FILECOUNT	同一モジュールのログファイル数の上限値を設定します。	×	×	UNIX 版 osagent 限定

（凡例）　○：サポート　×：未サポート　－：説明なし

2.9.3　エラーログ（定義句）

エラーログに関する定義句の詳細について説明します。

(1)　HVI_ORBLOG_SIZE

～<符号なし整数> ((45000～1073741824)) <<45000>>（単位：バイト）

エラーログの出力ファイルの上限サイズを設定します。複数の同じプロセスが同じエラーログファイルに出力するため，ほかのプロセス起動中にこの定義句の設定値を変更した場合，エラーログファイルが更新時間に関係なく削除されるおそれがあります。そのため，すべてのプロセスを停止してから設定値を変更してください。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB，osagent 以外のコマンド

2.9.4　メッセージログ（定義句）

メッセージログに関する定義句の詳細について説明します。

(1)　HVI_ORBMSGLOG_SIZE

～<符号なし整数> ((4096～16777216)) <<1048576>>（単位：バイト）

メッセージログの出力ファイルの上限サイズを設定します。複数の同じプロセスが同じメッセージログに出力するため，ほかのプロセス起動中にこの定義句の設定値を変更した場合，メッセージログファイルが更新時間に関係なく削除されるおそれがあります。そのため，すべてのプロセスを停止してから設定値を変更してください。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB 限定

2.9.5　バーボースログ（定義句）

バーボースログに関する定義句の詳細について説明します。

(1)　HVI_GTEE_INTERVAL

～<符号なし整数> ((0～14515200)) <<180>>　(単位：秒)

hvmgtee コマンドの稼働中に定義ファイルを読み取る間隔を設定します。

"0"を設定した場合，定義ファイルは読み取りません。そのため，"0"を設定したあと再び読み取りを有効にするためには，osagent の再起動が必要です。この設定値は，hvmgtee コマンドの稼働中に変更できます。

使用言語

UNIX 版 osagent 限定

(2)　HVI_GTEE_OUTPUT

～<文字列> { DISPLAY|FILE|ALL|OFF } <<FILE>>

hvmgtee コマンドを経由したデータの画面出力およびファイル出力をするかどうかを設定します。この設定値は，hvmgtee コマンドの稼働中に変更できます。

- DISPLAY
  画面出力だけを行い，ファイル出力を抑止します。

- FILE
  画面出力を抑止し，ファイル出力だけを行います。
- ALL
  画面出力およびファイル出力の両方を行います。
- OFF
  画面出力およびファイル出力の両方を抑止します。

使用言語

UNIX 版 osagent 限定

(3) HVI_GTEE_FILESIZE

～<符号なし整数> ((1～524288)) <<10240>>（単位：キロバイト）

バーボーズログの出力ファイルの上限サイズを設定します。

ファイルにログ出力されるときに，ファイルのサイズがチェックされます。そのときに，すでにこの設定値を超えていると，もう一つのファイルにスワップします。

この設定値は，hvmgtee コマンドの稼働中に変更できます。

使用言語

UNIX 版 osagent 限定

(4) HVI_GTEE_LOGPRESS

～<文字列> { true|false|Y|N } <<true>>

hvmgtee コマンドが正常終了したとき，または SIGINT，もしくは SIGTERM を受信して終了したときまでに，出力していたログファイルを圧縮するかどうかを設定します。この設定値は，hvmgtee コマンドの稼働中に変更できます。

- true|Y
  自動的に圧縮します。
- false|N
  自動的に圧縮しません。

設定値"Y"および"N"は，下位互換のために使用することはできますが，設定値"true"および"false"の使用を推奨します。

使用言語

UNIX 版 osagent 限定

(5) HVI_GTEE_FILECOUNT

～<符号なし整数> ((2～64)) <<4>>

同一モジュールのログファイル数の上限値を設定します。

ファイル数の上限値の変更は hvmgtee コマンドの起動時だけ有効とし，hvmgtee コマンドの稼働中の変更は無効となります。

使用言語

UNIX 版 osagent 限定

2.10　プロパティの設定

この節では，プロパティの設定について，説明します。

環境変数に対応するプロパティを設定することで，環境変数と同じ設定ができます。プロパティに設定できる値，および値の範囲は対応する環境変数と同じです。また，環境変数とプロパティを同時に設定した場合は，プロパティの設定が有効となります。

設定できるプロパティは次のとおりです。

- トレースプロパティ
  モジュールトレース，通信トレース，マイナーコード，およびバーボースログのプロパティです。
- マイナーコードプロパティ
  システム例外のマイナーコードのプロパティです。
- メッセージログプロパティ
  メッセージログのプロパティです。

2.10.1　プロパティの設定方法

使用している ORB に応じて，プロパティを設定します。

(1)　プロパティの設定方法（Java ORB）

EJB クライアントアプリケーションの場合は JavaVM の起動オプション（システムプロパティ）に，Web コンテナサーバまたは J2EE サーバの場合は，usrconf.cfg ファイルに設定します。

(2)　プロパティの設定方法（C++ ORB）

C++ ORB の場合のプロパティを設定する方法について説明します。

htc.props ファイルに記述することで，プロパティを設定します。htc.props ファイルは，トレースプロパティ，マイナーコードプロパティ，およびメッセージログプロパティを設定するためのプロパティファイルです。htc.props ファイルに設定したプロパティは，ユーザプロセス，サーバプロセス，および osagent で有効です。

なお，トレースプロパティ，およびマイナーコードプロパティは htc.props ファイルだけで設定します。メッセージログプロパティはコマンドの引数で設定することもできます。設定方法は，マニュアル「Borland Enterprise Server VisiBroker デベロッパーズガイド」を参照してください。

なお，Linux では，htc.props ファイルを使用できません。

次の手順で設定してください。

1. htc.props ファイルの作成

   次に示す形式で記述します。

   プロパティ名=このプロパティに対する定義値

   空白の行とコメント行（#で始まる行）は無視されます。

   記述例を示します。

```
#trace property
vbroker.orb.htc.mtr.trace=true
vbroker.orb.htc.mtr.entryCount=5000
```

2. htc.props ファイルの格納

   環境変数 VBROKER_ADM で設定されているディレクトリに，"htc.props"の名称で，テキスト形式にして格納してください。

2.10.2 プロパティと環境変数の対応

プロパティとそれに対応する環境変数を次の表に示します。

表 2-14 プロパティと環境変数の対応（ORB のトラブルシュート機能）

項番	種別	プロパティ	環境変数	サポート言語		備考
				Java ORB	C++ ORB	
1	トレースプロパティ	vbroker.orb.htc.mtr.trace	HVI_MDLTRACE	○	○	−
2		vbroker.orb.htc.mtr.entryCount	HVI_MTRENTRYCOUNT	○	○	−
3		vbroker.orb.htc.mtr.fileCount	HVI_MTRFILECOUNT	○	○	−
4		vbroker.orb.htc.comt.trace	HVI_COMTRACE	○	○	−
5		vbroker.orb.htc.comt.entryCount	HVI_COMTENTRYCOUNT	○	○	−
6		vbroker.orb.htc.comt.fileCount	HVI_COMTFILECOUNT	○	○	−
7		vbroker.orb.htc.comt.telegSize	HVI_COMTTELEGSIZE	○	○	−
8		vbroker.orb.htc.traceFilenameTod	HVI_TRACEFILENAME_TOD	○	○	−
9		vbroker.orb.htc.nameLog.output	HVI_NAMELOGOUTPUT	○	○	ネーミングサービス限定

項番	種別	プロパティ	環境変数	サポート言語		備考
				Java ORB	C++ ORB	
10		vbroker.orb.htc.nameLog.fileSize	HVI_NAMELOGFILESIZE	○	○	ネーミングサービス限定
11		vbroker.orb.htc.nameLog.fileCount	HVI_NAMELOGFILECOUNT	○	×	ネーミングサービス限定
12		vbroker.orb.htc.nameLog.snapshotInterval	HVI_NAMELOGSNAPSHOTINTERVAL	○	○	ネーミングサービス限定
13		vbroker.orb.htc.tracePath	HVI_TRACEPATH	○	○	"(ダブルクォーテーション)で囲まれたディレクトリを設定しないでください。設定した場合に動作の保証はできません。
14		vbroker.orb.htc.mapFileInit	HVI_MAPFILEINIT	○	○	UNIX 限定
15	マイナーコードプロパティ	vbroker.orb.htc.omgVmcid	HVI_OMGVMCID	×	○	−
16	メッセージログプロパティ	vbroker.orb.htc.msgLog.output	HVI_MSGLOG_OUTPUT	○	×	−
17		vbroker.orb.htc.msgLog.console	HVI_MSGLOG_CONSOLE	○	×	UNIX 限定
18		vbroker.orb.htc.msgLog.level	HVI_MSGLOG_LEVEL	○	○	−
19		vbroker.orb.htc.msgLog.security	HVI_MSGLOG_SECURITY	○	○	−
20		vbroker.orb.htc.msgLog.noPermission	HVI_MSGLOG_NO_PERMISSION	×	×	osagent 限定

(凡例) ○：サポート　×：未サポート　−：説明なし

2.11 hdumpns コマンドの使用方法（Cosminexus TPBroker）（Windows）

この節では，hdumpns コマンドの使用方法について説明します。hdumpns コマンドは，Cosminexus TPBroker（Windows）で，ネーミングサービスのスレッドダンプを取得するために使用します。

形式

hdumpns <ネーミングサービスのプロセス ID>

機能

ネーミングサービスから応答が返らない場合，ネーミングサービスで障害が発生した場合などに JavaVM のスレッドダンプを取得します。

このコマンドは，Management Server でネーミングサービスを起動した場合，または Management Server ではなく，J2EE サーバからネーミングサービスを自動起動した場合に使用します。ネーミングサービスが J2EE サーバのプロセス内（インプロセス）で起動されている場合は使用できません。

このコマンドを実行するとネーミングサービスの標準出力，および表 2-15 に示す<スレッドダンプの出力ディレクトリ>¥javacore*.txt にスレッドダンプが出力されます。なお，JavaVM の-XX:-HitachiThreadDumpToStdout オプションを設定してネーミングサービスを起動した場合，標準出力にスレッドダンプは出力されません。

表 2-15　スレッドダンプの出力ディレクトリ

ネーミングサービスの起動方法	設定値	スレッドダンプの出力ディレクトリ
Management Server	なし。	<Cosminexus のインストール先ディレクトリ>¥TPB¥logj
J2EE サーバのコマンド	デフォルト（次に示すプロパティがすべて設定されていない）。	<Cosminexus のインストール先ディレクトリ>¥CC¥server¥public¥ejb¥<J2EE サーバ名>¥logs¥TPB¥logj
	[usrconf.cfg] ejb.public.directory=<指定したディレクトリ>	<指定したディレクトリ>¥ejb¥<J2EE サーバ名>¥logs¥TPB¥logj
	[usrconf.cfg] ejb.server.log.directory=<指定したディレクトリ>	<指定したディレクトリ>¥ejb¥<J2EE サーバ名>¥logs¥TPB¥logj
	[usrconf.properties] vbroker.orb.htc.tracePath=<指定したディレクトリ>	<指定したディレクトリ>

表 2-15 の[usrconf.cfg]および[usrconf.properties]は，J2EE サーバで使用するプロパティファイルです。このファイルに J2EE サーバのプロパティやオプションを設定することができます。設定方法については，マニュアル「Cosminexus システム構築ガイド」を参照してください。

プロパティの優先順位を次に示します。

ejb.public.directory < ejb.server.log.directory < vbroker.orb.htc.tracePath

引数

<ネーミングサービスのプロセス ID>

スレッドダンプを取得したいネーミングサービスのプロセス ID を指定します。プロセス ID は，Windows のタスクマネジャを使用して確認してください。プロセス名は"nameserv.exe"です。

入力例

コマンドの入力例を示します。

（例）

```
hdumpns 1104
```

戻り値

0：コマンドの実行に成功しました。

1：指定されたネーミングサービスと通信できません。

2：指定されたネーミングサービスと通信できません。

戻り値が 0 以外の場合，メッセージが出力されます。メッセージについては，「8.6　hdumpns コマンドのエラーメッセージ」を参照してください。

注意事項

- 負荷が高い状態では，hdumpns コマンドが一時的にエラーになる場合があります。その場合，hdumpns コマンドを再度実行してください。
- hdumpns コマンド実行時にネーミングサービスから出力されるファイル javacore*.txt のファイル数には制限がありません。そのため，hdumpns コマンドを繰り返し実行するとディスク容量を圧迫することになります。javacore*.txt は定期的に退避し，削除してください。

2.12　メモリ所要量およびディスク占有量

この節では，ORB のトラブルシュート機能を使用するときの，メモリ所要量およびディスク占有量について説明します。

2.12.1　メモリ所要量

ORB のトラブルシュート機能を使用するときのメモリ所要量を次の表に示します。

表 2-16　ORB のトラブルシュート機能を使用するときのメモリ所要量

項番	項目	メモリ所要量（バイト）
1	モジュールトレースの取得	32 ビット用 OS の場合 Java ORB 16×＜環境変数 HVI_MTRENTRYCOUNT の値＞＋ 15,360 C++ ORB 32×＜環境変数 HVI_MTRENTRYCOUNT の値＞＋ 15,360 64 ビット用 OS の場合 Java ORB 32×＜環境変数 HVI_MTRENTRYCOUNT の値＞＋ 15,360 C++ ORB 48×＜環境変数 HVI_MTRENTRYCOUNT の値＞＋ 15,360
2	エラーログの取得	8,192
3	通信トレースの取得	32 ビット用 OS の場合 Java ORB 32×＜環境変数 HVI_COMTENTRYCOUNT の値＞＋ 15,360 C++ ORB 24×＜環境変数 HVI_COMTENTRYCOUNT の値＞＋ 15,360 64 ビット用 OS の場合 Java ORB 40×＜環境変数 HVI_COMTENTRYCOUNT の値＞＋ 15,360 C++ ORB 24×＜環境変数 HVI_COMTENTRYCOUNT の値＞＋ 15,360
4	メッセージログの取得	8,192
5	ネーミングサービス名前空間情報ログの取得	208×＜登録するコンテキストの数＞＋ 104×＜登録するオブジェクトの数＞＋ 8,192

注

UNIX の場合，モジュールトレースの取得はヒープメモリを使用しています。ヒープメモリが不足したときにはアプリケーションが異常終了するなど，動作が不安定になります。データサイズはモジュールトレースのサイズとプロセスで使用するデータのサイズを考慮して，limit コマンドなどで設定してください。

2.12.2　ディスク占有量

ORB のトラブルシュート機能を使用するときのディスク占有量について説明します。

(1)　ディスク占有量の算出式

ORB のトラブルシュート機能を使用するときのディスク占有量を次の表に示します。

表 2-17　ORB のトラブルシュート機能を使用するときのディスク占有量

項番	項目	ディスク占有量（バイト）
1	Windows %VBROKER_ADM%¥..¥log¥comtrc¥* UNIX※ $VBROKER_ADM/../log/comtrc/*	<C++ ORB 通信トレース容量>以上 詳細は次の個所を参照してください。 「(2)(a)　C++ ORB 通信トレース容量の算出式」
2	(UNIX だけ) ※ $VBROKER_ADM/../log/hgttrc/*	<osagent バーボースログファイル容量>以上 詳細は次の個所を参照してください。 「(2)(b)　osagent バーボースログファイル容量の算出式」
3	Windows %VBROKER_ADM%¥..¥log¥mdltrc¥* UNIX※ $VBROKER_ADM/../log/mdltrc/*	<C++ ORB モジュールトレース容量>以上 詳細は次の個所を参照してください。 「(2)(c)　C++ ORB モジュールトレース容量の算出式」
4	Windows(C++ORB)だけ %VBROKER_ADM%¥..¥log¥stktrc¥*	<C++ ORB スタックトレース容量>以上 詳細は次の個所を参照してください。 「(2)(d)　C++ ORB スタックトレース容量の算出式」
5	Windows %VBROKER_ADM%¥..¥logj¥* UNIX※ $VBROKER_ADM/../logj/*	<エラーログ容量>+<メッセージログ容量>以上 詳細は次の個所を参照してください。 •「(2)(e)　エラーログ容量の算出式」 「(2)(f)　メッセージログ容量の算出式」
6	Windows %VBROKER_ADM%¥..¥logj¥comtrc¥* UNIX※ $VBROKER_ADM/../logj/comtrc/*	<Java ORB 通信トレース容量>+<ネーミングサービス通信トレース容量>+<osfind 通信トレース容量>+<oad 通信トレース容量>+<irep 通信トレース容量>+<gatekeeper 通信トレース容量>以上 詳細は次の個所を参照してください。 •「(2)(g)　Java ORB 通信トレース容量の算出式」 •「(2)(h)　ネーミングサービス通信トレース容量の算出式」 •「(2)(i)　osfind 通信トレース容量の算出式」 •「(2)(j)　oad 通信トレース容量の算出式」 •「(2)(k)　irep 通信トレース容量の算出式」 •「(2)(l)　gatekeeper 通信トレース容量の算出式」

項番	項目	ディスク占有量（バイト）
7	Windows %VBROKER_ADM%¥..¥logj¥mdltrc¥* UNIX※ $VBROKER_ADM/../logj/mdltrc/*	<Java ORB モジュールトレース容量>＋<ネーミングサービスモジュールトレース容量>＋<osfind モジュールトレース容量>＋<oad モジュールトレース容量>＋<irep モジュールトレース容量>＋<gatekeeper モジュールトレース容量>以上 詳細は次の個所を参照してください。 • 「(2)(m)　Java ORB モジュールトレース容量の算出式」 • 「(2)(n)　ネーミングサービスモジュールトレース容量の算出式」 • 「(2)(o)　osfind モジュールトレース容量の算出式」 • 「(2)(p)　oad モジュールトレース容量の算出式」 • 「(2)(q)　irep モジュールトレース容量の算出式」 • 「(2)(r)　gatekeeper モジュールトレース容量の算出式」
8	Windows %VBROKER_ADM%¥..¥logj¥namelog¥* UNIX※ $VBROKER_ADM/../logj/namelog/*	<ネーミングサービス名前空間情報ログ容量> 詳細は次の個所を参照してください。 • 「(2)(s)　ネーミングサービス名前空間情報ログ容量の算出式」

注※

ORB アプリケーションプログラムが作成できるファイルサイズの上限値に，モジュールトレースまたは通信トレースのファイルサイズ以下の値を設定して ORB アプリケーションプログラムを実行すると，アプリケーションが異常終了するなど，動作が不安定になります。ファイルサイズの上限値は，limit コマンドなどでトレースファイルのサイズを下回る値に変更しないでください。

(2)　出力ファイルの算出式

出力ファイルの算出式について説明します。

(a)　C++ ORB 通信トレース容量の算出式

32 ビット用 OS の場合

C++ ORB 通信トレース容量 ＝ (24×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 128)×(<実行中の osagent の数> ＋ <実行中の OTS のデーモンプロセスおよびコマンド数>)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

64 ビット用 OS の場合

C++ ORB 通信トレース容量 ＝ (24×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 192)×(<実行中の osagent の数> ＋ <実行中の OTS のデーモンプロセスおよびコマンド数>)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

(b) osagent バーボースログファイル容量の算出式

osagent バーボースログファイル容量 = (1,024×<定義句 HVI_GTEE_FILESIZE の値>)×<実行中の osagent の数>×(<定義句 HVI_GTEE_FILECOUNT の値> + 2) / 0.8 （単位：バイト）

(c) C++ ORB モジュールトレース容量の算出式

32 ビット用 OS の場合

C++ ORB モジュールトレース容量 = (32×<環境変数 HVI_MTRENTRYCOUNT の値>)×(<実行中の osagent の数> + <実行中の OTS のデーモンプロセスおよびコマンド数>)×<環境変数 HVI_MTRFILECOUNT の値> / 0.8 （単位：バイト）

64 ビット用 OS の場合

C++ ORB モジュールトレース容量 = (48×<環境変数 HVI_MTRENTRYCOUNT の値> + 192) ×(<実行中の osagent の数> + <実行中の OTS のデーモンプロセスおよびコマンド数>)×<環境変数 HVI_MTRFILECOUNT の値> / 0.8 （単位：バイト）

(d) C++ ORB スタックトレース容量の算出式

C++ ORB スタックトレース容量 = 4,000×<環境変数 HVI_STKFILECOUNT の値> / 0.8 （単位：バイト）

(e) エラーログ容量の算出式

エラーログ容量 = (<定義句 HVI_ORBLOG_SIZE の値> + 512) ×2 / 0.8 （単位：バイト）

(f) メッセージログ容量の算出式

メッセージログ容量 = <定義句 HVI_ORBMSGLOG_SIZE の値>×2 / 0.8 （単位：バイト）

(g) Java ORB 通信トレース容量の算出式

32 ビット用 OS の場合

Windows

Java ORB 通信トレース容量 = (32×<環境変数 HVI_COMTENTRYCOUNT の値> + 128)×(<環境変数 HVI_COMTFILECOUNT の値> + <実行中の Java ORB UAP の数>) / 0.8 （単位：バイト）

UNIX

Java ORB 通信トレース容量 = (32×<環境変数 HVI_COMTENTRYCOUNT の値> + 128) ×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

64 ビット用 OS の場合

Windows

Java ORB 通信トレース容量 ＝ (40×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 192)×(<環境変数 HVI_COMTFILECOUNT の値> ＋ <実行中の Java ORB UAP の数>) / 0.8 （単位：バイト）

UNIX

Java ORB 通信トレース容量 ＝ (40×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 192) ×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

(h) ネーミングサービス通信トレース容量の算出式

32 ビット用 OS の場合

Windows

ネーミングサービス通信トレース容量 ＝ (32×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 128)×(<環境変数 HVI_COMTFILECOUNT の値> ＋ <実行中のネーミングサービスの数>) / 0.8 （単位：バイト）

UNIX

ネーミングサービス通信トレース容量 ＝ (32×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 128)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

64 ビット用 OS の場合

Windows

ネーミングサービス通信トレース容量 ＝ (40×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 192)×(<環境変数 HVI_COMTFILECOUNT の値> ＋ <実行中のネーミングサービスの数>) / 0.8 （単位：バイト）

UNIX

ネーミングサービス通信トレース容量 ＝ (40×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 192) ×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

(i) osfind 通信トレース容量の算出式

32 ビット用 OS の場合

Windows

osfind 通信トレース容量 ＝ (32×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 128)×(<環境変数 HVI_COMTFILECOUNT の値> ＋ <実行中の osfind の数>) / 0.8 （単位：バイト）

UNIX

osfind 通信トレース容量 ＝ (32×<環境変数 HVI_COMTENTRYCOUNT の値> ＋ 128)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

64 ビット用 OS の場合

Windows

osfind 通信トレース容量 = (40×<環境変数 HVI_COMTENTRYCOUNT の値> + 192)×(<環境変数 HVI_COMTFILECOUNT の値> + <実行中の osfind の数>) / 0.8 （単位：バイト）

UNIX

osfind 通信トレース容量 = (40×<環境変数 HVI_COMTENTRYCOUNT の値> + 192)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

(j) oad 通信トレース容量の算出式

32 ビット用 OS の場合

Windows

oad 通信トレース容量 = (32×<環境変数 HVI_COMTENTRYCOUNT の値> + 128)×(<環境変数 HVI_COMTFILECOUNT の値> + <実行中の oad の数>) / 0.8 （単位：バイト）

UNIX

oad 通信トレース容量 = (32×<環境変数 HVI_COMTENTRYCOUNT の値> + 128)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

64 ビット用 OS の場合

Windows

oad 通信トレース容量 = (40×<環境変数 HVI_COMTENTRYCOUNT の値> + 192)×(<環境変数 HVI_COMTFILECOUNT の値> + <実行中の oad の数>) / 0.8 （単位：バイト）

UNIX

oad 通信トレース容量 = (40×<環境変数 HVI_COMTENTRYCOUNT の値> + 192)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

(k) irep 通信トレース容量の算出式

32 ビット用 OS の場合

Windows

irep 通信トレース容量 = (32×<環境変数 HVI_COMTENTRYCOUNT の値> + 128)×(<環境変数 HVI_COMTFILECOUNT の値> + <実行中の irep の数>) / 0.8 （単位：バイト）

UNIX

irep 通信トレース容量 = (32×<環境変数 HVI_COMTENTRYCOUNT の値> + 128)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

64 ビット用 OS の場合

Windows

irep 通信トレース容量 = (40×<環境変数 HVI_COMTENTRYCOUNT の値> + 192)×(<環境変数 HVI_COMTFILECOUNT の値> + <実行中の irep の数>) / 0.8 （単位：バイト）

UNIX

irep 通信トレース容量 = (40×<環境変数 HVI_COMTENTRYCOUNT の値> + 192)×<環境変数 HVI_COMTFILECOUNT の値> / 0.8 （単位：バイト）

(l) gatekeeper 通信トレース容量の算出式

Windows だけサポートしています。

Windows

gatekeeper 通信トレース容量 = (32×<環境変数 HVI_COMTENTRYCOUNT の値> + 128)×(<環境変数 HVI_COMTFILECOUNT の値> + <実行中の gatekeeper の数>) / 0.8 （単位：バイト）

(m) Java ORB モジュールトレース容量の算出式

32 ビット用 OS の場合

Java ORB モジュールトレース容量 = (16×<環境変数 HVI_MTRENTRYCOUNT の値> + 128)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中の Java ORB UAP の数>) / 0.8 （単位：バイト）

64 ビット用 OS の場合

Java ORB モジュールトレース容量 = (32×<環境変数 HVI_MTRENTRYCOUNT の値> + 192)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中の Java ORB UAP の数>) / 0.8 （単位：バイト）

(n) ネーミングサービスモジュールトレース容量の算出式

32 ビット用 OS の場合

ネーミングサービスモジュールトレース容量 = (16×<環境変数 HVI_MTRENTRYCOUNT の値> + 128)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中のネーミングサービスの数>) / 0.8 （単位：バイト）

64 ビット用 OS の場合

ネーミングサービスモジュールトレース容量 = (32×<環境変数 HVI_MTRENTRYCOUNT の値> + 192)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中のネーミングサービスの数>) / 0.8 （単位：バイト）

(o) osfind モジュールトレース容量の算出式

32 ビット用 OS の場合

osfind モジュールトレース容量 = (16×<環境変数 HVI_MTRENTRYCOUNT の値> + 128) × (<環境変数 HVI_MTRFILECOUNT の値> + <実行中の osfind の数>) / 0.8 （単位：バイト）

64 ビット用 OS の場合

osfind モジュールトレース容量 = (32×<環境変数 HVI_MTRENTRYCOUNT の値> + 192)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中の osfind の数>) / 0.8 （単位：バイト）

(p) oad モジュールトレース容量の算出式

32 ビット用 OS の場合

oad モジュールトレース容量 = (16×<環境変数 HVI_MTRENTRYCOUNT の値> + 128)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中の oad の数>) / 0.8 （単位：バイト）

64 ビット用 OS の場合

oad モジュールトレース容量 = (32×<環境変数 HVI_MTRENTRYCOUNT の値> + 192)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中の oad の数>) / 0.8 （単位：バイト）

(q) irep モジュールトレース容量の算出式

32 ビット用 OS の場合

irep モジュールトレース容量 = (16×<環境変数 HVI_MTRENTRYCOUNT の値> + 128)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中の irep の数>) / 0.8 （単位：バイト）

64 ビット用 OS の場合

irep モジュールトレース容量 = (32×<環境変数 HVI_MTRENTRYCOUNT の値> + 192)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中の irep の数>) / 0.8 （単位：バイト）

(r) gatekeeper モジュールトレース容量の算出式

Windows だけサポートしています。

gatekeeper モジュールトレース容量 = (16×<環境変数 HVI_MTRENTRYCOUNT の値> + 128)×(<環境変数 HVI_MTRFILECOUNT の値> + <実行中の gatekeeper の数>) / 0.8 （単位：バイト）

(s) ネーミングサービス名前空間情報ログ容量の算出式

ネーミングサービス名前空間情報ログ容量 = (<環境変数 HVI_NAMELOGFILESIZE の値> + 1,024)×2×<環境変数 HVI_NAMELOGFILECOUNT の値> / 0.8 （単位：バイト）

3 ORB の拡張機能

この章では，ORB の運用に必要な拡張機能について説明します。

3.1 ORB の拡張機能の概要

この節では，ORB の運用に必要な拡張機能について説明します。拡張機能では次に示す動作の変更ができます。

- osagent
- ユーザプロセス
- サーバプロセス

ORB の拡張機能は，環境変数およびプロパティを設定することで使用できます。環境変数とプロパティの両方を設定した場合には，プロパティの設定が有効になります。

プロパティの設定方法は，Java ORB も C++ ORB も通常のプロパティと同じです。詳細は，マニュアル「Borland Enterprise Server VisiBroker デベロッパーズガイド」を参照してください。

注意

Java ORB の場合，プロパティだけで設定します。環境変数では設定できません。C++ ORB の場合は，環境変数，およびプロパティで設定できます。

3.2　環境変数とプロパティの一覧

この節では，ORB の拡張機能で設定する環境変数およびプロパティの一覧を示します。

3.2.1　osagent の設定の一覧

osagent に設定する環境変数およびプロパティの一覧を次の表に示します。

表 3-1　osagent の設定の一覧

環境変数	プロパティ	機能	備考
HVI_OSAGENT_CLIENTHANDLERADDR_FILE	なし。	osagent でのマルチホームホスト環境の設定	osagent 限定
HVI_OSAGENT_ALIVE_INTERVAL	なし。	HEARTBEAT メッセージおよび ARE_YOU_ALIVE メッセージの送信間隔	osagent 限定
HVI_OSAGENT_LOCATE_VERBOSE	なし。	バーボースログの出力抑止	osagent 限定
HVI_OSAGENT_TRIGGER_NOUSE	なし。	osagent 間のメッセージの出力抑止	osagent 限定
HVI_OSAGENT_RESEND_MAXCOUNT	なし。	osagent 間のメッセージ送信処理のリトライ回数	osagent 限定
HVI_OSAGENT_RESEND_INTERVAL	なし。	osagent 間のメッセージ送信処理のリトライ間隔	osagent 限定
HVI_DNS_TRYAGAIN_RETRY_COUNT	なし。	アドレス解決処理のリトライ回数	－
HVI_DNS_TRYAGAIN_RETRY_INTERVAL	なし。	アドレス解決処理のリトライ間隔	－
HVI_SIGXCPU_EXIT	なし。	SIGXCPU シグナル受信時の動作変更	UNIX 版 osagent 限定
HVI_SIGXFSZ_EXIT	なし。	SIGXFSZ シグナル受信時の動作変更	UNIX 版 osagent 限定

