HITACHI
Inspire the Next

Cosminexus V11 アプリケーションサーバ 機能解説 保守／移行編

解説書

3021-3-J11-50

前書き

■ 対象製品

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」の前書きの対象製品の説明を参照してください。

■ 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

HITACHI，Cosminexus，DABroker，HA モニタ，HiRDB，JP1，OpenTP1，TPBroker，uCosminexus は，株式会社 日立製作所の商標または登録商標です。

AIX は，世界の多くの国で登録された International Business Machines Corporation の商標です。

AMD は，Advanced Micro Devices,Inc.の商標です。

Eclipse および Jakarta は，Eclipse Foundation, Inc.の商標です。

Linux は，Linus Torvalds 氏の米国およびその他の国における登録商標です。

Microsoft，Active Directory，Azure，Excel，Internet Explorer，Microsoft Edge，SQL Server，Windows，Windows Server は，マイクロソフト 企業グループの商標です。

Oracle(R)，Java ，MySQL 及び NetSuite は，Oracle，その子会社及び関連会社の米国及びその他の国における登録商標です。

Red Hat, and Red Hat Enterprise Linux are registered trademarks of Red Hat, Inc. in the United States and other countries. Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.

Red Hat，および Red Hat Enterprise Linux は，米国およびその他の国における Red Hat, Inc.の登録商標です。Linux(R)は，米国およびその他の国における Linus Torvalds 氏の登録商標です。

UNIX は，The Open Group の登録商標です。

その他記載の会社名，製品名などは，それぞれの会社の商標もしくは登録商標です。

Eclipse は，開発ツールプロバイダのオープンコミュニティである Eclipse Foundation, Inc.により構築された開発ツール統合のためのオープンプラットフォームです。

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

■ マイクロソフト製品のスクリーンショットの使用について

マイクロソフトの許可を得て使用しています。

■ 発行

2024 年 2 月 3021-3-J11-50

■ 著作権

All Rights Reserved. Copyright (C) 2020, 2024, Hitachi, Ltd.

変更内容

変更内容(3021-3-J11-50) uCosminexus Application Server 11-40，uCosminexus Client 11-40，uCosminexus Developer 11-40，uCosminexus Service Architect 11-40，uCosminexus Service Platform 11-40

追加・変更内容	変更個所
アプリケーションサーバ 11-40 での主な機能変更についての説明を追加した。	1.4
ZGC を使用している場合の注意事項を追加した。	5.8.1(13)
シグナルが発生した場合に JavaVM が出力するメッセージログを追加した。	5.8.1(17)，5.8.1(23)，5.8.1(26)
JDK17 サポートに伴い，デフォルトで選択されるメモリ管理方式およびモジュール関連オプションについての説明を変更した。	9.19.1，9.19.3
アプリケーションサーバ 11-00，11-10，11-20，または 11-30 から，11-40 までの仕様変更についての説明を追加した。	10.3.3
マニュアル訂正の内容を反映した。	-

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルをお読みになる際の前提情報については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」のはじめにの説明を参照してください。

目次

付録 972

1 アプリケーションサーバの機能

この章では，アプリケーションサーバの機能の分類と目的，および機能とマニュアルの対応について説明します。また，このバージョンで変更した機能についても説明しています。

1.1 機能の分類

アプリケーションサーバは，Java EE 8 に対応した J2EE サーバを中心としたアプリケーションの実行環境を構築したり，実行環境上で動作するアプリケーションを開発したりするための製品です。Java EE の標準仕様に準拠した機能や，アプリケーションサーバで独自に拡張された機能など，多様な機能を使用できます。目的や用途に応じた機能を選択して使用することで，信頼性の高いシステムや，処理性能に優れたシステムを構築・運用できます。

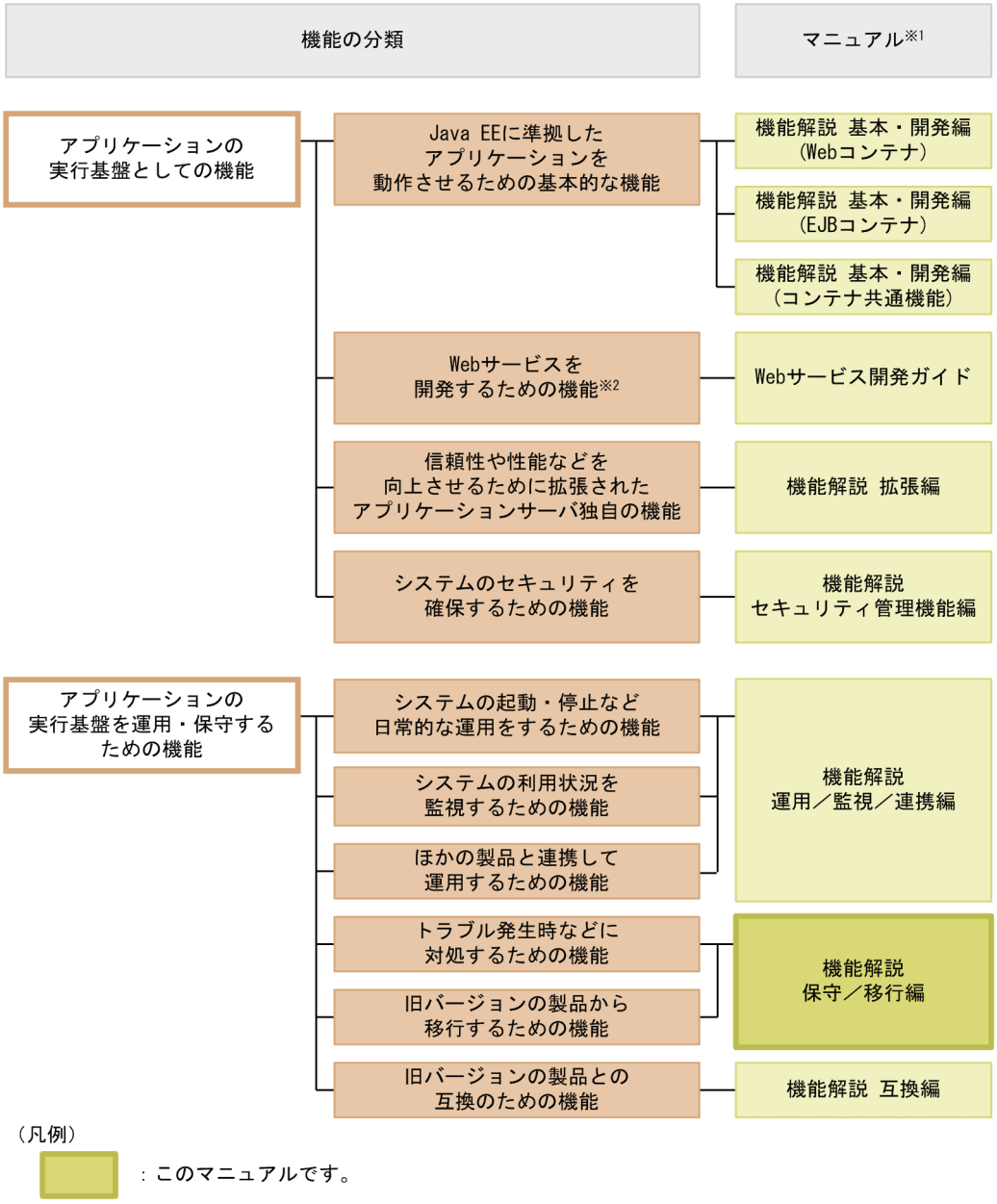
アプリケーションサーバの機能は，大きく分けて，次の二つに分類できます。

- アプリケーションの実行基盤としての機能
- アプリケーションの実行基盤を運用・保守するための機能

二つの分類は，機能の位置づけや用途によって，さらに詳細に分類できます。アプリケーションサーバのマニュアルは，機能の分類に合わせて提供しています。

アプリケーションサーバの機能の分類と対応するマニュアルについて，次の図に示します。

図 1-1　アプリケーションサーバの機能の分類と対応するマニュアル



注※1

マニュアル名称の「アプリケーションサーバ」を省略しています。

注※2

アプリケーションサーバでは，SOAP Web サービスと RESTful Web サービスを実行できます。目的によっては，マニュアル「アプリケーションサーバ Web サービス開発ガイド」以外の次のマニュアルも参照してください。

SOAP アプリケーションを開発・実行する場合

- アプリケーションサーバ SOAP アプリケーション開発の手引

SOAP Web サービスや SOAP アプリケーションのセキュリティを確保する場合

- XML Security - Core ユーザーズガイド

- アプリケーションサーバ Web サービスセキュリティ構築ガイド

XML の処理について詳細を知りたい場合

- XML Processor ユーザーズガイド

ここでは，機能の分類について，マニュアルとの対応と併せて説明します。

1.1.1 アプリケーションの実行基盤としての機能

アプリケーションとして実装されたオンライン業務やバッチ業務を実行する基盤となる機能です。システムの用途や求められる要件に応じて，使用する機能を選択します。

アプリケーションの実行基盤としての機能を使用するかどうかは，システム構築やアプリケーション開発よりも前に検討する必要があります。

アプリケーションの実行基盤としての機能について，分類ごとに説明します。

(1) アプリケーションを動作させるための基本的な機能（基本・開発機能）

アプリケーション（J2EE アプリケーション）を動作させるための基本的な機能が該当します。主に J2EE サーバの機能が該当します。

アプリケーションサーバでは，Java EE 8 に対応した J2EE サーバを提供しています。J2EE サーバでは，標準仕様に準拠した機能のほか，アプリケーションサーバ独自の機能も提供しています。

基本・開発機能は，機能を使用する J2EE アプリケーションの形態に応じて，さらに三つに分類できます。アプリケーションサーバの機能解説のマニュアルは，この分類に応じて分冊されています。

それぞれの分類の概要を説明します。

- Web アプリケーションを実行するための機能（Web コンテナ）
  Web アプリケーションの実行基盤である Web コンテナの機能，および Web コンテナと Web サーバが連携して実現する機能が該当します。
- Enterprise Bean を実行するための機能（EJB コンテナ）
  Enterprise Bean の実行基盤である EJB コンテナの機能が該当します。また，Enterprise Bean を呼び出す EJB クライアントの機能も該当します。
- Web アプリケーションと Enterprise Bean の両方で使用する機能（コンテナ共通機能）
  Web コンテナ上で動作する Web アプリケーションおよび EJB コンテナ上で動作する Enterprise Bean の両方で使用できる機能が該当します。

(2) Web サービスを開発するための機能

Web サービスの実行環境および開発環境としての機能が該当します。

アプリケーションサーバでは，次のエンジンを提供しています。

- JAX-WS 仕様に従った SOAP メッセージのバインディングを実現する JAX-WS エンジン
- JAX-RS 仕様に従った RESTful HTTP メッセージのバインディングを実現する JAX-RS エンジン

(3) 信頼性や性能などを向上させるために拡張されたアプリケーションサーバ独自の機能（拡張機能）

アプリケーションサーバで独自に拡張された機能が該当します。バッチサーバ，CTM，データベースなど，J2EE サーバ以外のプロセスを使用して実現する機能も含まれます。

アプリケーションサーバでは，システムの信頼性を高め，安定稼働を実現するための多様な機能が拡張されています。また，J2EE アプリケーション以外のアプリケーション（バッチアプリケーション）を Java の環境で動作させる機能も拡張しています。

(4) システムのセキュリティを確保するための機能（セキュリティ管理機能）

アプリケーションサーバを中心としたシステムのセキュリティを確保するための機能が該当します。不正なユーザからのアクセスを防止するための認証機能や，通信路での情報漏えいを防止するための暗号化機能などがあります。

1.1.2 アプリケーションの実行基盤を運用・保守するための機能

アプリケーションの実行基盤を効率良く運用したり，保守したりするための機能です。システムの運用開始後に，必要に応じて使用します。ただし，機能によっては，あらかじめ設定やアプリケーションの実装が必要なものがあります。

アプリケーションの実行基盤を運用・保守するための機能について，分類ごとに説明します。

(1) システムの起動・停止など日常的な運用をするための機能（運用機能）

システムの起動や停止，アプリケーションの開始や停止，入れ替えなどの，日常運用で使用する機能が該当します。

(2) システムの利用状況を監視するための機能（監視機能）

システムの稼働状態や，リソースの枯渇状態などを監視するための機能が該当します。また，システムの操作履歴など，監査で使用する情報を出力する機能も該当します。

(3) ほかの製品と連携して運用するための機能（連携機能）

JP1 やクラスタソフトウェアなど，ほかの製品と連携して実現する機能が該当します。

(4) トラブル発生時などに対処するための機能（保守機能）

トラブルシューティングのための機能が該当します。トラブルシューティング時に参照する情報を出力するための機能も含みます。

(5) 旧バージョンの製品から移行するための機能（移行機能）

旧バージョンのアプリケーションサーバから新しいバージョンのアプリケーションサーバに移行するための機能が該当します。

(6) 旧バージョンの製品との互換のための機能（互換機能）

旧バージョンのアプリケーションサーバとの互換用の機能が該当します。なお，互換機能については，対応する推奨機能に移行することをお勧めします。

1.1.3 機能とマニュアルの対応

アプリケーションサーバの機能解説のマニュアルは，機能の分類に合わせて分冊されています。

機能の分類と，それぞれの機能について説明しているマニュアルとの対応を次の表に示します。

表 1-1 機能の分類と機能解説のマニュアルの対応

分類	機能	マニュアル※1
基本・開発機能	Web コンテナ	基本・開発編(Web コンテナ)
	JSF および JSTL の利用	
	JAX-RS 2.1 の利用	
	WebSocket	
	NIO HTTP サーバ	
	サーブレットおよび JSP の実装	
	セッションマネージャの指定機能	
	EJB コンテナ	基本・開発編(EJB コンテナ)
	EJB クライアント	
	Enterprise Bean 実装時の注意事項	
	ネーミング管理	基本・開発編(コンテナ共通機能)
	リソース接続とトランザクション管理	
	OpenTP1 からのアプリケーションサーバの呼び出し（TP1 インバウンド連携機能）	

分類	機能	マニュアル※1
	JPA 2.2 の利用	
	CJMS プロバイダ	
	JavaMail の利用	
	アプリケーションサーバでの CDI の利用	
	アプリケーションサーバでの Bean Validation の利用	
	Java Batch	
	JSON-P	
	JSON-B	
	Concurrency Utilities	
	アプリケーションの属性管理	
	アノテーションの使用	
	J2EE アプリケーションの形式とデプロイ	
	コンテナ拡張ライブラリ	
	ライブラリ競合回避機能	
	パッケージ名変換機能	
拡張機能	バッチサーバによるアプリケーションの実行	拡張編
	CTM によるリクエストのスケジューリングと負荷分散	
	バッチアプリケーションのスケジューリング	
	J2EE サーバ間のセッション情報の引き継ぎ（セッションフェイルオーバ機能）	
	データベースセッションフェイルオーバ機能	
	明示管理ヒープ機能を使用した FullGC の抑止	
	アプリケーションのユーザログ出力	
セキュリティ管理機能	統合ユーザ管理による認証	セキュリティ管理機能編
	アプリケーションの設定による認証	
	SSL/TLS 通信での TLSv1.2 の使用	
	API による直接接続を使用する負荷分散機の運用管理機能からの制御	
運用機能	システムの起動と停止	運用／監視／連携編
	J2EE アプリケーションの運用	
監視機能	稼働情報の監視（稼働情報収集機能）	
	リソースの枯渇監視	

分類	機能	マニュアル※1
	監査ログ出力機能	
	データベース監査証跡連携機能	
	運用管理コマンドによる稼働情報の出力	
	Management イベントの通知と Management アクションによる処理の自動実行	
	CTM の稼働統計情報の収集	
	コンソールログの出力	
連携機能	JP1 と連携したシステムの運用	
	システムの集中監視（JP1/IM との連携）	
	ジョブによるシステムの自動運転（JP1/AJS との連携）	
	監査ログの収集および一元管理（JP1/Audit Management - Manager との連携）	
	クラスタソフトウェアとの連携	
	1:1 系切り替えシステム（クラスタソフトウェアとの連携）	
	相互系切り替えシステム（クラスタソフトウェアとの連携）	
	N:1 リカバリシステム（クラスタソフトウェアとの連携）	
	ホスト単位管理モデルを対象にした系切り替えシステム（クラスタソフトウェアとの連携）	
保守機能	トラブルシューティング関連機能	保守／移行編※2
	性能解析トレースを使用した性能解析	
	製品の JavaVM（以降，JavaVM と略す場合があります）の機能	
移行機能	旧バージョンのアプリケーションサーバからの移行	
	推奨機能への移行	
互換機能	基本・開発機能の互換機能	互換編
	拡張機能の互換機能	

注※1　マニュアル名称の「アプリケーションサーバ 機能解説」を省略しています。

注※2　このマニュアルです。

1.2　システムの目的と機能の対応

アプリケーションサーバでは，構築・運用するシステムの目的に合わせて，適用する機能を選択する必要があります。

この節では，このマニュアルで説明している次の機能をどのような場合に使用するとよいかを説明します。なお，これらの機能は，アプリケーションサーバが独自に拡張した機能です。

- システムの保守のための機能
- JavaVM の機能
- 旧バージョンの製品から移行するための機能

機能ごとに，次の項目への対応を示しています。

- **性能**
  性能を重視したシステムの場合に使用するとよい機能です。
  システムのパフォーマンスチューニングで使用する機能などが該当します。
- **運用・保守**
  効率の良い運用・保守をしたい場合に使用するとよい機能です。
- **そのほか**
  そのほかの個別の目的に対応するための機能です。

1.2.1　システムの保守のための機能

システムの保守のための機能を次の表に示します。システムの目的に合った機能を選択してください。機能の詳細については，参照先を確認してください。

表 1-2　システムの保守のための機能

機能	システムの目的			参照先
	性能	運用・保守	そのほか	
トラブルシューティング	－	○	－	2 章，3 章，4 章，5 章，6 章
性能解析トレースを使用した性能解析	○	○	－	7 章，8 章

（凡例）　○：対応する　　－：対応しない

1.2.2　製品の JavaVM の機能

製品の JavaVM の機能を次の表に示します。システムの目的に合った機能を選択してください。機能の詳細については，参照先を確認してください。

表 1-3　製品の JavaVM の機能

機能	システムの目的			参照先
	性能	運用・保守	そのほか	
クラス別統計機能	○	○	−	9.3
インスタンス統計機能	○	○	−	9.4
STATIC メンバ統計機能	○	○	−	9.5
参照関係情報出力機能	○	○	−	9.6
統計前の GC 選択機能	○	○	−	9.7
Tenured 領域内不要オブジェクト統計機能	○	○	−	9.8
Tenured 増加要因の基点オブジェクトリスト出力機能	○	○	−	9.9
クラス別統計情報解析機能	○	○	−	9.10
Survivor 領域の年齢分布情報出力機能	○	○	−	9.11
hndlwrap 機能	−	○	−	9.12
JIT コンパイル時の C ヒープ確保量の上限値設定機能	−	○	−	9.13
スレッド数の上限値設定機能	−	○	−	9.14

（凡例）　○：対応する　　−：対応しない

1.2.3　旧バージョンの製品から移行するための機能

旧バージョンの製品から移行するための機能を次の表に示します。システムの目的に合った機能を選択してください。機能の詳細については，参照先を確認してください。

表 1-4　旧バージョンの製品から移行するための機能

機能	システムの目的			参照先
	性能	運用・保守	そのほか	
旧バージョンのアプリケーションサーバからの移行	−	−	○	10 章

（凡例）　○：対応する　　−：対応しない

1.3　このマニュアルに記載している機能の説明

ここでは，このマニュアルで機能を説明するときの分類の意味と，分類を示す表の例について説明します。

1.3.1　分類の意味

このマニュアルでは，各機能について，次の五つに分類して説明しています。マニュアルを参照する目的によって，必要な個所を選択して読むことができます。

- 解説
  機能の解説です。機能の目的，特長，仕組みなどについて説明しています。機能の概要について知りたい場合にお読みください。
- 実装
  コーディングの方法や DD の記載方法などについて説明しています。アプリケーションを開発する場合にお読みください。
- 設定
  システム構築時に必要となるプロパティなどの設定方法について説明しています。システムを構築する場合にお読みください。
- 運用
  運用方法の説明です。運用時の手順や使用するコマンドの実行例などについて説明しています。システムを運用する場合にお読みください。
- 注意事項
  機能を使用するときの全般的な注意事項について説明しています。注意事項の説明は必ずお読みください。

1.3.2　分類を示す表の例

機能説明の分類については，表で説明しています。表のタイトルは，「この章の構成」または「この節の構成」となっています。

次に，機能説明の分類を示す表の例を示します。

機能説明の分類を示す表の例

表 X-1　この章の構成（○○機能）

分類	タイトル	参照先
解説	○○機能とは	X.1
実装	アプリケーションの実装	X.2

分類	タイトル	参照先
	DD および cosminexus.xml※での定義	X.3
設定	実行環境での設定	X.4
運用	○○機能を使用した運用	X.5
注意事項	○○機能使用時の注意事項	X.6

注※

cosminexus.xml については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「16. アプリケーションの属性管理」を参照してください。

ポイント

cosminexus.xml を含まないアプリケーションのプロパティ設定

cosminexus.xml を含まないアプリケーションでは，実行環境へのインポート後にプロパティを設定，または変更します。設定済みのプロパティも実行環境で変更できます。

実行環境でのアプリケーションの設定は，サーバ管理コマンドおよび属性ファイルで実施します。サーバ管理コマンドおよび属性ファイルでのアプリケーションの設定については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「3.5.2　J2EE アプリケーションのプロパティの設定手順」を参照してください。

属性ファイルで指定するタグは，DD または cosminexus.xml と対応しています。DD または cosminexus.xml と属性ファイルのタグの対応については，マニュアル「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)」の「3.　J2EE アプリケーションの設定で使用する属性ファイル」を参照してください。