（凡例）　－：説明なし

3.2.2　ユーザプロセスの設定の一覧

ユーザプロセスに設定する環境変数およびプロパティの一覧を次の表に示します。

表 3-2　ユーザプロセスの設定の一覧

環境変数	プロパティ	機能	使用できる言語		備考
			Java ORB	C++ ORB	
なし。	vbroker.orb.htc.requestTimer	リクエストの監視間隔	○	×	−
なし。	vbroker.orb.htc.strictSequenceAny	sequence<any>のマーシャリング方法の変更	○	×	−
なし。	vbroker.rmi.htc.disableCMClass	java.util.Vector クラスの下位互換の設定	○	×	−
なし。	vbroker.ce.iiop.ccm.htc.readerPerConnection	コネクションのクローズの抑止	○	×	−
なし。	vbroker.ce.iiop.ccm.htc.threadStarter	専用スレッドによる応答電文受信の設定	○	×	−
HVI_SURROGATE_CHECK_OFF	vbroker.orb.htc.surrogateCheckOff	文字コード範囲のチェック抑止	○※	○	−
HVI_CONNECTION_CACHE	vbroker.orb.htc.connectionCache	リファレンス解放時のコネクションキャッシュ抑止	×	○	−
HVI_CONNECTION_RETRY	vbroker.orb.htc.connectionRetry	connect()エラー時の再試行の抑止	×	○	−
HVI_AGENT_ADDR_ONLY	vbroker.agent.htc.addrOnly	osagent 探索方式の切り替え	×	○	−
HVI_TERMINATION_TIMER	vbroker.orb.htc.terminationTimer	終了処理での Sleep のタイマ値	×	○	−
HVI_DNS_TRYAGAIN_RETRY_COUNT	vbroker.orb.htc.tryAgainRetryCount	アドレス解決処理のリトライ回数	×	○	−
HVI_DNS_TRYAGAIN_RETRY_INTERVAL	vbroker.orb.htc.tryAgainRetryInterval	アドレス解決処理のリトライ間隔	×	○	−
HVI_CTRLHANDLER_RTN	vbroker.orb.htc.ctrlHandlerRtn	コントロールハンドラのリターン値	×	○	Windows 限定
HVI_CTRLHANDLER_NOCLEANUP	vbroker.orb.htc.ctrlHandlerNoCleanup	コントロールハンドラ内で行う ORB の終了処理	×	○	Windows 限定
HVI_STRICT_ANY_MARSHALLING	vbroker.orb.htc.strictAnyMarshalling	CORBA::Any 型のマーシャリング方法の変更	×	○	Windows 限定
HVI_UNLIMITED_QOS_TIMEOUT_POLICY	vbroker.orb.htc.unlimitedQosTimeoutPolicy	QoS ポリシーの上限値解除	×	○	32 ビット用

環境変数	プロパティ	機能	使用できる言語		備考
			Java ORB	C++ ORB	
					UNIX 限定
HVI_BIND_MAX	vbroker.orb.htc.bindMax	バインドのリトライ回数	×	○	－
HVI_BIND_SLEEP_TIME	vbroker.orb.htc.bindSleepTime	バインドのリトライ間隔	×	○	－
HVI_SPLIT_RW	vbroker.orb.htc.splitRw	GIOP メッセージの分割送受信	×	○	Windows 限定

（凡例） ○：サポート　×：未サポート　－：説明なし

注※

プロパティだけをサポートしています。環境変数はサポートしていません。

3.2.3　サーバプロセスの設定の一覧

サーバプロセスに設定する環境変数およびプロパティの一覧を次の表に示します。

表 3-3　サーバプロセスの設定の一覧

環境変数	プロパティ	説明	使用できる言語		備考
			Java ORB	C++ ORB	
HVI_OAD_NOUSE	vbroker.orb.htc.oadNoUse	OAD による自動検索の抑止	○※	○	－
HVI_NCATCHALL	vbroker.orb.htc.ncatchall	CORBA::UNKNOWN 例外発生の抑止	×	○	Windows 限定

（凡例） ○：サポート　×：未サポート　－：説明なし

注※

プロパティだけをサポートしています。環境変数はサポートしていません。

3.3 osagent の設定

この節では，osagent に設定する環境変数およびプロパティの詳細について説明します。

3.3.1 osagent でのマルチホームホスト環境の設定

環境変数	プロパティ	指定範囲	デフォルト値
HVI_OSAGENT_CLIENTHANDLERADDR_FILE	なし。	任意のパス名（絶対パスのファイル名）	Windows %VBROKER_ADM%¥htc.clienthandleraddr UNIX ${VBROKER_ADM}/htc.clienthandleraddr

この機能は，htc.clienthandleraddr ファイルとして使用するファイルを設定します。この機能を設定することで，任意のファイルを htc.clienthandleraddr ファイルとして使用できます。

htc.clienthandleraddr ファイルを使用することで，osagent とクライアントアプリケーション間の通信で使用する IP アドレスをカスタマイズし，マルチホームホスト環境で通信できるようになります。htc.clienthandleraddr ファイルには，各ネットワーク上のアプリケーションに対して，osagent が応答する IP アドレスを定義します。

osagent は各アプリケーションと UDP および TCP を使用して通信します。そのため，マルチホームホスト上で起動する osagent がアプリケーションとの通信に使用する IP アドレスは，通常 localaddr ファイルに定義されている適切なインタフェースまたはプライマリ IP アドレスです。プライマリ IP アドレスとは，gethostname()または sysinfo()を利用して得られるホスト名を使用して gethostbyname()を発行して得られた IP アドレスのことをいいます。

しかし，osagent と異なるサブネットに存在するアプリケーションと osagent がプライマリ IP アドレスではないネットワークインタフェースを使用して通信する場合は，osagent とアプリケーション間で，円滑に通信できないことがあります。そのため，htc.clienthandleraddr ファイルに，osagent が各アプリケーションに対して使用する IP アドレスを動的に指定することで，上記の動作で不都合がある環境で，osagent とアプリケーション間の通信が円滑に行えるようになります。

マルチホームホスト環境では，htc.clienthandleraddr ファイル内の osagent が応答する IP アドレスに，次のようにサービス IP アドレスを指定してください。

htc.clienthandleraddr ファイルの記述形式

ファイルの記述形式を次に示します。

```
通信相手のIPアドレス　　サブネットマスク　　osagentが応答するIPアドレス
XXX.XXX.XXX.XXX　　XXX.XXX.XXX.XXX　　XXX.XXX.XXX.XXX
```

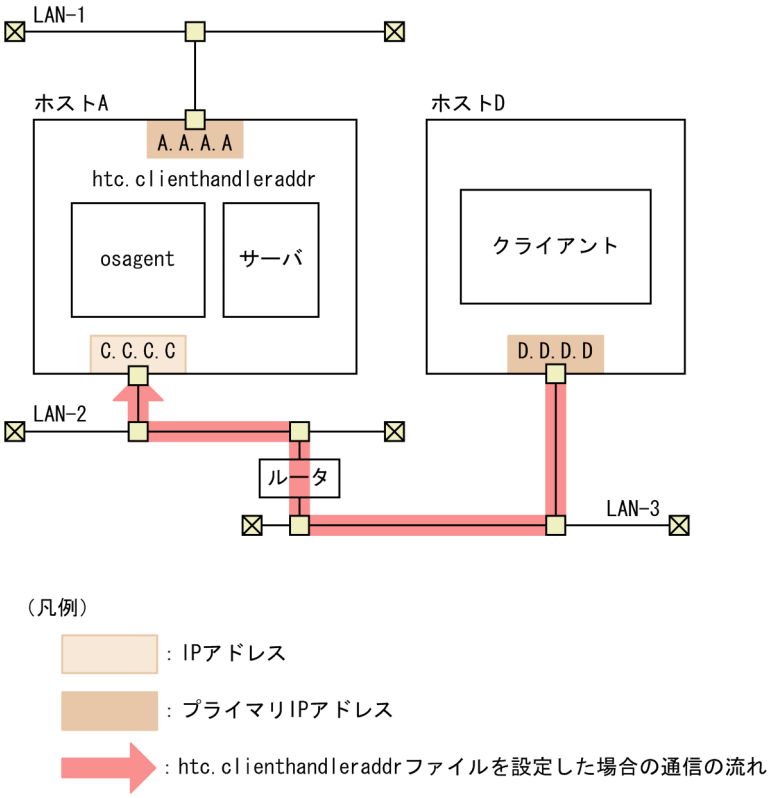
- X は任意の数字です。
- 「#」で始まるエントリは，コメント行と見なされます。

- 行は必ず改行文字で終了します。

htc.clienthandleraddr ファイルの設定例

次の図に示したシステム構成を例に説明します。

図 3-1　マルチホームホスト環境のシステム構成（osagent でのマルチホームホスト環境の設定）



条件

- マルチホームホスト環境で，osagent（ホスト A），クライアント（ホスト D）の間で通信します。htc.clienthandleraddr ファイルは，osagent を起動するホスト（ホスト A）に格納します。
- 設定されている IP アドレスは，次のとおりです。

  A.A.A.A：ホスト A の実 IP（プライマリ IP）アドレス

  C.C.C.C：ホスト A のサービス IP アドレス（osagent が応答する IP アドレス）

  D.D.D.D：ホスト D の実 IP（プライマリ IP）アドレス

htc.clienthandleraddr ファイルの内容

図 3-1 に示したシステム構成で，上記の条件の場合の htc.clienthandleraddr ファイルの内容を次に示します。

```
D.D.D.D <subnet-mask> C.C.C.C
```

注意事項

環境変数 HVI_OSAGENT_CLIENTHANDLERADDR_FILE を設定していない場合，デフォルト値に示した htc.clienthandleraddr ファイルを使用します。

備考

osagent 限定

3.3.2 HEARTBEAT メッセージおよび ARE_YOU_ALIVE メッセージの送信間隔

環境変数	プロパティ	指定範囲	デフォルト値
HVI_OSAGENT_ALIVE_INTERVAL	なし。	60～86400	300

osagent が定期的に自身の存在を通知するメッセージ（HEARTBEAT メッセージ），および認識しているほかの osagent の存在確認メッセージ（ARE_YOU_ALIVE メッセージ）の送信間隔を秒単位で設定します。

注意事項

送信間隔を短くした場合，送信量が多くなることで通信処理に負荷が掛かります。

備考

osagent 限定

3.3.3 バーボースログの出力抑止

環境変数	プロパティ	指定範囲	デフォルト値
HVI_OSAGENT_LOCATE_VERBOSE	なし。	true|false	true

osagent のバーボースログ取得時にスマートファインダコマンド（osfind コマンド）を実行すると，osagent のバーボースログのロケーションサービスに関連する大量のメッセージが出力されます。

この機能は，osagent のバーボースログのロケーションサービスに関連するメッセージの出力を抑止するかどうかを設定します。

- true
  ロケーションサービスに関連するメッセージを抑止しません。
- false
  ロケーションサービスに関連するメッセージを抑止します。

備考

osagent 限定

3.3.4　osagent 間のメッセージの出力抑止

環境変数	プロパティ	指定範囲	デフォルト値
HVI_OSAGENT_TRIGGER_NOUSE	なし。	true|false	false

osagent 間で行われるサーバ起動および終了時のメッセージを抑止するかどうかを設定します。

- true
  サーバ起動および終了時のメッセージを抑止します。
- false
  サーバ起動および終了時のメッセージを抑止しません。

注意事項

トリガー機能を使用しているシステムの場合は，環境変数 HVI_OSAGENT_TRIGGER_NOUSE を設定しないでください。

備考

osagent 限定

3.3.5　osagent 間のメッセージ送信処理のリトライ回数

環境変数	プロパティ	指定範囲	デフォルト値
HVI_OSAGENT_RESEND_MAXCOUNT	なし。	1～59	4

サーバオブジェクトの探索時に，osagent 間で送信する DSAMessage::GET_PROVIDER メッセージの送信回数の最大値を設定します。

備考

osagent 限定

3.3.6　osagent 間のメッセージ送信処理のリトライ間隔

環境変数	プロパティ	指定範囲	デフォルト値
HVI_OSAGENT_RESEND_INTERVAL	なし。	1～59	2

サーバオブジェクトの探索時に，osagent 間で送信する DSAMessage::GET_PROVIDER メッセージの送信間隔を秒単位で設定します。

備考

osagent 限定

3.3.7　アドレス解決処理のリトライ回数

環境変数	プロパティ	指定範囲	デフォルト値
HVI_DNS_TRYAGAIN_RETRY_COUNT	なし。	-1～65535	20

osagent を DNS 環境下で使用している場合に DNS 障害が発生すると，osagent は DNS へのアドレス解決処理を一定回数リトライします。この機能は，DNS へのアドレス解決処理のリトライ回数を設定します。

- -1
  無限にリトライします。
- 0
  リトライしません。
- 1～65535
  設定した数字の回数だけリトライします。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

備考

なし。

3.3.8　アドレス解決処理のリトライ間隔

環境変数	プロパティ	指定範囲	デフォルト値
HVI_DNS_TRYAGAIN_RETRY_INTERVAL	なし。	0～60	0

osagent を DNS 環境下で使用している場合に DNS 障害が発生すると，osagent は DNS へのアドレス解決処理を一定回数リトライします。この機能は，DNS へのアドレス解決処理のリトライ間隔を秒単位で設定します。

- 0
  リトライ間隔を空けません。
- 1～60
  設定した数字の秒数だけリトライ間隔を空けます。

注意事項

指定範囲外の値を設定した場合は，デフォルト値が設定されます。

備考

なし。

3.3.9　SIGXCPU シグナル受信時の動作変更

環境変数	プロパティ	指定範囲	デフォルト値
HVI_SIGXCPU_EXIT	なし。	true|false	false

環境変数 HVI_SIGXCPU_EXIT には osagent の SIGXCPU シグナル受信時の動作を無視にするか，停止にするかの指定をします。

- true
  SIGXCPU シグナル受信時に，プロセスを停止させます。
- false
  SIGXCPU シグナル受信時に，処理を続行します。

注意事項

- 指定範囲外の値が設定された場合，デフォルト値が設定されます。
- osagent に対して，limit などで CPU 時間の制限はしないようにしてください。CPU 時間を制限した場合，osagent の性能に影響を与える場合があります。
- SIGXCPU シグナルを無視する設定（デフォルト動作）とした場合でも，ハードリミットに達した場合は，プロセスは強制終了します。（AIX5L 版，Linux 版のみ）

備考

UNIX 版 osagent 限定

3.3.10　SIGXFSZ シグナル受信時の動作変更

環境変数	プロパティ	指定範囲	デフォルト値
HVI_SIGXFSZ_EXIT	なし	true|false	false

環境変数 HVI_SIGXFSZ_EXIT には osagent の SIGXFSZ シグナル受信時の動作を無視にするか，停止にするかの指定をします。

- true
  SIGXFSZ シグナル受信時に，プロセスを停止させます。
- false
  SIGXFSZ シグナル受信時に，処理を続行します。

注意事項

- 指定範囲外の値が設定された場合，デフォルト値が設定されます。
- osagent に対して，limit などでファイルサイズを制限する場合は，ログまたはトレースの出力ファイルサイズが制限値を越えないように設定してください。また，SIGXFSZ シグナルを無視した場

合（デフォルト動作），ファイルサイズが制限値を越えるとトレースファイルの出力はしません。ログファイルは制限値までのログ出力をします。各種ログ，トレースファイルの容量の詳細は「2.12.2 ディスク占有量」の各トレースファイル容量の算出式を参照してください。

備考

UNIX 版 osagent 限定

3.4　ユーザプロセスの設定

この節では，ORB のライブラリを使用して作成するユーザアプリケーションに設定する環境変数およびプロパティの詳細について説明します。

3.4.1　リクエストの監視間隔

環境変数	プロパティ	指定範囲	デフォルト値
なし。	vbroker.orb.htc.requestTimer	100～1566848000	3000

ORB は一定間隔でリクエストの送信状態（QoS::RelativeRequestTimeoutPolicy）を確認します。この機能は，この間隔をミリ秒単位で設定します。

注意事項

- プロパティ vbroker.orb.htc.requestTimer に設定した監視間隔分，QoS::RelativeRequestTimeoutPolicy によるタイムアウト発生が間延びすることがあります。
- プロパティ vbroker.orb.htc.requestTimer に小さ過ぎる値を設定すると，CPU の占有率が高くなることが要因となって，性能が劣化するおそれがあります。
- ORB プロパティ vbroker.orb.fragmentSize を同時に設定した場合，RelativeRequestTimeout の検知に時間が掛かることがあります。
- 指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB 限定

3.4.2　sequence<any>のマーシャリング方法の変更

環境変数	プロパティ	指定範囲	デフォルト値
なし。	vbroker.orb.htc.strictSequenceAny	true|false	false

sequence<any>について，CORBA 仕様に準拠したマーシャル/アンマーシャルで通信を行うかどうかを設定します。

- true

  sequence<any>について，CORBA 仕様に準拠したマーシャル/アンマーシャルで通信します。
- false

  sequence<any>について，TPBroker 独自のマーシャル/アンマーシャルができるかどうかを自動的に判断して通信します。

サーバ，クライアントのどちらかに"true"を設定することで，sequence<any>を CORBA 仕様に準拠したマーシャル/アンマーシャルで通信を行えます。プロパティ vbroker.orb.htc.strictSequenceAny はプロセス単位で有効になり，プロセスで最初に ORB.init()を発行したときの設定値が有効になります。

TPBroker は，通信相手によって sequence<any>について TPBroker 独自のマーシャル/アンマーシャルができるかどうかを自動的に判断します。そのため，通常はプロパティ vbroker.orb.htc.strictSequenceAny を設定する必要はありませんが，TPBroker Object Transaction Monitor を使用する場合にはこの機能の設定が必要です。TPBroker Object Transaction Monitor での指示に従ってください。

使用言語

Java ORB 限定

3.4.3 java.util.Vector クラスの下位互換の設定

環境変数	プロパティ	指定範囲	デフォルト値
なし。	vbroker.rmi.htc.disableCMClass	java.util.Vector	null

Cosminexus TPBroker 05-18 以降では，Cosminexus TPBroker 05-00，05-10，および 05-11 と java.util.Vector クラスについて互換性がありません。java.util.Vector クラスのオブジェクトを，Cosminexus TPBroker 05-00，05-10，および 05-11 との間で送受信する場合は，MARSHAL 例外またはデータの不整合が発生します。この機能は，java.util.Vector クラスを正しく送受信する機能を使用するかどうかを設定します。

EJB クライアントアプリケーションの場合はシステムプロパティに，Web コンテナサーバまたは J2EE サーバの場合は usrconf.properties に，次のとおりに設定してください。

```
vbroker.rmi.htc.disableCMClass=java.util.Vector
```

注意事項

プロパティ vbroker.rmi.htc.disableCMClass を設定した場合は，該当するプロセスと java.util.Vector クラスとの送受信を行う，Cosminexus のすべてのプロセスに，プロパティ vbroker.rmi.htc.disableCMClass を設定する必要があります。

使用言語

Java ORB 限定

3.4.4 コネクションのクローズの抑止

環境変数	プロパティ	指定範囲	デフォルト値
なし。	vbroker.ce.iiop.ccm.htc.readerPerConnection	true|false	false

Cosminexus TPBroker 05-12 以降，リクエストがタイムアウトした場合に，コネクションをクローズします。コネクションをクローズするため，タイムアウトが発生したときに同じ宛先のサーバと通信中のほかのスレッドには，通信障害が発生します。

この機能は，タイムアウト発生時のコネクションのクローズを抑止するかどうかを設定します。この機能で設定するプロパティはクライアント側で指定します。また，ここで指定した値は ORB ごとに有効になります。

- true
  タイムアウト発生時のコネクションのクローズを抑止します。
- false
  タイムアウト発生時のコネクションのクローズを抑止しません。

注意事項

- "true"を設定した場合，コネクションごとにリプライ受信専用スレッドを起動します。スレッド数を見積もるときに考慮してください。
- 指定範囲外の値を設定した場合は，デフォルト値が設定されます。

使用言語

Java ORB 限定

3.4.5 専用スレッドによる応答電文受信の設定

環境変数	プロパティ	指定範囲	デフォルト値
なし。	vbroker.ce.iiop.ccm.htc.threadStarter	true|false	false

リプライ受信専用スレッドを管理するスレッドを起動するかどうかを設定します。Web アプリケーション，もしくは EJB（J2EE サーバ）で vbroker.ce.iiop.ccm.htc.readerPerConnection=true を設定する場合は，"true"を設定してください。

- true
  リプライ受信専用スレッドを管理するスレッドを起動します。
- false
  リプライ受信専用スレッドを管理するスレッドを起動しません。

注意事項

"true"を設定した場合，リプライ受信専用スレッドを管理するスレッドを，ORB インスタンスごとに一つ起動します。スレッド数を見積もるときに考慮してください。

使用言語

Java ORB 限定

3.4.6　文字コード範囲のチェック抑止

環境変数	プロパティ	指定範囲	デフォルト値
HVI_SURROGATE_CHECK_OFF （C++ ORB 限定）	vbroker.orb.htc.surrogateCheckOff	auto|true|false	auto

コードセットが UTF-16 の場合（TPBroker は UTF-16 だけサポート），0xD800～0xDFFF の範囲にあるデータを，IDL の wchar 型または wstring 型で送受信するかどうかを設定します。RMI-IIOP 通信で，char 型の送受信は IDL の wchar 型と同等に，java.lang.String オブジェクトの送受信は IDL の wstring 型と同等に扱われます。

Unicode の補助文字は，UTF-16 ではサロゲートペアと呼ばれる 0xD800～0xDFFF の範囲にあるデータを対にした値で，表されます。Unicode の補助文字を送受信したい場合は，"auto"（デフォルト値）または"true"を設定してください。

- auto

  通信相手によって，0xD800～0xDFFF の範囲にあるデータを含む wchar 型または wstring 型を送受信するかどうかを自動的に切り替えます。

  次の条件がすべて重なる場合に送受信できます。それ以外は"false"を設定したときと同じ動作をします。

  - 通信相手が TPBroker 05-19 以降である。
  - 通信相手がこの機能に"auto"（デフォルト値）または"true"を設定している。
  - GIOP1.2（TPBroker のデフォルト値）または GIOP1.1 で通信している。
- true

  0xD800～0xDFFF の範囲にあるデータを含む IDL の wchar 型または wstring 型を送受信します。
- false

  TPBroker 05-18 以前と同じ動作です。

  次の条件のどれかの場合に，0xD800～0xDFFF の範囲にあるデータを含む IDL の wchar 型または wstring 型を送受信しません。そのため，例外 DATA_CONVERSION が発生します。

  - Java ORB の場合

    ・IDL の wchar 型を GIOP1.0，GIOP1.1，または GIOP1.2 で送信する。

    ・IDL の wstring 型を GIOP1.2，および vbroker.orb.embedCodeset に"true"（TPBroker 05-18 以前のデフォルト）を設定して送信する。
  - C++ ORB の場合

    ・IDL の wchar 型を GIOP1.1，または GIOP1.2 で送受信する。

    ・IDL の wstring 型を GIOP1.1，または GIOP1.2 で送信する。

注意事項

- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

- プロパティ vbroker.orb.htc.surrogateCheckOff に auto を指定（またはデフォルト）したサーバおよびクライアントが，IDL の wchar/wstring 型で Unicode の補助文字を含むデータを送受信する場合，以下に注意してください。

  設定した値と異なる動作になる場合があります。

  (a) 05-18 以前の TPBroker で作成した IOR ファイルと，05-19 以降の TPBroker で作成した IOR ファイルを混在させないでください。

  (b) 05-19 以降の TPBroker で，上記機能に auto または true を指定して作成した IOR ファイルと，false を指定して作成した IOR ファイルを混在させないでください。

  (c) 上記機能の設定を変更する場合は，サーバとクライアントを共に停止してから設定し直してください。

注　IOR ファイルの混在とは，vbroker.se.xxx.scm.yyy.listener.port を固定して複数の IOR ファイルを作成し，使用することを意味します。

使用言語

環境変数：C++ ORB 限定

プロパティ：Java ORB，C++ ORB

3.4.7　リファレンス解放時のコネクションキャッシュ抑止

環境変数	プロパティ	指定範囲	デフォルト値
HVI_CONNECTION_CACHE	vbroker.orb.htc.connectionCache	true|false	true

オブジェクトリファレンスの解放時にサーバとの接続をキャッシュするかどうかを設定します。

- true

  リファレンス解放時に，プロパティ vbroker.ce.iiop.ccm.connectionCacheMax，およびプロパティ vbroker.ce.iiop.ccm.connectionMaxIdle の指定に従って，サーバとのコネクションをキャッシュします。

- false

  サーバとのコネクションをキャッシュしません。プロパティ vbroker.ce.iiop.ccm.connectionCacheMax，およびプロパティ vbroker.ce.iiop.ccm.connectionMaxIdle の指定は無視されます。

注意事項

- 設定機能範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB 限定

3.4.8　connect()エラー時の再試行の抑止

環境変数	プロパティ	指定範囲	デフォルト値
HVI_CONNECTION_RETRY	vbroker.orb.htc.connectionRetry	true|false	true

コネクトのタイムアウトの指定に関係なく，connect()が ECONNREFUSED になった場合に connect()のリトライを行うかどうかを設定します。

- true
  リトライします。
- false
  リトライしません。QoS ポリシー（RelativeConnectionTimeoutPolicy）に指定した時間をおいて，ECONNREFUSED をリターンします。

使用言語

C++ ORB 限定

3.4.9　osagent 探索方式の切り替え

環境変数	プロパティ	指定範囲	デフォルト値
HVI_AGENT_ADDR_ONLY	vbroker.agent.htc.addrOnly	true|false	false

この機能では，osagent の探索方式を設定します。

通常は，アプリケーションは自身が接続する osagent は，プロパティ vbroker.agent.addr で設定されたホスト，自ホスト，agentaddr ファイルで指定されたホスト，自身のネットワークへのブロードキャストの順番で探索を行います。このとき，最初に応答を返した osagent に接続します。

この機能を設定すると，環境変数 OSAGENT_ADDR，またはプロパティ vbroker.agent.addr で設定されたホストだけに osagent の探索を行うことができます。

- true
  環境変数 OSAGENT_ADDR，またはプロパティ vbroker.agent.addr で設定されたホストだけが対象です。
- false
  プロパティ vbroker.agent.addr で設定されたホスト，自ホスト，agentaddr ファイルで指定されたホスト，自身のネットワークへのブロードキャストの順番で探索を行います。

注意事項

- "true"の場合，環境変数 OSAGENT_ADDR，またはプロパティ vbroker.agent.addr のどちらかを設定します。設定されていない場合，この機能は動作しません。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB 限定

3.4.10　終了処理での Sleep のタイマ値

環境変数	プロパティ	指定範囲	デフォルト値
HVI_TERMINATION_TIMER	vbroker.orb.htc.terminationTimer	0～10000	0

マルチスレッドで UAP が不正に終了したとき，UAP の終了に同期をとるために，ORB の DLL ファイルの終了処理内での Sleep のタイマ値をミリ秒単位で設定します。

- 0
  Sleep をしません。
- 1～10000
  設定した数字分だけ Sleep します。

この機能を設定すると，メインスレッドが子スレッドより先に終了することで発生する異常終了やハングアップを回避できます。

注意事項

- TPBroker 05-16-/A 以降（UNIX）では，この機能の設定は不要です。
- 指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB 限定

3.4.11　アドレス解決処理のリトライ回数

環境変数	プロパティ	指定範囲	デフォルト値
HVI_DNS_TRYAGAIN_RETRY_COUNT	vbroker.orb.htc.tryAgainRetryCount	-1～65535	20

TPBroker を DNS 環境下で使用している場合，DNS 障害が発生すると，TPBroker は DNS へのアドレス解決処理を一定回数リトライします。この機能は，DNS へのアドレス解決処理のリトライ回数を設定します。

- -1
  無限にリトライします。
- 0
  リトライしません。
- 1～65535
  設定した数字の回数だけリトライします。

注意事項

- 指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB 限定

3.4.12　アドレス解決処理のリトライ間隔

環境変数	プロパティ	指定範囲	デフォルト値
HVI_DNS_TRYAGAIN_RETRY_INTERVAL	vbroker.orb.htc.tryAgainRetryInterval	0～60	0

TPBroker を DNS 環境下で使用している場合，DNS 障害が発生すると，TPBroker は DNS へのアドレス解決処理を一定回数リトライします。この機能は，DNS へのアドレス解決処理のリトライ間隔を秒単位で設定することができます。

- 0
  リトライ間隔を空けません。
- 1～60
  設定した数字の秒数だけリトライ間隔を空けます。

注意事項

- 指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB 限定

3.4.13　CTRL_BREAK_EVENT 発生時のコントロールハンドラ内の動作

ORB では CORBA::ORB_init()で SetConsoleCtrlHandler()を用いてコントロールハンドラの登録を行い，コントロールハンドラ内で ORB の終了処理のあとに FALSE をリターンしています。この機能は，コントロールハンドラがリターンする値の設定，および CTRL_BREAK_EVENT の発生時にデフォルトの処理で行っている ORB の終了処理を行うかどうかの設定をする機能です。

(1)　コントロールハンドラのリターン値

環境変数	プロパティ	指定範囲	デフォルト値
HVI_CTRLHANDLER_RTN	vbroker.orb.htc.ctrlHandlerRtn	true|false	false

CTRL_BREAK_EVENT の発生時に，ORB のコントロールハンドラがリターンする値を設定します。

- true
  コントロールハンドラが TRUE をリターンします。
- false
  コントロールハンドラが FALSE をリターンします。

コントロールハンドラがリターンする値 TRUE および FALSE の意味については，Microsoft 社が提供する SetConsoleCtrlHandler 関数の仕様を MSDN などでご確認ください。

注意事項

- この機能を設定していない場合，および指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB（Windows）限定

(2)　コントロールハンドラ内で行う ORB の終了処理

環境変数	プロパティ	指定範囲	デフォルト値
HVI_CTRLHANDLER_NOCLEANUP	vbroker.orb.htc.ctrlHandlerNoCleanup	true|false	false

CTRL_BREAK_EVENT の発生時に，ORB のコントロールハンドラ内で ORB の終了処理を行うかどうかを設定します。ORB の終了処理を行う場合，以降のプロセス上で ORB の機能を使用できません。

- true
  コントロールハンドラ内で ORB の終了処理を行いません。
- false

コントロールハンドラ内で ORB の終了処理を行います。

注意事項

- この機能を使用しない場合（"true"を設定する場合）は，サーバアプリケーション，およびクライアントアプリケーション内で ORB の終了処理を行ってください。終了処理を行わない場合，osagent は終了したアプリケーションの情報を削除できないため，アプリケーションに対して HEARTBEAT メッセージを送信します。この結果，アプリケーションの数によっては osagent やネットワークに負荷が掛かり，クライアントアプリケーションがサーバアプリケーションを認識できないなどの問題が発生することがあります。
- この機能を設定していない場合，および指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB（Windows）限定

3.4.14 CORBA::Any 型のマーシャリング方法の変更

環境変数	プロパティ	指定範囲	デフォルト値
HVI_STRICT_ANY_MARSHALLING	vbroker.orb.htc.strictAnyMarshalling	true|false	true

CORBA::Any のマーシャリング時に，CORBA2.5 仕様に準拠した方法，VisiBroker 独自の方法のどちらかを使用するかを設定します。

TPBroker 05-15-/A（32 ビット用 Windows）または TPBroker 05-00～05-15（64 ビット用 Windows）の C++ ORB と，CORBA::Any に CORBA::WChar 型を格納して通信する場合には，"false"を設定する必要があります。

- true
  CORBA::Any のマーシャリング時に CORBA2.5 仕様に準拠した方法に従ってマーシャリングします。
- false
  CORBA::Any のマーシャリング時に VisiBroker 独自の方法に従ってマーシャリングします。

注意事項

- "false"を設定した場合，次のバージョンの ORB と CORBA::Any に CORBA::WChar 型を格納した通信は行えません。
  ・TPBroker Vesion 5 に対応した Java ORB すべて
  ・TPBroker 05-15 以前（UNIX）の C++ ORB
  ・TPBroker 05-15-/A（Linux）の C++ ORB

- 設定値はシステム全体で統一してください。
  この機能をサポートしていない C++ ORB（Windows）は，"false"を設定したときと同等に動作します。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB（Windows）限定

3.4.15 QoS ポリシーの上限値解除

環境変数	プロパティ	指定範囲	デフォルト値
HVI_UNLIMITED_QOS_TIMEOUT_POLICY	vbroker.orb.htc.unlimitedQosTimeoutPolicy	true|false	false

次に示す QoS ポリシーに設定できる値の上限値は 15668480000000（100 ナノ秒）（18 日 3 時間 14 分 8 秒）です。

- QoSExt::RelativeConnectionTimeoutPolicy
- Messaging::RelativeRequestTimeoutPolicy
- Messaging::RelativeRoundtripTimeoutPolicy

この機能では，QoS ポリシーの上限値を解除するかどうかを設定します。

- true
  上記の QoS ポリシーの上限値を解除します。
- false
  上記の QoS ポリシーの上限値を解除しません。上限値を超える値を設定すると，上限値が設定されます。

注意事項

- この機能に"true"を設定して，上記の QoS ポリシーに 15668480000000 を超えた値を設定した場合，次に示す条件を満たすとタイムアウトを正しく監視できなくなります。
  条件
  QoS ポリシーに設定した時間とタイムアウトイベントの発生時刻を足した時刻が，2038/01/19 03:14:07（UTC 時間）を超える。
- 指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB（32 ビット用 UNIX）限定

3.4.16　バインドのリトライ回数

環境変数	プロパティ	指定範囲	デフォルト値
HVI_BIND_MAX	vbroker.orb.htc.bindMax	-1～65535	-1

osagent からサーバ情報を取得したクライアントが，サーバとのコネクション確立に失敗すると，osagent からサーバ情報を取得して再度コネクトを試みます。この機能は，このリトライ処理を行う回数を設定します。リトライ回数の上限を超えた場合は，リトライの要因となった例外が発生します。

- -1
  無限にリトライします。
- 0
  リトライしません。
- 1～65535
  設定した数字の回数だけリトライします。

この機能が有効となる条件を次に示します。

- <interface_name>::_bind()を発行するとき
- CORBA::ORB::bind()を発行するとき
- サーバオペレーションを呼び出すとき
- 次に示す条件を満たしている場合でサーバ起動時に自身を OAD に登録するとき
  - 環境変数 HVI_OAD_NOUSE，またはプロパティ vbroker.orb.htc.oadNoUse に"true"を設定していない
  - oad が起動している

注意事項

- この機能を設定していない場合，および指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB 限定

3.4.17　バインドのリトライ間隔

環境変数	プロパティ	指定範囲	デフォルト値
HVI_BIND_SLEEP_TIME	vbroker.orb.htc.bindSleepTime	0～600000	50

osagent からサーバ情報を取得したクライアントが，サーバとのコネクション確立に失敗すると，osagent からサーバ情報を取得して再度コネクトを試みます。この機能は，このリトライ処理を行う前に Sleep する時間をミリ秒単位で設定します。

注意事項

- この機能を設定していない場合，および指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB 限定

3.4.18 GIOP メッセージの分割送受信

環境変数	プロパティ	指定範囲	デフォルト値
HVI_SPLIT_RW	vbroker.orb.htc.splitRw	true|false	false

Windows で TCP/IP を使用し，64 キロバイトを超えるデータを送受信しようとした場合，WSAENOBUFS エラーが発生してデータ送受信ができなくなることがあります。この機能は，GIOP メッセージを 1 回当たり 64 キロバイト単位に分割して送受信するかどうかを設定します。

- true
  GIOP メッセージを 1 回当たり 64 キロバイト単位に分割して送受信します。
- false
  GIOP メッセージを分割しないで送受信します。

注意事項

- 指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

C++ ORB（Windows）限定

3.5　サーバプロセスの設定

この節では，CORBA オブジェクトを持つサーバプロセスに設定する環境変数およびプロパティの詳細について説明します。

3.5.1　OAD による自動検索の抑止

環境変数	プロパティ	指定範囲	デフォルト値
HVI_OAD_NOUSE（C++ ORB だけ）	vbroker.orb.htc.oadNoUse	true|false	false

この機能は，OAD による自動検索を抑止するかどうかを設定します。OAD を使用する場合は，"true"を設定しないでください。

- true
  自動検索を抑止します。
- false
  自動検索を抑止しません。

注意事項

- この機能を設定していない場合，および指定範囲外の値を設定した場合は，デフォルト値が設定されます。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。

使用言語

環境変数：C++ ORB 限定
プロパティ：Java ORB，C++ ORB

3.5.2　CORBA::UNKNOWN 例外発生の抑止

環境変数	プロパティ	指定範囲	デフォルト値
HVI_NCATCHALL	vbroker.orb.htc.ncatchall	true|false	false

サーバオペレーション内で CORBA 例外以外の例外が発生した場合，ORB が例外をキャッチしないかどうかを設定します。

- true
  CORBA 例外以外の例外をキャッチしません。
- false

CORBA 例外以外の例外をキャッチします。

注意事項

- 指定範囲外の値を設定した場合の動作を次に示します。
  環境変数 HVI_NCATCHALL：デフォルト値が設定されます。
  プロパティ vbroker.orb.htc.ncatchall：例外 CORBA::BAD_PARAM が発生します。
- プロパティ vbroker.orb.htc.ncatchall を使用する場合は，"htc.props"に設定してください。コマンドラインで設定した場合，環境変数 HVI_NCATCHALL に設定した値が有効になります。
- オブジェクトラッパー機能を使用し，サーバオペレーション内で CORBA 例外以外の例外が発生した場合，この機能は有効にならないため，例外をキャッチします。
- 環境変数 HVI_NCATCHALL およびプロパティ vbroker.orb.htc.ncatchall に"true"を設定した場合，スタックトレースを取得しません。
- クライアントプロセスに対して環境変数 HVI_NCATCHALL およびプロパティ vbroker.orb.htc.ncatchall に"true"を設定した場合，スタックトレースを取得しません。クライアントプロセスにはこの機能を使用しないでください。
- 環境変数とプロパティの両方の設定がある場合はプロパティの設定が有効になります。
- Windows（Visual Studio 2005）版では，構造化例外は対象外です。false を指定してもキャッチされません。

使用言語

C++ ORB（Windows）限定

4 Microsoft Cluster Service との連携

この章では，TPBroker と Microsoft Cluster Service との連携方法について説明します。

4.1 Microsoft Cluster Service との連携でできること

この節では，Microsoft Cluster Service との連携でできることについて説明します。

Microsoft Cluster Service と連携することで，TPBroker をクラスタ構成で運用できます。連携できる OS は，Windows だけです。連携できる TPBroker のプログラムプロダクトは次のとおりです。

- Cosminexus TPBroker
- TPBroker Developer
- TPBroker

また，Microsoft Cluster Service と連携すると，TPBroker では次の場合にフェールオーバーができます。

- ハードウェア障害などクラスタサービスがシステム障害を検出した場合
- ユーザが計画的にフェールオーバーする場合

4.2 Microsoft Cluster Service の導入時の検討

この節では，Microsoft Cluster Service の導入時に検討する必要があることについて説明します。

ご使用のシステムに応じて，TPBroker の機能ごとに環境設定項目を決定します。次のことを検討してください。

- ORB 機能を使用するかどうか
- OTS 機能を使用するかどうか
- ADM 機能を使用するかどうか

基本的な検討事項は，OS のマニュアルを参照してください。

4.2.1 ORB 機能の使用

ご使用のシステムで，ORB 機能を使用するかどうかを決定してください。

Cosminexus TPBroker

Microsoft Cluster Service を使用して，osagent，osagent に接続するプロセス，およびサーバを運用する場合は，ORB 機能に関する IP アドレスとポート番号を設定する必要があります。詳細は「4.3.2 ORB 機能使用時の設定（Microsoft Cluster Service）」を参照してください。

なお，osagent に接続するプロセスとは，ネーミングサービス，CTM デーモン，および osagent を使用する J2EE サーバを指します。この場合，J2EE サーバを起動するときに，usrconf.properties ファイルの vbroker.agent.enableLocator キーに true を指定しています。

TPBroker Developer，または TPBroker

Microsoft Cluster Service を使用して，osagent，osagent に接続するプロセス，およびサーバを運用する場合は，ORB 機能に関する IP アドレスとポート番号を設定する必要があります。詳細は「4.3.2 ORB 機能使用時の設定（Microsoft Cluster Service）」を参照してください。

4.2.2 OTS 機能の使用

ご使用のシステムで，OTS 機能を使用するかどうかを決定してください。

Cosminexus TPBroker

インプロセストランザクションサービスを使用します。インプロセストランザクションサービスと Microsoft Cluster Service との連携方法については，マニュアル「Cosminexus システム構築ガイド」を参照してください。

TPBroker Developer，または TPBroker

クラスタサービスに対応していません。OTS 機能を使用しない運用にしてください。

4.2.3　ADM 機能の使用

ご使用のシステムで，ADM 機能を使用するかどうかを決定してください。

Cosminexus TPBroker

　クラスタサービスに対応していません。

　Microsoft Cluster Service を使用する場合は，Management Server を利用してください。

TPBroker Developer，または TPBroker

　クラスタサービスに対応していません。ADM 機能を使用しない運用にしてください。

4.3 Microsoft Cluster Service との連携時のセットアップ

この節では，Microsoft Cluster Service との連携時のセットアップの方法について説明します。

なお，Microsoft Cluster Service の基本的な設定は，OS のマニュアルを参照してください。ここでは，TPBroker との連携時に必要な手順だけを説明します。

4.3.1 Microsoft Cluster Service のセットアップ

osagent をフェールオーバーの対象とする場合には，Microsoft Cluster Service に「汎用アプリケーション」として登録します。

4.3.2 ORB 機能使用時の設定（Microsoft Cluster Service）

ORB 機能を使用するときに必要な IP アドレスとポート番号を設定します。

(1) osagent の設定（Microsoft Cluster Service）

osagent への IP アドレスの設定について説明します。

(a) IP アドレスの設定（Microsoft Cluster Service）

Microsoft Cluster Service 上で osagent を起動する場合，すべてのノードに次の定義ファイルを設定します。

- localaddr
- htc.clienthandleraddr

各定義ファイルの設定値は，シングルホームホスト環境か，またはマルチホームホスト環境かどうかによって異なります。以降，それぞれの環境での設定値について説明します。

シングルホームホスト環境

シングルホームホスト環境で，Microsoft Cluster Service を使用したクラスタ環境を構築する場合に設定する値を次の表に示します。

表 4-1 シングルホームホスト環境の設定値（Microsoft Cluster Service）

項番	条件	localaddr ファイル	htc.clienthandleraddr ファイル
1	osagent をフェールオーバーの対象にする	• 実 IP アドレス（プライマリ IP アドレス） • サービス IP アドレス	• サービス IP アドレス

項番	条件	localaddr ファイル	htc.clienthandleraddr ファイル
2	osagent をフェールオーバーの対象にしない	• 実 IP アドレス（プライマリ IP アドレス）	• 実 IP アドレス（プライマリ IP アドレス）

それぞれの条件について説明します。

osagent をフェールオーバーの対象にする場合

localaddr ファイルには，実 IP アドレス（プライマリ IP アドレス）および osagent が使用するサービス IP アドレスを必ず設定します。

また，osagent に接続するプロセスに対して，サービス IP アドレスを返却するように htc.clienthandleraddr ファイルを定義します。

osagent をフェールオーバーの対象にしない場合

localaddr ファイルには，実 IP アドレス（プライマリ IP アドレス）を必ず設定します。サービス IP アドレスは設定しないでください。

また，osagent に接続するプロセスに対して，実 IP アドレス（プライマリ IP アドレス）を返却するように htc.clienthandleraddr ファイルを定義します。

マルチホームホスト環境

マルチホームホスト環境で，Microsoft Cluster Service を使用したクラスタ環境を構築する場合に設定する値を次の表に示します。

表 4-2　マルチホームホスト環境の設定値（Microsoft Cluster Service）

項番	条件	localaddr ファイル	htc.clienthandleraddr ファイル
1	osagent をフェールオーバーの対象にする	• 実 IP アドレス（プライマリ IP アドレス） • osagent に明示的に指定したい IP アドレス • サービス IP アドレス	• サービス IP アドレス
2	osagent をフェールオーバーの対象にしない	• 実 IP アドレス（プライマリ IP アドレス） • osagent に明示的に指定したい IP アドレス	• 実 IP アドレス（プライマリ IP アドレス）

それぞれの条件について説明します。

osagent をフェールオーバーの対象にする場合

localaddr ファイルには，実 IP アドレス（プライマリ IP アドレス），osagent に明示的に指定したい IP アドレス，および osagent が使用するサービス IP アドレスを設定します。

また，osagent に接続するプロセスに対して，サービス IP アドレスを返却するように htc.clienthandleraddr ファイルを定義します。

osagent をフェールオーバーの対象にしない場合

localaddr ファイルには，実 IP アドレス（プライマリ IP アドレス）および osagent に明示的に指定したい IP アドレスを必ず設定します。サービス IP アドレスは設定しないでください。

また，osagent に接続するプロセスに対して，実 IP アドレス（プライマリ IP アドレス）を返却するように htc.clienthandleraddr ファイルを定義します。

注意

osagent は，localaddr ファイルに設定された IP アドレスだけを認識します。localaddr ファイルを設定しない場合，osagent はデフォルトで gethostbyname()から得られる IP アドレスを認識します。

(b) osagent への設定（Microsoft Cluster Service）

異なるネットワークドメインの osagent 間の通信で，一方の osagent をフェールオーバーの対象とする場合，対象の osagent と通信をする osagent の定義ファイル（agentaddr ファイル）に，サービス IP アドレスを設定します。

(c) osagent に接続するプロセスへの設定（Microsoft Cluster Service）

osagent と osagent に接続するプロセスを異なるネットワークドメインで起動し，osagent をフェールオーバーの対象とする場合，次に示す定義ファイル，およびオプションのどれかにサービス IP アドレスを設定してください。

- agentaddr ファイル
- 環境変数 OSAGENT_ADDR
- プロパティ vbroker.agent.addr

(2) osagent の接続に関する設定（Microsoft Cluster Service）

Microsoft Cluster Service を使用したクラスタ環境で osagent を起動する場合，次に示す環境変数およびオプションの設定が osagent に必要です。

- 環境変数名：OSAGENT_CLIENT_HANDLER_PORT
  環境変数の設定方法については，マニュアル「Borland Enterprise Server VisiBroker デベロッパーズガイド」を参照してください。
- オプション：-m
  osagent の起動時に，コマンドの引数としてこのオプションを設定します。
  このオプションは，Microsoft Cluster Service を使用したクラスタ環境で，osagent をフェールオーバーの対象とする場合だけ必須です。

(3) サーバの設定（Cosminexus TPBroker）（Microsoft Cluster Service）

Cosminexus TPBroker で，ネーミングサービスをフェールオーバーの対象とする場合，次の設定をしてください。

- プロパティ vbroker.se.iiop_tp.host
  エイリアス IP アドレスまたはエイリアスホスト名を設定します。

- プロパティ vbroker.se.iiop_tp.scm.iiop_tp.listener.port
  任意のポート番号を設定します。
  使用するポート番号は，すべてのノードで同じポート番号が使用できるように管理する必要があります。

(4) サーバの設定（TPBroker Developer，または TPBroker）（Microsoft Cluster Service）

TPBroker Developer，または TPBroker では，次の設定をしてください。

- プロパティ vbroker.se.<xxx>.host
  エイリアス IP アドレス，またはエイリアスホスト名を設定します。
- プロパティ vbroker.se.<xxx>.scm.<yyy>.listener.port
  任意のポート番号を設定します。
  使用するポート番号は，すべてのノードで同じポート番号が使用できるように管理する必要があります。

プロパティ名で，xxx はサーバエンジン，yyy はサーバコネクションマネージャを示します。

(5) ORB 機能使用時の設定例（Microsoft Cluster Service）

ここで示す設定例は，シングルホームホスト環境です。

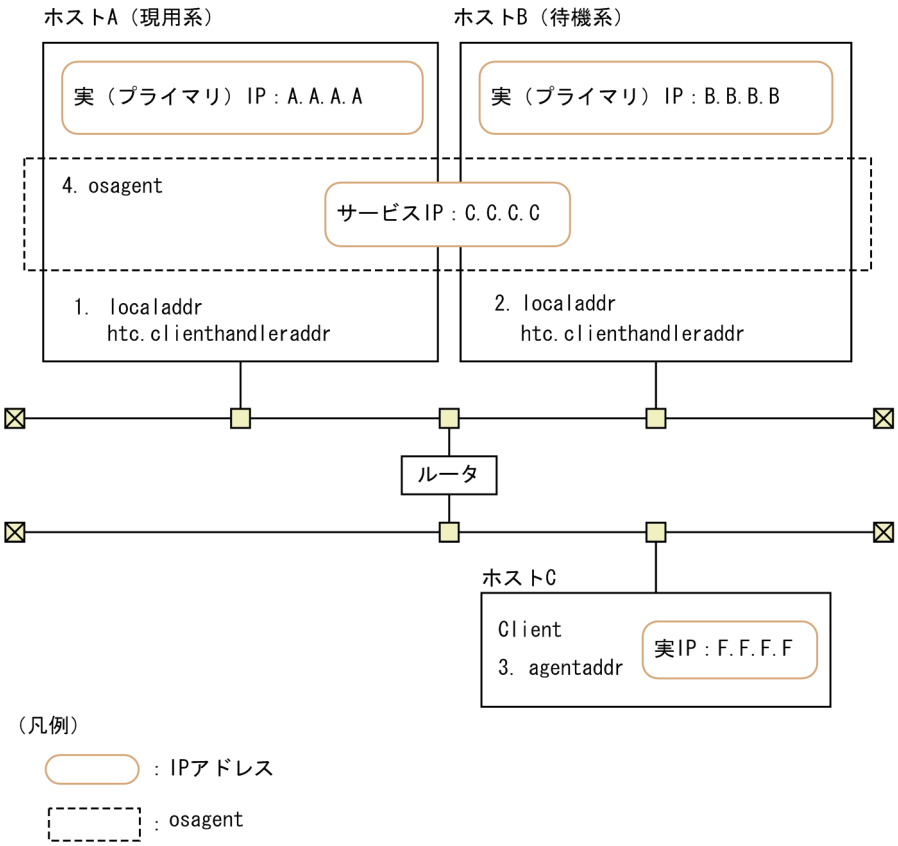
(a) osagent をフェールオーバーの対象にする場合（Microsoft Cluster Service）

osagent をフェールオーバーの対象にする場合の，ORB 機能の使用時の設定例について説明します。

システム構成の例

ORB 機能使用時のシステム構成の例を次の図に示します。

図 4-1　osagent をフェールオーバーの対象にする場合のシステム構成例（Microsoft Cluster Service）



IP アドレスの設定例

上記の図で示すシステム構成例での IP アドレスの設定例を示します。

1. ホスト A（現用系）の IP アドレスの設定
   ・localaddr ファイル
   "A.A.A.A subnet broadcast"（ホスト A の実 IP アドレスを設定）
   "C.C.C.C subnet broadcast"（サービス IP アドレスを設定）
   ・htc.clienthandleraddr ファイル
   "F.F.F.F subnet C.C.C.C"（クライアントの実 IP アドレスにサービス IP アドレスを設定）
2. ホスト B（待機系）の IP アドレスの設定
   ・localaddr ファイル
   "B.B.B.B subnet broadcast"（ホスト B の実 IP アドレスを設定）
   "C.C.C.C subnet broadcast"（サービス IP アドレスを設定）
   ・htc.clienthandleraddr ファイル
   "F.F.F.F subnet C.C.C.C"（クライアントの実 IP アドレスにサービス IP アドレスを設定）
3. ホスト C の IP アドレスの設定
   ・agentaddr ファイル
   "C.C.C.C"（サービス IP アドレスを設定）

4. ホスト A およびホスト B で起動する osagent
   ・環境変数 OSAGENT_CLIENT_HANDLER_PORT
   ・-m オプション（osagent の起動時にコマンドの引数として設定）

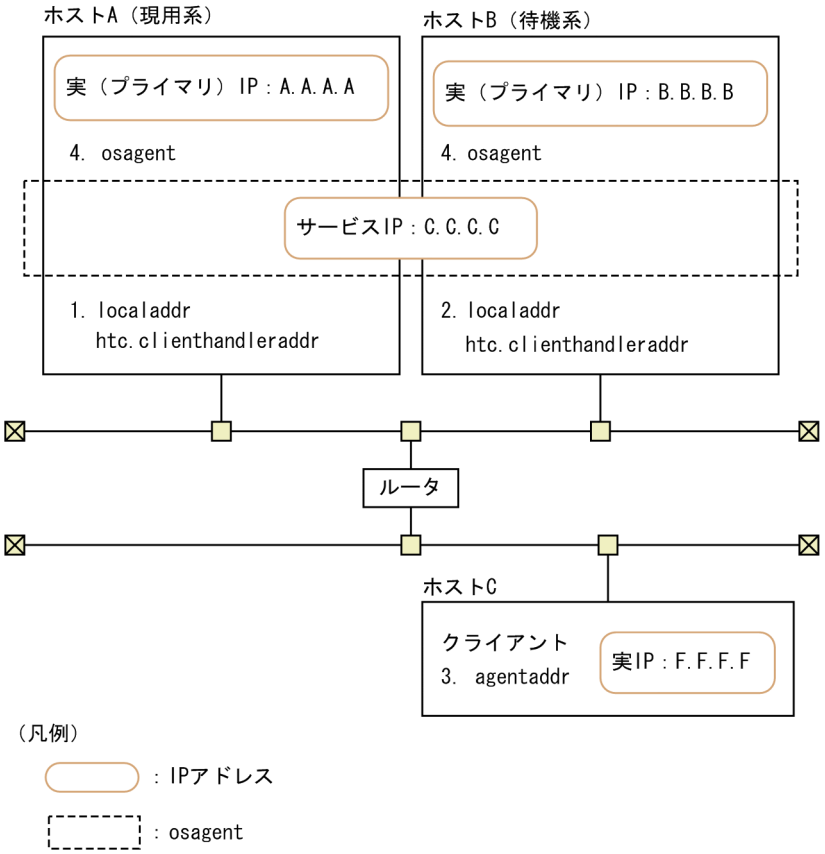
(b) osagent をフェールオーバーの対象にしない場合（Microsoft Cluster Service）

osagent をフェールオーバーの対象にしない場合の，ORB 機能の使用時の設定例について説明します。

システム構成の例

ORB 機能使用時のシステム構成の例を次の図に示します。

図 4-2 osagent をフェールオーバーの対象にしない場合のシステム構成例（Microsoft Cluster Service）



IP アドレスの設定例

上記の図で示すシステム構成例での IP アドレスの設定例を示します。

1. ホスト A（現用系）の IP アドレスの設定
   ・localaddr ファイル
   "A.A.A.A subnet broadcast"（ホスト A の実 IP アドレスを設定）
   ・htc.clienthandleraddr ファイル
   "F.F.F.F subnet A.A.A.A"（クライアントの実 IP アドレスにホスト A の実 IP アドレスを設定）
2. ホスト B（待機系）の IP アドレスの設定
   ・localaddr ファイル

"B.B.B.B subnet broadcast"（ホスト B の実 IP アドレスを設定）

・htc.clienthandleraddr ファイル

"F.F.F.F subnet B.B.B.B"（クライアントの実 IP アドレスにホスト B の実 IP アドレスを設定）

3. ホスト C の IP アドレスの設定

・agentaddr ファイル

"A.A.A.A"（ホスト A の実 IP アドレスを設定）

"B.B.B.B"（ホスト B の実 IP アドレスを設定）

4. ホスト A およびホスト B で起動する osagent

・環境変数 OSAGENT_CLIENT_HANDLER_PORT

5 HA モニタとの連携

この章では，TPBroker と HA モニタとの連携方法について説明します。

5.1　HA モニタとの連携でできること

この節では，HA モニタとの連携でできることについて説明します。

HA モニタと連携することで，TPBroker をクラスタ構成で運用できます。

連携できる OS は次のとおりです。

- AIX
- Linux

連携できる TPBroker のプログラムプロダクトは次のとおりです。

- Cosminexus TPBroker
- TPBroker Developer
- TPBroker

また，HA モニタと連携すると，TPBroker では，次の場合に系切り替えができます。

- ハードウェア障害など HA モニタがシステム障害を検出した場合
- ユーザが計画的に系切り替えする場合

TPBroker は系切り替えをしたときに，次の情報を待機系に引き継ぐことができます。

- 監視中のプロセス情報
- システムの構成情報
- 未決着のトランザクション情報

TPBroker は実行系で監視中のプロセスを，待機系で起動します。再開始用のコマンドが設定してある場合，再開始用のコマンドを起動します。系切り替えが起こった時点で未決着トランザクションがあると，待機系でトランザクションの回復をします。

TPBroker のシステムダウンでは系切り替えは行いません。TPBroker のシステムダウン時は，システムダウンした系で TPBroker を再開始します。

5.2　HA モニタの導入時の検討

この節では，HA モニタの導入時に検討する必要があることについて説明します。

ご使用のシステムに応じて，TPBroker の機能ごとに環境設定項目を決定します。次のことを検討してください。

- ORB 機能を使用するかどうか
- OTS 機能を使用するかどうか
- ADM 機能を使用するかどうか
- どのシステムが共有ディスクを使用するか

導入時の注意事項もありますので，必ず読んでください。基本的な検討事項は，マニュアル「HA モニタ」を参照してください。

5.2.1　ORB 機能の使用（HA モニタ）

ご使用のシステムで，ORB 機能を使用するかどうかを決定してください。

Cosminexus TPBroker

HA モニタを使用して，osagent，osagent に接続するプロセス，およびサーバを運用する場合は，ORB 機能に関する IP アドレスとポート番号を設定する必要があります。詳細は「5.3.5　ORB 機能使用時の設定（HA モニタ）」を参照してください。

なお，osagent に接続するプロセスとは，ネーミングサービス，CTM デーモン，および osagent を使用する J2EE サーバを指します。この場合，J2EE サーバを起動するときに，usrconf.properties ファイルの vbroker.agent.enableLocator キーに true を指定しています。

TPBroker Developer，または TPBroker

HA モニタを使用して，osagent，osagent に接続するプロセス，およびサーバを運用する場合は，ORB 機能に関する IP アドレスとポート番号を設定する必要があります。詳細は「5.3.5　ORB 機能使用時の設定（HA モニタ）」を参照してください。

5.2.2　OTS 機能の使用（HA モニタ）

ご使用のシステムで OTS 機能を使用するかどうかを決定してください。

Cosminexus TPBroker

インプロセストランザクションサービスを使用します。インプロセストランザクションサービスと HA モニタとの連携方法については，マニュアル「Cosminexus システム構築ガイド」を参照してください。

TPBroker Developer，または TPBroker

系切り替えで構成する場合，各系で情報を共有するため，共有ディスク上のディレクトリまたはキャラクタ型スペシャルファイルに環境変数 TPFS を設定します。系切り替えで構成しない場合，ローカルディスク上のディレクトリまたはキャラクタ型スペシャルファイルに環境変数 TPFS を設定します。

注意事項

環境変数 TPFS が設定されていない場合，デフォルトで環境変数 TPSPOOL に設定されたディレクトリが使用されます。環境変数 TPSPOOL では，共有しない情報を格納するため，ローカルディスク上のディレクトリを設定する必要があるので，注意してください。