なお，各属性ファイルで設定するプロパティは，アプリケーション統合属性ファイルでも設定できます。

1.4　アプリケーションサーバ 11-40 での主な機能変更

この節では，アプリケーションサーバ 11-40 での主な機能の変更について，変更目的ごとに説明します。

説明内容は次のとおりです。

- アプリケーションサーバ 11-40 で変更になった主な機能と，その概要を説明しています。機能の詳細については参照先の記述を確認してください。「参照先マニュアル」および「参照個所」には，その機能についての主な記載個所を記載しています。
- 「参照先マニュアル」に示したマニュアル名の「アプリケーションサーバ」は省略しています。

1.4.1　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 1-5　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Java SE 17 への対応	Java SE 17 の機能が使用できるようになりました。	システム設計ガイド	2.1
		このマニュアル	9 章
メモリ管理方式に ZGC を追加	メモリ管理方式に ZGC を追加しました。	システム設計ガイド	7.17
接続できるデータベースに Azure SQL Database を追加	DB Connector を使用して接続できるデータベースに Azure SQL Database を追加しました。	機能解説 基本・開発編 (コンテナ共通機能)	3.6.17

2 トラブルシューティング

アプリケーションサーバでは，システムの運用中に発生したトラブルの対処で利用できる，さまざまな資料を出力する機能を提供しています。この章では，トラブルが発生した場合に使用する資料の取得方法，対処手順，トラブルシューティングに関する留意事項などについて説明します。

2.1　この章の構成

トラブルシューティングに関する説明のうち，トラブルシューティングの資料取得と対処の概要について説明します。

運用中のシステムでトラブルが発生した場合，発生したトラブルの種別に応じて資料を収集する必要があります。アプリケーションサーバでは，システムの保守に利用できる資料として，ログを出力する機能を提供しています。

この章の構成を次の表に示します。

表 2-1　この章の構成（トラブルシューティング（資料取得と対処の概要））

分類	タイトル	参照先
解説	トラブルシューティングの概要	2.2
	資料の取得	2.3
	取得が必要な資料の種類	2.4
運用	トラブルへの対処と回復	2.5
注意事項	トラブルシューティングに関連する留意事項	2.6

なお，トラブルシューティングの資料の取得や出力に関する設定，資料を個別に出力する方法，資料の出力先や出力内容，およびトラブルシューティングの手順については，それぞれ次の個所を参照してください。

- 資料の取得や出力に関する設定
  「3.　トラブルシューティングのための準備」
- 資料のデフォルトでの出力先と，資料を個別に出力する方法
  「4.　トラブルシューティングで必要な資料の出力先と出力方法」
- 資料に出力される内容
  「5.　トラブルの分析」
- トラブルシューティングの手順
  「6.　トラブルシューティングの手順」

2.2　トラブルシューティングの概要

トラブル発生時，構成ソフトウェアでは，トラブルシュート情報としてその時点の状態をログに出力します。このログを収集・分析して，トラブルの発生要因を調査します。また，性能解析トレースを使用して，リクエストの処理状況からトラブルの発生個所を特定することもできます。

アプリケーションサーバで構築したシステムでは，構成ソフトウェアのトラブル発生時のログと定義ファイルを一括して収集できます。この情報を **snapshot ログ**といいます。運用管理ドメイン内の論理サーバが停止される直前，または J2EE サーバを手動で一括再起動する直前に，snapshot ログの一括収集が自動で実行されます。

また，CTM を利用すると，J2EE サーバが異常終了した場合でも，すぐにクライアントにエラーが返却されないように設定できます。J2EE サーバが再起動されるまでの間，CTM によって J2EE アプリケーションのスケジュールキューが閉塞制御され，すでに登録されているリクエストを保持して，さらにクライアントからのリクエストを受け付け続けます。これによって，すぐに再起動すれば，クライアントにトラブルを気づかせないで運用を続けられます。なお，CTM は，構成ソフトウェアに Component Transaction Monitor を含む製品だけで利用できます。利用できる製品については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」の「2.2.2　構成ソフトウェアの機能概要」を参照してください。

この節では，トラブル発生時の対処手順と，資料取得の流れについて説明します。

2.2.1　トラブルへの対処の手順

この節では，アプリケーションサーバのシステムの運用中にトラブルが発生した場合の，対処の手順について説明します。

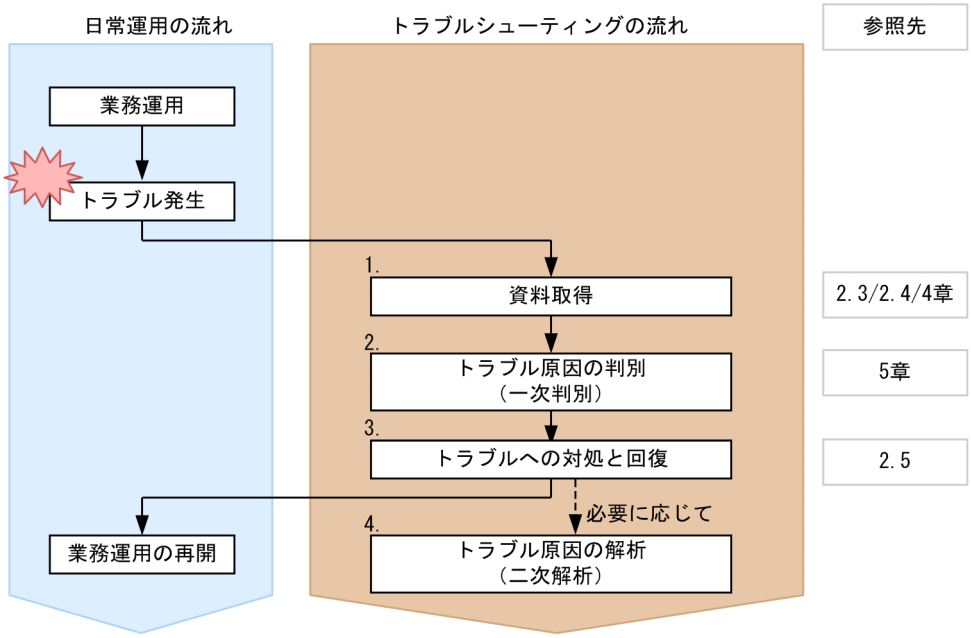
なお，トラブル発生時に取得する情報のうち，次に示す情報を取得するためには，運用開始前の準備が必要です。

- JavaVM ログ（JavaVM ログファイル）
  このファイルには，JavaVM の GC のログも出力されます。
- ユーザダンプ（Windows の場合）
- core ダンプ（UNIX の場合）
- OS の統計情報（Windows の場合）

これらの情報については，必要に応じてシステム構築時に取得準備をしてください。詳細は，「3.2　資料取得の設定の概要」および「3.3　実行環境での設定」を参照してください。

システムの運用中にトラブルが発生した場合は，次の図に示す手順で対処します。トラブル発生時の対処の流れを次の図に示します。

図 2-1　トラブル発生時の対処の流れ



1. 資料取得

トラブルシューティングに必要な資料をすべて取得します。

取得が必要な資料については，「2.4　取得が必要な資料の種類」を参照してください。なお，メモリダンプは，J2EE サーバまたは CORBA ネーミングサービスを再起動する場合にだけ必要な資料です。

また，資料の取得方法については，「2.3　資料の取得」，または「4.　トラブルシューティングで必要な資料の出力先と出力方法」を参照してください。

参考

- 資料の取得には，snapshot ログとして一括して取得する方法と，個別のログを取得する方法があります。
- snapshot ログの取得には，運用管理コマンドを使用して取得する方法，および設定によって自動で取得する方法があります。
- 論理サーバに異常が発生した場合の資料取得を自動化できます。詳細は，「2.3.1　トラブル発生時に自動的に取得できる資料」を参照してください。

2. トラブル原因の判別（一次判別）

トラブルの原因を判別します。判別には，取得した資料を使用します。資料の調査方法については，「5. トラブルの分析」を参照してください。

3. トラブルへの対処と回復

トラブルに対処して，回復します。

トラブルの種類ごとの対処方法については，「2.5　トラブルへの対処と回復」を参照してください。

4. トラブル原因の解析（二次解析）

必要に応じてトラブル原因をさらに分析します。なお，この解析作業は保守員が実施します。

ポイント

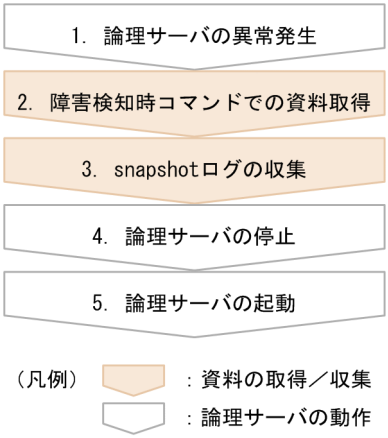
トラブルが発生したタイミングごとのトラブルシューティングについては，「6.　トラブルシューティングの手順」を参照してください。

2.2.2　トラブル発生時の資料取得の流れ

アプリケーションサーバで構築したシステムでは，自動でトラブルシューティングの資料を取得できます。論理サーバを起動すると，運用管理エージェントは論理サーバの監視を開始します。論理サーバに異常が発生すると，運用管理エージェントは異常を検知し，Management Server に通知します。Management Server は資料を取得および収集し，論理サーバを停止して再起動します。

自動で資料を取得する場合の流れを次の図に示します。

図 2-2　自動で資料を取得する場合の流れ



2.の障害検知時コマンドによって，トラブルシューティングに必要な情報が出力されます。このコマンドで出力した情報，およびそれ以外のトラブルシューティングに必要な情報を，3.で snapshot ログとしてまとめて収集します。なお，障害検知時コマンドを使用しないで，4.の論理サーバの停止後に snapshot ログを取得することもできますが，このとき取得できる情報は J2EE サーバだけとなります。このため，障害検知時コマンドを使用して，3.で snapshot ログを収集することをお勧めします。障害検知時コマンドを使用した資料の取得については，「2.3.2　障害検知時コマンドによる資料取得」を，snapshot ログについては，「2.3.3　snapshot ログの収集」を参照してください。

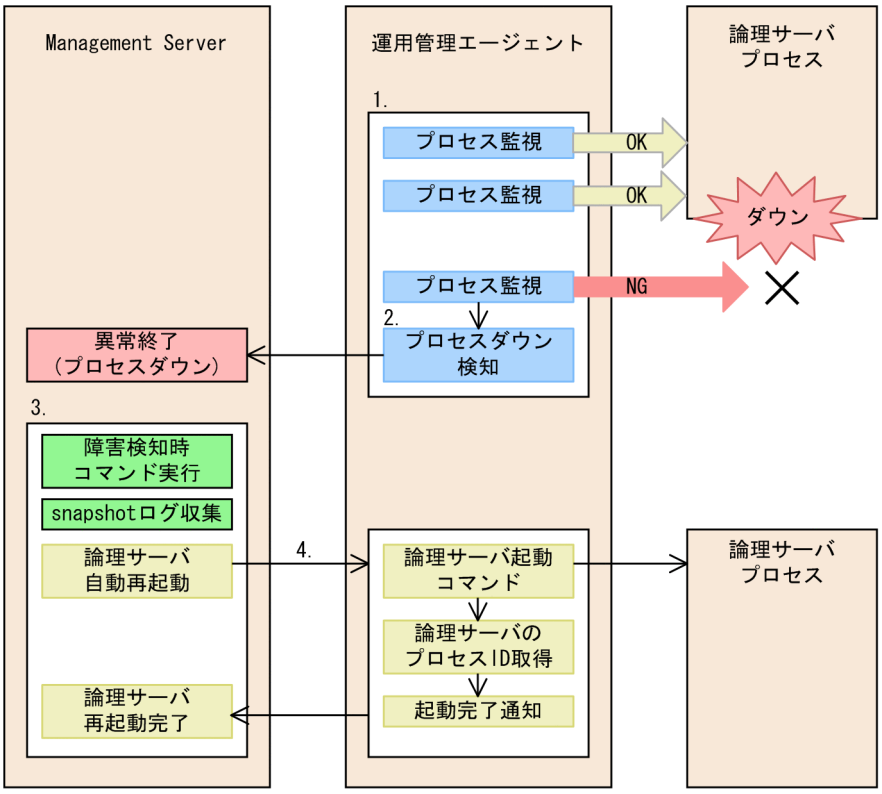
また，Management Server の運用管理コマンド（mngsvrutil）を使用して，snapshot ログを任意のタイミングで収集することもできます。運用管理コマンドを使用した snapshot ログの収集については，「2.3.3(4)　運用管理コマンドを使用した snapshot ログの収集」を参照してください。

論理サーバのプロセスがダウンしたとき，および論理サーバのプロセスがハングアップしたときの処理の流れについて説明します。

(1)　論理サーバのプロセスがダウンしたときの処理の流れ

論理サーバの起動後，運用管理エージェントのプロセス監視では，論理サーバプロセスのプロセス ID を使用して定期的にプロセスを監視します。論理サーバプロセスがダウンしたときの処理の流れを次の図に示します。

図 2-3　論理サーバのプロセスがダウンしたときの処理の流れ



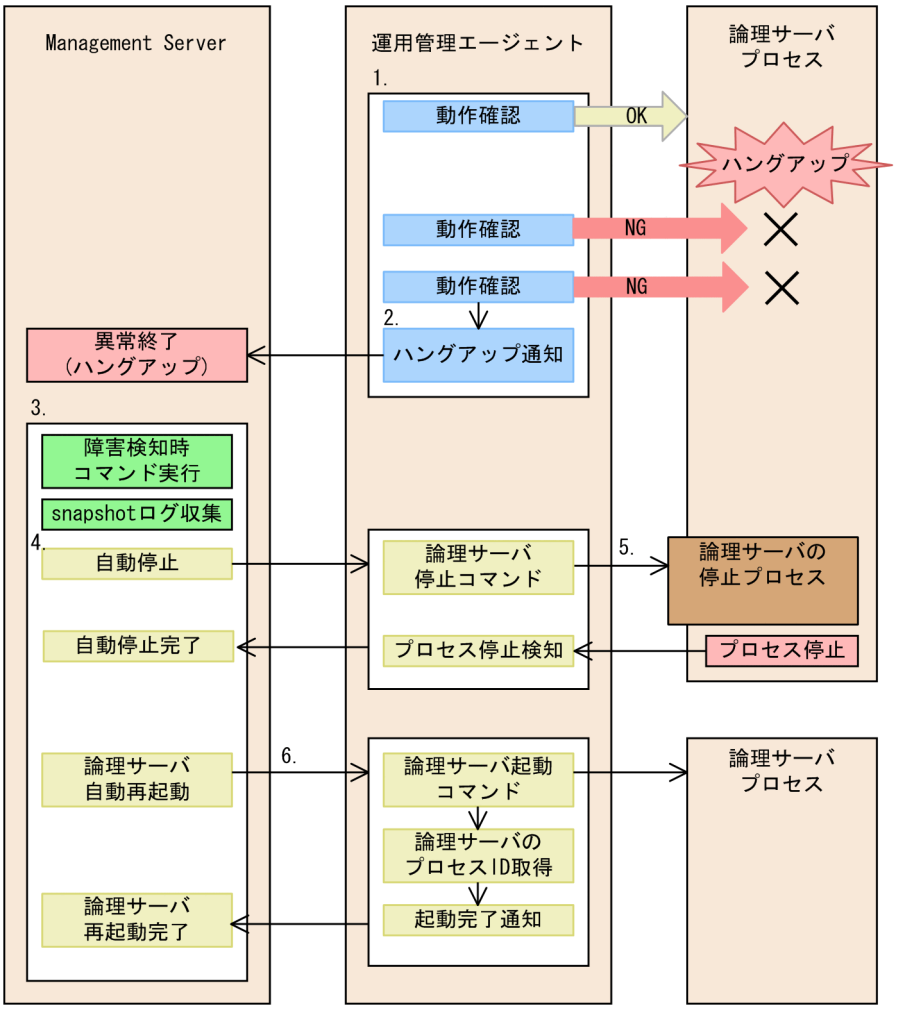
1. 論理サーバプロセスのプロセス ID で，定期的にプロセスを監視します。
2. 論理サーバプロセスが異常終了すると，運用管理エージェントはプロセスダウンを検知して，Management Server に通知します。

   プロセス監視では，プロセス ID の存在確認をします。確認内容は，論理サーバの種類によって異なります。詳細については，マニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「2.3.1　論理サーバの起動と稼働確認」，およびマニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「2.3.2　論理サーバの停止」を参照してください。
3. Management Server ではプロセスダウンを検知すると，障害検知時コマンドを実行し，snapshot ログを収集します。
4. 障害検知時コマンドと snapshot ログ収集後，論理サーバを自動再起動します。

(2)　論理サーバのプロセスがハングアップしたときの処理の流れ

論理サーバの起動後，運用管理エージェントのプロセス監視では，論理サーバプロセスに対して定期的に，論理サーバが動作しているかを確認します。動作確認中に論理サーバプロセスがハングアップしたときの処理の流れを次の図に示します。

図 2-4　論理サーバのプロセスがハングアップしたときの処理の流れ



1. 定期的に論理サーバプロセスの動作確認をします。

   動作確認は，プロセス監視でプロセス ID が存在することが確認されてから，実施されます。確認内容は，論理サーバの種類によって異なります。詳細については，マニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「2.3.1　論理サーバの起動と稼働確認」，およびマニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「2.3.2　論理サーバの停止」を参照してください。

2. 連続して動作確認に 2 回失敗すると（デフォルト値），運用管理エージェントはハングアップと判断して，Management Server に通知します。なお，ハングアップと判断する，動作確認の失敗回数は変更できます。

3. Management Server ではハングアップを検知すると，障害検知時コマンドを実行し，snapshot ログを収集します。

4. ハングアップの場合，プロセスは稼働中であるため，自動停止処理を実行します。

5. 運用管理エージェントでは論理サーバの停止コマンドを実行します。

   なお，一定時間経っても停止しない場合は，強制停止コマンドが実行されます。

6. 障害検知時コマンドと snapshot ログ収集後，論理サーバを自動再起動します。

2.3　資料の取得

トラブルシューティングに必要な資料は，次のどれかの方法で取得できます。

- トラブル発生時に自動的に取得する
- 運用管理コマンドを利用して一括取得する
- それぞれの情報を個別に収集する

トラブルシューティングに必要な情報は，snapshot ログとしてまとめて収集できます。**snapshot ログ**とは，システムの構成ソフトウェアのトラブル発生時に，そのときの状態を出力したログのことです。アプリケーションサーバで構築したシステムでは，各種構成ソフトウェアの snapshot ログを一括して収集して，ZIP 形式のログファイルとして出力できます。snapshot ログは，一次送付資料，二次送付資料に分けて取得できます。また，snapshotlog コマンドを使用して定義送付資料を収集することもできます。

この節では，トラブル発生時に必要な資料の取得方法として，資料を取得するときに実行する障害検知時コマンドと，資料を一括収集する snapshot ログについて説明します。

snapshot ログとして取得しない情報については，「2.4.3　取得方法および調査方法との対応」を，それぞれの情報を個別に収集する方法ついては，「4.　トラブルシューティングで必要な資料の出力先と出力方法」を参照してください。

なお，資料によっては，事前に取得のための設定をしておく必要があります。例えば，OS の統計情報，ユーザダンプなどは，システム構築時に取得のための設定をしておかないと，取得できません。これらの資料はトラブルシューティングで必要となるため，取得することをお勧めします。

デフォルトの設定で取得できる資料についても，ログファイルの出力先，ログファイルの面数や 1 ファイル当たりの最大サイズなどを変更できます。必要に応じて，資料取得の設定を変更してください。

また，デフォルトの設定では snapshot ログとして収集できない資料でも，その資料の取得先を snapshot ログの収集対象として定義することで，snapshot ログとして一括収集できるようになります。

注意事項

Windows の場合に資料取得の設定をするときの注意事項

Windows では，障害発生時の調査資料を取得するために，レジストリの設定が必要です。レジストリの設定はシステム全体に対して影響を与えるため，設定するときは十分に注意してください。

トラブルシューティングに必要な資料取得の設定については，「3.　トラブルシューティングのための準備」を参照してください。

2.3.1　トラブル発生時に自動的に取得できる資料

アプリケーションサーバで構築したシステムでは，論理サーバに異常が発生した場合に，自動的にトラブルシューティングに必要な情報を取得，収集できます。

ポイント

トラブル発生時に資料を自動で取得するためには，システムを構築する段階で必要な設定をしておく必要があります。トラブルシューティングの資料取得の設定については，「3.2　資料取得の設定の概要」，および「3.3　実行環境での設定」を参照してください。

障害検知時コマンドの実行と snapshot ログの収集によって，トラブルシューティングに必要な情報は，次の情報を除いてすべて自動収集するように設定できます。

- OS の統計情報（Windows の場合）

デフォルトの設定で収集されない情報については，次のように設定して収集します。

1. ユーザ作成の障害検知時コマンド内で，必要な情報を出力します。
2. 1.の情報の出力先を，snapshot ログの収集対象のパスに追加します。
   snapshot ログ取得の設定については，「3.3.3　snapshot ログ収集の設定（J2EE アプリケーションを実行するシステム）」，または「3.3.4　snapshot ログ収集の設定（バッチアプリケーションを実行するシステム）」を参照してください。

注意事項

snapshot ログで取得できる構成ソフトウェアのログのうち，次の表に示すログの取得については，このマニュアルでは説明していません。取得方法については，次の表の参照先マニュアルを参照してください。

表 2-2　このマニュアルでログの取得方法を記載していない構成ソフトウェア

ログの種類	参照先マニュアル	参照個所
Component Container の SOAP アプリケーション実行基盤のログ	アプリケーションサーバ SOAP アプリケーション開発の手引	14 章
Web Services - Security のログ	アプリケーションサーバ Web サービスセキュリティ構築ガイド	6 章
TPBroker のログ	TPBroker ユーザーズガイド	

2.3.2　障害検知時コマンドによる資料取得

障害検知時コマンドは，Management Server が論理サーバの障害を検知したときに，システムによって実行されるコマンドです。障害検知時コマンドは，システム構築時に mserver.properties（Management Server 環境設定ファイル）のキーに値を指定することで実行されます。

> **注意事項**
>
> 運用管理エージェントが停止している状態で，論理サーバのプロセスが異常終了した場合，そのあとに運用管理エージェントを起動しても障害検知時コマンドは実行されません。この場合，運用管理エージェントの起動時に，詳細情報が「The command is not defined.」の KEOS20071-E メッセージが出力されます。メッセージについては，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」を参照してください。

障害検知時コマンドを利用して，トラブル発生時のスレッドダンプやユーザダンプの取得などの取得処理を実行することで，トラブル発生時のタイムリーな資料を取得できるようになります。また，障害検知時コマンドを利用すると，次のどちらかのタイミングで snapshot ログを自動収集することもできます。収集するタイミングは設定によって決められます。

- 論理サーバにトラブルが発生した場合，障害検知時コマンドが実行されて，論理サーバの停止前に snapshot ログが収集されます。
- J2EE サーバまたはバッチサーバにトラブルが発生した場合，障害検知時コマンドが実行されて，J2EE サーバまたはバッチサーバの再起動前に snapshot ログが収集されます。

障害検知時コマンドには，システム提供の障害検知時コマンドとユーザ作成の障害検知時コマンドの 2 種類があります。

システム提供の障害検知時コマンド

あらかじめアプリケーションサーバで定義されているコマンドです。システム提供の障害検知時コマンドでは，論理サーバにトラブルが発生した場合に，トラブルが発生した論理サーバの JavaVM のスレッドダンプや性能解析トレースなどを取得します。システム提供の障害検知時コマンドで取得した資料は，snapshot ログとして収集できます。

デフォルトの設定では，論理サーバにトラブルが発生した場合にシステム提供の障害検知時コマンドが実行されて，トラブルが発生した論理サーバの停止前に snapshot ログが収集されます。

ユーザ作成の障害検知時コマンド

障害検知時コマンドはユーザが作成することもできます。資料取得に必要な任意の処理を記述したバッチファイルやシェルスクリプトを，障害検知時コマンドとして実行できます。

なお，ユーザ作成の障害検知時コマンドで取得した資料を snapshot ログとして収集するためには，その資料の取得先を snapshot ログの収集先として定義しておく必要があります。

システム提供の障害検知時コマンドの動作設定を変更する場合，またはユーザ作成の障害検知時コマンドを利用する場合に必要な設定については，「3.3.1　障害検知時コマンドによる資料取得の設定（J2EE アプ

リケーションを実行するシステム)」または「3.3.2　障害検知時コマンドによる資料取得の設定（バッチアプリケーションを実行するシステム)」を参照してください。

ポイント

障害検知時コマンドは，システム構築時に必要な設定がされていない場合，実行されません。次の設定内容を確認してください。

- mserver.properties（Management Server 環境設定ファイル）の com.cosminexus.mngsvr.sys_cmd.abnormal_end.enabled キーに「true」が設定されている場合，システムで提供されている障害検知時コマンドが実行されます。
- mserver.properties（Management Server 環境設定ファイル）の com.cosminexus.mngsvr.usr_cmd.abnormal_end.enabled キーに「true」が設定されている場合，ユーザが作成した障害検知時コマンドが実行されます。

ここでは，それぞれで取得できる情報について説明します。

(1)　システムで提供されている障害検知時コマンドで取得できる情報

システムで提供されている障害検知時コマンドで取得できる情報について，次の表に示します。障害検知時コマンドで取得できる情報は，障害の種類（システムダウン，またはハングアップ）および使用している OS によって異なります。

なお，snapshot ログでこれらの情報を取得するためには，adminagent.properties（運用管理エージェントプロパティファイル）に必要な設定がされていることが必要です。

障害検知時コマンドで取得できる情報を次に示します。

表 2-3　障害検知時コマンドで取得できる情報

論理サーバ	OS	取得できる情報		J2EE アプリケーション	バッチアプリケーション
		プロセスダウンを検出した場合	ハングアップを検出した場合		
論理パフォーマンストレーサ	Windows UNIX	• バッファの内容を出力した PRF トレースファイル	－	○	○
論理 J2EE サーバ※	Windows	• 性能解析トレースファイル	• スレッドダンプ • 性能解析トレースファイル	○	○
	UNIX	• JavaVM のスタックトレース情報 • 性能解析トレースファイル	• スレッドダンプ • 性能解析トレースファイル	○	○
論理 Web サーバ	Windows UNIX	• 性能解析トレースファイル	• 内部トレース • 性能解析トレースファイル	○	－

論理サーバ	OS	取得できる情報		J2EE アプリケーション	バッチアプリケーション
		プロセスダウンを検出した場合	ハングアップを検出した場合		
ほかの論理サーバ	Windows UNIX	• 性能解析トレースファイル	• 性能解析トレースファイル	○	−

（凡例）

J2EE アプリケーション：J2EE アプリケーションを実行するシステム

バッチアプリケーション：バッチアプリケーションを実行するシステム

−：該当しない

○：取得できる

注※　バッチサーバは論理 J2EE サーバとして定義します。

- **トラブルが発生した J2EE サーバまたはバッチサーバのスレッドダンプ**

  adminagent.properties（運用管理エージェントプロパティファイル）の adminagent.j2ee.sys_cmd.abnormal_end.threaddump キーに「true」が設定されている場合に取得できます。

- **性能解析トレース**

  adminagent.properties（運用管理エージェントプロパティファイル）の adminagent.sys_cmd.abnormal_end.prftrace キーに「true」が設定されている場合に取得できます。性能解析トレースファイルは<Application Server のインストールディレクトリ>/manager/tmp に出力されます。なお，出力先は adminagent.properties（運用管理エージェントプロパティファイル）の adminagent.prftrace_dir キーで変更できます。

注意事項

トラブルが発生した J2EE サーバのプロセスが存在しないと判断された場合，スレッドダンプは出力されません。また，パフォーマンストレーサのプロセスが存在しないと判断された場合，性能解析トレースは出力されません。

(2)　ユーザが作成した障害検知時コマンドで取得できる情報

ユーザが作成した障害検知時コマンドとして，資料取得に必要な任意の処理を記述したバッチファイルまたはシェルスクリプトを実行できます。例えば，ユーザダンプまたは core ダンプは，このコマンドの中で drwtsn32 コマンドなどを実行して取得できます。障害検知時コマンドの作成方法については，「3.3.1　障害検知時コマンドによる資料取得の設定（J2EE アプリケーションを実行するシステム）」または「3.3.2 障害検知時コマンドによる資料取得の設定（バッチアプリケーションを実行するシステム）」を参照してください。

なお，ユーザ作成の障害検知時コマンドは，adminagent.properties（運用管理エージェントプロパティファイル）の adminagent.<serverkind>.usr_cmd.abnormal_end キーに指定されているものが実行されます。

2.3.3　snapshot ログの収集

システムの構成ソフトウェアがトラブル発生時にそのときの状態を出力したログを，**snapshot ログ**といいます。snapshot ログには，各種構成ソフトウェアのログのほかに，スレッドダンプ，性能解析トレースなど，システム保守に必要な情報とアプリケーション保守に必要な情報が含まれます。アプリケーションサーバで構築したシステムでは，これらの情報を snapshot ログとして一括収集し，ZIP 形式のログファイルとして取得できます。運用管理者は，snapshot ログを収集・分析することで，トラブルに対処できます。

障害検知時コマンドの実行と snapshot ログの収集によって，トラブルシューティングに必要な資料は自動収集するように設定されています。デフォルトの設定で収集されない情報を snapshot ログとして収集したい場合には，その資料の取得先を snapshot ログの収集対象として定義してください。デフォルトの設定で収集されない情報については，「付録 A　snapshot ログの収集対象一覧」の snapshot ログの収集可否や収集に関する設定変更の説明を参照してください。

snapshot ログの収集対象，出力先ディレクトリや面数などは，システム構築時に設定を変更できます。

ここでは，snapshot ログの収集タイミング，収集できる資料，および収集の流れについて説明します。また，任意のタイミングで収集する方法として，運用管理コマンドを使用した snapshot ログの収集についても説明します。なお，snapshot ログ収集の設定については，「3.3.3　snapshot ログ収集の設定（J2EE アプリケーションを実行するシステム）」，または「3.3.4　snapshot ログ収集の設定（バッチアプリケーションを実行するシステム）」を参照してください。

(1)　snapshot ログの収集タイミング

snapshot ログは，自動または任意のタイミングで収集できます。snapshot ログの収集のタイミングを次の表に示します。なお，収集できるのは，論理サーバが稼働しているホスト内の snapshot ログです。

表 2-4　snapshot ログの収集のタイミング

分類	収集のタイミング
自動的に収集する※1	論理サーバが障害時に自動停止される直前※2
	運用管理ドメインで管理している J2EE サーバまたはバッチサーバが障害時に自動再起動される直前※2
	J2EE サーバまたはバッチサーバを手動で一括再起動する直前
任意のタイミングで収集する	Management Server の運用管理コマンド（mngsvrutil）で snapshot ログの収集を実行したとき（実行方法については，「2.3.3(4)　運用管理コマンドを使用した snapshot ログの収集」を参照してください）

注※1

次に示すどちらのタイミングで snapshot ログを収集するかは，システム構築時に設定を変更できます。デフォルトの設定では，論理サーバの停止前に snapshot ログが収集されます。

・論理サーバの停止前

・J2EE サーバの再起動前

注※2

このタイミングで snapshot ログを収集する場合には，Management Server によって障害検知時コマンドが実行されます。

参考

snapshot ログは，次のタイミングでも収集できます。

- 運用管理ポータルの「論理サーバの起動／停止」にある［snapshot ログ］画面で［収集］ボタンをクリックした時※1
- snapshotlog コマンド実行時※2

注※1

収集できるのは，収集を実行した J2EE サーバまたはバッチサーバ内の snapshot ログです。

注※2

Management Server を使用しているかどうかに関係なく，コマンドは実行できます。

［snapshot ログ］画面については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「11.9.5　J2EE サーバの snapshot ログの収集」を，snapshotlog コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「snapshotlog（snapshot ログの収集）」を参照してください。

(2)　snapshot ログとして収集できる資料

snapshot ログとして収集できる資料について説明します。snapshot ログには，システム保守に必要な情報とアプリケーション保守に必要な情報が含まれます。

トラブルシューティングに必要な資料は，資料を保守員へ送付するときのタイミングによって，一次送付資料と二次送付資料に分類されます。snapshot ログでは，一次送付資料および二次送付資料を収集できます。また，snapshotlog コマンドを使用して定義送付資料を収集できます。

- 一次送付資料

  メールの添付資料として送付できるファイルサイズの資料が該当します。メールなどで早急に保守員に送付できます。

- 二次送付資料

  一次送付資料に加えて，ファイルサイズが比較的大きい資料が該当します。ファイルサイズが比較的大きい資料は資料の送付に時間が掛かるため，別途，保守員に送付する必要があります。

- 定義送付資料

  定義ファイルの設定誤りを調査するため，定義ファイルだけを収集した資料です。ファイルサイズが小さく，メールなどで早急に保守員に送付できます。なお，この資料は snapshotlog コマンドを使用して，Management Server や J2EE サーバが起動していない状態でも収集できます。

注意事項

定義送付資料で収集される定義情報中にユーザ ID およびパスワードが含まれる場合があります。

snapshot ログとして収集する資料は，snapshot ログ収集対象定義ファイルの設定によって変更できます。例えば，ユーザ作成の障害検知時コマンドで出力した情報を収集したい場合には，設定を変更すれば収集対象に加えられます。snapshot ログ収集対象定義ファイルはシステム構築時に設定します。

実際の運用環境での収集対象は，次のファイルで確認できます。

- snapshotlog.conf
  一次送付資料として取得する内容が定義されています。
  snapshotlog.conf の格納場所を次に示します。
  - Windows の場合
    <Application Server のインストールディレクトリ>¥manager¥config¥snapshotlog.conf
  - UNIX の場合
    /opt/Cosminexus/manager/config/snapshotlog.conf
- snapshotlog.2.conf
  二次送付資料として取得する内容が定義されています。
  snapshotlog.2.conf の格納場所を次に示します。
  - Windows の場合
    <Application Server のインストールディレクトリ>¥manager¥config¥snapshotlog.2.conf
  - UNIX の場合
    /opt/Cosminexus/manager/config/snapshotlog.2.conf
- snapshotlog.param.conf
  定義送付資料として取得する内容が定義されています。
  snapshotlog.param.conf の格納場所を次に示します。
  - Windows の場合
    <Application Server のインストールディレクトリ>¥manager¥config¥snapshotlog.param.conf
  - UNIX の場合
    /opt/Cosminexus/manager/config/snapshotlog.param.conf

ただし，adminagent.properties（運用管理エージェントプロパティファイル）の指定内容によっては，収集できる情報が異なる場合があります。

デフォルトの設定の場合，トラブル発生時に自動的に取得される snapshot ログには，次の情報が含まれます。

一次送付資料

- snapshot ログの一次送付資料の収集対象として定義されているディレクトリ下の情報※1
- J2EE サーバのスレッドダンプ※2
- インストール情報
- OS の状態ログ

二次送付資料

- 性能解析トレース※2
- snapshot ログの二次送付資料の収集対象として定義されているディレクトリ下の情報※1
- インストール情報
- OS の状態ログ

注※1　snapshot ログの収集対象については，「付録 A　snapshot ログの収集対象一覧」を参照してください。

注※2　システムで提供されている障害検知時コマンドで，スレッドダンプおよび性能解析トレースを出力する設定にしている場合に収集できます。

参考

snapshotlog コマンドで収集できる snapshot ログと，Management Server の運用管理コマンド（mngsvrutil）などで収集できる snapshot ログでは，デフォルトの状態で収集できるファイルが異なります。Management Server の運用管理コマンド（mngsvrutil）などで収集できる snapshot ログは，デフォルトの状態でインストール情報や OS の状態・ログなどを収集できますが，snapshotlog コマンドでインストール情報や OS の状態・ログなどを収集するためには，コマンド実行時に指定する snapshot ログ収集対象定義ファイルを編集して，snapshot ログの収集先を追加する必要があります。

デフォルトの状態で収集できる情報と収集できない情報の例を次に示します。

(a)　デフォルトの状態で収集できる情報

デフォルトの状態で収集できる情報の例を次に示します。

システム保守に必要な情報

システム保守に必要な情報として，次に示す構成ソフトウェアの定義ファイルおよびログファイルが収集されます。

- Component Container※
- Component Transaction Monitor
- Developer's Kit for Java
- Performance Tracer

- TPBroker
- Web Services - Security
- HTTP Server

このほか，トレース共通ライブラリのログ，およびプログラムプロダクト情報（UNIX の場合）も収集します。

注※　SOAP アプリケーション実行基盤の情報も含みます。

アプリケーション保守に必要な情報

アプリケーション保守に必要な情報として，Component Container のメッセージログおよびユーザログを収集します。

次に示す情報についても，デフォルトの状態で収集されます。

Component Container 関連の情報

Windows の場合

<Application Server のインストールディレクトリ>¥CC¥server¥public 下すべて

UNIX の場合

/opt/Cosminexus/CC/server/public 下すべて

なお，デフォルトの状態では，構成ソフトウェアのインストール時にデフォルトで作成されたディレクトリが snapshot ログの収集対象として定義されています。ログの出力先を変更している場合は収集先も変更するようにしてください。

(b)　デフォルトの状態では収集できない情報

次に示す情報は，デフォルトの状態で運用している場合は収集されません。必要に応じて，取得するように設定してください。

Windows の場合

Component Container 関連

- EAR/JAR ファイル（デプロイ/インポートできなかった場合）

Microsoft IIS 関連

Microsoft IIS と連携している場合，次の資料を取得します。

- C:¥inetpub¥logs（C:の部分にはシステムドライブを指定します）

OS 関連

- システムモニタ関連の資料一式（4.13 参照）
- イベントログ（アプリケーション，システム）
- OS の稼働資料

  winmsd 起動画面から［操作］－［テキストファイルとして保存］メニューで資料採取します（5 分から 10 分掛かる場合があります）。

UNIX の場合

Component Container 関連

- EAR/JAR ファイル（デプロイ/インポートできなかった場合）

OS 関連

- OS の稼働資料

(3) snapshot ログ収集の流れ

論理サーバに異常が発生すると，Management Server から運用管理エージェントに対して snapshot ログ収集のサービス要求が実行されます。snapshot ログ収集時に実行される処理の流れを次に示します。なお，設定によって，実行される処理は異なります。

1. システム提供の障害検知時コマンドの実行
2. ユーザ作成の障害検知時コマンドの実行
3. snapshot ログ（一次送付資料）の収集
4. snapshot ログ（二次送付資料）の収集

マシンの動作環境や，障害の発生状況（想定外の障害が発生した場合など）によっては，snapshot ログの収集に時間が掛かることがあります。このため，これらの処理に対して，それぞれタイムアウトが設定できます。

なお，snapshot ログの収集に対するタイムアウトは，想定外の障害発生によって snapshot ログの収集に時間が掛かり過ぎる場合に，論理サーバの再起動を優先しなくてはならないような状況が見込まれるときなどに設定してください。snapshot ログの収集時にタイムアウトが実行されると，収集処理は強制的に中断され，トラブルシューティングに必要な情報が取得できなくなります。

(4) 運用管理コマンドを使用した snapshot ログの収集

snapshot ログは，自動実行されるほか，運用管理コマンド（mngsvrutil）を使用して任意のタイミングで収集できます。

なお，取得できる情報の種類は，トラブル発生時に自動的に収集される snapshot ログと同じです。「2.3.3(2) snapshot ログとして収集できる資料」を参照してください。

ログの出力先については，adminagent.properties ファイル（運用管理エージェントプロパティファイル）の adminagent.snapshotlog.log_dir キーの値を参照してください。デフォルトの出力先は次のディレクトリです。

- Windows の場合
  <Manager のログ出力ディレクトリ>¥snapshot
- UNIX の場合
  <Manager のログ出力ディレクトリ>/snapshot

snapshot ログを収集するには，運用管理コマンド（mngsvrutil）にサブコマンド「collect」を指定して実行します。-t オプションで指定したホスト内の snapshot ログが収集できます。

コマンドの実行形式と実行例を次に示します。この例では，一次送付ファイルと二次送付ファイルの両方を収集します。<n>には，一次送付資料または二次送付資料のどちらを収集するかを指定します。

実行形式

```
mngsvrutil -m <Management Serverのホスト名>[:<ポート番号>] -u <管理ユーザID> -p <管理パスワード> -t <ホスト名> -k host collect snapshot <n>
```

実行例

```
mngsvrutil -m mnghost -u user01 -p pw1 -t host01 -k host collect snapshot 1
mngsvrutil -m mnghost -u user01 -p pw1 -t host01 -k host collect snapshot 2
```

2.3.4　取得した情報の格納先

snapshot ログの収集によって自動で取得された情報は，adminagent.properties（運用管理エージェントプロパティファイル）の adminagent.snapshotlog.log_dir キーで指定したディレクトリに次の名称で格納されています。

- 一次送付資料として収集したファイル

  snapshot-<ホスト名※1>-log-<yyyyMMddHHmmss※2>.zip
- 二次送付資料として収集したファイル

  snapshot-<ホスト名※1>-log-<yyyyMMddHHmmss※2>.2.zip

注

定義送付資料は収集されません。

注※1

論理サーバを指定して収集した場合は，論理サーバ名になります。

注※2

yyyy：西暦　MM：月　dd：日　HH：時　mm：分　ss：秒

なお，出力先ディレクトリのデフォルトは，次のとおりです。

- Windows の場合

  <Manager のログ出力ディレクトリ>¥snapshot
- UNIX の場合

  <Manager のログ出力ディレクトリ>/snapshot

参考

snapshot ログとして収集しようとしたファイルの一覧，および実際に収集したファイルの一覧は，次のファイルで確認できます。これらのファイルは，アーカイブされた zip ファイルに含まれています。

- 収集しようとしたファイルの一覧

  snapshot-<ホスト名>-log-<yyyyMMddHHmmss※>/filelist_pre.txt

  snapshot-<ホスト名>-log-<yyyyMMddHHmmss※>.2/filelist_pre.txt

- 実際に収集したファイルの一覧

  snapshot-<ホスト名>-log-<yyyyMMddHHmmss※>/filelist_post.txt

  snapshot-<ホスト名>-log-<yyyyMMddHHmmss※>.2/filelist_post.txt

注※　yyyy：西暦　MM：月　dd：日　HH：時　mm：分　ss：秒

2.4　取得が必要な資料の種類

この節では，アプリケーションサーバのシステムで想定されるトラブルの種別と，それに応じて取得が必要な資料について説明します。

トラブルシューティングに必要な資料は，発生したトラブルの種別によって異なります。例えば，エラーメッセージが出力されたのか，アプリケーションサーバのサーバプロセスが異常終了したのかによって，必要な資料は異なります。

また，トラブルの内容によっては保守員に連絡する必要があります。**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。

保守員に連絡が必要なトラブル

- 出力されたエラーメッセージの対処に「保守員に連絡してください。」とある場合
- トラブルの要因がわからない場合
- エラーメッセージの内容に対処できない場合

保守員に連絡する場合には，次の内容をできるだけ正確に伝えてください。

- エラーの発生時期
- エラー発生の直前に実行した操作
- エラー発生時の画面の動作
- トラブル発生時に取得した資料

トラブルシューティングに必要な資料は，資料を保守員へ送付するときのタイミングによって，一次送付資料，二次送付資料，および定義送付資料に分類されます。

- **メールなどで早急に送付する資料（一次送付資料**および**定義送付資料）**
  メールの添付資料として送付できるファイルサイズの資料が該当します。メールなどで早急に保守員に送付できます。
  例えば，次の資料が一次送付資料として取得できます。
  - メッセージログ
    J2EE サーバなどの稼働状態およびエラー情報が出力されます。
  - ユーザログ
    アプリケーション中で出力される標準出力および標準エラー出力の情報が出力されます。
  - 例外ログ
    システムでトラブルが発生したときの例外情報が出力されます。
  - 保守用ログ
    システムでトラブルが発生したときの障害保守情報が出力されます。保守員が障害解析用に使用します。