環境変数 TPFS，および環境変数 TPSPOOL については，マニュアル「TPBroker ユーザーズガイド」を参照してください。

5.2.3 ADM 機能の使用（HA モニタ）

ご使用のシステムで，ADM 機能を使用するかどうかを決定してください。

Cosminexus TPBroker

ADM 機能を使用できません。Management Server で運用することを推奨します。

TPBroker Developer，または TPBroker

ADM 機能をクラスタリングシステムで使用する場合，各系で情報を共有させるため，環境変数 ADMFS を，共有ディスク上のディレクトリに設定します。ADM 機能をクラスタリングシステムで使用しない場合，環境変数 ADMFS を，ローカルディスク上のディレクトリに設定します。

注意事項

環境変数 ADMFS が設定されていない場合，デフォルトで環境変数 ADMSPOOL に設定されたディレクトリが使用されます。環境変数 ADMSPOOL では，共有しない情報を格納するため，ローカルディスク上のディレクトリに設定する必要があるので，注意してください。

環境変数 ADMFS および環境変数 ADMSPOOL については，マニュアル「TPBroker ユーザーズガイド」を参照してください。

5.2.4 共有ディスクを使用するシステム（HA モニタ）

ADM 機能またはアウトプロセストランザクションサービスを使用する場合，共有ディスクを TPBroker ファイルシステムで使用するか，UNIX ファイルシステムで使用するかを決定してください。TPBroker では，TPBroker ファイルシステムを使用することを推奨します。

TPBroker ファイルシステム

環境変数 ADMFS と環境変数 TPFS は同じキャラクタ型スペシャルファイルで共存できます。

TPBroker ファイルシステムを使用できるのは，次のプログラムプロダクトだけです。

- TPBroker Developer（AIX）

- TPBroker（AIX）

UNIX ファイルシステム

環境変数 ADMFS と環境変数 TPFS には異なるディレクトリパスを指定してください。同じディレクトリパスは指定できません。

5.2.5　導入時の注意事項（HA モニタ）

HA モニタとの連携をする場合，導入時には次の点に注意してください。

(1)　システム構成について

- TPBroker の ADM 機能の環境を一つのサーバにしてください。
- すべての系の TPBroker のバージョンを統一してください。
- TPBroker が起動するアプリケーションは，HA モニタの系切り替えに対応させてください。

(2)　設定値について

- すべての系の TPBroker のシステム環境定義を統一してください。
- ADM 機能で使用する定義ファイルをすべての系で統一してください。

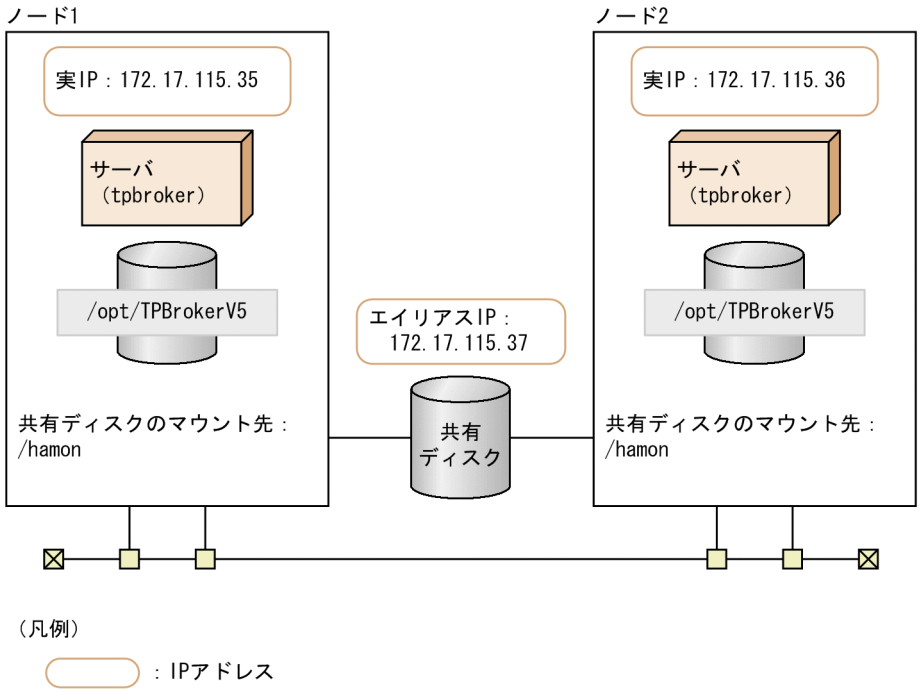
  そのためには，TPBroker が起動するアプリケーションの絶対パスを統一することと，環境変数がすべての系で統一されていることが必要です。
- 環境変数 ADMFS は，共有ディスク上のディレクトリパスまたはキャラクタ型スペシャルファイルを指定してください。
- 共有ディスクには，環境変数 ADMFS，および環境変数 TPFS で示される TPBroker のステータスファイルを格納してください。

  環境変数 TPSPOOL，および環境変数 ADMSPOOL で示される TPBroker 稼働情報格納ディレクトリは，各系のローカルディスクを指定します。また，TPBroker を含むプログラム群は各系のローカルディスクに格納する必要があります。
- アプリケーションで使用するポート番号が固定の場合，すべての系で同じポート番号が使用できるように管理してください。

5.2.6　システムの構成例（HA モニタ）

HA モニタを使用した TPBroker のクラスタリングシステムの構成例を次の図に示します。

図 5-1　HA モニタを使用した TPBroker のクラスタリングシステムの構成例



この構成例の条件を示します。

- 共有ディスクを 2 台で共有し，同じディレクトリでマウントする。
- 実 IP アドレスを 1 台ごとに設定する。
- エイリアス IP アドレスは各マシンに同じものを設定する。

5.3 以降では，ここで示した例に従って環境設定方法を説明します。記載しているスクリプトの値，およびコマンドのオプションの値も，この例に従っています。

5.3　HA モニタとの連携時のセットアップ

この節では，HA モニタとの連携時のセットアップの方法について説明します。

ここで説明するセットアップの方法は，HA モニタ，および TPBroker の基本的な設定を終えていることを前提とし，TPBroker との連携時に必要な手順だけを説明します。セットアップの詳細は，マニュアル「HA モニタ」，およびマニュアル「TPBroker ユーザーズガイド」を参照してください。

5.3.1　セットアップの流れ（HA モニタ）

HA モニタとの連携時のセットアップの流れを次の図に示します。

図 5-2　HA モニタとの連携時のセットアップの流れ

5.3.2　共有ディスクの設定（HA モニタ）

TPBroker では，共有ディスクとして TPBroker ファイルシステムと UNIX ファイルシステムに対応しています。使用するファイルシステムに応じて，共有ディスクに次の項目を設定します。

- 切り替える共有ディスク装置のスペシャルファイル名
- 切り替えるファイルシステムに対応する論理ボリュームの絶対パス名
- 切り替えるファイルシステムのマウント先ディレクトリの絶対パス名（UNIX ファイルシステム限定）

設定方法については，マニュアル「HA モニタ」を参照してください。

5.3.3　TPBroker のセットアップ（HA モニタ）

HA モニタを起動する前に，すべての系で TPBroker のセットアップをする必要があります。すべての系で，次に示す手順を繰り返してください。

(1) 環境変数の設定（HA モニタ）

TPBroker の動作に必要な環境変数を設定します。詳細は，マニュアル「TPBroker ユーザーズガイド」を参照してください。

(2) TPBroker ファイルシステムの作成（HA モニタ）

共有ディスクを TPBroker ファイルシステムで使用する場合，tsmkfs コマンドで TPBroker ファイルシステムを作成します。UNIX ファイルシステムで使用する場合はこの手順は必要ありません。tsmkfs コマンドについては，マニュアル「TPBroker ユーザーズガイド」を参照してください。

この手順は実行系で一度だけ行ってください。また，TPBroker が 05-12 以前のバージョンで作成した TPBroker ファイルシステムを使用する場合もこの手順を行う必要があります。

適用 OS が AIX の場合のコマンドの実行例を次に示します。

1. ボリュームグループの活動を開始します。

```
varyonvg vg00
```

2. TPBroker ファイルシステムを作成します。

```
tsmkfs -s 1024 -n 100 -l 10 /dev/rdisk0
```

tsmkfs コマンドのオプションは，使用しているシステムに応じて適切な値を設定してください。この実行例では，セクタ長 1024，容量 100MB，最大ファイル数 10 の領域を確保します。

3. ボリュームグループの活動を停止します。

```
varyoffvg vg00
```

(3) TPBroker の OTS 環境のセットアップ（HA モニタ）

tssetup コマンドで，TPBroker の OTS 環境のセットアップをします。必ず-i オプションを指定して，ディスクを共有させてください。

環境を再構築する場合など，共用ディスク上に存在する情報を引き継ぐ必要のない場合は，-n オプションを指定して tssetup コマンドを実行してください。OTS 環境が正常開始します。待機系の環境構築など，実行系の情報を引き継がせる場合は，-n オプションを指定しないでください。

tssetup コマンドについては，マニュアル「TPBroker ユーザーズガイド」を参照してください。

コマンドの実行例を次に示します。

共用ディスクの情報を引き継ぐ場合

1. TPBroker の OTS 環境のセットアップをします。

   tssetup -i

共用ディスクの情報を引き継がない場合

1. TPBroker の OTS 環境のセットアップをします。

   tssetup -n -i

(4) システム環境定義の設定（HA モニタ）

tsdefvalue コマンドで HA モニタとの連携に必要なシステム環境定義を設定します。定義パスは，次の表に示す値を設定します。

表 5-1 HA モニタとの連携時に必要なシステム環境定義

定義パス名	値
/ADM/ set_conf_mode	"MANUAL2"
/OTS/ completion_process_ipaddr_info	"<エイリアス IP アドレス>"
/OTS/ set_ipaddr_info	"<エイリアス IP アドレス>"
/SYSTEM/ system_id_info	"<4 バイトの文字列>"

上記以外の項目については，ご使用の環境に合わせた値を設定します。また，すべての系で同じ値を設定します。

tsdefvalue コマンドについては，マニュアル「TPBroker ユーザーズガイド」を参照してください。

コマンドの実行例を次に示します。

1. 開始モードを設定します。

```
tsdefvalue /ADM set_conf_mode -s ”MANUAL2”
```

2. 決着プロセスホスト名を設定します。

```
tsdefvalue /OTS completion_process_ipaddr_info -s ”172.17.115.37”
```

3. デーモンプロセスホスト名を設定します。

```
tsdefvalue /OTS set_ipaddr_info -s ”172.17.115.37”
```

4. システム識別子情報を設定します。

```
tsdefvalue /SYSTEM system_id_info -s ”TPB1”
```

(5) TPBroker の運用支援機能実行環境のセットアップ（HA モニタ）

admsetup コマンドで，TPBroker の運用支援機能実行環境のセットアップをします。必ず-i オプションを指定して，ディスクを共有させてください。なお，-c オプションはすべての系で同じ値（パス）を指定し，指定するプロセス監視定義ファイルは同じ内容としてください。

環境の再構築時など，共用ディスク上に存在する情報を引き継ぐ必要のない場合には，-n オプションを指定して admsetup コマンドを実行してください。ADM 環境が正常開始します。待機系の環境構築など，実行系の情報を引き継がせる場合は，-n オプションを指定しないでください。

admsetup コマンドについては，マニュアル「TPBroker ユーザーズガイド」を参照してください。

コマンドの実行例を次に示します。

共用ディスクの情報を引き継ぐ場合

1. TPBroker の運用支援機能実行環境のセットアップをします。
   admsetup -c /opt/TPBrokerV5/adm/admconf.cf -i

共用ディスクの情報を引き継がない場合

1. TPBroker の運用支援機能実行環境のセットアップをします。
   admsetup -c /opt/TPBrokerV5/adm/admconf.cf -n -i

5.3.4 HA モニタの設定

HA モニタの設定をします。

(1) 定義ファイルの作成（HA モニタ）

使用しているシステムに応じて，HA モニタの定義ファイル（sysdef）を作成します。

(2) LAN の状態設定（HA モニタ）

HA モニタがサーバ単位に LAN の接続，切り離しをするので，サーバごとにネットワークの設定用ファイルが必要です。HA モニタでは，LAN の状態設定ファイルを作成して設定します。

TPBroker が使用する LAN の状態設定ファイルは次のとおりです。エイリアス IP アドレスを設定してください。

- tpbroker.up：LAN を接続する場合に使用します。
- tpbroker.down：LAN の切り離しをする場合に使用します。

LAN の状態設定ファイルの設定例を次に示します。

tpbroker.up の例

```
/etc/ifconfig en0 inet 172.17.115.37 alias netmask 255.255.255.0 broadcast 172.17.115.255
```

tpbroker.down の例

```
# Network Configuration for OFFLINE
#

IFC_NAME="en0"
IP_ADDR="172.17.115.37"
IFCONFIG=/etc/ifconfig

if [ "$IFC_NAME" != "" -a "$IP_ADDR" != "" ]
then
    if [ "`$IFCONFIG $IFC_NAME | grep $IP_ADDR`" != "" ]
    then
       $IFCONFIG $IFC_NAME inet $IP_ADDR delete
    fi
fi
```

(3) TPBroker 開始スクリプトの作成（HA モニタ）

HA モニタから TPBroker を起動するためのスクリプトを作成します。

スクリプトの設定例を次に示します。この例では，TPBroker 開始スクリプト（tpbroker_start.sh）を/home/tpbroker/hamon/bin に格納し，/home/tpbroker/hamon/log にログを出力する設定になっています。

(例)

```
#!/bin/ksh
## ************************************************************
## TPBroker Start Script for HAMonitor
## ************************************************************

LOGDIR=/home/tpbroker/hamon/log
export TPDIR=/opt/TPBrokerV5
export TPSPOOL=${TPDIR}/otsspool
export TPFS=/dev/rdisk0
export ADMSPOOL=${TPDIR}/spool
export ADMFS=/dev/rdisk0
export OSAGENT_PORT=14000
export VBROKER_ADM=${TPDIR}/adm
export LIBPATH=${TPDIR}/lib

logg()
{
        echo `date '+[%Y/%m/%d %H:%M:%S]'`"[$$]: $1" ¥
          >> ${LOGDIR}/tpbroker.log 2>&1
}

logg "### tpbroker_start.sh: started. ###"

#============================================================
```

```
## Start TPBroker
#============================================================

${TPDIR}/bin/admstart >> ${LOGDIR}/tpbroker.log 2>&1
logg "### tpbroker_start.sh: stopped. ###"

exit 0
```

(4) TPBroker 停止スクリプトの作成（HA モニタ）

HA モニタから TPBroker を停止するためのスクリプトを作成します。このスクリプトでは，HA モニタの monend コマンドが実行されたときに，TPBroker が正常停止し，HA モニタの monswap コマンドが実行されたときに，TPBroker が強制停止するようにしています。

スクリプトの設定例を次に示します。この例では，TPBroker 停止スクリプト（tpbroker_stop.sh）を/home/tpbroker/hamon/bin に格納し，/home/tpbroker/hamon/log にログを出力する設定になっています。

(例)

```
#!/bin/ksh
## ************************************************************
## TPBroker Stop Script
## ************************************************************

LOGDIR=/home/tpbroker/hamon/log

export TPDIR=/opt/TPBrokerV5
export TPSPOOL=${TPDIR}/otsspool
export TPFS=/dev/rdisk0
export ADMSPOOL=${TPDIR}/spool
export ADMFS=/dev/rdisk0
export OSAGENT_PORT=14000
export VBROKER_ADM=${TPDIR}/adm
export LIBPATH=${TPDIR}/lib
logg()
{
       echo `date '+[%Y/%m/%d %H:%M:%S]'`"[$$]: $1" ¥
          >> ${LOGDIR}/tpbroker.log 2>&1
}

logg "### tpbroker_stop.sh: started. $* ###"
#============================================================
## Stop TPBroker
#============================================================
case $1 in
-e)
       logg "### tpbroker_stop.sh: TPBroker stop normally. ###"
       ${TPDIR}/bin/admstop >> ${LOGDIR}/tpbroker.log 2>&1
       ;;
-w)
       logg "### tpbroker_stop.sh: TPBroker stop forcibly. ###"
       ${TPDIR}/bin/admstop -fr >> ${LOGDIR}/tpbroker.log 2>&1
       ;;
```

```
-c)
        logg "### tpbroker_stop.sh: do nothing. ###"
        ;;
*)
        logg "### tpbroker_stop.sh: unknown option. $1 ###"
        exit 1

esac

logg "### tpbroker_stop.sh: stopped. ###"

exit 0
```

(5) TPBroker 監視スクリプトの作成（HA モニタ）

HA モニタから TPBroker を監視するスクリプトを作成します。

スクリプトの設定例を次に示します。この例では，TPBroker 監視スクリプト（tpbroker_monitor.sh）を/home/tpbroker/hamon/bin に格納し，/home/tpbroker/hamon/log にログを出力する設定になっています。

(例)

```
#!/bin/ksh
## ************************************************************
## TPBroker Daemon Monitor Script
## ************************************************************
trap break 15

LOGDIR=/home/tpbroker/hamon/log

export TPDIR=/opt/TPBrokerV5
export TPSPOOL=${TPDIR}/otsspool
export TPFS=/dev/rdisk0
export ADMSPOOL=${TPDIR}/spool
export ADMFS=/dev/rdisk0
export OSAGENT_PORT=14000
export VBROKER_ADM=${TPDIR}/adm
export LIBPATH=${TPDIR}/lib

logg()
{
        echo `date '+[%Y/%m/%d %H:%M:%S]'`"[$$]: $1" ¥
          >> ${LOGDIR}/tpbroker.log 2>&1
}

logg "### tpbroker_monitor.sh: started. ###"

loop=true
while $loop
do
       MESSAGE=`${TPDIR}/bin/admstat`
       echo ${MESSAGE} | grep KFCB29001-I > /dev/null
       if [ $? -eq 0 ]; then
             logg "### tpbroker_monitor.sh: TPBroker start. ###"
```

```
            ${TPDIR}/bin/admstart >> ${LOGDIR}/tpbroker.log 2>&1
        fi
        sleep 10
done
```

(6) 定義ファイルの作成（サーバ）（HA モニタ）

TPBroker 用のサーバの定義ファイル（servers）を作成します。TPBroker 用のサーバに固有の設定を次の表に示します。

表 5-2 定義ファイル（servers）の設定

項番	オペランド	設定値	備考
1	name	/home/tpbroker/hamon/bin/tpbroker_start.sh	TPBroker 開始スクリプトを設定します。 詳細は次を参照してください。 「(3) TPBroker 開始スクリプトの作成（HA モニタ）」
2	alias	tpbroker	固定値です。
3	acttype	monitor	固定値です。
4	termcommand	/home/tpbroker/hamon/bin/tpbroker_stop.sh	TPBroker 停止スクリプトを設定します。 詳細は次を参照してください。 「(4) TPBroker 停止スクリプトの作成（HA モニタ）」
5	disk	/dev/vg00	TPBroker で使用する共用ディスクを設定します。
6	lan_updown	use	固定値です。
7	fs_name	/dev/rdisk0	論理ボリュームの絶対パスを設定します。 $TPFS/$ADMFS を UNIX ファイルで使用する場合だけ設定が必要です。
8	fs_mount_dir	/hamon	共有ディスクのマウント先ディレクトリの絶対パス名を指定します。 $TPFS/$ADMFS を UNIX ファイルシステムで使用する場合だけ設定が必要です。
9	patrolcommand	/home/tpbroker/hamon/bin/tpbroker_monitor.sh	TPBroker 監視スクリプトを設定します。 詳細は次を参照してください。 「(5) TPBroker 監視スクリプトの作成（HA モニタ）」

5.3.5 ORB 機能使用時の設定（HA モニタ）

ORB 機能を使用するときに必要な IP アドレスとポート番号を設定します。

次に示す設定をすることで，HA モニタを起動するホスト上で osagent，nameserv，および CORBA アプリケーションを起動できるようになります。

(1) osagent の設定（HA モニタ）

osagent への IP アドレスの設定について説明します。

(a) IP アドレスの設定（HA モニタ）

HA モニタを起動するホスト上で osagent を起動する場合，すべての系に次の定義ファイルを設定します。

- localaddr
- htc.clienthandleraddr

各定義ファイルの設定値は，シングルホームホスト環境か，またはマルチホームホスト環境かどうかによって異なります。以降，それぞれの環境での設定値について説明します。

シングルホームホスト環境

シングルホームホスト環境で，HA モニタを使用したクラスタシステムを構築する場合に設定する値を次の表に示します。

表 5-3 シングルホームホスト環境の設定値（HA モニタ）

項番	条件	localaddr ファイル	htc.clienthandleraddr ファイル
1	osagent を系切り替えの対象にする	• 実 IP アドレス（プライマリ IP アドレス） • エイリアス IP アドレス	• エイリアス IP アドレス
2	osagent を系切り替えの対象にしない	• 実 IP アドレス（プライマリ IP アドレス）	• 実 IP アドレス（プライマリ IP アドレス）

それぞれの条件について説明します。

osagent を系切り替えの対象にする場合

localaddr ファイルには，実 IP アドレス（プライマリ IP アドレス）および osagent が使用するエイリアス IP アドレスを必ず設定してください。

また，osagent が各クライアント（osagent に接続するプロセス）に対して，エイリアス IP アドレスを返却するように htc.clienthandleraddr ファイルを定義してください。

osagent を系切り替えの対象にしない場合

localaddr ファイルには，実 IP アドレス（プライマリ IP アドレス）を必ず設定してください。エイリアス IP アドレスは設定しないでください。

また，osagent が各クライアント（osagent に接続するプロセス）に対して，実 IP アドレス（プライマリ IP アドレス）を返却するように htc.clienthandleraddr ファイルを定義してください。

マルチホームホスト環境

マルチホームホスト環境で，HA モニタを使用したクラスタシステムを構築する場合に設定する値を次の表に示します。

表 5-4　マルチホームホスト環境の設定値（HA モニタ）

項番	条件	localaddr ファイル	htc.clienthandleraddr ファイル
1	osagent を系切り替えの対象にする	• 実 IP アドレス（プライマリ IP アドレス） • osagent に明示的に指定したい IP アドレス • エイリアス IP アドレス	• エイリアス IP アドレス
2	osagent を系切り替えの対象にしない	• 実 IP アドレス（プライマリ IP アドレス） • osagent に明示的に指定したい IP アドレス	• 実 IP アドレス（プライマリ IP アドレス）

それぞれの条件について説明します。

osagent を系切り替えの対象にする場合

localaddr ファイルには，実 IP アドレス（プライマリ IP アドレス），osagent に明示的に指定したい IP アドレス，および osagent が使用するエイリアス IP アドレスを設定します。

また，osagent が各クライアント（osagent に接続するプロセス）に対して，エイリアス IP アドレスを返却するように htc.clienthandleraddr ファイルを定義します。

osagent を系切り替えの対象にしない場合

localaddr ファイルには，実 IP アドレス（プライマリ IP アドレス）および osagent に明示的に指定したい IP アドレスを必ず設定します。エイリアス IP アドレスは設定しないでください。

また osagent が各クライアント（osagent に接続するプロセス）に対して，実 IP アドレス（プライマリ IP アドレス）を返却するように htc.clienthandleraddr ファイルを定義します。

注意

osagent は，localaddr ファイルに設定された IP アドレスだけを認識します。localaddr ファイルを設定しない場合，osagent はデフォルトで gethostbyname()から得られる IP アドレスを認識します。osagent が認識する IP アドレスはバーボースモードで起動させることにより，確認できます。

(b)　osagent への設定（HA モニタ）

異なるネットワークドメインの osagent 間の通信で，一方の osagent を系切り替えの対象とする場合，系切り替え対象の osagent と通信をする osagent の定義ファイル（agentaddr ファイル）に，サービス IP アドレスを設定します。

(c) osagent に接続するプロセスへの設定（HA モニタ）

osagent と osagent に接続するプロセスを異なるネットワークドメインで起動し，osagent を系切り替えの対象とする場合，次に示す定義ファイル，およびオプションのどれかにサービス IP アドレスを設定してください。

- agentaddr ファイル
- 環境変数 OSAGENT_ADDR
- プロパティ vbroker.agent.addr

(2) osagent の接続に関する設定（HA モニタ）

HA モニタ上で osagent を起動する場合，次に示す環境変数の設定が osagent に必要です。

環境変数名：OSAGENT_CLIENT_HANDLER_PORT

環境変数の設定方法については，マニュアル「Borland Enterprise Server VisiBroker デベロッパーズガイド」を参照してください。

(3) サーバの設定（Cosminexus TPBroker）（HA モニタ）

Cosminexus TPBroker では次の設定をします。

ネーミングサービスを系切り替えの対象とする場合

- nameserv コマンドの起動時に次の値を設定します。
  -J-Dvbroker.se.iiop_tp.host=<エイリアス IP アドレス>または<エイリアスホスト名>
- ネーミングサービスが使用するポート番号は，すべての系で同じポート番号が使用できるよう管理する必要があります。

J2EE サーバ，および運用監視エージェントを系切り替えの対象とする場合

J2EE サーバ，および運用監視エージェントのプロパティファイルに次の設定をします。

- プロパティ vbroker.se.iiop_tp.host
  エイリアス IP アドレスまたはエイリアスホスト名を設定します。
- プロパティ vbroker.se.iiop_tp.scm.iiop_tp.listener.port
  任意のポート番号を設定します。

(4) サーバの設定（TPBroker Developer，および TPBroker）（HA モニタ）

TPBroker Developer，および TPBroker で，ネーミングサービスを含む CORBA アプリケーションを系切り替えの対象とする場合，次の設定をします。

- プロパティ vbroker.se.<xxx>.host
  エイリアス IP アドレス，またはエイリアスホスト名を設定します。

- プロパティ vbroker.se.<xxx>.scm.<yyy>.listener.port
  任意のポート番号を設定します。
  使用するポート番号は，すべての系で同じポート番号が使用できるように管理する必要があります。

プロパティ名で，xxx はサーバエンジン，yyy はサーバコネクションマネージャを示します。

(5)　ORB 機能使用時の設定例（HA モニタ）

ここで示す設定例は，シングルホームホスト環境です。

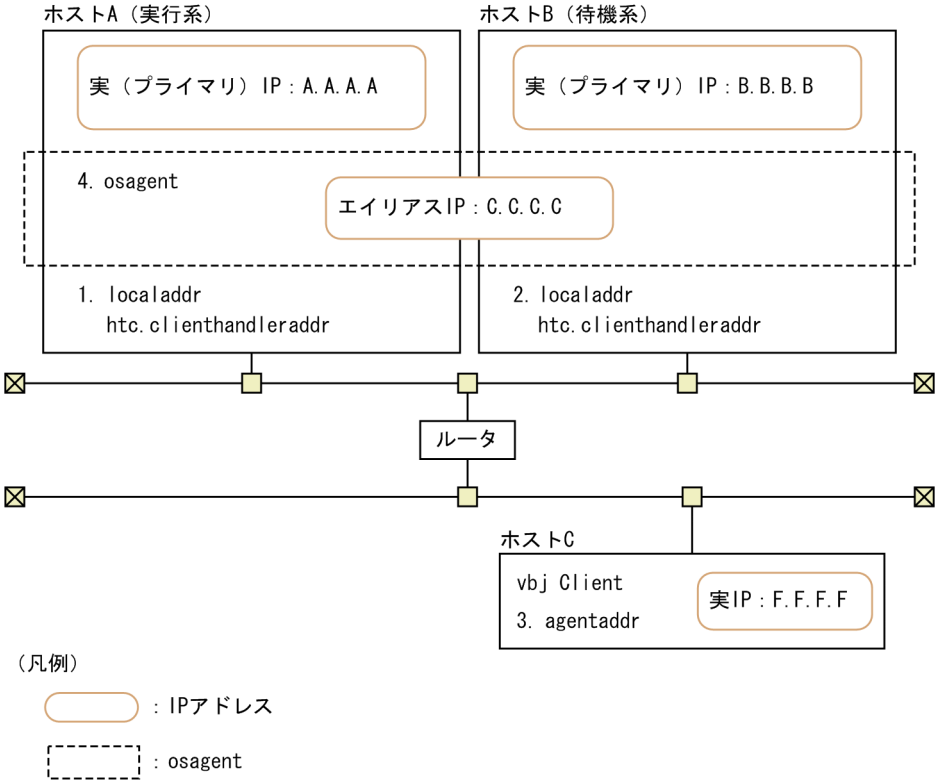
(a)　osagent を系切り替え対象にする場合（HA モニタ）

osagent を系切り替え対象にする場合の，ORB 機能の使用時の設定例について説明します。

システム構成の例

ORB 機能使用時のシステム構成の例を次の図に示します。

図 5-3　osagent を系切り替え対象にする場合のシステム構成例（HA モニタ）



IP アドレスの設定例

上記の図で示すシステム構成例での IP アドレスの設定例を示します。

1. ホスト A（実行系）の IP アドレスの設定

・localaddr ファイル

"A.A.A.A subnet broadcast"（ホスト A の実 IP アドレスを設定）

"C.C.C.C subnet broadcast"（エイリアス IP アドレスを設定）

・htc.clienthandleraddr ファイル

"F.F.F.F subnet C.C.C.C"（クライアントの実 IP アドレスにエイリアス IP アドレスを設定）

2. ホスト B（待機系）の IP アドレスの設定

・localaddr ファイル

"B.B.B.B subnet broadcast"（ホスト B の実 IP アドレスを設定）

"C.C.C.C subnet broadcast"（エイリアス IP アドレスを設定）

・htc.clienthandleraddr ファイル

"F.F.F.F subnet C.C.C.C"（クライアントの実 IP アドレスにエイリアス IP アドレスを設定）

3. ホスト C の IP アドレスの設定

・agentaddr ファイル

"C.C.C.C"（エイリアス IP アドレスを設定）

4. ホスト A およびホスト B で起動する osagent

・環境変数 OSAGENT_CLIENT_HANDLER_PORT

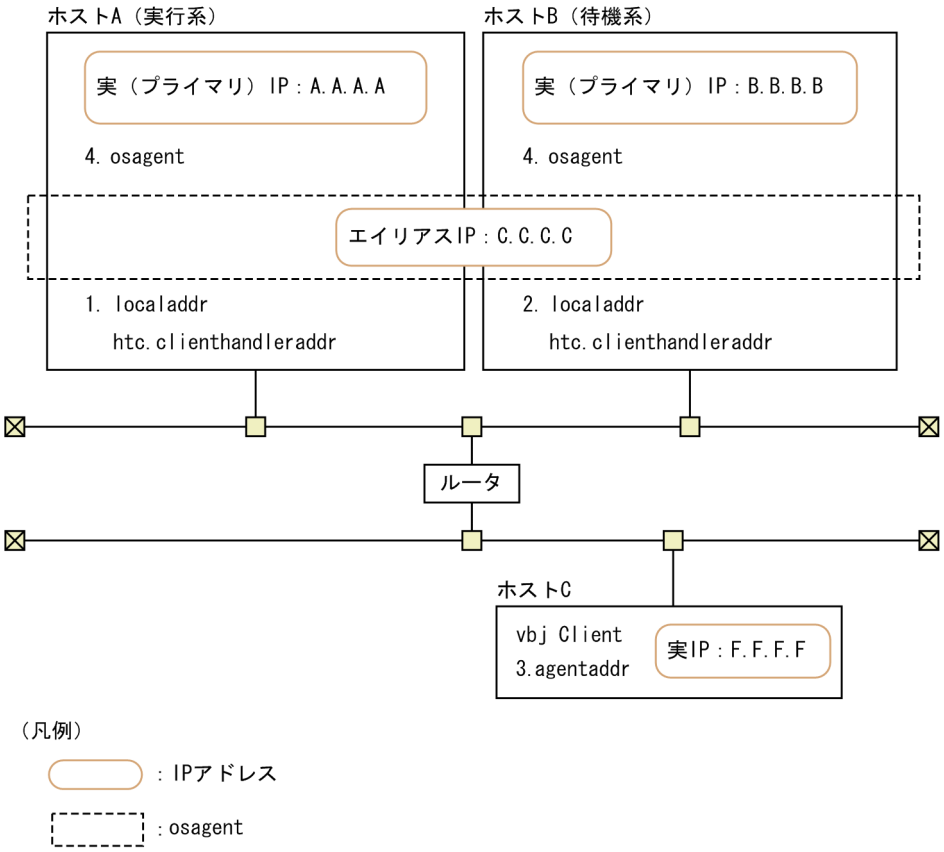
(b) osagent を系切り替え対象にしない場合（HA モニタ）

osagent を系切り替えの対象にしない場合の，ORB 機能の使用時の設定例について説明します。

システム構成の例

ORB 機能使用時のシステム構成の例を次の図に示します。

図 5-4　osagent を系切り替え対象にしない場合のシステム構成例（HA モニタ）



IP アドレスの設定例

上記の図で示すシステム構成例での IP アドレスの設定例を示します。

1. ホスト A（実行系）の IP アドレスの設定
   ・localaddr ファイル
   "A.A.A.A subnet broadcast"（ホスト A の実 IP アドレスを設定）
   ・htc.clienthandleraddr ファイル
   "F.F.F.F subnet A.A.A.A"（クライアントの実 IP アドレスにホスト A の実 IP アドレスを設定）
2. ホスト B（待機系）の IP アドレスの設定
   ・localaddr ファイル
   "B.B.B.B subnet broadcast"（ホスト B の実 IP アドレスを設定）
   ・htc.clienthandleraddr ファイル
   "F.F.F.F subnet B.B.B.B"（クライアントの実 IP アドレスにホスト B の実 IP アドレスを設定）
3. ホスト C の IP アドレスの設定
   ・agentaddr ファイル
   "A.A.A.A"（ホスト A の実 IP アドレスを設定）
   "B.B.B.B"（ホスト B の実 IP アドレスを設定）
4. ホスト A およびホスト B で起動する osagent
   ・環境変数 OSAGENT_CLIENT_HANDLER_PORT