また，定義送付資料として，定義ファイルだけを収集した情報が出力されます。定義誤りの調査に使用します。

- **別途送付する資料（二次送付資料）**

  ファイルサイズが比較的大きく，保守員への送付に時間が掛かるため，メール以外の方法などで別途送付する必要がある資料が該当します。なお，二次送付資料には，一次送付資料の内容が含まれます。

  例えば，次の資料が二次送付資料として取得できます。

  - 性能解析トレース

    リクエストの一連の処理で出力されるトレース情報が出力されます。
  - メモリダンプ

    プロセスのメモリイメージがダンプ出力されます。

2.4.1 トラブルの種別と取得が必要な資料

アプリケーションサーバで想定されるトラブルには，次に示す種別があります。

- **エラーメッセージ出力**

  エラーメッセージが出力されます。
- **システムダウン**

  アプリケーションサーバのサーバプロセスが異常終了します。
- **ハングアップ（無応答）**

  アプリケーション呼び出しの応答が返らなくなります。
- **スローダウン**

  アプリケーション呼び出しの応答は返りますが，応答時間が長くなります。

トラブル種別ごとに取得する資料を，次に示します。

表 2-5 トラブル種別ごとの取得資料一覧

項番	資料の種類	トラブル種別			
		エラーメッセージ出力	システムダウン	ハングアップ（無応答）	スローダウン
1	メッセージログ※1	○	○	○	○
2	ユーザログ※1	○	○	○	○
3	例外ログ※1	○	○	○	○
4	保守用ログ※1	○	○	○	○
5	性能解析トレース	○	○	○	○
6	JavaVM のスレッドダンプ	○	○	○	○

項番	資料の種類	トラブル種別			
		エラーメッセージ出力	システムダウン	ハングアップ（無応答）	スローダウン
7	JavaVM の GC のログ	△	○	○	○
8	メモリダンプ	－	○	○	○
9	JavaVM ログファイル※2	－	○	○	○
10	エラーリポートファイル	－	○	－	－
11	OS の状態・ログ	○	○	○	○
12	OS の統計情報	△	○	○	○
13	定義情報	○	○	○	○
14	作業ディレクトリ※4	△	△	△	△
15	リソースの設定	○	○	○	○
16	Web サーバのログ	○	○	○	○
17	JavaVM のスタックトレース※3	－	○	－	－
18	JavaVM の明示管理ヒープ機能のイベントログ※2	－	○	○	○

（凡例）

○：必須

△：取得するかどうかは，実際のエラーの内容によって異なる

－：取得しない

注※1

アプリケーションサーバの構成ソフトウェアが出力するログです。

注※2

JavaVM 起動オプションを指定している場合に出力されます。取得するためのオプションについては，「4.10　JavaVM ログ(JavaVM ログファイル)」を参照してください。また，オプションの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.　JavaVM 起動オプション」を参照してください。

注※3

UNIX を使用している場合だけ出力されます。

注※4

作業ディレクトリのデフォルトは，<Application Server のインストールディレクトリ>¥CC¥server¥public（Windows の場合），または/opt/Cosminexus/CC/server/public（UNIX の場合）です。

2.4.2　取得が必要な資料の一覧

アプリケーションサーバで，トラブルシューティングに必要な資料の一覧を次の表に示します。また，それぞれの資料を保守員に送付する場合の送付タイミングについても示します。

表 2-6　取得資料一覧

項番	資料の種類	説明	送付タイミング
1	メッセージログ※1	J2EE サーバなどの稼働状態およびエラー情報が出力されます。	一次／二次
2	ユーザログ※1	アプリケーション中で出力される標準出力および標準エラー出力の情報が出力されます。	一次
3	例外ログ※1	システムでトラブルが発生したときの例外情報が出力されます。	一次
4	保守用ログ※1	システムでトラブルが発生したときの障害保守情報が出力されます。保守員が障害解析用に使用します。	一次／二次
5	性能解析トレース	リクエストの一連の処理で出力されるトレース情報が出力されます。	二次
6	JavaVM のスレッドダンプ	JavaVM の稼働情報やスレッドのスタック状態が出力されます。	一次
7	JavaVM の GC のログ	JavaVM の GC の活動状態が出力されます。	一次
8	メモリダンプ	プロセスのメモリイメージがダンプ出力されます。	二次
9	JavaVM ログファイル※2	Developer's Kit for Java で提供される JavaVM のログが出力されます。	一次
10	エラーリポートファイル	JavaVM がダウンした場合に出力されるログファイルです。	一次
11	OS の状態・ログ	OS の稼働情報の記録です。	一次／二次
12	OS の統計情報	OS の統計情報の記録です。	一次
13	定義情報	アプリケーションサーバの各種設定の定義情報です。	一次／定義
14	作業ディレクトリ	アプリケーションサーバ稼働時に作業ファイルが格納されているディレクトリです。	一次／二次／定義
15	リソースの設定	アプリケーションサーバ上で使用するリソース（データソース，リソースアダプタ）の設定情報です。	二次／定義
16	Web サーバのログ	HTTP Server，Microsoft IIS のログです。	一次／二次
17	JavaVM のスタックトレース※3	JavaVM が異常終了した場合の原因究明に使用する情報です。	一次／二次
18	JavaVM の明示管理ヒープ機能のイベントログ※2	明示管理ヒープ機能で発生するイベント（Explicit メモリブロックの初期化，Explicit メモリブロックへのオブジェクトの生成，Explicit メモリブロックの解放処理など）の情報が出力されます。	一次

（凡例）

一次：一次送付資料

二次：二次送付資料

定義：定義送付資料

注※1

アプリケーションサーバの構成ソフトウェアが出力するログです。

注※2

JavaVM 起動オプションを指定している場合に出力されます。取得するためのオプションについては，「4.10　JavaVM ログ(JavaVM ログファイル)」を参照してください。また，オプションの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.　JavaVM 起動オプション」を参照してください。

注※3

UNIX を使用している場合だけ出力されます。

ログファイルは，各構成ソフトウェアの環境設定ファイルで指定したディレクトリに自動的に作成されます。ファイル名には，1～指定したログファイルの面数の通し番号が付きます。ログファイルのファイルサイズが一面当たりの最大サイズに達すると，次のログファイルに出力先を切り替えます。指定したログファイルの面数分のファイルに出力したあとは，ファイル名の番号が 1 のログファイルに再度ログを出力します。なお，ログファイルの 1 レコードの出力が途切れないようするために，ログファイルのファイルサイズが環境設定ファイルに指定したサイズより若干大きくなることがあります。1 レコードの最大長は約 4 キロバイトです。

2.4.3　取得方法および調査方法との対応

ここでは，取得が必要な資料の取得方法と調査方法の参照先を示します。また，snapshot ログとして取得できる情報の種類についても示します。

表 2-7　取得が必要な資料の取得方法と調査方法の参照先

項番	資料の種類	デフォルト設定の snapshot ログで取得可※1	参照先	
			個別の収集方法	調査方法
1	メッセージログ	○	4.3，4.4，4.5	5.2，5.3
2	ユーザログ	○		
3	例外ログ	○		
4	保守用ログ	○		
5	性能解析トレース	○	4.6	5.4，7.7
6	JavaVM のスレッドダンプ	○※2	4.7	5.5
7	JavaVM の GC のログ	×	4.8	5.6
8	メモリダンプ	×※2	4.9	＿※3
9	JavaVM ログファイル※4	×	4.10	5.7
10	エラーリポートファイル	×	4.11	5.8
11	OS の状態・ログ	○	4.12	5.9
12	OS の統計情報	×	4.13	＿※3

項番	資料の種類	デフォルト設定の snapshot ログで取得可※1	参照先	
			個別の収集方法	調査方法
13	定義情報	○	4.14	―※3
14	作業ディレクトリ	×	4.15	―※3
15	リソースの設定	×	4.16	―※3
16	Web サーバのログ	×	4.17	―※3
17	JavaVM のスタックトレース※3, ※5	×	4.18	5.10
18	JavaVM の明示管理ヒープ機能のイベントログ※4	×	4.19	5.11

（凡例）○：取得できる　×：取得できない

注※1

snapshot ログの収集対象ディレクトリを追加すれば取得できます。

注※2

あらかじめ障害検知時コマンドなどによって出力されている場合だけ収集できます（snapshot ログ収集時には，これらのダンプを出力するコマンドは実行されません）。

注※3

保守員が使用する資料です。

注※4

JavaVM 起動オプションを指定している場合に出力されます。取得するためのオプションについては，「4.10　JavaVM ログ（JavaVM ログファイル）」を参照してください。また，各オプションの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.　JavaVM 起動オプション」を参照してください。

注※5

UNIX を使用している場合だけ出力されます。

2.5　トラブルへの対処と回復

この節では，次のようなトラブルが発生した場合の対処方法および回復方法について説明します。

- 構成ソフトウェアのプロセス（論理サーバ）が異常終了した場合
- J2EE アプリケーションの強制停止に失敗した場合
- データベースセッションフェイルオーバ機能を使用時にトラブルが発生した場合
- JavaVM が異常終了した場合
- OutOfMemoryError 発生時に運用管理エージェントが強制終了した場合
- JP1 と連携したシステムでトラブルが発生した場合
- 1:1 系切り替えシステムでトラブルが発生した場合
- N:1 リカバリシステムでトラブルが発生した場合
- ホスト単位管理モデルを対象とした系切り替えシステムでトラブルが発生した場合
- EJB クライアントでトラブルが発生した場合

2.5.1　構成ソフトウェアのプロセス（論理サーバ）が異常終了した場合

ここでは，アプリケーションサーバの構成ソフトウェアのプロセスが異常終了したときに，システムを再起動する方法について説明します。

アプリケーションサーバで構築したシステムでは，アプリケーションサーバの構成ソフトウェアのプロセス（論理サーバ）が異常終了したとき，起動順序が設定されていれば Management Server によって自動的に再起動されます。Management Server では，構成ソフトウェアのプロセスを**論理サーバ**として管理しています。

手動で再起動を実行する場合，異常終了した構成ソフトウェアの起動順序に依存関係があるときは，前提となる構成ソフトウェアが起動していることを確認してから再起動する必要があります。前提となる構成ソフトウェアは，システムの運用形態によって異なります。以降で，構成ソフトウェアの起動順序の依存関係，および構成ソフトウェアの再起動方法について説明します。

ポイント

CTM を利用したシステムでは，CTM によって，すぐに再起動すれば，クライアントにエラーが返却される前にシステムを回復するように設定できます。ただし，リクエストキューの最大登録数を超えた場合は，クライアントにエラーが返却されます。

CTM を利用したシステムについては，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「3.　CTM によるリクエストのスケジューリングと負荷分散」を参照してください。

(1) 構成ソフトウェアのプロセスの起動順序の依存関係

アプリケーションサーバで構築したシステムを構成する構成ソフトウェアの，プロセスの起動順序の依存関係について説明します。

以降は，構成ソフトウェアのプロセスの起動順序の依存関係について説明します。

(a) プロセスの依存関係

プロセスの依存関係を次の表に示します。

表 2-8 プロセスの依存関係

プロセスの種類	前提プロセス
パフォーマンストレーサ	－
スマートエージェント※1，※2	－
CTM ドメインマネジャ※1	スマートエージェント※1，※2
CORBA ネーミングサービス	－
CTM デーモン※1	• パフォーマンストレーサ • スマートエージェント • CTM ドメインマネジャ • CORBA ネーミングサービス
J2EE サーバ	• パフォーマンストレーサ • スマートエージェント • CTM ドメインマネジャ※3 • CORBA ネーミングサービス※4 • CTM デーモン※3
Web サーバ	パフォーマンストレーサ

（凡例）

－：前提プロセスがない

注※1

CTM を使用する場合に起動するプロセスです。

注※2

トランザクションサービスを使用する場合に起動するプロセスです。

注※3

CTM を使用する場合に前提となるプロセスです。

注※4

CORBA ネーミングサービスをアウトプロセスで使用する場合に起動するプロセスです。CORBA ネーミングサービスをインプロセスで使用する場合は必要ありません。

(2)　プロセスの再起動方法

システムでプロセスが異常終了した場合のプロセスの再起動方法について説明します。

ポイント

CTM に関連するプロセスが異常終了して再起動できない場合は，次の手順で対処してください。

1. プロセスが再起動できない場合，出力されたエラーメッセージからトラブルの要因を調査する。
2. ctmrasget コマンドを実行して CTM の実行環境のバックアップを取得する。

(a)　再起動の手順

プロセスの再起動（回復）手順を次の表に示します。

表 2-9　プロセスの再起動（回復）手順

異常プロセス	開始コマンド	再起動（回復）手順
データベースサーバ	−	DB サーバを再起動してください。
OpenTP1	−	OpenTP1 を再起動してください。
パフォーマンストレーサ	cprfstart	パフォーマンストレーサを再起動してください。
スマートエージェント※1，※2	osagent	スマートエージェントを再起動してください。
CTM ドメインマネジャ※1	ctmdmstart	CTM ドメインマネジャを再起動してください。
CORBA ネーミングサービス	nameserv	次に示す手順で対処してください。なお，CTM を使用しない場合，手順 1 および手順 4 は不要です。 1. CTM デーモンの強制停止 2. J2EE サーバの強制停止 3. CORBA ネーミングサービスの再起動 4. CTM デーモンの再起動 5. J2EE サーバ再起動
CTM デーモン※1	ctmstart	次に示す手順で対処してください。 1. CORBA ネーミングサービスの強制停止 2. J2EE サーバの強制停止 3. CORBA ネーミングサービスの再起動 4. CTM デーモンの再起動 5. J2EE サーバの再起動
CTM レギュレータ	−	CTM デーモンによって自動再起動されるため，再起動は不要です。
J2EE サーバ	cjstartsv	J2EE サーバを再起動してください。
Web サーバ	−	Web サーバを再起動してください。

（凡例）

－：使用する製品によって開始コマンドが異なる，または該当する開始コマンドがない

注※1

CTM を使用する場合に起動するプロセスです。

注※2

トランザクションサービスを使用する場合に起動するプロセスです。

2.5.2　J2EE アプリケーションの強制停止に失敗した場合

J2EE アプリケーションの強制停止に失敗した場合，J2EE サーバによって，次の情報が出力されます。

- メッセージ

  メッセージログに出力されます。メッセージログの出力先については，「4.3　アプリケーションサーバのログ（J2EE アプリケーションを実行するシステム）」を参照してください。

- スタックトレース

  例外ログとスレッドダンプに出力されます。例外ログの出力先については，「4.3　アプリケーションサーバのログ（J2EE アプリケーションを実行するシステム）」を参照してください。スレッドダンプについては，「5.5　JavaVM のスレッドダンプ」を参照してください。

なお，J2EE アプリケーションの強制停止では，内部でメソッドキャンセルが実行されています。メソッドキャンセル実行時に出力される障害情報については，マニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「5.3.13　J2EE アプリケーション実行時間監視で出力されるログ情報」を参照してください。

2.5.3　データベースセッションフェイルオーバ機能でトラブルが発生した場合

ここでは，データベースセッションフェイルオーバ機能にトラブルが発生した場合の対処方法として，次の 2 種類のトラブルへの対処について説明します。

- J2EE サーバでトラブルが発生した場合
- データベースでトラブルが発生した場合

(1)　J2EE サーバでトラブルが発生した場合

データベース上のデータへの変更を含む処理中に，J2EE サーバの障害でプロセスがダウンした場合，データベースのロールバックで障害前の状態に戻るため，システムの整合性が保たれます。J2EE サーバの障害を取り除き，Web アプリケーションを再開することでトラブルを回復できます。

J2EE サーバのホストがハングアップした場合や，ネットワークに障害が発生した場合，データベースの操作が中断されレコードの排他（グローバルセッション情報のロック）が未解放の状態となるときがありま

す。この場合，J2EE サーバが回復し Web アプリケーションを再開する前に，未解放の排他を解放する必要があります。

> **参考**
>
> 未解放の排他の解放には，クライアントからデータベースへの無効接続を検出し，接続セッションを強制終了させるなどの方法があります。未解放の排他の解放について，HiRDB を使用する場合は，マニュアル「HiRDB UAP 開発ガイド」の UAP 処理時間監視機能に関する説明を参照してください。Oracle を使用する場合は，Oracle のマニュアルを参照してください。

(2) データベースでトラブルが発生した場合

データベースで障害が発生した場合，データベースを回復すればロールバックで障害前の状態に戻るため，データベースセッションフェイルオーバ機能を使用した Web アプリケーションは業務を再開できます。Web アプリケーションの再起動は必要ありません。

そのほかのデータベース障害時の対応については，各データベースのマニュアルを参照してください。

2.5.4 JavaVM が異常終了した場合

JavaVM で異常終了が発生した場合の対処方法と，異常終了が発生した場合に出力される情報について説明します。

(1) 異常終了時の対処方法

UNIX の場合，異常終了時には，次の手順で対処してください。なお，Windows の場合には，この対処は不要です。

1. **異常終了が発生したマシン上で javatrace コマンドを実行します。**

   javatrace.log ファイルが出力されます。javatrace コマンドの実行方法については，「4.18 JavaVM のスタックトレース情報」を参照してください。

2. **取得した javatrace.log ファイルを，エラーリポートファイル（hs_err_pid<プロセス ID>.log）とあわせて保守員に送付します。**

   エラーリポートファイルの取得方法については，「4.11 JavaVM 出力メッセージログ（標準出力またはエラーリポートファイル）」を参照してください。

3. **次のコマンドを実行して，JavaVM の実行ファイルやトラブル発生時にロードされていたライブラリ，および core ダンプのアーカイブファイルを作成します。**

   - AIX の場合

     snapcore コマンドを実行してください。pax コマンドで圧縮されたアーカイブファイルが作成されます。

- Linux の場合

  compress コマンドがインストールされている場合は car_tar_Z コマンドを実行してください。compress コマンドで圧縮されたアーカイブファイルが作成されます。

  gzip コマンドがインストールされている場合は car_tar_gz コマンドを実行してください。gzip コマンドで圧縮されたアーカイブファイルが作成されます。

4. 作成されたアーカイブファイルを，保守員に送付します。

(2) 異常終了時に出力される情報

JavaVM で次の異常終了が発生した場合に出力される情報について説明します。

- JavaVM 処理中に C ヒープ不足が発生した場合
- JavaVM 処理中の C ヒープ不足以外の OutOfMemoryError が発生した場合
- 内部論理エラーが発生した場合

(a) JavaVM 処理中に C ヒープ不足が発生した場合

C ヒープが不足すると，次のようなメッセージに続いて，メモリの状態，Java ヒープ情報，およびスタックトレース情報が標準出力およびエラーリポートファイル（hs_err_pid<プロセス ID>.log）に出力されます。そのあとで，JavaVM が強制終了されます。

出力された情報を確認して対処してください。

```
Exception in thread <スレッド名称> java.lang.OutOfMemoryError:requested <メモリ確保要求サイズ> bytes [ for <内部調査用メッセージ>] .
```

(b) JavaVM 処理中の C ヒープ不足以外の OutOfMemoryError が発生した場合

J2EE サーバを起動するときのオプションとして-XX:+HitachiOutOfMemoryAbort を設定している場合，次の要因によって OutOfMemoryError が発生したときに，メッセージを出力して JavaVM が強制終了します。出力された情報を確認して対処してください。

また，OutOfMemory ハンドリング機能が有効の場合（-XX:+HitachiOutOfMemoryHandling を設定している場合），OutOfMemoryError 発生の要因が Java ヒープ不足または Metaspace 領域不足のときには，強制終了しないことがあります。

強制終了する要因

- Java ヒープ不足
- Metaspace 領域不足
- J2SE クラスライブラリでの C ヒープ不足

なお，JavaVM 処理中に C ヒープ不足になった場合は，このオプションの指定に関係なく強制終了します。

終了時には，次のメッセージが出力されます。

```
java.lang.OutOfMemoryError occurred.
JavaVM aborted because of specified -XX:+HitachiOutOfMemoryAbort options.
```

強制終了するタイミング

JavaVM が強制終了するタイミングは，オプションの設定によって異なります。

- オプションとして-XX:+HitachiOutOfMemoryStackTrace が指定されている場合は，スタックトレースが出力されたあとで終了します。ただし，あらかじめ java.io.File.deleteOnExit メソッドや java.lang.Runtime.addShutdownHook メソッドによって JavaVM 終了時に実行する処理を登録していた場合も，それらは実行されないで強制終了します。
- オプションとして-XX:+HitachiOutOfMemoryAbortThreadDump が指定されている場合は，スレッドダンプを出力したあとで終了します。特に，「Java ヒープ不足」または「Metaspace 領域不足」の場合は，オプションとして-XX:+HitachiOutOfMemoryAbortThreadDumpWithJHeapProf も設定されているときには，クラス別統計情報付きスレッドダンプを出力したあとで終了します。

OutOfMemory ハンドリング機能が有効な場合の動作

J2EE サーバを起動するときのオプションとして-XX:+HitachiOutOfMemoryAbort に加えて-XX:+HitachiOutOfMemoryHandling を設定すると，OutOfMemory ハンドリング機能が有効になります。この場合，Java ヒープまたは Metaspace 領域不足が発生した際に，OutOfMemory のハンドリングによって強制終了するかどうかの判定処理が実行されます。判定処理の結果，次に示す OutOfMemoryError スロー条件のすべてに合致した場合は，強制終了が実行されません。

OutOfMemoryError スロー条件

- Java ヒープ不足，または Metaspace 領域不足が原因の OutOfMemory である。
- Web コンテナ上の Web アプリケーション（Servlet／JSP）が実行中のリクエスト処理，EJB クライアントアプリケーションから呼び出された Enterprise Bean が実行中の処理，Message-driven Bean が実行中の処理，または Timer Service から呼び出された Enterprise Bean が実行中の処理で発生した OutOfMemory である。
- OutOfMemoryError スロー除外条件に該当しない。

OutOfMemoryError スロー除外条件

今回の OutOfMemory が発生した時刻から過去 1 時間以内の Java ヒープ不足が原因の OutOfMemory の発生回数と，Metaspace 領域不足が原因の OutOfMemory の発生回数の合計値（今回の OutOfMemory を含む）が，-XX:HitachiOutOfMemoryHandlingMaxThrowCount オプションに指定した値よりも大きい。