5.4　HA モニタとの連携時の運用

この節では，HA モニタとの連携時の運用について説明します。

次の運用について説明します。

- TPBroker の開始
- TPBroker の停止
- TPBroker の定義の変更

これ以外の運用については，マニュアル「TPBroker ユーザーズガイド」，およびマニュアル「HA モニタ」を参照してください。

5.4.1　TPBroker の開始（HA モニタ）

HA モニタの運用コマンドで，TPBroker 用のサーバ（tpbroker）を開始します。共有ボリュームグループの活動を開始するタイミングは，HA モニタが管理しているため，TPBroker の admstart コマンドで TPBroker を開始させないようにします。

5.4.2　TPBroker の停止（HA モニタ）

HA モニタの運用コマンドで，TPBroker 用のサーバ（tpbroker）を停止します。TPBroker の admstop コマンドでは，TPBroker は一時的にしか停止しません。

5.4.3　TPBroker の定義の変更（HA モニタ）

TPBroker のシステム環境定義，および監視対象プロセスのカレントディレクトリが作成，削除されるタイミングの定義を変更する場合の手順について説明します。

変更手順は，TPBroker 停止スクリプトの内容によって異なります。

monend 投入時に TPBroker が正常停止するスクリプトの場合

1. 定義を変更します。
   必ずすべての系で同じ設定をしてください。
   システム環境定義
   tsdefvalue コマンドで定義を変更します。
   監視対象プロセスのカレントディレクトリが作成，削除されるタイミングの定義
   admsetup コマンドの-c オプションで指定したプロセス監視定義ファイルを編集します。
2. サーバ（tpbroker）を停止します。

monend コマンドを実行します。

monend 投入時に TPBroker が強制停止するスクリプトの場合

1. 定義を変更します。

   必ずすべての系で同じ設定をしてください。

   システム環境定義

   tsdefvalue コマンドで定義を変更します。

   監視対象プロセスのカレントディレクトリが作成，削除されるタイミングの定義

   admsetup コマンドの-c オプションで指定したプロセス監視定義ファイルを編集します。

2. TPBroker 用のサーバ（tpbroker）が起動している系で，TPBroker がオンラインであることを確認します。

   オンラインのときには，admstat コマンドを実行すると，TPBroker の稼働状況が表示されます。表示されないときには，開始中なので，オンラインになるまで待ってください。

3. TPBroker を正常停止します。

   admstop コマンドを実行します。

5.5　HA モニタとの連携時のアンセットアップ

この節では，HA モニタとの連携時のアンセットアップについて説明します。

TPBroker をアンセットアップする系ごとに次の手順をしてください。

1. HA モニタを停止します。
2. TPBroker の運用支援実行環境をアンセットアップします。
   次のコマンドを実行します。

```
admsetup -d
```

3. TPBroker の OTS 環境をアンセットアップします。
   次のコマンドを実行します。

```
tssetup -d
```

6 HACMP との連携

この章では，TPBroker と HACMP との連携方法について説明します。

6.1 HACMP との連携でできること

この節では，HACMP との連携でできることについて説明します。

HACMP と連携することで，TPBroker をクラスタ構成で運用できます。HACMP の詳細は，HACMP のマニュアルを参照してください。なお，TPBroker では HACMP5.3，HACMP5.4，HACMP5.4.1 との連携を確認しています。連携できる OS は AIX だけです。連携できる TPBroker のプログラムプロダクトは次のとおりです。

- TPBroker Developer
- TPBroker

また，HACMP と連携すると，TPBroker では，次の場合にノード引き継ぎができます。

- ハードウェア障害など HACMP がシステム障害を検出した場合
- ユーザが計画的にノード引き継ぎする場合

TPBroker はノード引き継ぎをしたときに，次の情報を現用系から待機系に引き継ぐことができます。

- 監視中のプロセス情報
- システムの構成情報
- 未決着のトランザクション情報

TPBroker は現用系で監視中のプロセスを，待機系で起動します。再開始用のコマンドが設定してある場合，再開始用のコマンドを起動します。ノード引き継ぎが起こった時点で未決着トランザクションがあると，待機系でトランザクションの回復をします。

TPBroker のシステムダウンではノード引き継ぎは行いません。TPBroker のシステムダウン時は，システムダウンした系で TPBroker を再開始します。

6.2　HACMP の導入時の検討

この節では，HACMP の導入時に検討する必要があることについて説明します。

ご使用のシステムに応じて，TPBroker の機能ごとに環境設定項目を決定します。次のことを検討してください。

- ORB 機能を使用するかどうか
- OTS 機能を使用するかどうか
- ADM 機能を使用するかどうか
- どのシステムが共有ディスクを使用するか

導入時の注意事項もありますので，必ず読んでください。基本的な検討事項は，HACMP のマニュアルを参照してください。

6.2.1　ORB 機能の使用（HACMP）

ご使用のシステムで，ORB 機能を使用するかどうかを決定してください。

ORB 機能をクラスタ構成で使用する場合，osagent や CORBA サーバを起動する環境で，各種 IP アドレスを設定する必要があります。詳細は「6.3.5　ORB 機能使用時の設定（HACMP）」を参照してください。

6.2.2　OTS 機能の使用（HACMP）

ご使用のシステムで OTS 機能を使用するかどうかを決定してください。

OTS 機能をクラスタ構成で使用する場合，環境変数 TPFS を，共有ディスク上のディレクトリまたはキャラクタ型スペシャルファイルに設定し，各系で情報を共有します。

OTS 機能をクラスタ構成で使用しない場合，環境変数 TPFS を，ローカルディスク上のディレクトリまたはキャラクタ型スペシャルファイルに設定します。

環境変数 TPFS が設定されていないときには，デフォルトで環境変数 TPSPOOL に設定されたディレクトリが使用されます。ただし，共有しない情報を格納するため，環境変数 TPSPOOL の値は，ローカルディスク上のディレクトリに設定する必要があります。

6.2.3　ADM 機能の使用（HACMP）

ご使用のシステムで，ADM 機能を使用するかどうかを決定してください。

ADM 機能をクラスタ構成で使用する場合，環境変数 ADMFS を，共有ディスク上のディレクトリに設定し，各系で情報を共有します。

ADM 機能をクラスタ構成で使用しない場合，環境変数 ADMFS を，ローカルディスク上のディレクトリに設定します。

環境変数 ADMFS が設定されていないときには，デフォルトで環境変数 ADMSPOOL に設定されたディレクトリが使用されます。ただし，共有しない情報を格納するため，環境変数 ADMSPOOL の値は，ローカルディスク上のディレクトリに設定する必要があります。

6.2.4　共有ディスクを使用するシステム（HACMP）

ADM 機能または OTS 機能を使用する場合，共有ディスクを TPBroker ファイルシステムか，UNIX ファイルシステムで使用するかを決定してください。

TPBroker ファイルシステム

環境変数 ADMFS と環境変数 TPFS は，同じキャラクタ型スペシャルファイルで共存できます。

UNIX ファイルシステム

環境変数 ADMFS と環境変数 TPFS には，異なるディレクトリパスを指定してください。同じディレクトリパスは指定できません。

6.2.5　導入時の注意事項（HACMP）

HACMP との連携をして，クラスタ構成で使用する場合，導入時には次の点に注意してください。

(1)　システム構成について

- TPBroker の ADM 機能の環境を一つのリソース・グループにしてください。
- すべての系の TPBroker のバージョンを統一してください。
- TPBroker が起動するアプリケーションは，HACMP のノード引き継ぎに対応させてください。

(2)　設定値について

- すべての系の TPBroker のシステム環境定義を統一してください。
- OTS 機能で使用する RM の情報をすべての系で統一してください。
- ADM 機能で使用する定義ファイルをすべての系で統一してください。

  そのためには，TPBroker が起動するアプリケーションの絶対パスを統一することと，環境変数がすべての系で統一されていることが必要です。

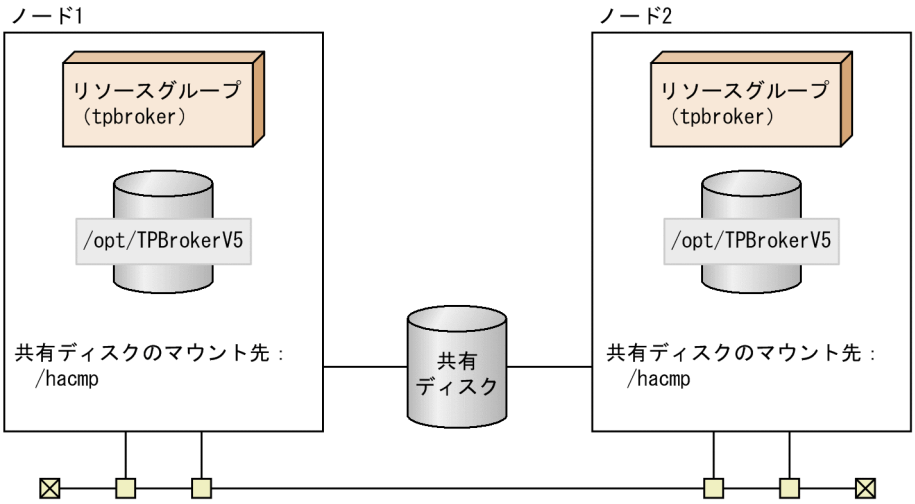
- 環境変数 ADMFS は，共有ディスク上のディレクトリパスまたはキャラクタ型スペシャルファイルを指定してください。
- 共有ディスクには，環境変数 ADMFS，および環境変数 TPFS で示される TPBroker のステータスファイルを格納してください。

  環境変数 TPSPOOL，および環境変数 ADMSPOOL で示される TPBroker 稼働情報格納ディレクトリは，各系のローカルディスクを指定します。また，TPBroker を含むプログラム群は各系のローカルディスクに格納する必要があります。
- アプリケーションで使用するポート番号が固定の場合，すべての系で同じポート番号が使用できるように管理してください。

6.2.6 システムの構成例（HACMP）

HACMP を使用した TPBroker のクラスタ構成の例を次の図に示します。

図 6-1 HACMP を使用した TPBroker のクラスタ構成の例



この構成例の条件を示します。

- 共有ディスクを 2 台で共有し，同じディレクトリでマウントする。
- サービス IP アドレス，およびサービス IP ラベルを共有ディスクに設定する。

6.3 以降では，ここで示した例に従って環境設定方法を説明します。記載しているスクリプトの値，およびコマンドのオプションの値も，この例に従っています。

6.3　HACMP との連携時のセットアップ

この節では，HACMP との連携時のセットアップの方法について説明します。

ここで説明するセットアップの方法は，HACMP，および TPBroker の基本的な設定を終えていることを前提とし，TPBroker との連携時に必要な手順だけを説明します。セットアップの詳細は，HACMP のマニュアル，およびマニュアル「TPBroker ユーザーズガイド」を参照してください。

6.3.1　セットアップの流れ（HACMP）

HACMP との連携時のセットアップの流れを次の図に示します。

図 6-2　HACMP との連携時のセットアップの流れ

6.3.2　共有ディスクの設定（HACMP）

TPBroker は，共有ディスクとして TPBroker ファイルシステムと UNIX ファイルシステムに対応しています。使用するファイルシステムに応じて，共有ディスクに次の項目を設定します。

- 切り替える共有ディスク装置のスペシャルファイル名
- 切り替えるファイルシステムに対応する論理ボリュームの絶対パス名
- 切り替えるファイルシステムのマウント先ディレクトリの絶対パス名（UNIX ファイルシステム限定）

設定方法については，HACMP のマニュアルを参照してください。

6.3.3　TPBroker のセットアップ（HACMP）

HACMP を起動する前に，すべての系で TPBroker のセットアップをする必要があります。すべての系で，次に示す手順を繰り返してください。

(1) 環境変数の設定（HACMP）

TPBroker の動作に必要な環境変数を設定します。詳細は，マニュアル「TPBroker ユーザーズガイド」を参照してください。

(2) TPBroker ファイルシステムの作成（HACMP）

共有ディスクを TPBroker ファイルシステムで使用する場合，tsmkfs コマンドで TPBroker ファイルシステムを作成します。UNIX ファイルシステムで使用する場合はこの手順は必要ありません。tsmkfs コマンドについては，マニュアル「TPBroker ユーザーズガイド」を参照してください。

この手順は現用系で一度だけ行ってください。また，TPBroker が 05-12 以前のバージョンで作成した TPBroker ファイルシステムを使用する場合もこの手順を行う必要があります。

コマンドの実行例を次に示します。

1. ボリュームグループの活動を開始します。

```
varyonvg vg00
```

2. TPBroker ファイルシステムを作成します。

```
tsmkfs -s 1024 -n 100 -l 10 /dev/rdisk0
```

tsmkfs コマンドのオプションは，使用しているシステムに応じて適切な値を設定してください。この実行例では，セクタ長 1024，容量 100MB，最大ファイル数 10 の領域を確保します。

3. ボリュームグループの活動を停止します。

```
varyoffvg vg00
```

(3) TPBroker の OTS 環境のセットアップ（HACMP）

tssetup コマンドで，TPBroker の OTS 環境のセットアップをします。必ず-i オプションを指定して，ディスクを共有させてください。

環境を再構築する場合など，共用ディスク上に存在する情報を引き継ぐ必要のない場合は，-n オプションを指定して tssetup コマンドを実行してください。OTS 環境が正常開始します。待機系の環境構築など，現用系の情報を引き継がせる場合は，-n オプションを指定しないでください。

tssetup コマンドについては，マニュアル「TPBroker ユーザーズガイド」を参照してください。

コマンドの実行例を次に示します。

共用ディスクの情報を引き継ぐ場合

1. TPBroker の OTS 環境のセットアップをします。

   tssetup -i

共用ディスクの情報を引き継がない場合

1. TPBroker の OTS 環境のセットアップをします。

   tssetup -n -i

(4) システム環境定義の設定（HACMP）

tsdefvalue コマンドで HACMP との連携に必要なシステム環境定義を設定します。定義パスは，次の表に示す値を設定します。

表 6-1 HACMP との連携時に必要なシステム環境定義

定義パス名	値
/ADM/ set_conf_mode	"MANUAL2"
/OTS/ completion_process_ipaddr_info	"<サービス IP アドレス>"
/OTS/ set_ipaddr_info	"<サービス IP アドレス>"
/SYSTEM/ system_id_info	"<4 バイトの文字列>"

上記以外の項目については，ご使用の環境に合わせた値を設定します。また，すべての系で同じ値を設定します。

tsdefvalue コマンドについては，マニュアル「TPBroker ユーザーズガイド」を参照してください。

コマンドの実行例を次に示します。

1. 開始モードを設定します。

```
tsdefvalue /ADM set_conf_mode -s ”MANUAL2”
```

2. 決着プロセスホスト名を設定します。

```
tsdefvalue /OTS completion_process_ipaddr_info -s ”172.17.115.37”
```

3. デーモンプロセスホスト名を設定します。

```
tsdefvalue /OTS set_ipaddr_info -s ”172.17.115.37”
```

4. システム識別子情報を設定します。

```
tsdefvalue /SYSTEM system_id_info -s ”TPB1”
```

(5) TPBroker の運用支援機能実行環境のセットアップ（HACMP）

admsetup コマンドで，TPBroker の運用支援機能実行環境のセットアップをします。必ず-i オプションを指定して，ディスクを共有させてください。なお，-c オプションはすべての系で同じ値（パス）を指定し，指定するプロセス監視定義ファイルは同じ内容としてください。

環境の再構築時など，共用ディスク上に存在する情報を引き継ぐ必要のない場合には，-n オプションを指定して admsetup コマンドを実行してください。ADM 環境が正常開始します。待機系の環境構築など，現用系の情報を引き継がせる場合は，-n オプションを指定しないでください。

admsetup コマンドについては，マニュアル「TPBroker ユーザーズガイド」を参照してください。

コマンドの実行例を次に示します。

共用ディスクの情報を引き継ぐ場合

1. TPBroker の運用支援機能実行環境のセットアップをします。
   admsetup -c /opt/TPBrokerV5/adm/admconf.cf -i

共用ディスクの情報を引き継がない場合

1. TPBroker の運用支援機能実行環境のセットアップをします。
   admsetup -c /opt/TPBrokerV5/adm/admconf.cf -n -i

6.3.4 HACMP の設定

HACMP の設定をします。

(1) サービス IP ラベル/アドレスの設定（HACMP）

サービス IP ラベル/アドレスを設定します。設定例を次の表に示します。

表 6–2 サービス IP ラベル，およびサービス IP アドレスの設定例

項番	項目	値	備考
1	IP ラベル/アドレス	node1_svc	サービス IP ラベルを設定します。
2	ネットワーク名	net_either_01	HACMP 環境で，ネットワークを識別するための値を任意に設定します。

(2) TPBroker 開始スクリプトの作成（HACMP）

HACMP から TPBroker を起動するためのスクリプトを作成します。

スクリプトの設定例を次に示します。この例では，TPBroker 開始スクリプト（tpbroker_start.sh）を/home/tpbroker/hacmp/bin に格納し，/home/tpbroker/hacmp/log にログを出力する設定になっています。

(例)

```
#!/bin/ksh
## ************************************************************
## TPBroker Start Script for HACMP
## ************************************************************

#============================================================
#  Execute TPBroker monitor script
#============================================================
printf "### tpbroker_start.sh: PID:%d DATE:%s ###¥n" $$ "`date`" ¥
  >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
/home/tpbroker/hacmp/bin/tpbroker_monitor.sh &
printf "### tpbroker_start.sh: monitor-PID:%d ###¥n" $! ¥
  >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1

exit 0
```

(3) TPBroker 停止スクリプトの作成（HACMP）

HACMP から TPBroker を停止するためのスクリプトを作成します。

このスクリプトでは，共有ディスクをアクセスしているプロセスを確実に停止するため，fuser コマンドで共有ディスクにアクセス中のプロセスを停止するようにしています。fuser コマンドは，共有ディスクに TPBroker ファイルシステムを使用する場合と，UNIX ファイルシステムを使用する場合とで指定するオプションが異なりますので，使用する環境に合わせて指定してください。fuser コマンドについては，OS のマニュアルを参照してください。

スクリプトの設定例を次に示します。この例では，TPBroker 停止スクリプト（tpbroker_stop.sh）を/home/tpbroker/hacmp/bin に格納し，/home/tpbroker/hacmp/log にログを出力する設定になっています。

(例)

```
#!/bin/ksh
## ************************************************************
## TPBroker Stop Script
## ************************************************************

export TPDIR=/opt/TPBrokerV5
export TPSPOOL=/opt/TPBrokerV5/otsspool
export TPFS=/dev/rdisk0
export ADMSPOOL=/opt/TPBrokerV5/spool
export ADMFS=/dev/rdisk0
export OSAGENT_PORT=14000
export VBROKER_ADM=$TPDIR/adm
export LIBPATH=$TPDIR/lib

printf "### tpbroker_stop.sh: PID:%d DATE:%s ###¥n" $$ "`date`" ¥
  >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
#============================================================
## Stop TPBroker monitor script
```

```
#============================================================
fuser -k /home/tpbroker/hacmp/bin/tpbroker_monitor.sh ¥
  >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
#============================================================
## Stop TPBroker
#============================================================
printf "### tpbroker_stop.sh: TPBroker stop DATE:%s ###¥n" "`date`" ¥
  >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
$TPDIR/bin/admstop -fr >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
#============================================================
## Kill all process accessing shared disk
#============================================================
fuser -k /dev/rdisk0 >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
#fuser -kc /hacmp >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
exit 0
```

(4) TPBroker 監視スクリプトの作成（HACMP）

HACMP から TPBroker のダウンを監視し，再起動をするスクリプトを作成します。

このスクリプトは TPBroker 開始スクリプトで TPBroker が実行され，TPBroker 停止スクリプトで TPBroker が停止されるようにしています。設定例を使用する場合，fuser コマンドを使用して停止するため，リソース・グループごとにスクリプトを用意する必要があります。また，リソース・グループを停止するときには，このスクリプトにアクセスしているプロセスは停止されますので，TPBroker の監視以外の目的でアクセスしないようにしてください。

スクリプトの設定例を次に示します。この例では，TPBroker 監視スクリプト（tpbroker_monitor.sh）を/home/tpbroker/hacmp/bin に格納し，/home/tpbroker/hacmp/log にログを出力する設定になっています。

(例)

```
#!/bin/ksh
## ************************************************************
## TPBroker Daemon Monitor Script
## ************************************************************
trap break 15

export TPDIR=/opt/TPBrokerV5
export TPSPOOL=/opt/TPBrokerV5/otsspool
export TPFS=/dev/rdisk0
export ADMSPOOL=/opt/TPBrokerV5/spool
export ADMFS=/dev/rdisk0
export OSAGENT_PORT=14000
export VBROKER_ADM=$TPDIR/adm
export LIBPATH=$TPDIR/lib

printf "### tpbroker_monitor.sh: PID:%d DATE:%s ###¥n" $$ "`date`" ¥
  >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1

loop=true
while $loop
do
```

```
  MESSAGE=`$TPDIR/bin/admstat`
  echo $MESSAGE | grep KFCB29001-I > /dev/null
  if [ $? -eq 0 ]; then
printf "### tpbroker_monitor.sh: TPBroker start: DATE:%s ###¥n" "`date`" ¥
      >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
    $TPDIR/bin/admstart >> /home/tpbroker/hacmp/log/tpbroker.log 2>&1
  fi
  sleep 10
done
```

(5) アプリケーション・サーバの設定（HACMP）

TPBroker 用のアプリケーション・サーバの設定をします。設定例を次の表に示します。

表 6-3 アプリケーション・サーバの設定例

項番	項目	値	備考
1	サーバ名	tpbroker	−
2	始動スクリプト	[/home/tpbroker/hacmp/bin/tpbroker_start.sh]	TPBroker 開始スクリプトの絶対パス名を設定します。 詳細は次を参照してください。 「(2) TPBroker 開始スクリプトの作成（HACMP）」
3	停止スクリプト	[/home/tpbroker/hacmp/bin/tpbroker_stop.sh]	TPBroker 停止スクリプトの絶対パス名を設定します。 詳細は次を参照してください。 「(3) TPBroker 停止スクリプトの作成（HACMP）」

（凡例）−：説明なし

(6) リソース・グループの設定（HACMP）

TPBroker 用のリソース・グループ，およびリソースの設定をします。TPBroker では同時にオンラインにできるのは一つの系だけです。設定例を次の表に示します。この例は，カスケード構成です。

表 6-4 リソース・グループの設定の設定例

項番	項目	値	備考
1	リソース・グループ名	tpbroker	−
2	参加ノード	node1 node2	−
3	サービス IP ラベル	[node1_svc]	次に示す手順で設定した値と同じものを設定します。 「(1) サービス IP ラベル/アドレスの設定（HACMP）」
4	アプリケーション・サーバ	[tpbroker]	次に示す手順で設定した値と同じものを設定します。 「(5) アプリケーション・サーバの設定（HACMP）」

項番	項目	値	備考
5	ファイルシステム	[/hacmp]	共有ディスクのマウント先ディレクトリの絶対パス名を指定します。 共有ディスクを UNIX ファイルシステムで使用する場合だけ設定します。
6	ボリューム・グループ	[vg00]	−

（凡例）−：説明なし

6.3.5　ORB 機能使用時の設定（HACMP）

ORB 機能を使用するときに必要な設定をします。設定方法は使用するシステムの IP アドレスの種類によって異なります。詳細は，マニュアル「Borland Enterprise Server VisiBroker デベロッパーズガイド」を参照してください。

注意

- エイリアス IP アドレスと永続 IP アドレスは同一の LAN に対して同時に構成することはできません。
- 設定された永続 IP アドレスを使用しないで，サービス IP アドレスを使用して ORB の通信をする場合は，通常の IP アドレスを使用する設定と同じ方法になります。設定方法については，「(1)　IP 交換による IP アドレス・テークオーバーの場合」を参照してください。

(1)　IP 交換による IP アドレス・テークオーバーの場合

IP 交換による IP アドレス・テークオーバーの場合の設定について説明します。

(a)　osagent の設定（IP 交換による IP アドレス・テークオーバー）

IP アドレスの設定（HACMP）

HACMP を起動するホスト上で osagent を起動する場合，HACMP 上ですべての系に次の定義ファイルを設定します。

- localaddr ファイル

  HACMP 上で osagent を使用する場合，すべての系に localaddr ファイルを作成し，その系で osagent の起動時に有効なすべての IP アドレスを指定してください。

  osagent がノード引き継ぎ対象ではなく，有効な IP アドレスをすべて設定できない場合は，自分自身をほかのホストで起動した osagent と認識し，通信量が増加してしまうことがあります。

- htc.clienthandleraddr ファイル

  HACMP 上で osagent が起動する場合，すべての系に htc.clienthandleraddr ファイルを作成し，すべてのホストに対して，サービス IP アドレスを応答するように設定してください。

osagent への設定（HACMP）

異なるネットワークドメインの osagent 間の通信で，一方の osagent をノード引き継ぎの対象とする場合，ノード引き継ぎ対象の osagent と通信をする osagent の定義ファイル（agentaddr ファイル）に，サービス IP アドレスを設定します。

osagent に接続するプロセスへの設定（HACMP）

osagent と osagent に接続するプロセスを異なるネットワークドメインで起動し，osagent をノード引き継ぎの対象とする場合，次に示す定義ファイル，およびオプションのどれかにサービス IP アドレスを設定してください。

- agentaddr ファイル
- 環境変数 OSAGENT_ADDR
- プロパティ vbroker.agent.addr

(b) osagent の接続に関する設定（IP 交換による IP アドレス・テークオーバー）

HACMP 上で osagent を起動する場合，次に示す環境変数の設定が osagent に必要です。

環境変数名：OSAGENT_CLIENT_HANDLER_PORT

環境変数の設定方法については，マニュアル「Borland Enterprise Server VisiBroker デベロッパーズガイド」を参照してください。

(c) サーバの設定（IP 交換による IP アドレス・テークオーバー）

ネーミングサービスを含む CORBA アプリケーションをノード引き継ぎの対象とする場合，次の設定をします。

- プロパティ vbroker.se.<xxx>.host
  サービス IP アドレス，またはホスト名を設定します。
- プロパティ vbroker.se.<xxx>.scm.<yyy>.listener.port
  任意のポート番号を設定します。
  使用するポート番号は，すべての系で同じポート番号が使用できるように管理する必要があります。

プロパティ名で，xxx はサーバエンジン，yyy はサーバコネクションマネージャを示します。

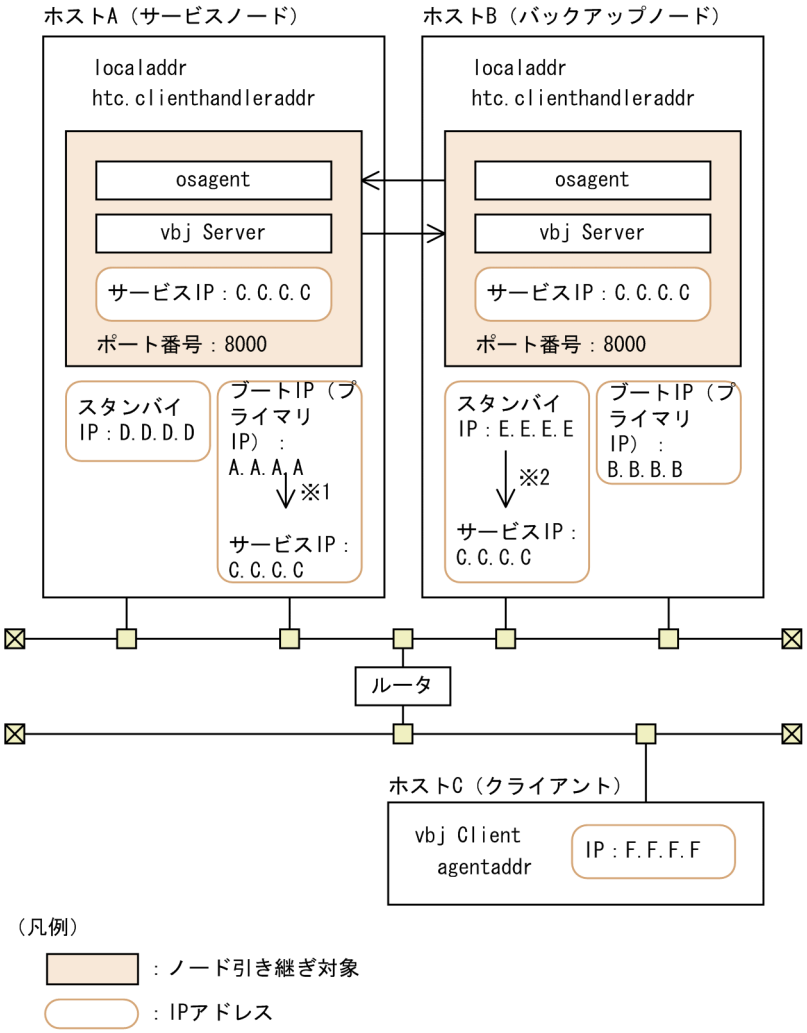
(d) ORB 機能使用時の設定例（IP 交換による IP アドレス・テークオーバー）

IP 交換による IP アドレス・テークオーバーの場合の，ORB 機能の使用時の設定例について説明します。

システム構成の例

ORB 機能使用時のシステム構成の例を次の図に示します。

図 6-3　IP 交換による IP アドレス・テークオーバーの場合のシステム構成例



注※1　HACMP起動時にサービスIPアドレスに書き換わります。
注※2　テークオーバー時にサービスIPアドレスに書き換わります。

IP アドレスの設定例

上記の図で示すシステム構成例での IP アドレスの設定例を次の図に示します。

図 6-4　IP 交換による IP アドレス・テークオーバーの場合の IP アドレス設定例

ホスト	ファイルまたは環境変数				
	localaddrファイル	htc.clienthandleraddr ファイル	agentaddr ファイル	環境変数 OSAGENT_PORT	環境変数 OSAGENT_CLIENT_HANDLER_PORT
ホストA	C.C.C.C subnet broadcast D.D.D.D subnet broadcast	F.F.F.F subnet C.C.C.C C.C.C.C subnet C.C.C.C	－	14000	16666
ホストB	C.C.C.C subnet broadcast B.B.B.B subnet broadcast	F.F.F.F subnet C.C.C.C C.C.C.C subnet C.C.C.C	－	14000	16666
ホストC	－	－	C.C.C.C	14000	－

（凡例）－：説明なし

ホストごとに設定値について説明します。

ホスト A の設定

- localaddr ファイルに，osagent 起動時に有効なサービス IP アドレス"C.C.C.C"とスタンバイ IP アドレス"D.D.D.D"を記述します。また，そのほかに有効な IP アドレスがある場合は記述します。
- htc.clienthandleraddr ファイルに，ホスト C に対する応答 IP アドレスにサービス IP アドレス"C.C.C.C"を記述します。
- サーバプロセスにプロパティ vbroker.se.<xxx>.host でサービス IP アドレス"C.C.C.C"を，プロパティ vbroker.se.<xxx>.scm.<yyy>.listener.port に"8000"を設定します。

  次のコマンドを実行します。

  vbj Server - vbroker.se.<xxx>.host.C.C.C.C ¥ -
  vbroker.xe.<xxx>.scm.<yyy>.listener.port 8000

  プロパティ名で，xxx はサーバエンジン，yyy はサーバコネクションマネージャを示します。
- 環境変数 OSAGENT_PORT に"14000"を設定します。
- 環境変数 OSAGENT_CLIENT_HANDLER_PORT に"16666"を設定します。

ホスト B の設定

- localaddr ファイルに，osagent 起動時に有効なサービス IP アドレス"C.C.C.C"とブート IP アドレス"B.B.B.B"を記述します。また，そのほかに有効な IP アドレスがある場合は記述します。
- htc.clienthandleraddr ファイルに，ホスト C に対する応答 IP アドレスにサービス IP アドレス"C.C.C.C"を記述します。
- 環境変数 OSAGENT_PORT に"14000"を設定します。
- 環境変数 OSAGENT_CLIENT_HANDLER_PORT に"16666"を設定します。

ホスト C の設定

- agentaddr ファイルに，サービス IP アドレス"C.C.C.C"を記述します。
- 環境変数 OSAGENT_PORT に"14000"を設定します。

(2) IP エイリアスによる IP アドレス・テークオーバーの場合

IP エイリアスによる IP アドレス・テークオーバーの場合の設定について説明します。

(a) osagent の設定（IP エイリアスによる IP アドレス・テークオーバー）

IP アドレスの設定（HACMP）

HACMP を起動するホスト上で osagent を起動する場合，HACMP 上ですべての系に次の定義ファイルを設定します。

- localaddr ファイル

  HACMP 上で osagent を使用する場合，すべての系に localaddr ファイルを作成し，その系で osagent の起動時に有効なすべての IP アドレスを指定してください。

- htc.clienthandleraddr ファイル
  HACMP 上で osagent が起動する場合，すべての系に htc.clienthandleraddr ファイルを作成し，すべてのホストに対して，エイリアス IP アドレスを応答するように設定してください。

osagent への設定（HACMP）

異なるネットワークドメインの osagent 間の通信で，一方の osagent をノード引き継ぎの対象とする場合，ノード引き継ぎ対象の osagent と通信をする osagent の定義ファイル（agentaddr ファイル）に，エイリアス IP アドレスを設定します。

osagent に接続するプロセスへの設定（HACMP）

osagent と osagent に接続するプロセスを異なるネットワークドメインで起動し，osagent をノード引き継ぎの対象とする場合，次に示す定義ファイル，およびオプションのどれかにエイリアス IP アドレスを設定してください。

- agentaddr ファイル
- 環境変数 OSAGENT_ADDR
- プロパティ vbroker.agent.addr

(b) osagent の接続に関する設定（IP エイリアスによる IP アドレス・テークオーバー）

HACMP 上で osagent を起動する場合，次に示す環境変数の設定が osagent に必要です。

環境変数名：OSAGENT_CLIENT_HANDLER_PORT

環境変数の設定方法については，マニュアル「Borland Enterprise Server VisiBroker デベロッパーズガイド」を参照してください。

(c) サーバの設定（IP エイリアスによる IP アドレス・テークオーバー）

ネーミングサービスを含む CORBA アプリケーションをノード引き継ぎの対象とする場合，次の設定をします。

- プロパティ vbroker.se.<xxx>.host
  サービス IP アドレス，またはホスト名を設定します。
- プロパティ vbroker.se.<xxx>.scm.<yyy>.listener.port
  任意のポート番号を設定します。
  使用するポート番号は，すべての系で同じポート番号が使用できるように管理する必要があります。

プロパティ名で，xxx はサーバエンジン，yyy はサーバコネクションマネージャを示します。

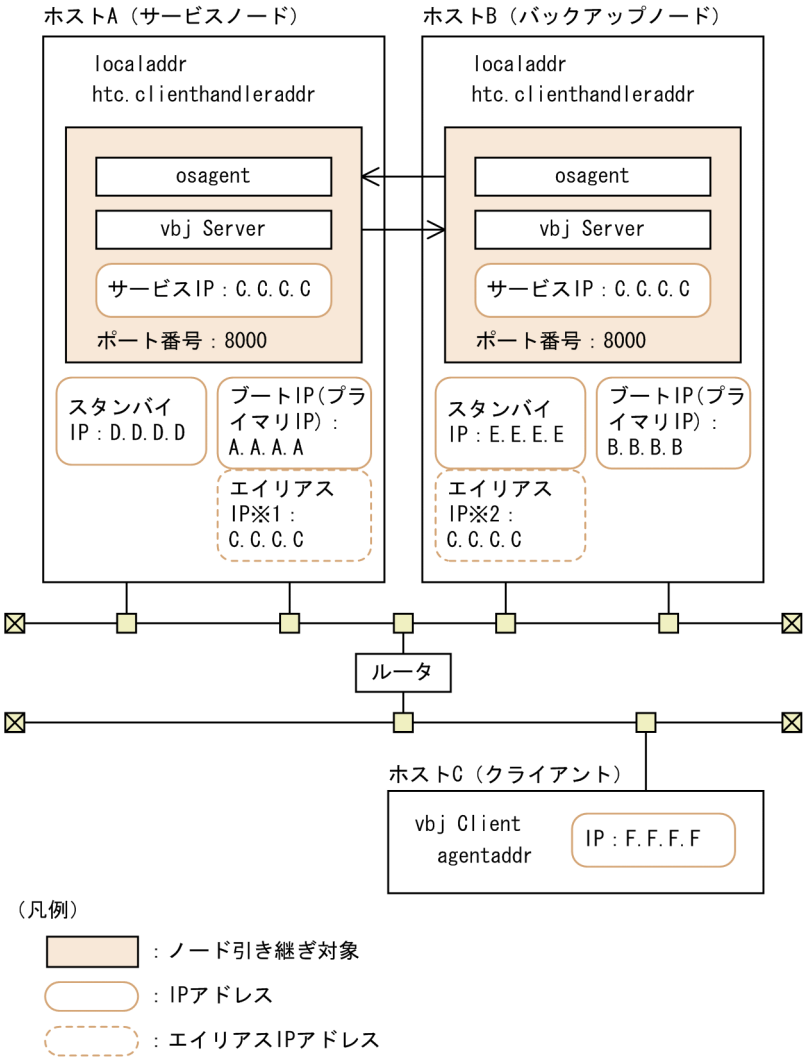
(d) ORB 機能使用時の設定例（IP エイリアスによる IP アドレス・テークオーバー）

IP エイリアスによる IP アドレス・テークオーバーの場合の，ORB 機能の使用時の設定例について説明します。

システム構成の例

ORB 機能使用時のシステム構成の例を次の図に示します。

図 6-5　IP エイリアスによる IP アドレス・テークオーバーの場合のシステム構成例



注※1　HACMP起動時に付加されます。

注※2　テークオーバー時に付加されます。

IP アドレスの設定例

上記の図で示すシステム構成例での IP アドレスの設定例を次の図に示します。

図 6-6　IP エイリアスによる IP アドレス・テークオーバーの場合の IP アドレス設定例

ホスト	ファイルまたは環境変数				
	localaddrファイル	htc.clienthandleraddr ファイル	agentaddr ファイル	環境変数 OSAGENT_PORT	環境変数 OSAGENT_CLIENT_HANDLER_PORT
ホストA	C.C.C.C subnet broadcast D.D.D.D subnet broadcast A.A.A.A subnet broadcast	F.F.F.F subnet C.C.C.C C.C.C.C subnet C.C.C.C	－	14000	16666
ホストB	C.C.C.C subnet broadcast B.B.B.B subnet broadcast E.E.E.E subnet broadcast	F.F.F.F subnet C.C.C.C C.C.C.C subnet C.C.C.C	－	14000	16666
ホストC	－	－	C.C.C.C	14000	－

（凡例）－：説明なし

ホストごとに設定値について説明します。

ホスト A の設定

- localaddr ファイルに，HACMP 起動時に有効なエイリアス IP アドレス"C.C.C.C"，スタンバイ IP アドレス"D.D.D.D"，およびブート IP アドレス"A.A.A.A"を記述します。そのほかに有効な IP アドレスがある場合は記述します。
- htc.clienthandleraddr ファイルに，現用系に対する応答 IP アドレスにエイリアス IP アドレス"C.C.C.C"，ホスト C に対する応答 IP アドレスにエイリアス IP アドレス"C.C.C.C"を記述します。
- サーバプロセスにプロパティ vbroker.se.<xxx>.host でエイリアス IP アドレス"C.C.C.C"を，vbroker.se.<xxx>.scm.<yyy>.listener.port に"8000"を設定します。

  次のコマンドを実行します。

  vbj Server - vbroker.se.<xxx>.host.C.C.C.C ¥ -
  vbroker.xe.<xxx>.scm.<yyy>.listener.port 8000

  プロパティ名で，xxx はサーバエンジン，yyy はサーバコネクションマネージャを示します。
- 環境変数 OSAGENT_PORT に"14000"を設定します。
- 環境変数 OSAGENT_CLIENT_HANDLER_PORT に"16666"を設定します。

ホスト B の設定

- localaddr ファイルに，osagent 起動時に有効なエイリアス IP アドレス"C.C.C.C"とブート IP アドレス"B.B.B.B"を記述します。そのほかに有効な IP アドレスがある場合は記述します。
- htc.clienthandleraddr ファイルに，ホスト C に対する応答 IP アドレスにエイリアス IP アドレス"C.C.C.C"を記述します。
- 環境変数 OSAGENT_PORT に"14000"を設定します。
- 環境変数 OSAGENT_CLIENT_HANDLER_PORT に"16666"を設定します。

ホスト C の設定

- agentaddr ファイルに，エイリアス IP アドレス"C.C.C.C"を記述します。

- 環境変数 OSAGENT_PORT に"14000"を設定します。

(3) 永続 IP アドレスの場合

永続 IP アドレスの場合の設定について説明します。

(a) osagent の設定（永続 IP アドレス）

IP アドレスの設定（HACMP）

HACMP を起動するホスト上で osagent を起動する場合，HACMP 上ですべての系に次の定義ファイルを設定します。IP アドレスの設定は，osagent の起動タイミングによって異なります。設定例は，「(c) ORB 機能使用時の設定例（永続 IP アドレス）」を参照してください。

- localaddr ファイル

  HACMP 上で osagent を使用する場合，すべての系に localaddr ファイルを作成し，その系で osagent の起動時に有効なすべての IP アドレスを指定してください。

  osagent がノード引き継ぎ対象ではなく，有効な IP アドレスをすべて設定できない場合は，自分自身をほかのホストで起動した osagent と認識し，通信量が増加してしまうことがあります。

- htc.clienthandleraddr ファイル

  HACMP 上で osagent が起動する場合，すべての系に htc.clienthandleraddr ファイルを作成し，すべてのホストに対して，永続 IP アドレスを応答するように設定してください。

osagent への設定（HACMP）

異なるネットワークドメインの osagent 間の通信で，一方の osagent をノード引き継ぎの対象とする場合，ノード引き継ぎ対象の osagent と通信をする osagent の定義ファイル（agentaddr ファイル）に，永続 IP アドレスを設定します。

osagent に接続するプロセスへの設定（HACMP）

osagent と osagent に接続するプロセスを異なるネットワークドメインで起動し，osagent をノード引き継ぎの対象とする場合，次に示す定義ファイル，およびオプションのどれかに永続 IP アドレスを設定してください。

- agentaddr ファイル
- 環境変数 OSAGENT_ADDR
- プロパティ vbroker.agent.addr

(b) サーバの設定（永続 IP アドレス）

ネーミングサービスを含む CORBA アプリケーションをノード引き継ぎの対象とする場合，次の設定をします。

- プロパティ vbroker.se.<xxx>.host

  永続 IP アドレス，またはホスト名を設定します。

- プロパティ vbroker.se.<xxx>.scm.<yyy>.listener.port

任意のポート番号を設定します。

使用するポート番号は，すべての系で同じポート番号が使用できるように管理する必要があります。

プロパティ名で，xxx はサーバエンジン，yyy はサーバコネクションマネージャを示します。

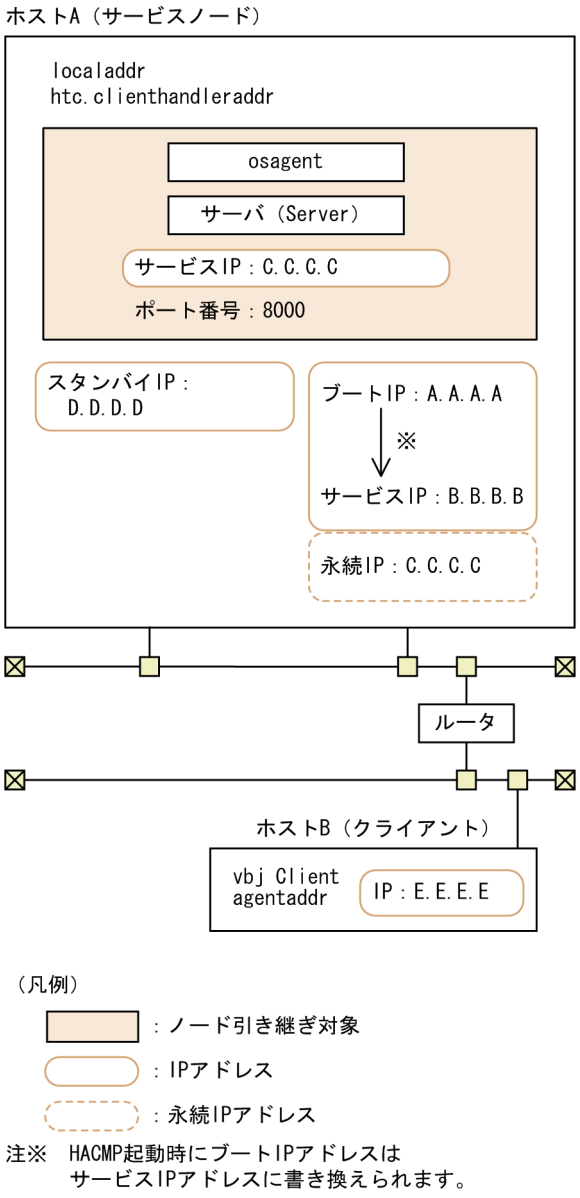
(c)　ORB 機能使用時の設定例（永続 IP アドレス）

永続 IP アドレスの場合の，ORB 機能の使用時の設定例について説明します。

システム構成の例

ORB 機能使用時のシステム構成の例を次の図に示します。

図 6-7　永続 IP アドレスの場合のシステム構成例



IP アドレスの設定例

上記の図で示すシステム構成例での IP アドレスの設定例を次の図に示します。

HACMP 起動前に osagent を起動させる場合

HACMP 起動前に osagent を起動させる場合の設定例です。

図 6-8　永続 IP アドレスの場合の IP アドレス設定例（HACMP 起動前に osagent を起動させる場合）

ホスト	ファイルまたは環境変数				
	localaddrファイル	htc.clienthandleraddr ファイル	agentaddr ファイル	環境変数 OSAGENT_PORT	環境変数 OSAGENT_CLIENT_HANDLER_PORT
ホストA	C.C.C.C subnet broadcast A.A.A.A subnet broadcast D.D.D.D subnet broadcast	E.E.E.E subnet C.C.C.C C.C.C.C subnet C.C.C.C	－	14000	－
ホストB	－	－	C.C.C.C	14000	－

（凡例）－：説明なし

HACMP 起動後に osagent を起動させる場合

HACMP 起動後に osagent を起動させる場合の設定例です。

図 6-9　永続 IP アドレスの場合の IP アドレス設定例（HACMP 起動後に osagent を起動させる場合）

ホスト	ファイルまたは環境変数				
	localaddrファイル	htc.clienthandleraddr ファイル	agentaddr ファイル	環境変数 OSAGENT_PORT	環境変数 OSAGENT_CLIENT_HANDLER_PORT
ホストA	C.C.C.C subnet broadcast B.B.B.B subnet broadcast D.D.D.D subnet broadcast	E.E.E.E subnet C.C.C.C C.C.C.C subnet C.C.C.C	－	14000	－
ホストB	－	－	C.C.C.C	14000	－

（凡例）－：説明なし

ホスト A の設定

- localaddr ファイルの設定

  1. HACMP 起動前に osagent を起動させる場合

  osagent 起動時に有効な永続 IP アドレス"C.C.C.C"，ブート IP アドレス"A.A.A.A"，およびスタンバイ IP アドレス"D.D.D.D"を記述します。そのほかに有効な IP アドレスがある場合は記述します。

  2. HACMP 起動後に osagent を起動させる場合

  osagent 起動時に有効な永続 IP アドレス"C.C.C.C"，サービス IP アドレス"B.B.B.B"，およびスタンバイ IP アドレス"D.D.D.D"を記述します。そのほかに有効な IP アドレスがある場合は記述します。

- htc.clienthandleraddr ファイルに，現用系に対する応答 IP アドレスに永続 IP アドレス"C.C.C.C"，ホスト B に対する応答 IP アドレスに永続 IP アドレス"C.C.C.C"を記述します。

- サーバプロセスにプロパティ vbroker.se.<xxx>.host で永続 IP アドレス"C.C.C.C"を，プロパティ vbroker.se.<xxx>.scm.<yyy>.listener.port に"8000"を設定します。
  次のコマンドを実行します。
  vbj Server - vbroker.se.<xxx>.host.C.C.C.C ¥ -
  vbroker.xe.<xxx>.scm.<yyy>.listener.port 8000
  プロパティ名で，xxx はサーバエンジン，yyy はサーバコネクションマネージャを示します。
- 環境変数 OSAGENT_PORT に"14000"を設定します。

ホスト B の設定

- agentaddr ファイルに永続 IP アドレス"C.C.C.C"を記述します。
- 環境変数 OSAGENT_PORT に"14000"を設定します。

6.4 HACMP との連携時の運用

この節では，HACMP との連携時の運用について説明します。

次の運用について説明します。

- TPBroker の開始
- TPBroker の停止
- TPBroker の定義の変更
- 待機系での TPBroker の移行

これ以外の運用については，マニュアル「TPBroker ユーザーズガイド」，HACMP のマニュアルを参照してください。

6.4.1 TPBroker の開始（HACMP）

HACMP の運用コマンドで，リソース・グループ（tpbroker）を開始します。共有ボリュームグループの活動を開始するタイミングは，HACMP が管理しているため，TPBroker の admstart コマンドで TPBroker を開始させないようにします。

6.4.2 TPBroker の停止（HACMP）

HACMP の運用コマンドで，リソース・グループ（tpbroker）を停止します。TPBroker の admstop コマンドでは，TPBroker は一時的にしか停止しません。

6.4.3 TPBroker の定義の変更（HACMP）

TPBroker をクラスタ構成で使用し，TPBroker のシステム環境定義，および監視対象プロセスのカレントディレクトリが作成，削除されるタイミングの定義を変更する場合の手順について説明します。

1. 定義を変更します。
   必ずすべての系で同じ設定をしてください。
   - システム環境定義
     tsdefvalue コマンドで定義を変更します。
   - 監視対象プロセスのカレントディレクトリが作成，削除されるタイミングの定義
     admsetup コマンドの-c オプションで指定したプロセス監視定義ファイルを編集します。
2. リソース・グループ tpbroker が起動している系で，TPBroker がオンラインであることを確認します。

オンラインのときには，admstat コマンドを実行すると，TPBroker の稼働状況が表示されます。表示されないときには，開始中なので，オンラインになるまで待ってください。

3. TPBroker を正常停止します。
   admstop コマンドを実行します。

6.4.4 待機系での TPBroker の移行（HACMP）

TPBroker のバージョンアップなどで，TPBroker が動作していない系の TPBroker を入れ替える手順について説明します。

TPBroker を入れ替えるには，OTS 機能を正常停止させる必要があります。次の手順に従ってください。

1. OTS 機能を停止します。
   -i オプションを指定して admstopprc コマンドを実行することで，OTS 機能を正常停止させます。
2. 計画的に系を切り替えます。
   切り替えた先の系で OTS 機能を起動してもかまいません。
3. ADM 機能をアンセットアップします。
4. TPBroker を入れ替えます。
5. OTS 機能で使用するリソースマネージャを登録します。
   コマンドの実行例を示します。

   ```
   tslnkrm -nlm
   ```

   -f オプションは指定しないでください。
6. ADM 機能をセットアップします。
7. 計画的に系を切り替えます
   セットアップした環境に戻します。
8. OTS 機能を開始します。
   同様にすべての系の TPBroker を入れ替え，すべての系の TPBroker のバージョンを同一にしてください。

6.5 HACMP との連携時のアンセットアップ

この節では，HACMP との連携時のアンセットアップについて説明します。

TPBroker をアンセットアップする系ごとに次の手順をしてください。

1. TPBroker のリソース・グループ（tpbroker）を停止します。
2. TPBroker の運用支援実行環境をアンセットアップします。
   次のコマンドを実行します。

```
admsetup -d
```

3. TPBroker の OTS 環境をアンセットアップします。
   次のコマンドを実行します。

```
tssetup -d
```

7 ディスク複製インストール方法

この章では，TPBroker と JP1/ServerConductor/Deployment Manager との連携方法，および TPBroker と仮想化プラットフォームの複製機能について説明します。

7.1 JP1/ServerConductor/Deployment Manager または仮想化プラットフォームでできること

この節では，JP1/ServerConductor/Deployment Manager または仮想化プラットフォームとの連携でできることについて説明します。

JP1/ServerConductor/Deployment Manager または仮想化プラットフォームと連携させることで，ディスク複製インストールをすることができます。

連携できる OS は次のとおりです。

- Windows
- Linux

連携できる TPBroker のプログラムプロダクトは次のとおりです。

- Cosminexus TPBroker
- TPBroker Developer
- TPBroker
- TPBroker Client（Windows 限定）

JP1/ServerConductor/Deployment Manager の詳細は，マニュアル「JP1/ServerConductor/Deployment Manager」を参照してください。

仮想化ソフトウェアの詳細は，仮想化ソフトウェアのマニュアルを参照してください。

7.2 JP1/ServerConductor/Deployment Manager または仮想化プラットフォームとの連携時のセットアップ

この節では，JP1/ServerConductor/Deployment Manager または仮想化プラットフォームとの連携時のセットアップの方法について説明します。

ここで説明するセットアップの方法は，JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム，および TPBroker の基本的な設定を終えていることを前提とし，TPBroker との連携時に必要な手順だけを説明します。セットアップの詳細は，マニュアル「JP1/ServerConductor/Deployment Manager」または仮想化ソフトウェアのマニュアル，およびマニュアル「TPBroker ユーザーズガイド」を参照してください。

7.2.1 セットアップの流れ（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）

JP1/ServerConductor/Deployment Manager または仮想化プラットフォームとの連携時のセットアップの流れを次の図に示します。

図 7-1 JP1/ServerConductor/Deployment Manager または仮想化プラットフォームとの連携時のセットアップの流れ

なお，Cosminexus TPBroker では，OTS 機能，および ADM 機能が使用できません。そのため，Cosminexus TPBroker を使用している場合のセットアップは，次に示す手順だけです。

1. ログ・トレースの削除（7.2.2(6)）
2. シナリオの設定（7.2.3）
3. ディスク複製による OS インストール（7.2.4）
4. 環境変数の設定（7.2.5(1)）
5. ORB 機能の設定（7.2.5(2)(a)）

インストール中に TPBroker に関連するサービスが起動した場合，ほかのコンピュータ上で起動している osagent と連携して，システム全体として問題が発生する可能性があるため，7.2.2 以降で示すセットアップ手順に従って，ディスク複製中は TPBroker が起動しないようにしてください。

7.2.2　マスタコンピュータのセットアップ

JP1/ServerConductor/Deployment Manager または仮想化プラットフォームで，TPBroker を含む OS をディスク複製する場合，次に示す手順に従ってください。

(1)　OTS 機能の停止

OTS 機能を使用している場合は，OTS 機能を正常停止します。

コマンドの実行例を次に示します。

1. OTS 機能を正常停止します。
   次のコマンドを実行します。
   - ADM 機能と連携している場合
     admstopprc -i 0001
   - ADM 機能と連携していない場合
     tsstop

Cosminexus TPBroker で，J2EE サーバ上でインプロセストランザクションサービスを使用している場合は，J2EE サーバの停止方法に従って，正常停止してください。

(2)　TPBroker のシステム環境定義のバックアップ

次の機能を使用している場合は，TPBroker のシステム環境定義のバックアップを取得します。

- ADM 機能
- TPBroker の OTS 機能

コマンドの実行例を次に示します。

1. TPBroker の定義のバックアップを取得します。
   tslsconf > tslsconf.log

(3)　ADM 機能のアンセットアップ

ADM 機能を使用している場合は，ADM 機能をアンセットアップします。

コマンドの実行例を次に示します。

Windows

1. TPBroker のサービスを停止します。
   net stop TPBroker
2. TPBroker の ADM 機能をアンセットアップします。
   admsetup -d

Linux

1. システム環境定義の開始モードの値に，"MANUAL"を設定します。
   tsdefvalue /ADM set_conf_mode -s "MANUAL"
2. TPBroker を正常停止します。
   admstop
3. TPBroker の ADM 機能をアンセットアップします。
   admsetup -d

(4) TPBroker のアンセットアップ

次の機能を使用している場合は，TPBroker をアンセットアップします。

- ADM 機能
- TPBroker の OTS 機能

コマンドの実行例を次に示します。

1. TPBroker をアンセットアップします。
   tssetup -d

(5) ステータスファイルの削除

Cosminexus TPBroker で，J2EE サーバ上でインプロセストランザクションサービスを使用している場合，次に示す J2EE サーバのプロパティで，指定されているディレクトリを削除します。

- ejbserver.distributedtx.ots.status.directory1
- ejbserver.distributedtx.ots.status.directory2