詳細は，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]HitachiOutOfMemoryHandling（OutOfMemory ハンドリングオプション）」を参照してください。

OutOfMemory ハンドリング機能が有効な場合，Java ヒープ不足，および Metaspace 領域不足が原因の OutOfMemory 発生時に，OutOfMemory の発生頻度に関する情報が JavaVM ログファイルに出力されます。

(c) 内部論理エラーが発生した場合

内部論理エラーが発生すると，エラーが発生した JavaVM の情報，調査用のエラー ID および問題が発生したスレッドを示すメッセージが，標準出力およびエラーリポートファイル（hs_err_pid<プロセスID>.log）に出力されます。出力された情報を保守員に送付してください。

2.5.5 OutOfMemoryError 発生時に運用管理エージェントが強制終了した場合

運用管理エージェント稼働中に java.lang.OutOfMemoryError が発生した場合，ログを出力して，運用管理エージェントは強制終了されます。ログは次の場所に出力されます。

- Windows の場合
  <Manager のログ出力ディレクトリ>¥adminagent.javalog[nn].log
- UNIX の場合
  <Manager のログ出力ディレクトリ>/adminagent.javalog[nn].log
  注　[nn]は 0～99 の番号を示します。

なお，運用管理エージェントの終了処理ですべての論理サーバを停止する設定をしている場合に，運用管理エージェントが強制終了しても論理サーバは停止しません。運用管理エージェントの終了処理で論理サーバを停止する設定は，adminagent.properties（運用管理エージェントプロパティファイル）の adminagent.finalization.stop_servers キーに「true」を設定します。

また，アプリケーションサーバで提供している JDK 以外を使用している場合は，OutOfMemoryError の発生による運用管理エージェントの強制終了はしません。

OutOfMemoryError の発生によって運用管理エージェントが強制終了した場合は次の作業を実施してください。

1. adminagentuser.cfg の次のオプションの設定値を見直します。
   次のオプションの設定値を見直してください。
   add.jvm.arg=-Xmx
2. 運用管理エージェントを再起動します。

2.5.6 JP1 と連携したシステムでトラブルが発生した場合

JP1 と連携したシステムでトラブルが発生した場合，次に示す対処を実施する必要があります。

(1) JP1/IM と連携したシステムでのトラブルへの対処

JP1/IM と連携したシステムで予想されるトラブルとその対処方法を次に示します。

表 2-10　JP1/IM と連携したシステムで予想されるトラブルとその対処方法

トラブルの種類	対処方法
監視ツリーの自動生成時にトラブルが発生した	トラブルが発生したら，JP1/Base のプラグインサービスログのファイルに出力されたアダプタコマンドのメッセージを基にトラブルの発生要因を調査してください。JP1/Base のプラグインサービスログのファイルの詳細については，マニュアル「JP1/Base 運用ガイド」を参照してください。
JP1 イベントが JP1/IM に通知されない	アプリケーションサーバで構築したシステムから JP1/IM に JP1 イベントが発行されているかどうか，運用管理エージェント・運用監視エージェント・Management Server のログを確認してください。運用管理エージェント・運用監視エージェント・Management Server のログの格納場所については，「4.3.1　Component Container のログの取得」，または「4.4.1 Component Container のログの取得（バッチアプリケーションを実行するシステム）」を参照してください。 運用管理エージェント・運用監視エージェント・Management Server のログの内容に応じて，次の対処を実施してください。 • JP1 イベント発行のログが出力されていない場合，運用管理サーバ※での JP1 イベント発行の設定内容を確認してください。 • JP1 イベント発行のログが出力されている場合，JP1 統合運用管理サーバでの JP1/Base の構成定義の作成内容を確認してください。また，運用管理サーバ※および J2EE サーバでの JP1/Base のイベントサービスの動作環境設定内容を確認してください。
JP1/IM-View のモニタ起動操作を実行したあと Web ブラウザが起動しない	モニタ起動コマンドをコピーしたディレクトリに格納されている mngsvrmonitor.log に出力されたメッセージを基にトラブルの発生要因を調査してください。

注※　アプリケーションサーバの運用管理サーバのことです。

(2)　JP1/AJS と連携したシステムでのトラブルへの対処

JP1/AJS 連携による運用管理コマンド（mngsvrutil）の自動実行でエラーが発生した場合，JP1/AJS-View のジョブネットモニタウィンドウから，トラブルが発生したジョブの詳細情報を表示し，実行結果を確認してください。実行結果詳細に表示された mngsvrutil コマンドのエラーメッセージを基にトラブルの発生要因を調査してください。JP1/AJS-View での操作の詳細については，マニュアル「JP1/Automatic Job Management System 操作ガイド」を参照してください。

2.5.7　1:1 系切り替えシステムでトラブルが発生した場合

1:1 系切り替えシステムで，待機系のホストへの系切り替え処理がデータベースサーバの障害（サーバダウンやデッドロックなど）によってタイムアウトした場合の対処について，OS ごとに説明します。

(1)　Windows の場合

ログを取得してから，手動で待機系ホストをオンラインにしてください。

(a)　1:1 系切り替えシステムのログの取得

1:1 系切り替えシステムでトラブルが発生した場合，クラスタログを取得する必要があります。Windows を標準パスにインストールした場合のクラスタログの出力先を次に示します。

```
C:¥WINDOWS¥cluster¥cluster.log
```

クラスタログには次の情報が出力されます。

- クラスタサービスの稼働ログ
- VBScript の構文にエラーがあった場合のエラーメッセージ
- 汎用スクリプトの Resource.LogInformation メソッド
- その他のメッセージ

クラスタログの詳細については，Windows のドキュメントを参照してください。

また，J2EE サーバが動作しない場合は，Component Container のログも参照してください。

(b)　1:1 系切り替えシステムの手動リカバリ

次の手順で，1:1 系切り替えシステムを手動でリカバリしてください。

1. データベースを再起動するなどして，タイムアウトした原因を解消します。
2. 待機系のホストの対象リソースをオンラインにします。

(2)　UNIX の場合

データベースを再起動するなどして，タイムアウトした原因を解消してください。

2.5.8　N:1 リカバリシステムでトラブルが発生した場合

N:1 リカバリシステムでトラブルが発生した場合のリカバリ手順について，OS ごとに説明します。

(1)　Windows の場合

N:1 リカバリシステムで，待機系のホスト（リカバリ専用サーバ）でのリカバリ処理がデータベースサーバの障害（サーバダウンやデッドロックなど）によってタイムアウトした場合，ログを取得して，手動でリカバリを実行してください。リカバリ手順を次に示します。

1. N:1 系切り替えシステムのログを取得します。

   N:1 リカバリシステムでトラブルが発生した場合，クラスタログを取得する必要があります。取得するログは，1:1 系切り替えシステムでトラブルが発生した場合に取得するログと同じです。取得するログの詳細については，「2.5.7　1:1 系切り替えシステムでトラブルが発生した場合」を参照してください。

2. N:1 系切り替えシステムを手動でリカバリします。

次のどちらかの方法で，N:1 リカバリシステムを手動でリカバリしてください。

- 待機系のホストの対象リソースをオンラインにする
- J2EE サーバのトランザクション回復コマンド（cjstartrecover）を実行する

(a) 待機系のホストの対象リソースをオンラインにしてリカバリを実行する

待機系のホストの対象リソースをオンラインにしてリカバリを実行する手順について説明します。

1. データベースを再起動するなどして，タイムアウトした原因を解消します。
2. 待機系のホストの対象リソースをオンラインにします。

(b) J2EE サーバのトランザクション回復コマンド（cjstartrecover）を実行してリカバリを実行する

J2EE サーバのトランザクション回復コマンド（cjstartrecover）を実行してリカバリを実行する手順について説明します。

1. データベースを再起動するなどして，タイムアウトした原因を解消します。
2. 待機系のホストの汎用スクリプトの「Dir_Name」に指定したパスにフォルダを作成します。

すでにフォルダが存在する場合は，削除してからフォルダを作成してください。

3. 待機系のホストの汎用スクリプトがオンラインのときに，クラスタのログを参考に，cjstartrecover を実行します。
4. リカバリに成功したら，「Dir_Name」に指定したパスに作成したフォルダを削除します。

cjstartrecover コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjstartrecover（J2EE サーバのトランザクション回復）」を参照してください。

(2) UNIX の場合

N:1 リカバリシステムで，待機系のホスト（リカバリ専用サーバ）でのリカバリ処理がデータベースサーバの障害（サーバダウンやデッドロックなど）によってタイムアウトした場合，次の手順に従って手動でリカバリを実行します。

1. データベースを再起動するなどして，タイムアウトした原因を解消します。
2. 待機系のホストで，ダウンした実行系のホストに対応する monbegin を実行します。

```
# monbegin サーバの識別名
```

下線部分には，servers ファイルのオペランド「alias」に指定されている実行系のサーバの識別名を指定します。

3. 待機系のホストで，ダウンした実行系のホストに対応する monact を実行します。

```
# monact サーバの識別名
```

下線部分には，servers ファイルのオペランド「alias」に指定されている実行系のサーバの識別名を指定します。

これによって，待機系のホスト（リカバリ専用サーバ）で，ダウンした実行系のホストで未決着だったトランザクションのリカバリ処理が実行されます。

参考

servers ファイルの定義など，HA モニタでのサーバ対応の環境設定については，マニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「19.5.4　HA モニタの環境設定」を参照してください。

2.5.9　ホスト単位管理モデルを対象とした系切り替えシステムでトラブルが発生した場合

ホスト単位管理モデルを対象とした系切り替えシステムでトラブルが発生した場合の対処は，1:1 系切り替えシステムでトラブルが発生した場合と同じです。詳細については，「2.5.7　1:1 系切り替えシステムでトラブルが発生した場合」を参照してください。

2.5.10　EJB クライアントでトラブルが発生した場合

グローバルトランザクションを使用している EJB クライアントがダウンした場合，グローバルトランザクションをリカバリする必要があります。グローバルトランザクションのリカバリは，EJB クライアントを再起動することで実行できます。

グローバルトランザクションのリカバリの処理が完了したかどうかを確認する方法を次に示します。

- アプリケーションサーバで cjlisttrn コマンドを実行して，トランザクションがアクティブな状態になっているかを確認します。cjlisttrn コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjlisttrn（稼働中の J2EE サーバのトランザクション情報の表示)」を参照してください。
- リソースアダプタを停止できるかどうかを確認してください。停止できる場合はグローバルトランザクションのリカバリ処理が完了しています。
- アプリケーションサーバが正常停止できるかどうかを確認してください。正常停止できる場合はグローバルトランザクションのリカバリ処理が完了しています。
- 各リソースが提供するツール，コマンドなどを使用して確認してください。

また，複数の EJB クライアントが存在する場合に，どの EJB クライアントがグローバルトランザクションのリカバリに必要なのかを調査する手順を次に示します。

1. cjlisttrn コマンドを-pending オプションを指定して実行して未決着のトランザクションを確認します。

   実行形式

   ```
   cjlisttrn [<サーバ名称>] -pending -bqual
   ```

   実行例

   cjlisttrn MyServer -pending -bqual

2. 手順 1.で確認した未決着のトランザクションから，次のすべての条件に当てはまるトランザクションがあるかどうかを確認します。

   - cjlisttrn コマンドを発行するたびに経過時間が増加している
   - ブランチの種類が「Sub」または「Sub(recovered)」である
   - グローバルトランザクション ID に，停止した EJB クライアントの起動時に出力される KFCB40051-I メッセージの TmHash の値が含まれる

   これらのすべての条件に当てはまるトランザクションが，グローバルトランザクションのリカバリに必要です。

2.6　トラブルシューティングに関連する留意事項

ここでは，トラブルシューティングに関連する留意事項について説明します。

2.6.1　EJB クライアントアプリケーションのシステムログに関する留意事項

ここでは，サブディレクトリ共有モードの EJB クライアントアプリケーションで出力されるシステムログを参照する場合，およびサブディレクトリ共有モードの EJB クライアントアプリケーションを運用する場合の留意事項について説明します。

- ログファイル内で有効なデータは，ファイルの先頭から EOF までです。

  EJB クライアントアプリケーションのシステムログでは，ログファイルがラップアラウンドしたときにラップアラウンド以前のログのデータが削除されないで，先頭から順番に上書きされていきます。このため，ログファイルを参照するときには，EOF 以降のデータは無視してください。EOF 以降のデータは，ラップアラウンド以前のログファイルの無効なデータです。

  有効なログファイルのデータの末尾は，次に示すデータになります。

```
EOF CRLF CRLF CRLF CRLF----------< End of Data >----------CRLF CRLF
```

  EOF はトレースデータの終端を表す文字（0x1A）です。CRLF は，改行（0x0D，0x0A）を表します。

  出力例を次に示します。なお，トレースの終端を表す文字は，[EOF] と表記します。

  - Windows の場合

```
**** "OS名（OSバージョンなど含む）"                TZ="タイムゾーン名"
    xxxx/xx/xx xx:xx:xx.xxx
     yyyy/mm/dd hh:mm:ss.sss              pid      tid      message-id           mes
sage(LANG=ja)
0000 xxxx/xx/xx xx:xx:xx.xxx     HEJB       BE3F6FE9 015EE671 KDJEXXXXX-W          xx
xxxxxxx
0001 xxxx/xx/xx xx:xx:xx.xxx     HEJB       BE3F6FE9 015EE671 KDJEYYYYY-I          yy
yyyyyyy
0002 xxxx/xx/xx xx:xx:xx.xxx     HEJB       BE3F6FE9 015EE671 KDJEZZZZZ-I          zz
zzzzzzz
[EOF]


----------< End of Data >----------

<<ラップアラウンド前の無効なデータ>>…
…
…
```

  - UNIX の場合

```
**** "OS名（OSバージョンなど含む）"                TZ=Asia/Tokyo                      xx
xx/xx/xx xx:xx:xx.xxx
     yyyy/mm/dd hh:mm:ss.sss                  pid      tid      message-id
    message(LANG=ja)
0000 xxxx/xx/xx xx:xx:xx.xxx     HEJB           BE3F6FE9 015EE671 KDJEXXXXX-W
```

```
    xxxxxxxxx
0001 xxxx/xx/xx xx:xx:xx.xxx     HEJB             BE3F6FE9 015EE671 KDJEYYYYY-I
    yyyyyyyyy
0002 xxxx/xx/xx xx:xx:xx.xxx     HEJB             BE3F6FE9 015EE671 KDJEZZZZZ-I
    zzzzzzzzz
[EOF]



----------< End of Data >----------

<<ラップアラウンド前の無効なデータ>>…
…
…
```

- EJB クライアントアプリケーションのプロセス起動時に，システムプロパティの ejbserver.logger.channels.define.<チャネル名>.filenum キーで指定した面数のトレースファイルはすべて作成されます。このとき，トレースファイルは空白（0x20）で初期化されます。
- ユーザログファイルの容量は，システムプロパティの ejbserver.logger.channels.define.<チャネル名>.filesize キーで指定した容量で固定です。指定したサイズのトレースファイルがプロセス起動時に作成されます。このため，ログが出力されるのに伴って，容量が増減することはありません。
- ログファイルの容量または面数を変更する場合，該当するログファイルに出力しているプロセスをすべて停止して，ログファイルと mmap ディレクトリ以下のログ管理ファイルを別のディレクトリへ移動，または削除する必要があります。
- ログファイルの容量または面数を変更する場合以外では，ログファイル，およびログ管理ファイルを変更または削除しないでください。変更または削除した場合，以降のログが正しく出力されなくなるおそれがあります。
- サブディレクトリ共有モードで動作する EJB クライアントアプリケーションがログを出力しているサブディレクトリを，cjcldellog コマンドで削除しないでください。削除した場合，以降のログが正しく出力されなくなるおそれがあります。
- すでにサブディレクトリ専有モードで動作している EJB クライアントアプリケーションがある環境でサブディレクトリ共有モードで EJB クライアントアプリケーションを開始する場合，システムプロパティの ejbserver.client.ejb.log キーに，サブディレクトリ専有モードで動作している EJB クライアントアプリケーションとは異なる値を指定してください。同じ値を指定した場合，サブディレクトリ専有モードで動作する EJB クライアントアプリケーションのサブディレクトリ数を正しく管理できません。なお，サブディレクトリ専有モードは，旧バージョンとの互換用のモードです。

2.6.2 CTM 使用時の留意事項

ここでは，CTM 使用時の留意事項について説明します。CTM では次のことに注意してください。

- CTM の障害情報は，CTM ドメイン単位に CTMSPOOL 環境変数の下に取得されています。CTM デーモンおよび CTM ドメインマネジャのダウンの場合には再開始時にも障害情報を取得するため，障害発生後に障害情報を退避してください。

- CTMSPOOL 環境変数下のディレクトリは，製品の運用ディレクトリのため，ファイルやディレクトリを削除しないでください。

2.6.3 PRF 使用時の留意事項

ここでは，PRF 使用時の留意事項について説明します。PRF では次のことに注意してください。

- PRF の障害情報は，PRF 識別子単位に PRFSPOOL 環境変数の下に取得されています。PRF デーモンのダウンの場合には再開始時にも障害情報を取得するため，障害発生後に障害情報を退避してください。
- PRFSPOOL 環境変数下のディレクトリは，製品の運用ディレクトリのため，ファイルやディレクトリを削除しないでください。

2.6.4 JavaVM の資料に関する留意事項

ここでは，JavaVM の資料に関する留意事項について説明します。

- 次に示す JavaVM の資料は，マルチバイト文字をサポートしていません。日本語のクラス名称など，該当する文字の部分は文字化けします。
  - JavaVM のスレッドダンプログファイル
  - JavaVM ログファイル
  - エラーリポートファイル（JavaVM 出力メッセージログ）
  - 明示管理ヒープ機能のイベントログファイル
- 次に示すオプションで JIS X0213:2004 の第三水準および第四水準の文字が含まれる名称を指定すると，製品の JavaVM ログファイルや明示管理ヒープ機能のイベントログファイルは出力されません。
  - -XX:HitachiJavaLog オプション
  - -XX:HitachiExplicitMemoryJavaLog オプション

  また，拡張スレッドダンプのファイルの出力先ディレクトリに，JIS X0213:2004 の第三水準および第四水準の文字が含まれる場合，拡張スレッドダンプはファイルに出力されません。

3 トラブルシューティングのための準備

アプリケーションサーバで構築したシステムでは，トラブル発生時に自動でトラブルシューティングに必要な情報を取得できます。トラブルシューティングに必要な資料には，運用開始前に準備が必要なものもあります。この章では，それぞれの資料取得の設定方法について説明します。

3.1　この章の構成

トラブルシューティングに関する説明のうち，トラブルシューティングに必要な資料を取得するための設定について説明します。

トラブルシューティングに必要な資料の一部は，運用を開始する前に設定が必要です。また，デフォルトで設定されているログの出力先，サイズなどを変更することもできます。

トラブル発生時に取得する情報のうち，次に示す情報を取得するためには，運用を開始する前に設定が必要です。

- JavaVM ログ（JavaVM ログファイル）
  このファイルには，JavaVM の GC のログも出力されます。
- ユーザダンプ（Windows の場合）
- core ダンプ（UNIX の場合）
- OS の統計情報（Windows の場合）

これらの情報については，必要に応じてシステム構築時に取得するための設定をしてください。

この章の構成を次の表に示します。

表 3-1　この章の構成（トラブルシューティング（資料取得の設定））

分類	タイトル	参照先
解説	資料取得の設定の概要	3.2
設定	実行環境での設定	3.3

なお，トラブルシューティングの概要，資料の出力方法および出力内容については，それぞれ次の個所を参照してください。

- トラブルシューティングの概要と，資料を自動で出力する方法
  「2.　トラブルシューティング」
- 収集する資料の出力先と，資料を個別に出力する方法
  「4.　トラブルシューティングで必要な資料の出力先と出力方法」
- 資料に出力される内容
  「5.　トラブルの分析」

3.2 資料取得の設定の概要

トラブルシューティングに必要な資料のうち，一部の資料は，運用を開始する前に資料取得のための設定をしておく必要があります。例えば，OS の統計情報，ユーザダンプ（Windows の場合）または core ダンプ（UNIX の場合），JavaVM の GC のログなどは，事前に取得のための設定をしておかないと，取得できません。これらの資料はトラブルシューティングで必要となるため，取得することをお勧めします。

また，デフォルトの設定で取得できるようになっている資料については，特に設定は不要ですが，ログの出力先やサイズなどを変更したい場合には，簡易構築定義ファイルやユーザ定義ファイルを編集して，設定を変更してください。

トラブルシューティングに必要な資料と，事前設定または設定変更の対象となる資料との対応を次の表に示します。

表 3-2 トラブルシューティングに必要な資料と，事前設定または設定変更の対象となる資料との対応

<table>
<tr><th colspan="2">トラブルシューティングに必要な資料</th><th>事前設定または設定変更の対象となる資料</th></tr>
<tr><td rowspan="4">アプリケーションサーバのログ</td><td>メッセージログ</td><td rowspan="4">• snapshot ログ
• Management Server のログ
• J2EE サーバのログ
• バッチサーバのログ
• Web サーバのログ
• アプリケーションのユーザログ
• Manager のログ
• コンソールログ
• リソースアダプタのログ
• TPBroker のトレースファイル
• CJMS プロバイダのログ
• サーバ管理コマンドのログ
• NIO HTTP サーバのログ</td></tr>
<tr><td>ユーザログ</td></tr>
<tr><td>例外ログ</td></tr>
<tr><td>保守用ログ</td></tr>
<tr><td rowspan="4">EJB クライアントアプリケーションのシステムログ</td><td>メッセージログ</td><td rowspan="4">• システムログ※
• アプリケーションのユーザログ</td></tr>
<tr><td>ユーザログ</td></tr>
<tr><td>例外ログ</td></tr>
<tr><td>保守用ログ</td></tr>
<tr><td colspan="2">性能解析トレース</td><td>性能解析トレースファイル</td></tr>
<tr><td colspan="2">JavaVM のスレッドダンプ</td><td>JavaVM の資料</td></tr>
<tr><td colspan="2">JavaVM の GC のログ</td><td>JavaVM の資料</td></tr>
<tr><td colspan="2">メモリダンプ</td><td>• ユーザダンプ（Windows の場合）
• core ダンプ（UNIX の場合）</td></tr>
</table>

トラブルシューティングに必要な資料	事前設定または設定変更の対象となる資料
JavaVM ログファイル	JavaVM の資料
エラーリポートファイル	－
コンパイルリプレイファイル	－
OS の状態・ログ	－
OS の統計情報	OS の統計情報
定義情報	－
作業ディレクトリ	－
リソースの設定	－
Web サーバのログ	－
JavaVM のスタックトレース	－
明示管理ヒープ機能のイベントログ	JavaVM の資料

（凡例）－：事前設定，設定変更の対象となる資料なし

注

この表のほかに，事前設定または設定変更の対象となる資料に，稼働情報ファイルがあります。稼働情報ファイルは，機能ごとにサーバ性能，リソースの情報などの稼働情報を取得できるファイルで，システムの運用監視に利用できます。稼働情報ファイルはデフォルトで取得できます。

注※

EJB クライアントアプリケーションのシステムログについては，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(EJB コンテナ)」の「3.8　EJB クライアントアプリケーションのシステムログ出力」を参照してください。

この節では，アプリケーションを実行するシステムごとに，事前設定または設定変更の対象となる資料の種類と設定の概要について説明します。

3.2.1　設定できる内容

ここでは，資料取得方法として設定できる内容について説明します。

トラブルシューティングで使用する資料の一部では，取得方法として次の内容を設定できます。設定できる内容は，資料によって異なります。

- **資料を取得するかどうか**
- **資料の出力先**
- **出力先ファイルの面数**
- **出力先ファイルの切り替え方法**

なお，次のプロセスが出力する資料の一部では，出力先ファイルを切り替えるタイミング，および切り替えの際のファイル名の付与規則を選択できます。

- J2EE サーバ
- 運用管理エージェント
- Management Server※
- 仮想サーバマネージャの内部構築ツール
- サーバ通信エージェント

注※　統合ログに対してシフトモードを設定した場合に対象になります。

ここでは，これらのプロセスで選択できる出力先ファイルを切り替えるタイミング，およびファイル名の付与規則について説明します。

(1)　出力先ファイルを切り替えるタイミング

一部の資料では，資料の出力先ファイルを切り替える方法を次の 2 種類から選択できます。

ファイルサイズで切り替える方法

出力先ファイルのファイルサイズが一定のサイズになったタイミングで，ログの出力先を次のファイルに切り替える方法です。ファイルサイズは資料ごとに指定できます。ただし，一部の資料ではファイルサイズは固定です。

時刻とファイルサイズで切り替える方法

指定した時刻になったタイミング，または出力先のファイルサイズが一定のサイズになったタイミングで，ログの出力先を次のファイルに切り替える方法です。

切り替え方法は，プロセス単位で設定できます。

時刻とファイルサイズで切り替える方法を選択した場合に，時刻だけで切り替えるようにしたいときは，ファイルサイズとして，想定しているログの出力量よりも大きい値を指定してください。また，時刻で出力先ファイルが切り替わった場合に作成される新しいファイルのヘッダ情報は，ログを出力するプロセスがログの初期化を行った日時になります。

なお，時刻とファイルサイズで切り替える方法の対象になるファイルについては，「5.2　アプリケーションサーバのログ」を参照してください。

注意事項

時刻とファイルサイズで切り替える方法を選択した場合，OS のリソースの負荷状況によって，出力先ファイルを切り替えるタイミングが，指定した時刻よりも数ミリ秒～数秒遅れることがあります。

また，指定した時刻にログ出力を行うプロセス（J2EE サーバなど）が起動していない場合，指定した時刻を過ぎたあとの次回起動時に切り替えは行わず，既存ファイルに追加書きします。例えば，切り替える時刻を 12:00:00 に指定し，11:00:00 にプロセスを停止した場合，13:00:00 に起動したプロセスは，既存のファイルにログを追加書きします。

(2)　出力先ファイル切り替え時のファイル名付与規則

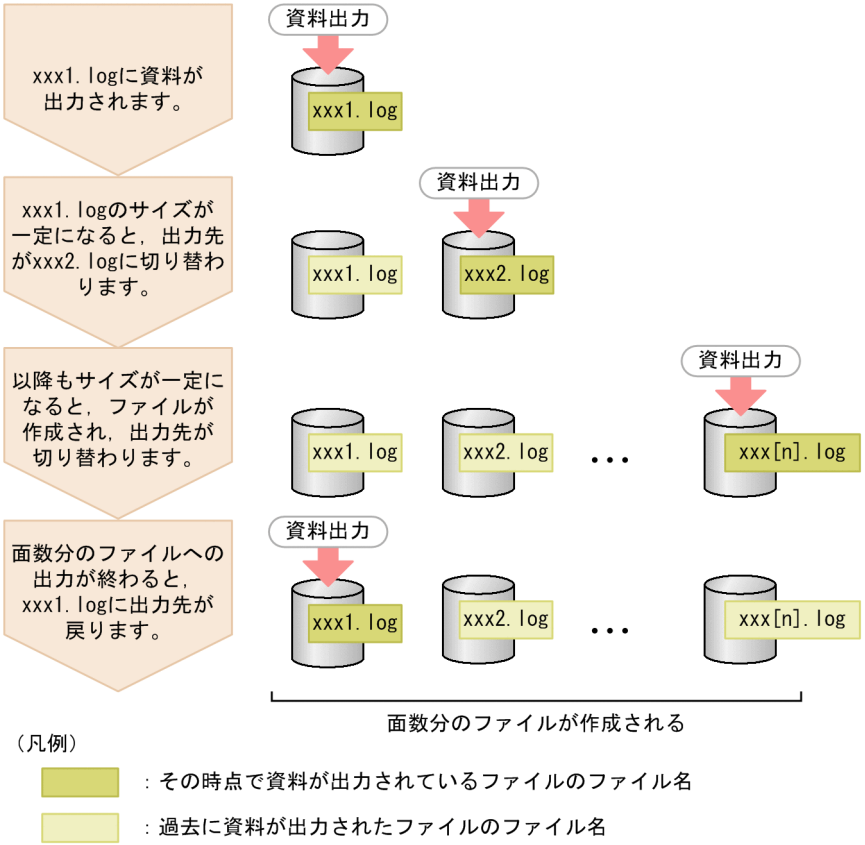
出力先ファイルには，資料ごとに固有の文字列と通番で構成されたファイル名が設定されます。通番の付与規則は，次の 2 種類のモードから選択できます。

ラップアラウンドモード

資料固有の文字列が xxx の場合，xxx1.log，xxx2.log…のように通番が付与されたファイルが作成されます。通番の範囲は，1～（出力先ファイル面数）です。出力先ファイル面数分のファイルが作成されたあとで出力先変更が発生すると，資料の出力先が通番 1 のファイルに切り替わります。

ラップアラウンドモードの場合のファイル名付与規則を次の図に示します。

図 3-1　ラップアラウンドモードの場合のファイル名付与規則



シフトモード

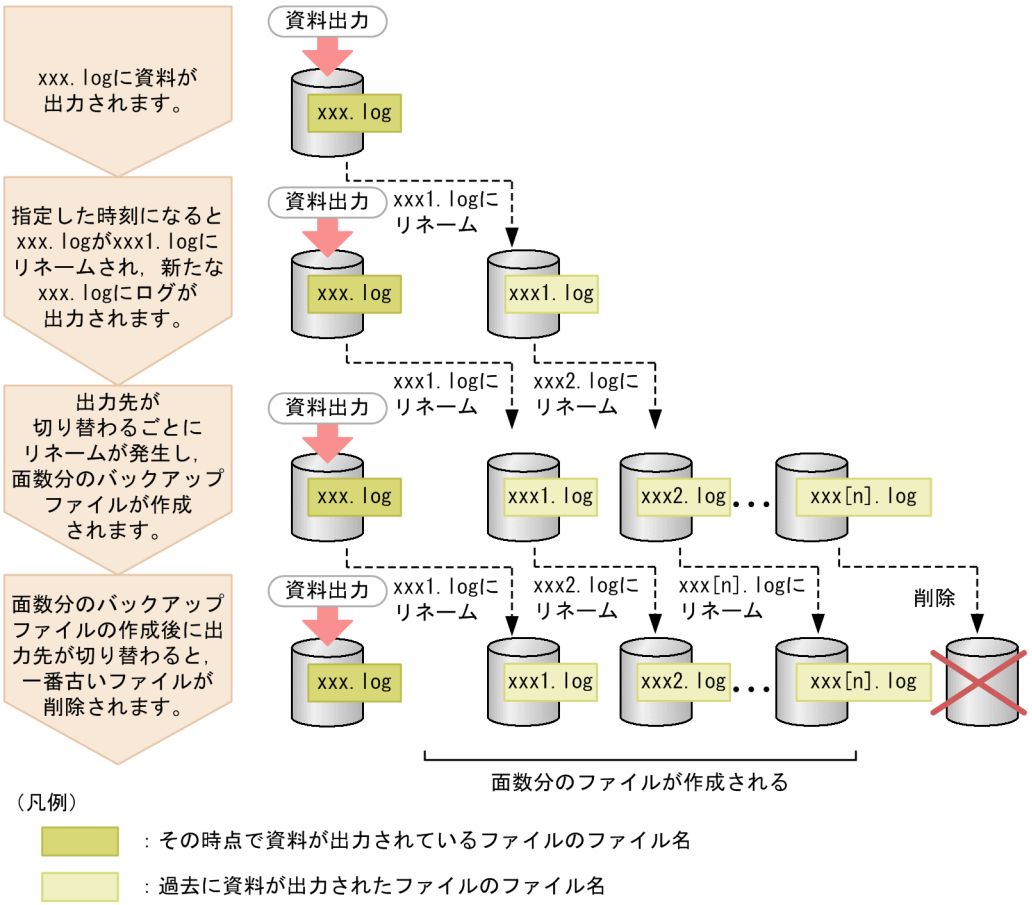
その時点で資料が出力されているファイルは，資料固有の文字列だけのファイル名になります。通番は付与されません。指定した時刻に出力先が切り替わったあとで，過去のファイルに対して通番が付与されます。

資料固有の文字列が xxx の場合，その時点で資料が出力されるファイルは，xxx.log（通番なしのファイル名）になります。過去の出力先ファイル名は，xxx1.log，xxx2.log…のように通番が付与されたファイル名に変更（リネーム）されます。このとき，通番が小さいファイルから順に，新しいファイルとなります。

通番の範囲は，1～（出力先ファイル面数）です。このため，ファイルの合計数は，指定した出力先ファイル面数＋ 1（現在の出力先ファイル分）となります。

シフトモードの場合の出力先切り替えの流れを次の図に示します。

図 3-2　シフトモードの場合の出力先切り替えの流れ



シフトモードの対象になるファイルについては，「5.2　アプリケーションサーバのログ」を参照してください。

なお，Windows の場合，ファイルが削除される際に，ほかのプロセスが使用中のファイルが一時的に退避されることがあります。この場合，元のファイル名の末尾に「#removed#[n]」（n は 0 以上の整数）が付与されたファイルが一時的に作成されます。このファイルは，すべてのプロセスからのそのファイルに対する処理が完了したあとで，自動的に削除されます。

3.2.2　資料取得の設定の概要（J2EE アプリケーションを実行するシステム）

資料のデフォルトの設定（ログの出力先やサイズ）を変更する場合や，デフォルトで取得できない資料を取得する場合には，簡易構築定義ファイルやユーザ定義ファイルを編集して設定します。ここでは，J2EE アプリケーションを実行するシステムで，事前に資料（ログ）を取得するための設定が必要かどうかについて説明します。また，設定方法の概要についても説明します。

資料の種類ごとに資料取得のために設定が必要なもの，デフォルトの資料取得の設定を変更する場合だけ設定が必要なものに分けて，次の表に示します。

- **資料取得のために設定が必要な資料**

表 3-3　資料取得のための設定（J2EE アプリケーションを実行するシステム）

資料の種類	資料取得のための設定	参照先マニュアル	参照個所
アプリケーションのユーザログ	簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ロガーやハンドラの設定，ログの出力レベル，サイズ，面数などを設定します。また，server.policy で，セキュリティポリシーを設定します。	アプリケーションサーバ 機能解説 拡張編	8 章
OS の統計情報	Windows の場合，Windows のシステムモニタでシステムリソースのパフォーマンスデータ取得の設定をします。	このマニュアル	3.3.14
ユーザダンプ	Windows の場合，タスクマネージャ，Windows のデバッグツール，または環境変数（CJMEMDUMP_PATH）で，ユーザダンプ取得の設定をします。	このマニュアル	3.3.15
core ダンプ	UNIX の場合，簡易構築定義ファイルやシェルコマンドで，core ファイル取得の設定をします。	このマニュアル	3.3.16
JavaVM の資料	簡易構築定義ファイルで，JavaVM のスレッドダンプや JavaVM ログ（JavaVM ログファイル）の出力方法や出力内容などの設定をします。また，明示管理ヒープ機能のイベントログを出力するファイル名や，ログの出力レベルなどを設定します。	このマニュアル	3.3.17

- デフォルトの資料取得の設定を変更する場合だけ設定が必要な資料

表 3-4　資料取得のための設定（J2EE アプリケーションを実行するシステム）

資料の種類	資料取得のための設定	参照先マニュアル	参照個所
snapshot ログ	snapshot ログの収集先，収集方法や収集のタイミングを変更する場合には，ユーザ定義ファイルを編集します。	このマニュアル	3.3.1，3.3.3
Management Server のログ	mserver.properties（Management Server 環境設定ファイル）で，ログの出力レベルや，ログファイルの面数などを設定します。	このマニュアル	3.3.5
性能解析トレースファイル	mserver.properties（Management Server 環境設定ファイル）で，性能解析トレースファイルの面数を設定します。 また，簡易構築定義ファイルで，論理パフォーマンストレーサ（performance-tracer）の<configuration>タグ内に，パフォーマンストレーサのトレース取得レベルや，PFR トレースファイルの面数などを指定します。	このマニュアル	7.5
J2EE サーバのログ	簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ログの出力レベル，サイズ，面数などを設定します。 また，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejbserver.logger.systemlog.enabled で，システムログ出力の設定が有効になっている場合，J2EE サーバの起動，停止および異常終了のメッセージがイベントログ（UNIX の場合，syslog）に出力されます。	このマニュアル	3.3.6

資料の種類	資料取得のための設定	参照先マニュアル	参照個所
Web サーバのログ	簡易構築定義ファイルで，論理 Web サーバ（web-server）の<configuration>タグ内に，Web サーバのログの出力レベル，出力先などを設定します。	このマニュアル	3.3.8
NIO HTTP サーバのログ	簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，NIO HTTP サーバのログ出力の有無，ファイル面数などを設定します。また，アクセスログは，フォーマットを定義することで，出力形式をカスタマイズできます。 NIO HTTP サーバのアクセスログのカスタマイズについては，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「7.11.2　NIO HTTP サーバのアクセスログのカスタマイズ」を参照してください。	このマニュアル	3.3.9
稼働情報ファイル	簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，稼働情報ファイルの出力先や面数などを設定します。	アプリケーションサーバ 機能解説 運用／監視／連携編	3.3.3
Manager のログ	manager.cfg で統合ログの面数やサイズを指定します。	このマニュアル	3.3.10
コンソールログ	adminagent.properties でコンソールログの出力の有無，面数やサイズを設定します。	アプリケーションサーバ 機能解説 運用／監視／連携編	11.3
リソースアダプタのログ	サーバ管理コマンドを使用して，リソースアダプタ単位でのログ出力の有無を設定します。 また，簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ログの出力レベル，サイズ，面数を設定します。	このマニュアル	3.3.11
TPBroker のトレースファイル	簡易構築定義ファイル（論理 J2EE サーバ（j2ee-server）の<configuration>タグ内），サーバ管理コマンド用の usrconf.bat（UNIX の場合は usrconf）と usrconf.properties で，トレースファイルの出力先や面数などを設定します。	このマニュアル	3.3.12
CJMS プロバイダのログ	commonconfig.properties または config.properties と，admin.properties で，CJMSP ブローカーおよび管理コマンド（cjmsicmd）のログの出力レベル，面数，ログファイルサイズを設定します。 また，Connector 属性ファイルで，CJMSP リソースアダプタのログの出力レベル，面数，ログファイルサイズを設定します。	このマニュアル	3.3.13
サーバ管理コマンドのログ	サーバ管理コマンド用の usrconf.bat（UNIX の場合は usrconf）と usrconf.properties で，ログの出力レベルなどを設定できます。	アプリケーションサーバ アプリケーション設定操作ガイド	3.4

ここに示したログは，snapshot ログで一括収集できます。ただし，TPBroker のトレースファイルについては，収集できるものと収集できないものが混在しています。このほかにも，snapshot ログのデフォルトの設定で取得できないログについては，取得のための設定をしたり，snapshot ログの収集先に追加したりする必要があります。また，ユーザダンプ（Windows の場合）または core ダンプ（UNIX の場合）については，固定のファイル名に対して収集します。障害発生時に収集するためには，ユーザ作成の障害検知時コマンドを利用する必要があります。

なお，次のログはログ出力先を変更できません。

- セットアップ時に作成される install.log，移行コマンドなどのログファイル
- 次のディレクトリに出力される Java のスレッドダンプファイル
  - Windows の場合
    <作業ディレクトリ>¥ejb¥<サーバ名称>
  - UNIX の場合
    <作業ディレクトリ>/ejb/<サーバ名称>

ログの種類やデフォルト値，チャネル名，取得できるログの詳細や取得方法については，「2.4　取得が必要な資料の種類」を参照してください。

3.2.3　資料取得の設定の概要（バッチアプリケーションを実行するシステム）

資料のデフォルトの設定（ログの出力先やサイズ）を変更する場合や，デフォルトで取得できない資料を取得する場合には，簡易構築定義ファイルやユーザ定義ファイルを編集して設定します。ここでは，バッチアプリケーションを実行するシステムで，事前に資料（ログ）を取得するための設定が必要かどうかについて説明します。また，設定方法の概要についても説明します。

資料の種類ごとに資料取得のために設定が必要なもの，デフォルトの資料取得の設定を変更する場合だけ設定が必要なものに分けて，次の表に示します。

- **資料取得のために設定が必要な資料**

表 3-5　資料取得のための設定（バッチアプリケーションを実行するシステム）

資料の種類	資料取得のための設定	参照先マニュアル	参照個所
アプリケーションのユーザログ	簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ロガーやハンドラの設定，ログの出力レベル，サイズ，面数などを設定します。また，server.policy で，セキュリティポリシーを設定します。	アプリケーションサーバ 機能解説 拡張編	8 章
OS の統計情報	Windows の場合，Windows のシステムモニタでシステムリソースのパフォーマンスデータ取得の設定をします。	このマニュアル	3.3.14※
ユーザダンプ	Windows の場合，Windows のデバッグツールで，ユーザダンプ取得の設定をします。	このマニュアル	3.3.15※

資料の種類	資料取得のための設定	参照先マニュアル	参照個所
core ダンプ	UNIX の場合，簡易構築定義ファイルやシェルコマンドで，core ファイル取得の設定をします。	このマニュアル	3.3.16※
JavaVM の資料	簡易構築定義ファイルで，JavaVM のスレッドダンプや JavaVM ログ（JavaVM ログファイル）の出力方法や出力内容などの設定をします。また，明示管理ヒープ機能のイベントログを出力するファイル名や，ログの出力レベルなどを設定します。	このマニュアル	3.3.17※

注※

J2EE サーバの場合と設定方法に差異はありません。参照先の記述の「J2EE サーバ」を「バッチサーバ」に置き換えてお読みください。

- デフォルトの資料取得の設定を変更する場合だけ設定が必要な資料

表 3-6　資料取得のための設定（バッチアプリケーションを実行するシステム）

資料の種類	資料取得のための設定	参照先マニュアル	参照個所
snapshot ログ	snapshot ログの収集先，収集方法や収集のタイミングを変更する場合には，ユーザ定義ファイルを編集します。	このマニュアル	3.3.2，3.3.4
Management Server のログ	mserver.properties（Management Server 環境設定ファイル）で，ログの出力レベルや，ログファイルの面数などを設定します。	このマニュアル	3.3.5※
性能解析トレースファイル	mserver.properties（Management Server 環境設定ファイル）で，性能解析トレースファイルの面数を設定します。 また，簡易構築定義ファイルで，論理パフォーマンストレーサ（performance-tracer）の<configuration>タグ内に，パフォーマンストレーサのトレース取得レベルや，PFR トレースファイルの面数などを指定します。	このマニュアル	7.5※
バッチサーバのログ	簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ログの出力レベル，サイズ，面数などを設定します。 また，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejbserver.logger.systemlog.enabled で，システムログ出力の設定が有効になっている場合，バッチサーバの起動，停止および異常終了のメッセージがイベントログ（UNIX の場合，syslog）に出力されます。	このマニュアル	3.3.7
稼働情報ファイル	簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，稼働情報ファイルの出力先や面数などを設定します。	アプリケーションサーバ 機能解説 運用／監視／連携編	3.3.3
Manager のログ	manager.cfg で統合ログの面数やサイズを指定します。	このマニュアル	3.3.10※
コンソールログ	adminagent.properties でコンソールログの出力の有無，面数やサイズを設定します。	アプリケーションサーバ 機能解説 運用／監視／連携編	11.3※
リソースアダプタのログ	サーバ管理コマンドを使用して，リソースアダプタ単位でのログ出力の有無を設定します。	このマニュアル	3.3.11※