(6) ログ・トレースの削除

次に示すディレクトリ以下のファイルをすべて削除します。

Windows

- <TPBroker のインストール先ディレクトリ>¥log
- <TPBroker のインストール先ディレクトリ>¥log¥mdltrc

- <TPBroker のインストール先ディレクトリ>¥log¥comtrc
- <TPBroker のインストール先ディレクトリ>¥log¥stktrc
- <TPBroker のインストール先ディレクトリ>¥logj
- <TPBroker のインストール先ディレクトリ>¥logj¥mdltrc
- <TPBroker のインストール先ディレクトリ>¥logj¥comtrc
- <TPBroker のインストール先ディレクトリ>¥logj¥namelog
- %VBROKER_ADM%¥..¥log
- %VBROKER_ADM%¥..¥log¥mdltrc
- %VBROKER_ADM%¥..¥log¥comtrc
- %VBROKER_ADM%¥..¥log¥stktrc
- %VBROKER_ADM%¥..¥logj
- %VBROKER_ADM%¥..¥logj¥mdltrc
- %VBROKER_ADM%¥..¥logj¥comtrc
- %VBROKER_ADM%¥..¥logj¥namelog
- %HVI_TRACEPATH%
- %HVI_TRACEPATH%¥mdltrc
- %HVI_TRACEPATH%¥comtrc
- %HVI_TRACEPATH%¥stktrc
- %HVI_TRACEPATH%¥namelog
- <TPBroker のインストール先ドライブ>¥vbroker¥log
- %OSAGENT_LOG_DIR%

Linux

- <TPBroker のインストール先ディレクトリ>/log
- <TPBroker のインストール先ディレクトリ>/log/mdltrc
- <TPBroker のインストール先ディレクトリ>/log/comtrc
- <TPBroker のインストール先ディレクトリ>/log/hgttrc
- <TPBroker のインストール先ディレクトリ>/logj
- <TPBroker のインストール先ディレクトリ>/logj/mdltrc
- <TPBroker のインストール先ディレクトリ>/logj/comtrc
- <TPBroker のインストール先ディレクトリ>/logj/namelog
- $VBROKER_ADM/../log
- $VBROKER_ADM/../log/mdltrc

- $VBROKER_ADM/../log/comtrc
- $VBROKER_ADM/../log/hgttrc
- $VBROKER_ADM/../logj
- $VBROKER_ADM/../logj/mdltrc
- $VBROKER_ADM/../logj/comtrc
- $VBROKER_ADM/hgtfifo
- $VBROKER_ADM/../logj/namelog
- $HVI_TRACEPATH
- $HVI_TRACEPATH/mdltrc
- $HVI_TRACEPATH/comtrc
- $HVI_TRACEPATH/hgttrc
- $HVI_TRACEPATH/namelog

Windows の場合，次に示すディレクトリを削除します。

%VBROKER_ADM%¥log

7.2.3 シナリオの設定

JP1/ServerConductor/Deployment Manager との連携の場合はシナリオを作成します。詳細は，マニュアル「JP1/ServerConductor/Deployment Manager」を参照してください。

また，JP1/ServerConductor/Deployment Manager と TPBroker を連携させる場合，次の設定を行います。

- 複製後のコンピュータで，固定 IP アドレスを使用する。
  DHCP で IP アドレスを割り当てないようにしてください。

7.2.4 ディスク複製による OS インストール

ディスク複製インストールを実行します。詳細は，マニュアル「JP1/ServerConductor/Deployment Manager」，または仮想化ソフトウェアのマニュアルを参照してください。

注意事項

- ディスク複製時にセットアップパラメタファイルのコピーに失敗した場合，固定 IP アドレスを使用するように手動でネットワークの設定を行ってください。

- TPBroker は DHCP に対応できません。ディスク複製後は DHCP を使用しないように設定してください。
- 仮想化プラットフォームが提供するイメージファイル化による複製機能を使用する場合，以下を使用している場合は，指定したホスト名や IP アドレスをシステムに合わせて変更してください。
  - agentaddr ファイル（環境変数 OSAGENT_ADDR_FILE またはプロパティ vbroker.agent.addrFile でも指定できます）
  - localaddr ファイル（環境変数 OSAGENT_LOCAL_FILE またはプロパティ vbroker.agent.localFile でも指定できます）
  - htc.clienthandleraddr ファイル（環境変数 HVI_OSAGENT_CLIENTHANDLERADDR_FILE でも指定できます）
  - プロパティ vbroker.agent.addr
  - プロパティ vbroker.se.xxx.host（xxx はサーバエンジンです）
  - プロパティ vbroker.se.xxx.proxyHost（xxx はサーバエンジンです）
  - プロパティ vbroker.orb.initRef
  - プロパティ vbroker.orb.defaultInitRef（C++のみ）
  - コマンドラインオプション ORBInitRef
  - コマンドラインオプション ORBDefaultInitRef（C++のみ）
  - コマンドラインオプション ORBagentAddr
  - osagent のコマンドラインオプション -a
  - 環境変数 OSAGENT_ADDR

  また，ホスト名や IP アドレスを変更する前に作成した IOR ファイルは，変更後に再作成し再配布してください。

7.2.5　複製先コンピュータのセットアップ

ディスク複製インストール後，次の設定を行ってください。

(1)　環境変数の設定（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）

次の表に示す環境変数を確認し，値を再設定してください。

表 7-1　環境変数の一覧（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）

項番	環境変数名	設定内容
1	OSAGENT_PORT	マスタコンピュータと ORB のドメインが異なる場合，ドメインを構成するポート番号を指定します。
2	OSAGENT_CLIENT_HANDLER_PORT	複製先コンピュータの環境に合わせます。
3	OSAGENT_ADDR	複製先コンピュータの環境に合わせます。

(2)　定義の設定（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）

ORB 機能，および ADM 機能に設定されている定義を確認し，値を再設定してください。

(a)　ORB 機能の設定（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）

ORB 機能に関する定義ファイル，およびプロパティを確認し，値を再設定してください。

再設定する定義ファイルを次の表に示します。

表 7-2　定義ファイルの設定の一覧

項番	定義ファイル	ファイルの格納場所	設定内容
1	agentaddr ファイル	次のどちらかです。 • $VBROKER_ADM/agentaddr（%VBROKER_ADM%¥agentaddr） • $OSAGENT_ADDR_FILE（%OSAGENT_ADDR_FILE%），またはプロパティ vbroker.agent.addrFile に指定されたファイル	複製先コンピュータの連携先の osagent の IP アドレスに合わせます。
2	localaddr ファイル	次のどちらかです。 • $VBROKER_ADM/localaddr（%VBROKER_ADM%¥localaddr） • $OSAGENT_ADDR_FILE（%OSAENT_LOCAL_FILE%），またはプロパティ vbroker.agent.localFile に指定されたファイル	複製先コンピュータの環境（自 IP アドレス）に合わせます。
3	htc.clienthandleraddr ファイル	次のどちらかです。 • $VBROKER_ADM/htc.clienthandleraddr（%VBROKER_ADM%¥htc.clienthandleraddr） • $HVI_OSAGENT_CLIENTHANDLERADDR_FILE	複製先コンピュータの環境（自 IP アドレス）に合わせます。

項番	定義ファイル	ファイルの格納場所	設定内容
		（%HVI_OSAGENT_CLIENTHANDLERADDR_FILE%）に指定されたファイル	

再設定するプロパティを次の表に示します。

なお，プロパティ vbroker.orb.propStorage（ORBpropStorage）で指定した定義ファイルに次の表に示すプロパティが指定されている場合は，定義ファイルの内容も変更する必要があります。

表 7-3　プロパティの設定の一覧

項番	オプション	設定内容
1	vbroker.se.<xxx>.host※1	複製先コンピュータの自 IP アドレスを指定します。
2	vbroker.se.<xxx>.proxyHost※1	複製先コンピュータで解決できる IP アドレスを指定します。
3	vbroker.se.<xxx>x.scm.<yyy>.listener.port※1,※2	複製先コンピュータの自ポート番号を指定します。
4	vbroker.agent.addr	複製先コンピュータから接続する osagent の IP アドレスを指定します。
5	vbroker.agent.port	マスタコンピュータと ORB のドメインが異なる場合，ドメインを構成するポート番号を指定します。

注※1

xxx はサーバエンジンを示します。

注※2

yyy はサーバコネクションマネージャを示します。

(b)　システム環境定義の設定（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）

OTS 機能，および ADM 機能を使用している場合は，システム環境定義を設定します。

コマンドの実行例を次に示します。

1. TPBroker をセットアップします。

   tssetup

2. システム環境定義の値を設定します。

   tsdefvalue /ADM set_conf_mode -s "MANUAL"

   設定する値については，事前に取得したシステム環境定義のバックアップに従います。詳細は，「7.2.2(2)　TPBroker のシステム環境定義のバックアップ」を参照してください。

   また，次の表に示すシステム環境定義をマスタコンピュータで設定していた場合は，複製先コンピュータの環境に合わせて，再設定してください。マスタコンピュータで設定していなかった場合は，設定する必要はありません。

表 7-4　システム環境定義の設定

項番	システム環境定義	設定内容
1	/OTS/completion_process_ipaddr_info	複製先コンピュータ上の決着デーモンが使用するホスト名に変更します。
2	/OTS/set_ipaddr_info	複製先コンピュータ上の OTS のデーモンが使用するホスト名に変更します。

(3)　ADM 機能の設定（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）

ADM 機能を使用している場合は，ADM 機能の設定をします。

コマンドの実行例を次に示します。

1. プロセス監視定義ファイルを変更します。
   再設定したシステム環境定義の値に合わせて，プロセス監視定義ファイルの値を変更します。
   この手順は再設定したシステム環境定義の値が，プロセス監視定義ファイルでも設定されている場合にだけ必要です。再設定したシステム環境定義の値については，「(2)　定義の設定（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）」を参照してください。
2. TPBroker の実行環境のセットアップをします。
   admsetup -c c:¥admconf.cf

(4)　サービスの開始（JP1/ServerConductor/Deployment Manager または仮想化プラットフォーム）

ADM 機能を使用する場合は，ADM 機能のサービスを開始します。この手順は，Windows だけ必要です。

コマンドの実行例を次に示します。

1. TPBroker の ADM 機能のサービスを開始します。
   net start TPBroker

8 メッセージ

この節では，TPBroker の運用時に出力されるメッセージについて説明します。

8.1　メッセージの概要

この節では，TPBroker の運用時に出力されるメッセージの種類，およびメッセージの表記について説明します。

8.1.1　メッセージの種類

TPBroker の運用時に出力されるメッセージの種類を示します。

表 8-1　TPBroker の運用時に出力されるメッセージの種類

メッセージの種類	参照先
KFCB91000～KFCB91999 のメッセージ	8.2
KFCB92000～KFCB92999 のメッセージ	8.3
トレース情報取得ができない場合に出力されるメッセージ	8.4
hvmgtee コマンドのエラーメッセージ	8.5
hdumpns コマンドのエラーメッセージ	8.6

このマニュアルでのメッセージの並び順について説明します。

メッセージ ID がある場合は，メッセージ ID の順に並んでいます。メッセージ ID がある場合のメッセージの出力形式については，「8.1.2　メッセージの表記」を参照してください。

メッセージ ID がない場合は，アルファベット順に並んでいます。

8.1.2　メッセージの表記

メッセージ ID がある場合のメッセージの表記について説明します。

(1)　メッセージの出力形式

メッセージの出力形式を次に示します。

```
KFCBnnnnn-X  YY.....YY.

    KFCBnnnnn：メッセージID（半角英数字9文字）
    X：エラーレベル
      E：エラー
      W：警告
      I：情報
    YY.....YY：メッセージテキスト
```

(2) メッセージの記述形式

メッセージの記述形式を次に示します。

メッセージ ID

メッセージテキスト

説明

[要因]

メッセージの出力要因を示します。

[システムの処理]

システムがメッセージを出力したあとにする主な処理を示します。

[対策]

メッセージ確認時の TPBroker 管理者の処置を示します。

8.2 KFCB91000～KFCB91999 のメッセージ

KFCB91000-I

Now starting Naming Service.

[要因]

ネーミングサービスを起動中です。

[システムの処理]

処理を続行します。

[対策]

なし。

KFCB91001-I

Naming Service is now online. FactoryName=aa....aa, Port=bb....bb

aa....aa：ネーミングサービス起動時に指定した名前

bb....bb：ネーミングサービスの待ち受けポート番号

vbroker.orb.htc.msgLog.security=true が指定されている場合，ポート番号は固定文字列"*****"が表示されます。

[要因]

ネーミングサービスが起動されました。

[システムの処理]

処理を続行します。

[対策]

なし。

KFCB91002-I

Naming Service stopped. FactoryName=aa....aa, Port=bb....bb

aa....aa：ネーミングサービス起動時に指定した名前

bb....bb：ネーミングサービスの待ち受けポート番号

vbroker.orb.htc.msgLog.security=true が指定されている場合，ポート番号は固定文字列"*****"が表示されます。

[要因]

ネーミングサービスユーティリティ（nsutil）の shutdown コマンドによって，ネーミングサービスが終了しました。

[システムの処理]

処理を続行します。

[対策]

なし。

KFCB91003-I

Naming Service terminated. FactoryName=aa....aa, Port=bb....bb

aa....aa：ネーミングサービス起動時に指定した名前

bb....bb：ネーミングサービスの待ち受けポート番号

vbroker.orb.htc.msgLog.security=true が指定されている場合，ポート番号は固定文字列"*****"が表示されます。

また，名前，ポート番号が取得できなかった場合，名前およびポート番号は固定文字列"*****"が表示されます。

[要因]

ネーミングサービスが終了しました。

ただし，このメッセージはネーミングサービスが強制終了された場合には出力されません。

[システムの処理]

処理を続行します。

[対策]

なし。

KFCB91004-W

A CORBA::TIMEOUT occurred. aa....aa=bb....bb, info=cc....cc

aa....aa：発生したタイムアウトの種類

RelativeConnectionTimeout，RelativeRequestTimeout，または RelativeRoundtripTimeout のどれかが出力されます。

bb....bb：設定したタイムアウト値（単位：100 ナノ秒）

cc....cc：保守情報

RelativeConnectionTimeout

プロセス間通信のコネクション接続時にタイムアウトが発生しました。

RelativeRequestTimeout

データ送信時にタイムアウトが発生しました。

RelativeRoundtripTimeout

リクエスト送信処理からリクエストの応答受信までの間にタイムアウトが発生しました。

[要因]

タイムアウトが発生しました。

[システムの処理]

処理を続行します。

[対策]

RelativeConnectionTimeout

ネットワークに問題があるため，コネクション接続に時間が掛かっている可能性があります。ネットワークの状態を見直してください。

指定したタイムアウト時間が短い場合は，
com.inprise.vbroker.QoSExt.RELATIVE_CONN_TIMEOUT_POLICY_TYPE の値を大きくしてください。

RelativeRequestTimeout

サーバ側のリソース不足，ネットワークの問題などによって，サーバにリクエストを送信できない可能性があります。サーバ側の状態，およびネットワークの状態を見直してください。

指定したタイムアウト時間が短い場合は，
org.omg.Messaging.RELATIVE_REQ_TIMEOUT_POLICY_TYPE の値を大きくしてください。

RelativeRoundtripTimeout

次の状態になっている可能性があります。

- サーバ処理に時間が掛かっている
- リソース不足などによって，サーバがリクエストを受信できない
- 再接続処理が繰り返し行われている

サーバの状態を見直してください。

また，ネットワークの問題によって，次の状態になっている可能性があります。

- サーバにリクエストが到達していない
- 再接続処理が繰り返し行われている

ネットワークの状態を見直してください。

指定したタイムアウト値が小さい場合には，
org.omg.Messaging.RELATIVE_RT_TIMEOUT_POLICY_TYPE の値を大きくしてください。

KFCB91005-W

ORB trace is unavailable due to failure to create directory. reason=aa....aa, path=bb....bb.

aa....aa：エラーコード

Windows は GetLastError()，UNIX は errno の値です。

bb....bb：パス名

作成時にエラーが発生したパス名です。

[要因]

トレース出力ディレクトリの生成に失敗しました。そのため，生成に失敗したディレクトリ下の Java ORB トレースは取得されません。

[システムの処理]

処理を続行します。

[対策]

指定されたパスの，どのディレクトリまで作成できたかを確認した上で，エラーコードからパスやアクセス権限などを見直してください。

出力ディレクトリのパスを再設定し，プロセスを再起動することをお勧めします。

問題を解決したあとプロセスを再起動しない場合には，障害発生時にトレース情報がないので，調査が困難になります。

8.3 KFCB92000～KFCB92999 のメッセージ

KFCB92000-I

Now starting OSAgent.

[要因]

osagent が起動しました。

[システムの処理]

処理を継続します。

[対策]

なし。

KFCB92001-I

OSAgent is now online. OSAGENT_PORT=aa....aa ,
OSAGENT_CLIENT_HANDLER_PORT=bb....bb

aa....aa：OSAGENT_PORT の値

bb....bb：OSAGENT_CLIENT_HANDLER_PORT の値

指定されていない場合は，それぞれのデフォルト値（14000 および 0）が表示されます。

vbroker.orb.htc.msgLog.security=true が指定されている場合は，それぞれ"******"が表示されます。

[要因]

osagent がサービスを開始しました。

[システムの処理]

処理を継続します。

[対策]

なし。

KFCB92002-I

OSAgent terminated. SIGNAL=aa....aa

aa....aa：シグナル番号

[要因]

（Windows）

osagent が終了しました。

(UNIX)

osagent がシグナル aa....aa 番によって終了しました。

"SIGNAL=aa....aa"は UNIX の場合だけ出力されます。Windows では該当個所のメッセージは出力されません。

[システムの処理]

osagent プロセスを終了します。

[対策]

なし。

KFCB92003-I

OSAgent stopped.

[要因]

osagent が WM_CLOSE メッセージによって終了しました。

このメッセージは Windows で-m オプションを指定した osagent だけ出力されます。

[システムの処理]

osagent プロセスを終了します。

[対策]

なし。

KFCB92004-E

OSAgent abnormal terminated.

[要因]

osagent がエラーによって終了しました。

[システムの処理]

osagent プロセスを終了します。

[対策]

osagent に指定したパラメタ（ポート番号，引数など），環境などに問題がないか確認してください。

KFCB92005-E

Lack of the memory occurred in OSAgent.

[要因]

osagent でメモリ不足が発生しました。

[システムの処理]

osagent プロセスを終了します。

[対策]

システムで osagent を複数起動するなどして，一つの osagent プロセスに対する負荷を軽減してください。

KFCB92006-W

OSAgent detected client's going down. Host Name:[aa....aa] Process id:[bb....bb]

aa....aa：osagent と通信があったプロセスが起動されていたホスト名

bb....bb：osagent と通信があったプロセス ID

[要因]

osagent と一定期間の通信がなかったため，osagent は該当プロセスの登録情報を削除しました。

登録情報を削除する条件は次のとおりです。

- 該当プロセスが異常終了している（C++ ORB の場合，［Ctrl］＋［C］での割り込みは含まれません）。
- 該当プロセスと osagent が通信できない状態になっている。
- 該当プロセスが，ORB.shutdown（Java ORB）や ORB.destroy（Java ORB），または CORBA::ORB::shutdown（C++ ORB）や CORBA::ORB::destroy（C++ ORB）を実行しないままプロセスを終了している。

注

アプリケーションプログラムが Java ORB の場合，プロセス ID はプロセスを識別するために TPBroker が割り当てた整数値になります。

Java ORB で起動したときのプロセス ID に該当する，アプリケーションプログラムの情報は，osagent のバーボースログで確認してください。

[システムの処理]

処理を継続します。

[対策]

該当プロセスの登録情報削除が予期しない動作である場合，ネットワーク構成，該当プロセスの起動状態，および該当プロセスの処理を見直してください。

KFCB92007-W

Invalid host is specified in agentaddr. (aa....aa)

aa....aa：agentaddr ファイルに指定されているホスト

[要因]

osagent は agentaddr ファイルに指定されているホスト aa....aa を，ホストとして解決できませんでした。

[システムの処理]

処理を継続します。

[対策]

agentaddr ファイルに指定されているホストが正しいか確認して修正してください。修正を有効にするためには，osagent を再起動してください。

8.4　トレース情報取得ができない場合に出力されるメッセージ

この節では，トレース情報取得ができない場合に出力されるメッセージについて説明します。このメッセージは，C++ ORB の場合だけ，標準出力に出力されます。

ORB trace is unavailable due to failure to create directory. reason=aa....aa, path=bb....bb.

aa....aa：エラーコード

Windows は GetLastError()，UNIX は errno の値です。

bb....bb：パス名

作成時にエラーが発生したパス名です。

[要因]

トレース出力ディレクトリの生成に失敗しました。そのため，生成に失敗したディレクトリ下の C++ ORB トレースは取得されません。

[システムの処理]

処理を続行します。

[対策]

指定されたパスの，どのディレクトリまで作成できたかを確認した上で，エラーコードからパスやアクセス権限などを見直してください。

出力ディレクトリのパスを再設定し，プロセスを再起動することをお勧めします。

問題を解決したあとプロセスを再起動しない場合には，障害発生時にトレース情報がなく，調査が困難になります。

ORB trace is unavailable due to system call failure. trace=aa....aa, func=bb....bb, reason=cc....cc.

aa....aa：トレース種別

mdl：モジュールトレース

comt：通信トレース

stk：スタックトレース

hgt：バーボースログ

bb....bb：エラーが発生したシステムコール名

cc....cc：システムコールのエラーコード

[要因]

システムコールでエラーが発生したため，トレース情報を取得できません。

[システムの処理]

処理を続行します。

[対策]

エラーコードから問題を解決し，プロセスを再起動することをお勧めします。

問題を解決したあとプロセスを再起動しない場合には，障害発生時にトレース情報がなく，調査が困難になります。

8.5 hvmgtee コマンドのエラーメッセージ

この節では，hvmgtee コマンドのエラーメッセージについて説明します。このエラーメッセージは，UNIX の場合だけ出力されます。

hvmgtee コマンドでエラーが発生した場合，エラーメッセージは次のフォーマットで標準エラー出力に出力されます。

```
*** hvmgtee error ,cause=<XXX> ,errno=<XXX> ,line=<XXX> ***

（凡例）　XXX：任意の文字列
```

XXX に出力される情報を項目ごとに次に示します。

- cause
  エラーの内容（関数名など）が出力されます。この情報で要因を特定できます。詳細は「表 8-2　cause に出力される情報とその要因」を参照してください。
- errno
  関数のエラーが要因となる場合に，errno にセットされた値が出力されます。
- line
  エラーが発生した hvmgtee コマンドの行番号が出力されます。

cause に出力される情報でエラー要因を特定します。errno や line の情報を参考にして，要因を取り除いたあとに，hvmgtee コマンドを再起動してください。

cause に出力される情報とその要因を次の表に示します。

表 8-2　cause に出力される情報とその要因

項番	cause に出力される情報	考えられる要因
1	$VBROKER_ADM not defined	環境変数 VBROKER_ADM が指定されていません。
2	$VBROKER_ADM over length	環境変数 VBROKER_ADM に指定されたパス名が 980 バイトを超えています。
3	child process dead	hvmgtee コマンドのプロセスが，起動後に何らかの要因で終了しました。
4	dup error1	osagent の標準出力のファイルディスクリプタを FIFO へ移すのに失敗しました。
5	dup error2	osagent の標準エラー出力のファイルディスクリプタを FIFO へ移すのに失敗しました。
6	execlp error	hvmgtee コマンドの起動に失敗しました。
7	FIFO can't be opened	hvmgtee コマンドが FIFO のアクセスに失敗しました。

項番	cause に出力される情報	考えられる要因
8	FIFO can't be opened in the time	osagent が作成した FIFO のアクセスを一定時間内に成功させることができませんでした。
9	fifo_dir opendir error	hgtfifo ディレクトリのアクセスに失敗しました。
10	fork error	hvmgtee コマンドを自動実行させるために必要な fork の処理で失敗しました。
11	malloc error	hvmgtee コマンドのプロセス内で malloc 処理が失敗しました。
12	mkfifo error	hgtfifo ディレクトリの下で FIFO の作成に失敗しました。
13	output_dir can't open	hvmgtee コマンドがファイル出力するディレクトリのアクセスに失敗しました。
14	outputfile fopen error	バーボースログの出力ファイルの作成に失敗しました。
15	parents process dead	hvmgtee コマンドのプロセスを自動起動した osagent が，ログ出力前に終了しました。

8.6 hdumpns コマンドのエラーメッセージ

この節では，hdumpns コマンドのエラーメッセージについて説明します。このエラーメッセージは，Cosminexus TPBroker（Windows）だけで，戻り値が 0 以外の場合に，出力されます。

```
hdumpns error:can't communicate with specified process.
ERROR CODE = <エラーコード>
```

[要因]

指定されたネーミングサービスと通信できません。

[対策]

引数に指定されたプロセス ID が正しいかを確認してください。

[戻り値]

1

```
hdumpns error:can't send request.
ERROR CODE = <エラーコード>
```

[要因]

指定されたネーミングサービスと通信できません。

[対策]

hdumpns コマンドを再度実行してください。

負荷が高い状態では hdumpns コマンドが一時的にエラーになる場合があります。その場合，しばらくしてから hdumpns コマンドを再度実行してください。

[戻り値]

2

9 Cosminexus のバージョンアップ時の移行

この章では，Cosminexus のバージョンアップ時の TPBroker の移行について説明します。

9.1　Cosminexus のバージョンアップ時の移行の流れ

この節では，Cosminexus のバージョンアップ時の移行の流れについて説明します。

移行の流れを次の図で示します。

図 9-1　Cosminexus のバージョンアップ時の移行の流れ

9.2　データのバックアップ

この節では，Cosminexus のバージョンアップ時に必要なデータのバックアップについて説明します。

バージョンアップする前に，使用している機能に応じて，必要な情報やファイルのバックアップを取得する必要があります。

9.2.1　ORB 機能を使用している場合

ORB 機能を使用している場合，使用している Cosminexus TPBroker に応じて，バックアップを取得します。

(1)　Cosminexus TPBroker Version 5

次のファイルを作成しているときには，バックアップを取得します。ファイルの格納先は，環境変数 VBROKER_ADM に指定されたディレクトリです。

- agentaddr
- localaddr
- htc.clienthandleraddr
- htc.props（Linux 以外の場合）
- HVMGTEE_DEF（AIX および Linux の場合）
- HVIORB_DEF

なお，agentaddr ファイル，localaddr ファイル，および htc.clienthandleraddr ファイルについては，環境変数 VBROKER_ADM とは別の環境変数でファイルの格納先を変更できます。その場合，ファイルの格納先は次の表に示す環境変数を確認してください。

表 9-1　格納先を変更できるファイルと環境変数の対応（Cosminexus TPBroker Version 5）

ファイル	環境変数
agentaddr	OSAGENT_ADDR_FILE
localaddr	OSAGENT_LOCAL_FILE
htc.clienthandleraddr	HVI_OSAGENT_CLIENTHANDLERADDR_FILE

9.3 TPBroker のインストール

この節では，Cosminexus TPBroker Version 5 のインストールについて説明します。

使用している OS，および Cosminexus のバージョンに応じて，Cosminexus TPBroker Version 5 をインストールしてください。

9.3.1 Windows の場合

Windows の場合は，使用している Cosminexus のバージョンに応じて，次に示す手順で Cosminexus TPBroker Version 5 をインストールします。

(1) Cosminexus Version 8 からのバージョンアップの場合

Cosminexus Version 9 製品のインストーラで，Cosminexus TPBroker を選択し，インストールします。

9.3.2 AIX，Linux の場合

AIX，Linux の場合は，使用している Cosminexus のバージョンに応じて，次に示す手順で Cosminexus TPBroker Version 5 をインストールします。

(1) Cosminexus Version 8 からのバージョンアップの場合

Cosminexus TPBroker Version 5 をインストールします。

9.4 TPBroker のセットアップ（32 ビット用 Windows）

この節では，32 ビット用 Windows の場合の TPBroker のセットアップ方法について説明します。

使用している Cosminexus TPBroker に応じて，TPBroker をセットアップする必要があります。

9.4.1 Cosminexus TPBroker Version 5 を使用していた場合（32 ビット用 Windows）（TPBroker のセットアップ）

TPBroker Version 5 を使用していた場合，使用している機能に応じて，次に示す手順で TPBroker のセットアップをします。

(1) ORB 機能を使用していた場合（32 ビット用 Windows）（TPBroker のセットアップ）

使用している環境に応じて，セットアップをします。

- 移行後に環境変数 VBROKER_ADM の値を変更し，agentaddr ファイル，localaddr ファイル，htc.clienthandleraddr ファイル，および htc.props ファイルを使用するとき

  次に示すファイルを変更後の環境変数 VBROKER_ADM で指定したディレクトリに合わせて格納し直します。

  - %VBROKER_ADM%¥agentaddr
  - %VBROKER_ADM%¥localaddr
  - %VBROKER_ADM%¥htc.clienthandleraddr
  - %VBROKER_ADM%¥htc.props
- 環境変数 VBROKER_ADM に<TPBroker のインストール先ディレクトリ>¥adm と異なるディレクトリを指定しているとき

  <TPBroker のインストール先ディレクトリ>¥adm 以下のファイルを環境変数 VBROKER_ADM で指定したディレクトリにコピーします。
- 移行前の環境で，agentaddr ファイル，localaddr ファイル，htc.clienthandleraddr ファイル，および htc.props ファイルの格納位置を環境変数 VBROKER_ADM とは別の環境変数で指定していたとき

  移行前の環境変数で指定していたディレクトリを基にファイルを格納し直します。

9.5 TPBroker のセットアップ（64 ビット用 Windows）

この節では，64 ビット用 Windows の場合の TPBroker のセットアップ方法について説明します。

使用している機能に応じて，TPBroker をセットアップします。

9.5.1 ORB 機能を使用していた場合（64 ビット用 Windows）（TPBroker のセットアップ）

使用している環境に応じて，セットアップをします。

- 移行後に環境変数 VBROKER_ADM の値を変更し，agentaddr ファイル，localaddr ファイル，htc.clienthandleraddr ファイル，および htc.props ファイルを使用するとき

  次に示すファイルを変更後の環境変数 VBROKER_ADM で指定したディレクトリに合わせて格納し直します。

  - %VBROKER_ADM%¥agentaddr
  - %VBROKER_ADM%¥localaddr
  - %VBROKER_ADM%¥htc.clienthandleraddr
  - %VBROKER_ADM%¥htc.props
- 環境変数 VBROKER_ADM に<TPBroker のインストール先ディレクトリ>¥adm と異なるディレクトリを指定しているとき

  <TPBroker のインストール先ディレクトリ>¥adm 以下のファイルを環境変数 VBROKER_ADM で指定したディレクトリにコピーします。
- 移行前の環境で，agentaddr ファイル，localaddr ファイル，および htc.clienthandleraddr ファイルの格納位置を環境変数 VBROKER_ADM とは別の環境変数で指定していたとき

  移行前の環境変数で指定していたディレクトリを基にファイルを格納し直します。

9.6 TPBroker のセットアップ（AIX，Linux）

この節では，AIX，Linux の場合の TPBroker のセットアップ方法について説明します。

使用していた Cosminexus TPBroker に応じて，TPBroker をセットアップする必要があります。

9.6.1 Cosminexus TPBroker Version 5 を使用していた場合（AIX，Linux）（TPBroker のセットアップ）

TPBroker Version 5 を使用していた場合，使用している機能に応じて，次に示す手順で TPBroker のセットアップをします。

(1) ORB 機能を使用していた場合（AIX，Linux）（TPBroker のセットアップ）

使用している環境に応じて，セットアップをします。

- ファイルをバックアップしていて，移行後もそのファイルを使用するとき
  バックアップしたファイルを環境変数 VBROKER_ADM に指定したディレクトリに格納します。バックアップしたファイルについては，「9.2.1 ORB 機能を使用している場合」を参照してください。
- 環境変数 VBROKER_ADM に$TPDIR/adm と異なるディレクトリを指定しているとき
  $TPDIR/adm 以下のファイルを環境変数 VBROKER_ADM で指定したディレクトリにコピーします。

10 接続性に関する注意事項

この章では，TPBroker の接続性に関する注意事項について説明します。

10.1 Cosminexus TPBroker 05-24 と TPBroker Version 3 を接続させる場合の注意事項

Cosminexus TPBroker 05-24 と TPBroker Version 3 を接続させる場合，引数および戻り値のデータ型によって問題が発生します。次の表に，引数および戻り値のデータ型ごとに発生する現象，回避策を示します。

表 10–1　データ型と現象・回避策

項番	引数や戻り値のデータ	発生する現象	回避策
1	wchar，wstring	文字化け	次のどちらかの対処をしてください。 • sequence<octet>を使用し，アプリケーションで使用する文字コードを統一する（推奨）。 • wstring または wchar を使用し，アプリケーションで使用する文字コードを UTF-16 にする。
2	sequence<any>に構造型(enum，struct，union，sequence，array)を複数設定したデータ	MARSHAL 例外	TPBroker for C++ V3 のアプリケーションを起動する場合 -ORBtypecodebounds 1 を指定する。 TPBroker for Java V3 のアプリケーションを起動する場合 -DORBtypecodeBounds=true を指定する。

10.2　CORBA::Any のマーシャリングに関する注意事項

CORBA::Any のマーシャリング時に，CORBA2.5 仕様に準拠した方法，VisiBroker 独自の方法のどちらかを使用することを設定します。

TPBroker 05-15-/A（32 ビット用 Windows）または TPBroker 05-00～05-15（64 ビット用 Windows）の C++ ORB と，CORBA::Any に CORBA::WChar 型を格納して通信する場合には，"false"を設定する必要があります。詳細は，「3.4.14　CORBA::Any 型のマーシャリング方法の変更」を参照してください。

10.3 Messaging::SyncScopePolicy に関する注意事項

TPBroker Version 5 クライアントから IDL に oneway 属性として定義したリクエストを，TPBroker Version 3 オブジェクトの呼び出しに使用する場合，次を指定した同期リクエストは正しく動作しません。

- Messaging::SyncScopePolicy に Messaging::SYNC_WITH_SERVER を指定
- Messaging::SyncScopePolicy に Messaging::SYNC_WITH_TARGET を指定

なお，クライアントおよびサーバの双方が TPBroker Version 5 の場合は，正しく動作します。サーバが TPBroker Version 3 の場合，oneway リクエストには次を指定してください。

- Messaging::SyncScopePolicy に Messaging::SYNC_WITH_TRANSPORT を指定
- Messaging::SyncScopePolicy に Messaging::SYNC_NONE を指定

10.4 引数，戻り値，またはユーザー例外に使用するクラスに関する注意事項

Cosminexus TPBroker 05-17 より前の Version 5 Java ORB の間で RMI-IIOP 通信する場合，次のクラスを引数，戻り値，またはユーザー例外に使用しないでください。

- java.lang.StringBuffer
- java.math.BigDecimal
- java.net.Inet6Address
- java.security.SecureRandom
- java.text.DecimalFormat
- java.util.Locale
- javax.naming.Binding
- javax.naming.directory.SearchResult
- javax.naming.NameClassPair

10.5　日立プログラムプロダクト向けの接続性に関する注意事項

TPBroker for C++ V3 との通信では，IDL で定義した wstring 型に NULL 文字列を格納して送信しないでください。なお，この注意事項は，TPBroker 上で動作する日立プログラムプロダクトの場合に該当します。

11 障害発生時の対応

この章では，障害発生時の対応について説明します。

11.1　障害が発生した場合に取得および退避するトラブルシュート情報

TPBroker がバンドルする VisiBroker では，トラブルシュート用に次の情報を取得しています。

- モジュールトレース
- エラーログ
- 通信トレース

障害が発生した場合は，VisiBroker が取得するこれらの情報を退避してください。各トラブルシュート情報の取得先を次に示します。

モジュールトレース※

（Windows）

%VBROKER_ADM%¥..¥log¥mdltrc または%VBROKER_ADM%¥..¥logj¥mdltrc

（Unix）

${VBROKER_ADM}/../log/mdltrc または${VBROKER_ADM}/../logj/mdltrc

エラーログ

（Windows）

%VBROKER_ADM%¥..¥logj

（Unix）

${VBROKER_ADM}/../logj

通信トレース※

（Windows）

%VBROKER_ADM%¥..¥log¥comtrc または%VBROKER_ADM%¥..¥logj¥comtrc

（Unix）

${VBROKER_ADM}/../log/comtrc または${VBROKER_ADM}/../logj/comtrc

注※

モジュールトレース，通信トレースについては，取得前に，それぞれのディレクトリ下のファイルに次のコマンドを実行してください。

hmapfsync *.dat

また，次の二つの出力結果もあわせて取得し，退避してください。

（Windows）

%TPDIR%¥bin¥vbver.exe %TPDIR%¥bin¥orb_r.dll

%TPDIR%¥bin¥vbver.exe %TPDIR%¥lib¥vbjorb.jar

（Unix）

${TPDIR}/bin/vbver ${TPDIR}/bin/liborb_r.*

```
${TPDIR}/bin/vbver ${TPDIR}/lib/vbjorb.jar
```

なお，HVI_TRACEPATH 環境変数を設定して，トラブルシュートファイルの出力ディレクトリを変更している場合は，変更先のディレクトリにあるすべてのトレースファイルを退避してください。

12 J2EE 環境で TPBroker for C++/Java Version 3 と連携する方法

この章では，J2EE 環境で TPBroker for C++/Java Version 3 と連携する方法について説明します。

12.1 Cosminexus TPBroker for Java Version 4 および Cosminexus TPBroker Version 5 で提供する ORB 機能

Cosminexus TPBroker for Java Version 4 および Cosminexus TPBroker Version 5 で提供する ORB 機能を次に示します。

ORB Version 3 機能

Cosminexus TPBroker for Java Version 4 で提供していた，CORBA 2.1 に準拠した TPBroker for Java Version 3 と互換性のある ORB 機能です。

ORB Version 4 機能

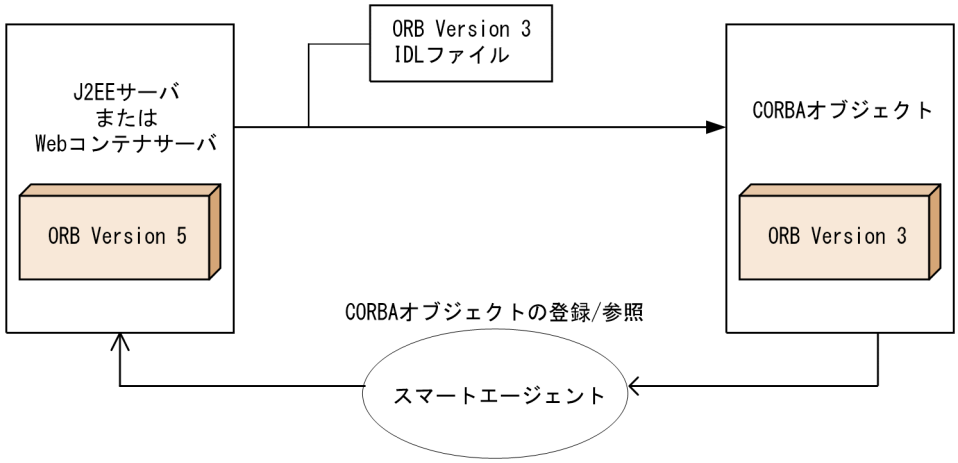
Cosminexus TPBroker for Java Version 4 で提供していた，J2EE 環境（EJB，Servlet など）で使用する ORB 機能です。

ORB Version 5 機能

Cosminexus TPBroker Version 5 で提供する，J2EE 環境（EJB，Servlet など）で使用する ORB 機能です。

J2EE 環境（J2EE サーバ，または Web コンテナサーバの実行環境）で，TPBroker for C++/Java Version 3（ORB Version 3：CORBA 2.1 に準拠した ORB）上で動作している CORBA オブジェクトを呼び出す仕組みを次の図に示します。

図 12-1 TPBroker for C++/Java Version 3 上で動作している CORBA オブジェクトを呼び出す仕組み



12.2　環境変数の設定

開発および実行する環境ごとに，ORB Version 3 または ORB Version 5 の環境変数を設定します。ORB Version 5 の環境変数についてはマニュアル「TPBroker ユーザーズガイド」の「2.2　環境変数を設定する」を，ORB Version 3 の環境変数については，ORB Version 3 のソフトウェアマニュアルおよびソフトウェア添付資料を参照してください。

12.3　アプリケーションの作成手順

次の手順でアプリケーションを作成してください。

1. ORB Version 3 の IDL ファイルを用意します。
   IDL ファイルは ORB Version 3 のサポート範囲に従って記述してください。
2. J2EE 環境で，ORB Version 5 の IDL コンパイラによってスタブを生成します。
3. CORBA オブジェクト環境で，ORB Version 3 の IDL コンパイラによってスケルトンを生成します。
4. Enterprise Bean やサーブレットで，CORBA オブジェクト呼び出しコードをプログラミングします。

12.4　ORB Version 3 の CORBA オブジェクトの呼び出し手順

次の手順で J2EE サーバ，または Web コンテナサーバを起動してください。

1. スマートエージェントを起動します。
   osagent に関しては，TPBroker で提供する osagent を使用してください。
2. J2EE 環境でスマートエージェントが利用するポート番号を設定します。
3. J2EE サーバ，または Web コンテナサーバを起動します。

J2EE サーバおよび Web コンテナサーバについては，マニュアル「Cosminexus V11 アプリケーションサーバ システム構築・運用ガイド」の「5.2　システムの構築・運用時に使用するコマンド」を参照してください。

13 RMI-IIOP アプリケーションを JDK9 以降でコンパイルまたは実行する場合の注意事項

RMI-IIOP アプリケーションを Cosminexus Developer's Kit for Java (TM) 09-80 以降でコンパイルまたは実行する場合の注意事項について説明します。

13.1 現象

JDK9 では，Java Module System の考え方を導入し，内部 API（フィールドアクセスを含む）のカプセル化によって，Java プログラム実行環境の安全性を高めています。

JDK9 と JDK8 以前との互換性については，内部 API やフィールドに対するアクセスを可能とする手段として，--add-opens などのオプションを用意しています。試行錯誤的な方法で，必要最小限の場所を指定できます。そのため，互換性を維持しながら，安全性を保った実行環境を実現しています。この仕様や考え方など，JDK9 の詳細については次の Oracle 社のページをご参照ください。

- Oracle JDK9 ドキュメント
  https://docs.oracle.com/javase/jp/9/
- Java Platform, Standard Edition Oracle JDK 9 移行ガイド：JDK9 への移行
  https://docs.oracle.com/javase/jp/9/migrate/toc.htm#JSMIG-GUID-7744EF96-5899-4FB2-B34E-86D49B2E89B6
  このページにある「ランタイム・アクセス警告の理解」の部分を参照してください。

JDK9 で導入された Java Module System の影響によって，JDK8 以前でコンパイルまたは実行できていた RMI-IIOP アプリケーションを，JDK9 以降でコンパイルまたは実行しようとした場合，アプリケーションの実装によって次の現象が発生することがあります。

項番	現象が発生する契機	現象
1	コンパイル時	java2iiop コマンド実行時に，次のメッセージが標準エラー出力に出力されます。クライアントスタブやサーバスケルトンなどの生成に失敗します。 java2iiop: fatal error: internal problem; please contact Inprise Customer Support
2	実行時	リクエストコールで，java.lang.reflect.InaccessibleObjectException 例外が発生します。

13.2 コンパイル時の対応

java2iiop コマンド実行時に，次のメッセージが標準エラー出力に出力された場合，クライアントスタブやサーバスケルトンなどの生成に失敗します。

java2iiop: fatal error: internal problem; please contact Inprise Customer Support

次の手順に従ってください。

1. java2iiop コマンドを-XXX_debug オプションを付けて実行してください。

   (例)

   ```
   > java2iiop -XXX_debug HelloRemote
   ```

2. java.lang.reflect.InaccessibleObjectException 例外のメッセージが，標準エラー出力に出力されます。例外メッセージからモジュール名とパッケージ名を見つけてください。

   (例外メッセージの例)

   ```
   java2iiop: fatal error: internal problem: unexpected exception: java.lang.reflect.Inacces
   sibleObjectException: Unable to make field private final java.util.List aaa.bbb.Ccc.ddd a
   ccessible: module xxx.yyy does not ”opens aaa.bbb” to unnamed module @735b478
   ```

   モジュール名とパッケージ名は，メッセージ中の「accessible: module」の直後に以下の形式で出力されています。

   accessible: module モジュール名

   does not "opens パッケージ名"

   例外メッセージの例の場合，モジュール名は「xxx.yyy」，パッケージ名は「aaa.bbb」となります。

3. 手順 2.で見つけたモジュール名とパッケージ名を--add-opens オプションに指定して，java2iiop コマンドを実行してください。

   (例)

   ```
   > java2iiop -J--add-opens=xxx.yyy/aaa.bbb=ALL-UNNAMED -XXX_debug HelloRemote
   ```

4. java.lang.reflect.InaccessibleObjectException 例外が発生しなくなるまで，手順 2.～手順 3.を繰り返して--add-opens オプションを java2iiop コマンドに追加してください。

   (例)

   ```
   > java2iiop -J--add-opens=xxx.yyy/aaa.bbb=ALL-UNNAMED -J--add-opens=xxx.yyy/ccc.ddd=ALL-U
   NNAMED -XXX_debug HelloRemote
   ```

5. java.lang.reflect.InaccessibleObjectException 例外が発生しなくなった場合，java2iiop コマンドから-XXX_debug オプションの指定を削除してください。

13.3　実行時の対応

java2iiop コマンド実行時に java.lang.reflect.InaccessibleObjectException 例外が発生しても，実行時には発生しない場合があります。

実行時に現象が発生した場合だけ，「13.2　コンパイル時の対応」の手順 3.で java2iiop コマンドに指定した--add-opens オプションを，クライアントアプリケーションおよびサーバアプリケーションの実行時に指定してください。

(例)

```
> vbj -J--add-opens=xxx.yyy/aaa.bbb=ALL-UNNAMED HelloServer
```

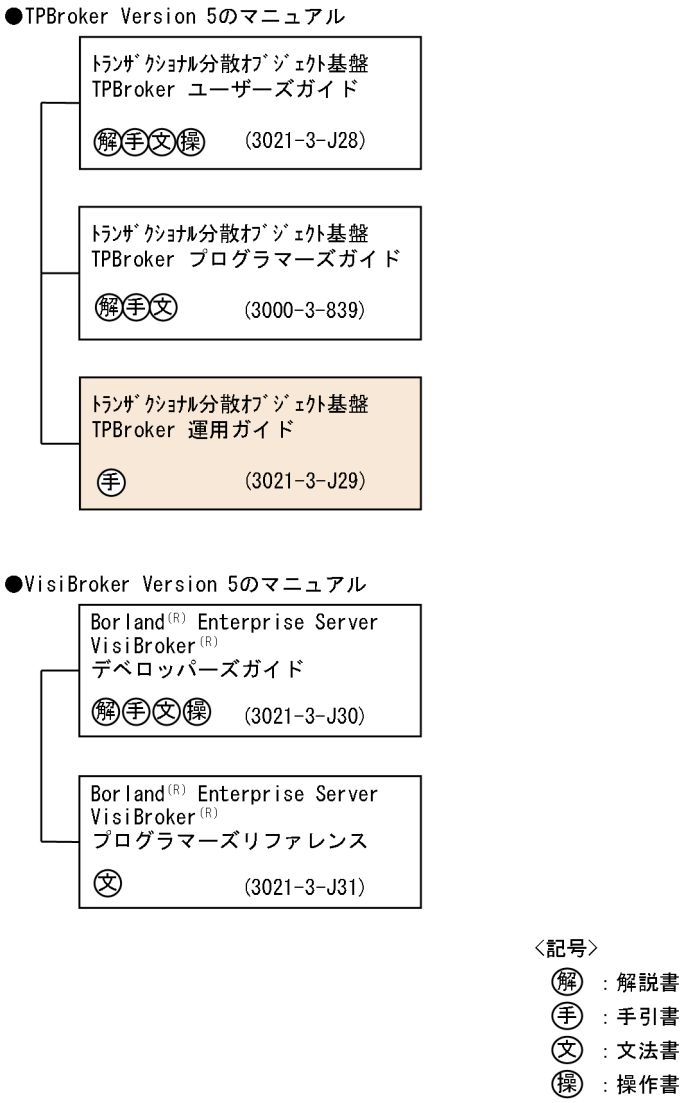
付録

付録 A　このマニュアルの参考情報

このマニュアルを読むに当たっての参考情報を示します。

付録 A.1　関連マニュアル

関連マニュアルを次に示します。必要に応じてお読みください。



- HA モニタのマニュアル
  - 高信頼化システム監視機能 HA モニタ AIX(R)編（3000-9-130）
  - 高信頼化システム監視機能 HA モニタ Linux(R)編（3000-9-132）
  - 高信頼化システム監視機能 HA モニタ メッセージ（3000-9-134）
- Cosminexus のマニュアル
  - Cosminexus V11 アプリケーションサーバ システム構築・運用ガイド（3021-3-J02）

- Cosminexus V11 アプリケーションサーバ リファレンス 定義編（アプリケーション/リソース定義）（3021-3-J17）

なお，JP1/ServerConductor/Deployment Manager の詳細については，JP1/ServerConductor/Deployment Manager のマニュアルを参照してください。

このマニュアルでは，次のマニュアルを省略して表記しています。マニュアルの正式名称とこのマニュアルでの表記を次の表に示します。

マニュアルの正式名称	このマニュアルでの表記
TPBroker Version 5 トランザクショナル分散オブジェクト基盤 TPBroker ユーザーズガイド	TPBroker ユーザーズガイド
VisiBroker Version 5 Borland (R) Enterprise Server VisiBroker (R) デベロッパーズガイド	Borland Enterprise Server VisiBroker デベロッパーズガイド
JP1 Version 8 JP1/ServerConductor/Deployment Manager	JP1/ServerConductor/Deployment Manager
高信頼化システム HA モニタ AIX(R)編	HA モニタ
高信頼化システム HA モニタ Linux(R)編	

付録 A.2　このマニュアルでの表記

このマニュアルでは，製品名を次のように表記しています。

表記			製品名
AIX			AIX V7.1
			AIX V7.2
Java			Java™
Java SE			Java™ Platform, Standard Edition
JavaVM			Java™ Virtual Machine
Linux			Red Hat Enterprise Linux 7.1 (AMD/Intel 64)
			Red Hat Enterprise Linux 8.1 (AMD/Intel 64)
VisiBroker			Borland(R) Enterprise Server VisiBroker(R)
Windows	Windows Server 2016	Windows Server 2016 Standard	Windows Server(R) 2016 Standard 日本語版

表記			製品名
		Windows Server 2016 Datacenter	Windows Server(R) 2016 Datacenter 日本語版
	Windows Server 2019	Windows Server 2019 Standard	Windows Server(R) 2019 Standard 日本語版
		Windows Server 2019 Datacenter	Windows Server(R) 2019 Datacenter 日本語版
	Windows 10	Windows 10 x64	Windows(R) 10 Enterprise 日本語版 (64 ビット版)

下記に示すプログラムプロダクトで仕様差がない場合，TPBroker と表記しています。

- Cosminexus TPBroker
- TPBroker
- TPBroker Developer
- TPBroker Client

また，ご使用になるプログラミング言語または OS によって説明が異なる場合，次の記号を使用しています。

記号	意味
(C++ ORB)	C++言語，または TPBroker で提供する C++インタフェースを使用する ORB 機能に該当
(Cosminexus TPBroker)	Cosminexus TPBroker に該当
(Java ORB)	Java 言語，または TPBroker で提供する Java インタフェースを使用する ORB 機能に該当
(UNIX)	すべての UNIX プラットフォームに該当
(Visual Studio 2005)	プログラムプロダクト「P-2A64-F124，P-2A64-F224，P-2A64-F324」に該当
(Windows)	Windows に該当

なお，環境変数の設定を説明する文中で，Windows と UNIX の両方が考えられる場合，UNIX の表記を示し，その後ろに Windows の表記を（　）で囲んでいます。

例　${VBROKER_ADM}/../logj（%VBROKER_ADM%¥..¥logj）

付録 A.3　英略語

このマニュアルで使用する英略語を次に示します。

英略語	英字での表記
ADM	Administration

英略語	英字での表記
API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CTM	Cosminexus Component Transaction Monitor
DB	Database
DBMS	Database Management System
DHCP	Dynamic Host Configuration Protocol
DLL	Dynamic Linking Library
DNS	Domain Name System
DTP	Distributed Transaction Processing
EB	Enterprise Bean
EJB	Enterprise JavaBeans™
ES	Embedded System
FIFO	First-In First-Out
GIOP	General Inter - ORB Protocol
GUI	Graphical User Interface
HACMP	High Availability Clustering Multi-Processing
ID	Identification
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
IOR	Interoperable Object Reference
IP	Internet Protocol
J2EE	Java™ 2 Platform, Enterprise Edition
JSP	Java Server Pages
LAN	Local Area Network
LIOP	Local Inter-ORB Protocol
MSDN	Microsoft Developer Network
OAD	Object Activation Daemon
OMG	Object Management Group
ORB	Object Request Broker
OS	Operating System

英略語	英字での表記
OTS	Object Transaction Service
OTSCRM	Object Transaction Service Communication Resource Manager
PID	Process Identification
POA	Portable Object Adapter
QoS	Quality of Service
RM	Reliable Messaging
RMI	Java Remote Method Invocation
SDK	Software Development Kit
TCP	Transmission Control Protocol
TCS	Transaction Context Server
UAP	User Application Program
UDP	User Datagram Protocol
UTC	Universal Time Coordinated
UTF	UCS Transformation Format
VM	Virtual Machine

付録 A.4　KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

付録 B　用語解説

(英字)

ADM

TPBroker が提供する，システム運用時のアプリケーションの起動，停止，監視，および再起動を制御する運用支援機能のことです。

C++ ORB

VisiBroker を用いて C++言語で作成されたプログラムに有効になる ORB 機能のことです。

VisiBroker が提供する osagent やネーミングサービスなどのデーモンやコマンドのうち，C++ ORB は，osagent の機能があります。

CORBA

分散オブジェクト技術の標準化を目的として，OMG が策定する業界標準です。

Cosminexus

アプリケーションサーバを中核とした，性能や信頼性の高い業務アプリケーションを開発・実行するためのシステム構築基盤製品です。

Cosminexus TPBroker

Cosminexus 用の TPBroker です。J2EE サーバ上での分散トランザクション処理を実現します。

ただし，CORBA アプリケーションを開発・実行する機能はありません。

GIOP

CORBA 環境で使用される汎用的な上位プロトコルです。データ送受信時の変換復元規則，メッセージフォーマットなどが規定されています。特定のトランスポートプロトコルには依存しません。

HA モニタ

HA クラスタシステムを実現するクラスタソフトウェアです。

HA とは High Availability の略で，高可用性，つまりシステムを連続して稼働できることを意味します。HA クラスタシステムとは，高可用性を実現するシステム構成のことです。

Java ORB

VisiBroker を用いて Java 言語で作成されたプログラムに有効になる ORB 機能のことです。

VisiBroker が提供する osagent やネーミングサービスなどのデーモンやコマンドのうち，Java ORB は，ネーミングサービスなどのデーモンやコマンドの機能があります。

JP1/ServerConductor/Deployment Manager

日立アドバンストサーバ HA8000 シリーズを一括管理するためのソフトウェアです。

資産管理機能，障害管理機能，遠隔操作機能，およびプログラム連携機能があります。

Management Server

運用管理ドメインを構成するサーバプログラムです。運用管理ドメイン単位に一つ配置します。

Management Server は運用管理ドメイン内の各ホストに配置した運用管理エージェントに指示を出して，運用管理ドメイン全体の運用管理を実行します。

Microsoft Cluster Service

Microsoft 社が提供するクラスターサービスのことです。

OAD

インプリメンテーションリポジトリを実現する，オブジェクト活性化デーモンのことです。

インプリメンテーションリポジトリは，サーバがサポートしているクラス，実体化されているオブジェクトとそれらに関する情報をランタイム時にリポジトリとして提供します。また，OAD はクライアントがオブジェクトを参照するときにインプリメンテーションを自動的に活性化する目的にも使用されます。

OMG

オブジェクト指向技術の普及と標準化のために設立された非営利団体です。

ORB

TPBroker が提供する，分散オブジェクト同士が通信するためのソフトウェア・バス（通信ミドルウェア）です。すべてのオブジェクトは ORB を介して通信します。

通信プロトコル（通信データ形式），通信形態の違いを ORB が吸収するため，システムの拡張や変更が簡単にできます。

osagent

Visibroker が提供するスマートエージェントのことです。

スマートエージェントとは，クライアントアプリケーションとオブジェクトインプリメンテーションの両方に機能を提供する動的な分散ディレクトリサービスです。ネットワークで使用できるオブジェクトを管理し，起動時にクライアントアプリケーションが要求するオブジェクトを探します。

OTS

CORBA が提供するサービスの一つで，トランザクションを制御する機能を持ちます。

QoS

各ポリシーを利用して，クライアントアプリケーションとそれに接続されているサーバとのコネクションの定義と管理を行うサービスです。

TPBroker

OMG が策定した CORBA の機能を提供する分散オブジェクト基盤です。オブジェクト管理やオブジェクト間通信の制御機能（ORB），トランザクション制御機能（OTS），および運用支援機能（ADM）があります。

（ア行）

アウトプロセス

プロセスの起動のしかたです。アウトプロセスで起動させると，J2EE サーバのプロセス外で実行します。アウトプロセスでネーミングサービスを使用する場合，ネーミングサービスはユーザが起動する必要があります。

アンマーシャリング

ORB の共通フォーマットに変換されたリクエストの内容を，サーバで処理できる形式に変換することです。

インプロセス

プロセスの起動のしかたです。インプロセスで起動させると，J2EE サーバのプロセス内で実行するように最適化されるので，パフォーマンスの高いシステムが実現できます。

Application Server では，トランザクションサービスを J2EE サーバのインプロセスで起動できます。インプロセスでトランザクションサービスを起動すると，トランザクション処理を J2EE サーバのプロセス内で実行するように最適化されるので，パフォーマンスの高いシステムを実現できます。

（カ行）

系切り替え

業務を実行しているシステム（系）やサーバに障害が発生した場合に，待機しているシステム（系）やサーバに業務を引き継ぐ機能のことです。

系には，実行系，待機系があります。

（サ行）

シナリオ

JP1/ServerConductor/Deployment Manager で，Linux パッチファイルの適用，バックアップ，およびリストアなどの実行に使用する設定ファイルです。

（タ行）

トラブルシュートファイル

ORB のトラブルシュート機能で出力できるファイルのことです。トラブルシュートファイルには次に示すファイルがあります。

- モジュールトレース
- エラーログ
- 通信トレース
- メッセージログ
- バーボースログ
- スタックトレース
- ネーミングサービス名前空間情報ログ

トランザクションサービス

グローバルトランザクションを使用する場合に，トランザクションを管理するサービスです。TPBroker OTS によって提供されるサービス全体を表します。トランザクションサービスは，J2EE サーバのインプロセスで起動されます。

（ナ行）

名前空間

インタフェース名などの名前（識別子）の重複を回避するための仕組みのことです。インタフェース名が重複した場合に，module キーワードを用いて，名前空間を定義することで，インタフェース名の重複を避けることができます。

ネーミングサービス

Application Server で，リモートオブジェクトの格納場所を管理するためのサービスであり，CORBA 2.5 の仕様に従って実装されています。Cosminexus TPBroker によって提供される機能です。

（ハ行）

プライマリ IP アドレス

gethostname()または sysinfo()を利用して得られるホスト名を使用して，gethostbyname()を発行して得られた IP アドレスのことをいいます。

（マ行）

マーシャリング

クライアントごとに個別に処理されたリクエストの内容を，ORB の共通フォーマットに変換することです。

マルチホーム

インターネットに接続するときに，複数のネットワークインタフェースから，複数のネットワークに接続されることです。負荷分散や障害対策を目的とします。

索引

I

J

K

L

M

O