資料の種類	資料取得のための設定	参照先マニュアル	参照個所
	また，簡易構築定義ファイルで，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ログの出力レベル，サイズ，面数を設定します。		
TPBroker のトレースファイル	簡易構築定義ファイル（論理 J2EE サーバ（j2ee-server）の<configuration>タグ内），サーバ管理コマンド用の usrconf.bat（UNIX の場合は usrconf）と usrconf.properties で，トレースファイルの出力先や面数などを設定します。	このマニュアル	3.3.12※
サーバ管理コマンドのログ	サーバ管理コマンド用の usrconf.bat（UNIX の場合は usrconf）と usrconf.properties で，ログの出力レベルなどを設定できます。	アプリケーションサーバ アプリケーション設定操作ガイド	3.4

注※

J2EE サーバの場合と設定方法に差異はありません。参照先の記述の「J2EE サーバ」を「バッチサーバ」に置き換えてお読みください。

ここに示したログは，snapshot ログで一括収集できます。ただし，TPBroker のトレースファイルについては，収集できるものと収集できないものが混在しています。このほかにも，snapshot ログのデフォルトの設定で取得できないログについては，取得のための設定をしたり，snapshot ログの収集先に追加したりする必要があります。また，ユーザダンプ（Windows の場合）または core ダンプ（UNIX の場合）については，固定のファイル名に対して収集します。障害発生時に収集するためには，ユーザ作成の障害検知時コマンドを利用する必要があります。

なお，次のログはログ出力先を変更できません。

- セットアップ時に作成される install.log，移行コマンドなどのログファイル
- 次のディレクトリに出力される Java のスレッドダンプファイル
  - Windows の場合
    <作業ディレクトリ>¥ejb¥<サーバ名称>
  - UNIX の場合
    <作業ディレクトリ>/ejb/<サーバ名称>

ログの種類やデフォルト値，チャネル名，取得できるログの詳細や取得方法については，「2.4　取得が必要な資料の種類」を参照してください。

3.3　実行環境での設定

トラブルシューティングに必要な資料のうち，運用を開始する前に設定が必要な資料や，デフォルトの設定を変更できる資料があります。この節では，各資料の設定方法について説明します。なお，事前設定または設定変更の対象となる資料については，「3.2　資料取得の設定の概要」を参照してください。

3.3.1　障害検知時コマンドによる資料取得の設定（J2EE アプリケーションを実行するシステム）

ここでは，トラブルシューティングの資料を障害検知時コマンドで取得するための設定について説明します。障害検知時コマンドで取得した資料は snapshot ログとして収集できます。

障害検知時コマンドには，システムで提供しているコマンドとユーザが作成したコマンドの 2 種類があります。デフォルトの設定では，論理サーバにトラブルが発生した場合にシステム提供の障害検知時コマンドが実行されて，スレッドダンプや性能解析トレースなどが取得されます。また，トラブルが発生した論理サーバの停止前に snapshot ログが収集されます。システム提供の障害検知時コマンドで取得できる情報については，「2.3.2(1)　システムで提供されている障害検知時コマンドで取得できる情報」を参照してください。

システム提供の障害検知時コマンドの動作設定を変更する場合には，「3.3.1(1)　Management Server での環境設定」および「3.3.1(2)　運用管理エージェントでの環境設定」が必要です。また，ユーザ作成の障害検知時コマンドを利用する場合は，「3.3.1(1)　Management Server での環境設定」，「3.3.1(2)　運用管理エージェントでの環境設定」および「3.3.1(3)　ユーザ作成の障害検知時コマンドのコマンドファイル作成」が必要です。それぞれの設定内容について，(1)～(3)で説明します。

注意事項

ユーザ作成の障害検知時コマンドで取得した資料を snapshot ログとして収集するためには，その資料の取得先を snapshot ログの収集先に追加する必要があります。snapshot ログの収集先の追加については，「3.3.3(3)　snapshot ログの収集先のカスタマイズ」を参照してください。

(1)　Management Server での環境設定

mserver.properties（Management Server 環境設定ファイル）で，障害検知時コマンドの動作を設定します。

次のキーで，障害検知時コマンドの動作を設定します。

キー	説明	設定の要否	
		システム	ユーザ
com.cosminexus.mngsvr.sys_cmd.abnormal_end.enabled	システム提供の障害検知時コマンドを利用するかどうかを指定します。デフォルトは，true（利用する）です。	△	−
com.cosminexus.mngsvr.usr_cmd.abnormal_end.enabled	ユーザ作成の障害検知時コマンドを利用するかどうかを指定します。デフォルトは，false（利用しない）です。	−	○
com.cosminexus.mngsvr.sys_cmd.abnormal_end.timeout	システム提供の障害検知時コマンドの終了を待つ時間を指定します。指定時間を経過してもコマンドが終了しない場合は，ユーザ回復処理を続行します。	△	−
com.cosminexus.mngsvr.usr_cmd.abnormal_end.timeout	ユーザ作成の障害検知時コマンドの終了を待つ時間を指定します。	−	△
com.cosminexus.mngsvr.snapshot.auto_collect.enabled	障害発生時，または一括再起動時に snapshot ログを取得するかどうかを指定します。デフォルトは，true（snapshot ログを取得する）です。	△	△
com.cosminexus.mngsvr.snapshot.collect.point	snapshot ログの収集タイミングとして，次のどちらかを指定します。 • 論理サーバの停止前 • J2EE サーバの再起動前 デフォルトは，論理サーバの停止前です。	△	△

（凡例）

システム：システム提供の障害検知時コマンドでの設定の要否を示す

ユーザ：ユーザ作成の障害検知時コマンドでの設定の要否を示す

○：設定は必要

△：デフォルトの設定を変更する場合だけ必要

−：設定は不要

(2)　運用管理エージェントでの環境設定

adminagent.properties（運用管理エージェントプロパティファイル）で，障害検知時コマンドで取得する資料を設定します。

adminagent.properties の次のキーで，障害検知時コマンドで取得する資料の数や取得の有無，障害検知時コマンドのパスなどを設定します。なお，snapshot ログの収集先を定義するための snapshot ログ収集対象定義ファイルについては，「3.3.3(3)　snapshot ログの収集先のカスタマイズ」を参照してください。

キー	説明	設定の要否	
		システム	ユーザ
adminagent.snapshotlog.num_snapshots	一次送付資料として収集する，論理サーバごとの snapshot ログファイルの数を指定します。	△	△

キー	説明	設定の要否	
		システム	ユーザ
adminagent.snapshotlog.listfile.2.num_snapshots	二次送付資料として収集する，論理サーバごとの snapshot ログファイルの数を指定します。	△	△
adminagent.j2ee.sys_cmd.abnormal_end.threaddump	システム提供の障害検知時コマンドで，スレッドダンプを取得するかどうかを指定します。	△	−
adminagent.sys_cmd.abnormal_end.prftrace	システム提供の障害検知時コマンドで，性能解析トレースファイルを取得するかどうかを指定します。	△	−
adminagent.<論理サーバの種類>.usr_cmd.abnormal_end	論理サーバの種類ごとに，実行する障害検知時コマンドのパスを指定します。	−	○

（凡例）

システム：システム提供の障害検知時コマンドでの設定の要否を示す

ユーザ：ユーザ作成の障害検知時コマンドでの設定の要否を示す

○：設定は必要

△：デフォルトの設定を変更する場合だけ必要

−：設定は不要

(3) ユーザ作成の障害検知時コマンドのコマンドファイル作成

ユーザ作成の障害検知時コマンドは，コマンドファイル（バッチファイルまたはシェルスクリプトファイル）に記述できます。このとき，次の表に示す環境変数をコマンドファイルに記述しておくことで，障害の発生した論理サーバの情報や障害に関する情報を利用してコマンドを実行できます。

表 3-7 ユーザ作成の障害検知時コマンドのコマンドファイルに記述できる環境変数

環境変数	説明
COSMI_MNG_LSNAME	障害の発生した論理サーバの論理サーバ名。なお，論理 CTM 内のネーミングサービスで障害が発生した場合は，論理 CTM の論理サーバ名が設定されます。
COSMI_MNG_RSNAME	障害の発生した論理サーバの実サーバ名。J2EE サーバ以外の論理サーバの場合，論理サーバ名が設定されます。
COSMI_MNG_LSPID	論理サーバ起動時に監視対象となったプロセス ID。間接起動の論理ユーザサーバで複数のプロセス ID が監視対象である場合，論理ユーザサーバ起動時に実行されるプロセス ID 取得用コマンドで取得された順に，コンマ（,）で区切って設定されます。
COSMI_MNG_LSARGS	論理サーバを起動したときのコマンドライン。
COSMI_MNG_TIME_SUSPENDED	ハングアップを検知した時刻。万国標準時（UTC）の 1970 年 1 月 1 日午前 0 時からの経過時間（単位：ms）。なお，無応答を検知しない場合，値は設定されません。

環境変数	説明
COSMI_MNG_TIME_TERMINATED	異常停止（プロセスダウン）を検知した時刻。万国標準時（UTC）の 1970 年 1 月 1 日午前 0 時からの経過時間（単位：ms）。なお，ハングアップの場合，値は設定されません。
COSMI_MNG_WEB_SYSTEM	障害が発生した論理サーバが所属する Web システム。Smart Composer 機能を使用しない場合，値は設定不要です。
COSMI_MNG_TIER	障害が発生した論理サーバが所属する物理ティア。Smart Composer 機能を使用しない場合，値は設定不要です。
COSMI_MNG_UNIT	障害が発生した論理サーバが所属するサービスユニット。Smart Composer 機能を使用しない場合，値は設定不要です。
COSMI_MNG_HWS	HTTP Server のインストールディレクトリ。

なお，障害検知コマンドとして実行するコマンドから出力される標準出力，標準エラー出力は，Management Server では取得しません。このため，コマンドの標準出力，標準エラー出力を取得する場合は，コマンドの中でファイルに出力する必要があります。

(a) core ダンプの取得例

J2EE サーバの障害発生時に kill コマンドを実行して core ダンプを取得する場合の例を次に示します。

- UNIX の場合

```
#!/bin/sh

# 障害がプロセスダウンかハングアップかを，プロセスダウンを検知した時刻から判断する。
if [ "$COSMI_MNG_TIME_TERMINATED" = "" ] ; then

# 障害がハングアップであるため，coreダンプを取得する。
/bin/kill -6 $COSMI_MNG_LSPID
fi
```

(b) スレッドダンプの取得例

Web サーバの障害発生時に，cjdumpsv コマンドを実行して J2EE サーバ（実サーバ名：J2EEServer）のスレッドダンプを取得する場合の例を次に示します。

この例では，時間の経過に応じた各スレッドの状態遷移を確認するため，複数回 cjdumpsv コマンドを実行しています。cjdumpsv コマンドは，目安として 3 秒おきに 10 回程度実行します。

- Windows の場合

```
rem 障害がプロセスダウンかハングアップかを環境変数から判断する。
if defined COSMI_MNG_TIME_TERMINATED goto END

rem 障害がハングアップであるため，スレッドダンプを取得する。
set COUNT=10
set INTERVAL=3000
for /l %%n in (1,1,%COUNT%) do (
"C:¥Program Files¥Hitachi¥Cosminexus¥CC¥server¥bin¥cjdumpsv.exe" J2EEServer
```

```
    if not "%%n" == "%COUNT%" (
        rem 次のスレッドダンプ取得まで待機する。(ミリ秒)
        echo WScript.sleep %INTERVAL% > sleep.vbs
        "C:¥WINDOWS¥¥system32¥cscript.exe" sleep.vbs > NUL
        del sleep.vbs
    )
)
:END
```

- UNIX の場合

```
#!/bin/sh

# 障害がプロセスダウンかハングアップかを環境変数から判断する。
if [ "$COSMI_MNG_TIME_TERMINATED" = "" ] ; then

# 障害がハングアップであるため，スレッドダンプを取得する。
COUNT=10
INTERVAL=3
for num in `seq $COUNT`
do
  /opt/Cosminexus/CC/server/bin/cjdumpsv J2EEServer
  if [ "$num" -ne  "$COUNT" ]; then
      # 次のスレッドダンプ取得まで待機する。(秒)
      sleep $INTERVAL
  fi
done
fi
```

(c) ユーザ作成の障害検知時コマンドの動作

論理 CTM では，グローバル CORBA ネーミングサービスと CTM デーモンの二つのプロセスが起動，停止，監視されます。そのため，論理 CTM 内のグローバル CORBA ネーミングサービスの障害を検知した場合と CTM デーモンの障害を検知した場合とでは，実行するコマンドが異なります。

- グローバル CORBA ネーミングサービスの障害を検出した場合

  adminagent.naming.usr_cmd.abnormal_end キーで指定したコマンドが実行されます。

- CTM デーモンの障害を検出した場合

  adminagent.ctm.usr_cmd.abnormal_end キーで指定したコマンドが実行されます。

また，論理 CTM 内の二つのプロセス（CTM デーモン，グローバル CORBA ネーミングサービス）のどちらのプロセスで障害を検知しても，Management Server のログには，論理サーバ（CTM）の障害検知時コマンドの開始を通知するログが出力されます。

3.3.2 障害検知時コマンドによる資料取得の設定（バッチアプリケーションを実行するシステム）

ここでは，トラブルシューティングの資料を障害検知時コマンドで取得するための設定について説明します。障害検知時コマンドで取得した資料は snapshot ログとして収集できます。

障害検知時コマンドには，システムで提供しているコマンドとユーザが作成したコマンドの 2 種類があります。デフォルトの設定では，論理サーバにトラブルが発生した場合にシステム提供の障害検知時コマンドが実行されて，スレッドダンプや性能解析トレースなどが取得されます。また，トラブルが発生した論理サーバの停止前に snapshot ログが収集されます。システム提供の障害検知時コマンドで取得できる情報については，「2.3.2(1)　システムで提供されている障害検知時コマンドで取得できる情報」を参照してください。

システム提供の障害検知時コマンドの動作設定を変更する場合には，Management Server および運用管理エージェントでの環境設定が必要です。また，ユーザ作成の障害検知時コマンドを利用する場合は，Management Server および運用管理エージェントでの環境設定と，ユーザ作成の障害検知時コマンドのコマンドファイル作成が必要です。それぞれの設定内容については，「3.3.1　障害検知時コマンドによる資料取得の設定（J2EE アプリケーションを実行するシステム）」を参照してください。その際に，「J2EE サーバ」を「バッチサーバ」に置き換えてお読みください。

注意事項

ユーザ作成の障害検知時コマンドで取得した資料を snapshot ログとして収集するためには，その資料の取得先を snapshot ログの収集先に追加する必要があります。snapshot ログの収集先の追加については，「3.3.3(3)　snapshot ログの収集先のカスタマイズ」を参照してください。

3.3.3　snapshot ログ収集の設定（J2EE アプリケーションを実行するシステム）

ここでは，snapshot ログ収集のための設定について説明します。snapshot ログとして収集するファイルの設定や，収集した snapshot ログの格納先などの設定を変更できます。

(1)　snapshot ログを収集するタイミングごとに収集できる資料

snapshot ログの収集方法を変更するための設定を収集タイミングごとに次の表に示します。

表 3-8　snapshot ログ収集を変更するための設定（J2EE アプリケーションを実行するシステムの場合）

分類	収集のタイミング	デフォルトの設定を変更するために必要な設定
自動的に収集する※	論理サーバが障害時に自動停止される直前	• 障害検知時コマンドの設定 • snapshot ログの収集先のカスタマイズ • snapshot ログ収集のタイムアウトの設定
	J2EE サーバが障害時に自動再起動される直前	
	J2EE サーバを手動で一括再起動する直前	• snapshot ログの収集先のカスタマイズ

分類	収集のタイミング	デフォルトの設定を変更するために必要な設定
任意のタイミングで収集する	Management Server の運用管理コマンド（mngsvrutil）で snapshot ログの収集を実行したとき	

注※　論理サーバの停止前，または J2EE サーバの再起動前のどちらのタイミングで snapshot ログを収集するかは，mserver.properties の com.cosminexus.mngsvr.snapshot.collect.point キーで変更できます。デフォルトの設定では，論理サーバの停止前に snapshot ログが収集されます。

次に，タイミングごとに収集できる資料について説明します。

(a)　自動的に収集する場合

次に示すどちらかのタイミングで snapshot ログを自動的に収集する場合には，Management Server によって障害検知時コマンドが実行されて，スレッドダンプと性能解析トレースなどの資料が取得されます。

- 論理サーバが障害時に自動停止される直前（デフォルトの設定）
- J2EE サーバが障害時に自動再起動される直前

障害検知時コマンドで取得された資料は，snapshot ログとして収集できます。障害検知時コマンドの動作や設定の変更については，「3.3.1　障害検知時コマンドによる資料取得の設定（J2EE アプリケーションを実行するシステム）」を参照してください。

(b)　任意のタイミングで収集する場合

任意のタイミングで snapshot ログを収集する場合には，スレッドダンプファイルやユーザダンプ（Windows の場合）または core ダンプ（UNIX の場合）が出力されているときだけ，snapshot ログとして収集できます。任意のタイミングで snapshot ログを収集する際に，コマンドが実行されてスレッドダンプファイルやユーザダンプまたは core ダンプが出力されることはありません。

Management Server の運用管理コマンド（mngsvrutil）を実行して任意のタイミングで snapshot ログを収集する場合は，コマンド実行時に収集先の種別（種別 1，種別 2）を指定して収集できます。それ以外のタイミングでは，種別 1 と種別 2 それぞれの収集対象として定義されているファイルがすべて収集されます。

(2)　snapshot ログで収集できるファイル

トラブルシューティングに必要な資料は，資料を保守員へ送付するときのタイミングによって，一次送付資料と二次送付資料に分類されます。snapshot ログでは，一次送付資料と二次送付資料を収集できます。各資料については，「2.3.3(2)　snapshot ログとして収集できる資料」を参照してください。

一次送付資料としてどのファイルを収集するかは，一次送付資料用の snapshot ログ収集対象定義ファイル（snapshotlog.conf）で指定します。mngsvrutil コマンドで引数 snapshot 1 を指定して snapshot ログを収集する場合は，snapshotlog.conf に指定されているファイルが収集されます。二次送付資料としてどのファイルを収集するかは，二次送付資料用の snapshot ログ収集対象定義ファイル

（snapshotlog.2.conf）で指定します。mngsvrutil コマンドで引数 snapshot 2 を指定して snapshot ログを収集する場合は，snapshotlog.conf と snapshotlog.2.conf に指定されているファイルが収集されます。

デフォルトで snapshot ログの収集先として定義されていないファイルを snapshot ログとして収集したい場合には，そのファイルの出力先を snapshot ログ収集対象定義ファイルに追加してください。また，資料によっては，次のような設定も必要になります。

- **運用開始前の資料取得のための設定**

  運用開始前に，資料取得のための設定をしておく必要があります。

- **コマンドなどによる資料の取得**

  snapshot ログとして収集する前に，コマンドを実行して収集対象の資料を取得しておく必要があります。資料の取得については，「4.　トラブルシューティングで必要な資料の出力先と出力方法」を参照してください。

（例）

- Windows の場合

  ユーザダンプは，運用開始前に環境変数（CJMEMDUMP_PATH）の設定が必要です。また，環境変数（CJMEMDUMP_PATH）を設定した場合には snapshot ログの収集先を変更する必要があります。

- UNIX の場合

  core ダンプは，運用開始前に core ファイルのサイズの設定が必要です。運用開始前に取得のための設定をした上で，ユーザ作成の障害検知時コマンドを使用して障害発生時の core ダンプを取得し，その core ダンプを snapshot ログとして収集することをお勧めします。また，デフォルトの snapshot ログ収集対象定義ファイルには，core ダンプのデフォルトの出力先が指定されています。

運用開始前に実施する取得のための設定については，「3.3.15　ユーザダンプ取得の設定」，または「3.3.16　core ダンプ取得の設定」を参照してください。ユーザ作成の障害検知時コマンドについては，「3.3.1　障害検知時コマンドによる資料取得の設定（J2EE アプリケーションを実行するシステム）」を参照してください。

snapshotlog.conf および snapshotlog.2.conf の指定については，「3.3.3(3)　snapshot ログの収集先のカスタマイズ」を参照してください。デフォルトの設定で一次送付資料および二次送付資料として収集できる資料については，「付録 A　snapshot ログの収集対象一覧」を参照してください。

(3)　snapshot ログの収集先のカスタマイズ

snapshot ログの収集先は，snapshot ログ収集対象定義ファイルでカスタマイズできます。また，adminagent.properties では，論理サーバごとの snapshot ログのファイル数を指定できます。ファイルの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「10.2.1 snapshot ログ収集対象定義ファイル」を参照してください。

(a) snapshot ログの収集先の指定

snapshot ログ収集対象定義ファイルを編集して，snapshot ログの収集先を指定してください。snapshot ログ収集対象定義ファイルの格納場所を次に示します。

- Windows の場合
  <製品のインストールディレクトリ>¥manager¥config¥snapshotlog.conf
  <製品のインストールディレクトリ>¥manager¥config¥snapshotlog.2.conf
- UNIX の場合
  /opt/Cosminexus/manager/config/snapshotlog.conf
  /opt/Cosminexus/manager/config/snapshotlog.2.conf

snapshotlog.conf では，一次送付資料として収集するファイルの格納ディレクトリを指定します。また，snapshotlog.2.conf では，二次送付資料として収集するファイルの格納ディレクトリを指定します。

snapshot ログ収集対象定義ファイルでは，収集対象のパスに変数を使用できます。例えば，製品のインストールディレクトリを表す変数「${cosminexus.home}」を使用して，snapshot ログ収集対象定義ファイルに「${cosminexus.home}/manager/log/.+」（ピリオド（.）は任意の文字，プラス（+）は 1 回以上を表します）と指定すると，<製品のインストールディレクトリ>¥manager¥log（Windows の場合），または/opt/Cosminexus/manager/log ディレクトリ（UNIX の場合）直下のファイルがすべて収集されます。なお，snapshot ログ収集対象定義ファイルの変数の値には，ドル記号（$）を含めないでください。

(b) snapshot ログのファイル数の指定

論理サーバごとの snapshot ログファイルの数は，adminagent.properties の次のキーで変更できます。デフォルトの設定は 10 です。adminagent.properties については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.1　adminagent.properties（運用管理エージェントプロパティファイル)」を参照してください。

- adminagent.snapshotlog.num_snapshots
  一次送付資料として収集する，論理サーバごとの snapshot ログファイルの数を指定します。
- adminagent.snapshotlog.listfile.2.num_snapshots
  二次送付資料として収集する，論理サーバごとの snapshot ログファイルの数を指定します。

(c) snapshot ログの格納先の指定

snapshot ログの収集によって自動で取得された情報の格納先は，adminagent.properties の adminagent.snapshotlog.log_dir キーで変更できます。デフォルトの設定では，次のディレクトリに格納されます。

- Windows の場合
  <Manager のログ出力ディレクトリ>¥snapshot
- UNIX の場合

<Manager のログ出力ディレクトリ>/snapshot

adminagent.properties については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.1　adminagent.properties（運用管理エージェントプロパティファイル)」を参照してください。

(4)　snapshot ログ収集のタイムアウトの設定

snapshot ログ収集時に実行する処理に対してタイムアウトを設定できます。タイムアウトは，mserver.properties で設定します。snapshot ログ収集時に実行する処理とタイムアウトの設定を次の表に示します。

表 3-9　snapshot ログ収集時に実行する処理とタイムアウトの設定

処理	タイムアウトの設定 (mserver.properties のキー)	説明
システム提供の障害検知時コマンドの実行	com.cosminexus.mngsvr.sys_cmd.abnormal_end.timeout	次に示すそれぞれの処理の終了を待つ時間です。 • システム提供の障害検知時に実行したコマンド • 性能解析トレースの収集 指定した時間を経過してもコマンド，または性能解析トレースの収集が終了しない場合は，実行したコマンド，または性能解析トレースの収集を無視して処理を続行します。
ユーザ作成の障害検知時コマンドの実行	com.cosminexus.mngsvr.usr_cmd.abnormal_end.timeout	ユーザ作成の障害検知時に実行したコマンドの終了を待つ時間です。指定した時間を経過してもコマンドが終了しない場合は，実行したコマンドを無視して処理を続行します。
snapshot ログ（一次送付資料）の収集	com.cosminexus.mngsvr.snapshot.auto_collect.timeout	一次送付資料および二次送付資料の収集の終了を待つ時間です。指定した時間を経過しても収集が終了しない場合は，Management Server から運用管理エージェントに snapshot ログ収集中止のサービス要求が実行され，Management Server のログに KEOS20052-E メッセージが出力されます。
snapshot ログ（二次送付資料）の収集		

mserver.properties については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.6　mserver.properties（Management Server 環境設定ファイル)」を参照してください。

3.3.4 snapshot ログ収集の設定（バッチアプリケーションを実行するシステム）

ここでは，バッチサーバを使用している場合の，snapshot ログ収集のための設定について説明します。snapshot ログとして収集するファイルの設定や，収集した snapshot ログの格納先などの設定を変更できます。snapshot ログの収集方法を変更するための設定を収集タイミングごとに次の表に示します。なお，このほかの説明については，J2EE サーバの場合と同じです。詳細は，「3.3.3 snapshot ログ収集の設定（J2EE アプリケーションを実行するシステム）」を参照してください。

表 3-10 snapshot ログ収集を変更するための設定（バッチアプリケーションを実行するシステムの場合）

分類	収集のタイミング	デフォルトの設定を変更するために必要な設定
自動的に収集する※	論理サーバが障害時に自動停止される直前	• 障害検知時コマンドの設定 • snapshot ログの収集先のカスタマイズ • snapshot ログ収集のタイムアウトの設定
	バッチサーバが障害時に自動再起動される直前	
	バッチサーバを手動で一括再起動する直前	• snapshot ログの収集先のカスタマイズ
任意のタイミングで収集する	Management Server の運用管理コマンド（mngsvrutil）で snapshot ログの収集を実行したとき	

注※ 論理サーバの停止前，またはバッチサーバの再起動前のどちらのタイミングで snapshot ログを収集するかは，mserver.properties の com.cosminexus.mngsvr.snapshot.collect.point キーで変更できます。デフォルトの設定では，論理サーバの停止前に snapshot ログが収集されます。

3.3.5 Management Server のログ取得の設定

ここでは，Management Server が出力するログを取得するための設定について説明します。

Management Server のログの出力先ディレクトリを次に示します。Management Server の出力先は manager.cfg（Manager ログ設定ファイル）で変更できます。

- Windows の場合
  <製品のインストールディレクトリ>¥manager¥log
- UNIX の場合
  /opt/Cosminexus/manager/log

Management Server のログの出力レベルや，ログファイルの面数などを変更する場合は，mserver.properties（Management Server 環境設定ファイル）で，次のキーを指定します。

- com.cosminexus.mngsvr.log.level

Management Server のログの出力レベルを指定します。

- com.cosminexus.mngsvr.log.rotate
  Management Server のログのファイル面数を指定します。
- com.cosminexus.mngsvr.log.size
  Management Server のログのファイルサイズを指定します。

mserver.properties および各キーの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.6　mserver.properties（Management Server 環境設定ファイル）」を参照してください。

注意事項

UNIX の場合，Manager のログ出力ディレクトリは，アクセス権に 777（rwxrwxrwx）が設定されて，自動的に作成されます。あらかじめ手動で作成したディレクトリをログ出力先に指定する場合は，ディレクトリのアクセス権に 777（rwxrwxrwx）を設定してください。ログ出力先に指定したディレクトリのアクセス権に 777（rwxrwxrwx）が設定されていない場合，コマンド実行時にログが出力されないことがあります。

3.3.6　J2EE サーバのログ取得の設定

ここでは，J2EE サーバのログ取得で設定できる項目について説明します。

J2EE サーバのログは，ログの出力先，ログサイズ，ログレベル，ログの出力先ファイルの切り替え方法およびログの出力先ファイルの切り替え時刻を変更できます。変更できる項目と，項目に対応する簡易構築定義ファイルのパラメタを次の表に示します。

表 3-11　J2EE サーバのログ取得の設定項目

項目	対応する簡易構築定義ファイルのパラメタ
ログの出力先	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejb.server.log.directory
ログサイズ	**ログファイル面数** 論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.channels.define.<チャネル名>.filenum **ログファイル 1 面当たりの最大サイズ** 論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.channels.define.<チャネル名>.filesize
ログレベル	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.enabled.*
ログの出力先ファイルの切り替え方法	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.rotationStyle

項目	対応する簡易構築定義ファイルのパラメタ
ログの出力先ファイルの切り替え時刻	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.rotationTime

なお，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejbserver.logger.systemlog.enabled パラメタで true（デフォルト値）を指定した場合，またはこのパラメタの指定を省略した場合は，J2EE サーバの起動，停止および異常終了のメッセージがイベントログ（UNIX の場合は，syslog）に出力されます。

注意事項（UNIX の場合）

J2EE サーバの起動，停止および異常終了のメッセージを syslog に出力するためには，syslog の設定で，facility「daemon」に対する priority を「info」または「debug」に設定する必要があります。また，syslog のログ出力先およびログファイル名は，syslog の設定に依存します。

syslog および syslog の設定に関する詳細については，OS 付属のマニュアルで，syslogd または syslog.conf の説明を参照してください。

簡易構築定義ファイルに指定するパラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

(1)　ログの出力先の変更

J2EE サーバのログの出力先を変更する場合は，簡易構築定義ファイルでログの出力先ディレクトリを指定します。

ログの出力先の変更

デフォルトのログ出力先を次に示します。

- Windows の場合
  <作業ディレクトリ>¥ejb¥<サーバ名>¥logs
- UNIX の場合
  <作業ディレクトリ>/ejb/<サーバ名>/logs

なお，作業ディレクトリのデフォルトは，<製品のインストールディレクトリ>¥CC¥server¥public（Windows の場合），または/opt/Cosminexus/CC/server/public（UNIX の場合）です。

作業ディレクトリおよび J2EE サーバのログの出力先は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，次のパラメタを指定すると変更できます。

- ejb.public.directory
  J2EE サーバの作業ディレクトリのパスを指定します。
- ejb.server.log.directory
  J2EE サーバのログの出力先ディレクトリを指定します。

設定例（物理ティアの定義の場合）

- Windows の場合

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejb.server.log.directory</param-name>
    <param-value>C:¥CClogs¥server¥MyServer</param-value>
  </param>
   :
</configuration>
```

- UNIX の場合

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejb.server.log.directory</param-name>
    <param-value>/CClogs/server/MyServer</param-value>
  </param>
   :
</configuration>
```

カレントディレクトリ

ログ出力先を相対パスで指定する場合のカレントディレクトリを次に示します。

- Windows の場合

  <作業ディレクトリ>¥ejb¥<サーバ名称>

- UNIX の場合

  <作業ディレクトリ>/ejb/<サーバ名称>

注意事項

- ログの出力先を変更した場合は，J2EE サーバを起動する前に，変更後のログの出力先ディレクトリを作成しておいてください。

  変更後のログの出力先ディレクトリがない場合は，J2EE サーバの起動時に KDJE40024-E のメッセージが出力されて異常終了します。また，サーバ管理コマンドの実行時に KDJE37209-E，KDJE37210-E，KDJE37211-E のメッセージが出力されて異常終了します。

- 同一ホスト内で複数の J2EE サーバを起動させている場合は，ログ出力先が同じディレクトリにならないように，ディレクトリにサーバ名称を含めるなど，サーバごとにユニークなディレクトリ名になるようにしてください。なお，パラメタの値に同じディレクトリを指定した場合は，動作の保証はしません。

- ログの出力先を変更して作業ディレクトリ以外にログを出力する場合，ログファイルはサーバのアンセットアップ時に削除されません。ログファイルを削除したい場合には，手動で削除してください。

- 簡易構築定義ファイルでログの出力先を設定していても，JavaVM 起動パラメタで JavaVM の保守情報および GC のログ出力先が設定されていると，JavaVM 起動パラメタの設定が優先されますので，注意してください。

  JavaVM 起動パラメタは，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の <configuration>タグ内に定義します。JavaVM 起動パラメタの指定内容を次に示します。

<param-name>タグ

add.jvm.arg

<param-value>タグ

XX:HitachiJavaLog:<JavaVM の保守情報および GC のログ出力先>

JavaVM 起動パラメタを指定している場合は，JavaVM の保守情報および GC のログ出力先に設定したディレクトリに，JavaVM の保守情報および GC のログファイルが出力されます。

- ログ出力先には，UNC 名を含むパスは指定できません。

(2) ログサイズの変更

J2EE サーバのログサイズを変更する場合は，簡易構築定義ファイルで，ログファイルの面数，およびログファイル 1 面当たりの最大サイズを設定します。

ログファイル面数の変更

J2EE サーバのログファイルの面数は，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejbserver.logger.channels.define.<チャネル名>.filenum パラメタで指定します。

設定例（物理ティアの定義の場合）

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejbserver.logger.channels.define.MessageLogFile.filenum</param-name>
    <param-value>3</param-value>
  </param>
   :
</configuration>
```

ログファイル 1 面当たりの最大サイズの変更

J2EE サーバのログファイル 1 面当たりの最大サイズ（単位：バイト）は，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejbserver.logger.channels.define.<チャネル名>.filesize パラメタで指定します。

設定例（物理ティアの定義の場合）

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejbserver.logger.channels.define.MessageLogFile.filesize</param-name>
    <param-value>2097152</param-value>
  </param>
   :
</configuration>
```

(3) ログレベルの変更

J2EE サーバのログレベルは，ログの重要度を表します。ログレベルには，「Error」，「Warning」，「Information」，「Debug」の四つがあります。ログレベルを設定すると，設定したレベルのログが出力さ

れます。デフォルトでは，Error レベルのログだけが取得されます。通常はデフォルトのまま利用してください。

ログレベルは，簡易構築定義ファイルの論理 J2EE サーバの<configuration>タグ内に，ejbserver.logger.enabled.*パラメタで設定します。ejbserver.logger.enabled.*の<param-value>タグにはレベル名を，「Error」，「Warning」，「Information」，「Debug」の文字列で一つ，または複数設定します。複数設定する場合には，レベル名の文字列の間をコンマ（,）で区切ります。

設定例（物理ティアの定義の場合）

1.

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejbserver.logger.enabled.*</param-name>
    <param-value>Error</param-value>
  </param>
   :
</configuration>
```

2.

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejbserver.logger.enabled.*</param-name>
    <param-value>Error,Warning</param-value>
  </param>
   :
</configuration>
```

3.

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejbserver.logger.enabled.*</param-name>
    <param-value>Error,Warning,Information</param-value>
  </param>
   :
</configuration>
```

4.

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejbserver.logger.enabled.*</param-name>
    <param-value>Error,Warning,Information,Debug</param-value>
  </param>
   :
</configuration>
```

注意事項

- 記述例の 1.，2.，3.，4.の順に，取得できるログの件数が増加していきます。複数のログレベルを設定してログを取得すると，性能が劣化し，ログファイルの面の切り替えが頻繁に起こるようになります。
- レベル名に「Error」，「Warning」，「Information」，「Debug」以外の文字列，または空の値を設定した場合は，KDJE90009-W のメッセージが出力されます。Error レベルのログは取得されます。

ログレベルの推奨設定

ログレベルの推奨設定を次に示します。

- 通常運用時
  レベル名に「Error」を指定します。
- 通常運用時（verbose）
  通常運用時よりも詳細な情報を取得する場合には，レベル名に「Error,Warning」を指定します。
- テスト時
  レベル名に「Error,Warning,Information」を指定します。
- 障害調査時
  レベル名に「Error,Warning,Information,Debug」を指定します。

3.3.7　バッチサーバのログ取得の設定

バッチサーバのログは，ログの出力先，ログサイズ，ログレベルを変更できます。変更できる項目と，項目に対応する簡易構築定義ファイルのパラメタを次の表に示します。なお，これらのパラメタは，簡易構築定義ファイルの，バッチサーバ用のユーザプロパティに設定します。

表 3-12　バッチサーバのログ取得の設定項目

項目	対応する簡易構築定義ファイルのパラメタ
ログの出力先	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejb.server.log.directory
ログサイズ	**ログファイル面数** 論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.channels.define.<チャネル名>.filenum **ログファイル 1 面当たりの最大サイズ** 論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.channels.define.<チャネル名>.filesize
ログレベル	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.enabled.*

なお，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejbserver.logger.systemlog.enabled パラメタで true（デフォルト値）を指定した場合，またはこのパラ

メタの指定を省略した場合は，バッチサーバの起動，停止および異常終了のメッセージがイベントログ（UNIX の場合は，syslog）に出力されます。

それぞれの設定項目の詳細については，「3.3.6　J2EE サーバのログ取得の設定」を参照してください。その際に，「J2EE サーバ」を「バッチサーバ」に置き換えてお読みください。

また，簡易構築定義ファイルに指定するパラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

注意事項（UNIX の場合）

バッチサーバの起動，停止および異常終了のメッセージを syslog に出力するためには，syslog の設定で，facility「daemon」に対する priority を「info」または「debug」に設定する必要があります。また，syslog のログ出力先およびログファイル名は，syslog の設定に依存します。

syslog および syslog の設定に関する詳細については，OS 付属のマニュアルで，syslogd または syslog.conf の説明を参照してください。

3.3.8　Web サーバのログ取得の設定

ここでは，Web サーバのログ取得で設定できる項目について説明します。ログ取得の設定対象となる Web サーバは，HTTP Server です。

Web サーバが出力するログには，エラーログ，アクセスログ，リクエストログなどがあります。出力するログの詳細については，マニュアル「HTTP Server」を参照してください。Web サーバの出力するログのうち，エラーログ，アクセスログおよびリクエストログは，出力先やログの出力方式などを簡易構築定義ファイルで変更できます。

Web サーバのログ取得で変更できる設定項目と，項目に対応する簡易構築定義ファイルのパラメタを次の表に示します。

表 3-13　Web サーバのログ取得の設定項目

ログ	項目	対応する簡易構築定義ファイルのパラメタ
エラーログ	出力するエラーログのレベル	論理 Web サーバ（web-server）の<configuration>タグ内の LogLevel
	エラーログの出力方式	論理 Web サーバ（web-server）の<configuration>タグ内の HttpsdErrorMethod
	エラーログの出力先ディレクトリ	論理 Web サーバ（web-server）の<configuration>タグ内の HttpsdErrorLogFileDir
アクセスログ	アクセスログの出力方式	論理 Web サーバ（web-server）の<configuration>タグ内の HttpsdCustomMethod
	アクセスログの出力先ディレクトリ	論理 Web サーバ（web-server）の<configuration>タグ内の HttpsdCustomLogFileDir

ログ	項目	対応する簡易構築定義ファイルのパラメタ
	アクセスログ出力時のフォーマット	論理 Web サーバ（web-server）の<configuration>タグ内の HttpsdCustomlogFormat
リクエストログ	トレースの採取有無	論理 Web サーバ（web-server）の<configuration>タグ内の HWSRequestLogLevel
	リクエストログの出力方式	論理 Web サーバ（web-server）の<configuration>タグ内の HttpsdRequestMethod
	リクエストログの出力先ディレクトリ	論理 Web サーバ（web-server）の<configuration>タグ内の HttpsdRequestLogFileDir

また，エラーログ，アクセスログおよびリクエストログに出力する時刻ならびに時間の単位を，論理 Web サーバ（web-server）の<configuration>タグ内の HWSLogTimeVerbose パラメタで指定できます。

注意事項

Management Server を利用して Web サーバの動作確認をする場合に，動作確認用のログを通常のログ（アクセスログ）と別に出力するときには，簡易構築定義ファイルでの設定が必要です。論理 Web サーバ（web-server）の<configuration>タグ内で SetBy パラメタに「item」を指定して，AppendDirectives パラメタと HttpsdCustomlogFormat パラメタを設定してください。

AppendDirectives パラメタと HttpsdCustomlogFormat パラメタの設定例を次に示します。ここでは，動作確認用のログをラップアラウンド方式で取得する場合を例にして説明します。ログの出力方式に応じて，AppendDirectives パラメタ内の CustomLog ディレクティブの記述を変更してください。

AppendDirectives パラメタの設定例

Windows の場合

```
<param>
  <param-name>AppendDirectives</param-name>
  <param-value>
<![CDATA[
SetEnvIf Remote_Addr ^127¥.0¥.0¥.1$ Env_ManagerHealthCheck
CustomLog "|¥"¥"<製品のインストールディレクトリ>/httpsd/sbin/rotatelogs2.exe¥" ¥"<製品のインストールディレクトリ>/httpsd/servers/HWS_<論理Webサーバの実サーバ名>/logs/access_manager¥" 8192 5¥"" hws_std env=Env_ManagerHealthCheck
]]>
  </param-value>
</param>
```

UNIX の場合

```
<param>
  <param-name>AppendDirectives</param-name>
  <param-value>
<![CDATA[
SetEnvIf Remote_Addr ^127¥.0¥.0¥.1$ Env_ManagerHealthCheck
CustomLog "|/opt/hitachi/httpsd/sbin/rotatelogs2 ¥"/opt/hitachi/httpsd/servers/HWS_<論理Webサーバの実サーバ名>/logs/access_manager¥" 8192 5" hws_std env=Env_ManagerHealthCheck
]]>
```

```
  </param-value>
</param>
```

HttpsdCustomlogFormat パラメタの設定例

```
<param>
  <param-name>HttpsdCustomlogFormat</param-name>
  <param-value>hws_std env=!Env_ManagerHealthCheck</param-value>
</param>
```

3.3.9　NIO HTTP サーバのログ取得の設定

ここでは，NIO HTTP サーバのログ取得で設定できる項目について説明します。

NIO HTTP サーバのログ取得で変更できる設定項目と，項目に対応する簡易構築定義ファイルのパラメタを次の表に示します。

表 3-14　NIO HTTP サーバのログ取得の設定項目

ログおよびトレース	項目	対応する簡易構築定義ファイルのパラメタ
アクセスログ	アクセスログのフォーマットの形式	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.access_log.nio_http.format※
	アクセスログのファイルサイズ	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filesize
	アクセスログのファイル面数	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filenum
性能解析トレース	－	ほかの性能解析トレースと同様に，日常的なシステム運用の作業で，cprfed コマンドを実行するときに取得条件などを指定します。性能解析トレースファイルの取得については，「7. 性能解析トレースを使用した性能解析」を参照してください。

（凡例）－：該当しない

注※　アクセスログでは，これらのキーでフォーマットを定義することで，ログの出力形式をカスタマイズできます。NIO HTTP サーバのアクセスログのカスタマイズについては，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「7.11.2　NIO HTTP サーバのアクセスログのカスタマイズ」を参照してください。

3.3.10　Manager のログ取得の設定

運用管理エージェント，運用監視エージェント，および Management Server のログには，個別に取得する以外に，**統合ログ**としてまとめて取得できるものがあります。統合ログとして取得できるのは，統合メッセージログファイル，統合トレースログファイル，コマンド保守ログファイルです。なお，統合ログとし

て取得できるログの詳細については，「4.3　アプリケーションサーバのログ（J2EE アプリケーションを実行するシステム）」を参照してください。

ここでは，統合ログの設定の変更について説明します。統合ログは，manager.cfg で設定します。manager.cfg の格納場所を次に示します。

- Windows の場合
  <製品のインストールディレクトリ>¥manager¥config¥manager.cfg
- UNIX の場合
  /opt/Cosminexus/manager/config/manager.cfg

統合ログの設定を変更する場合には，manager.cfg で次のキーを設定してください。

- com.cosminexus.manager.log.dir
  統合ログの出力先を指定します。なお，デフォルトの設定では，次の場所に出力されます。
  - Windows の場合
    <製品のインストールディレクトリ>¥manager¥log
  - UNIX の場合
    /opt/Cosminexus/manager/log
- com.cosminexus.manager.messagelog.size
  統合メッセージログファイルの 1 ファイル当たりの最大サイズを指定します。
- com.cosminexus.manager.messagelog.fnum
  統合メッセージログファイルの面数を指定します。
- com.cosminexus.manager.messagelog.style
  統合メッセージログファイルの出力先の切り替え方法を指定します。SHIFT（シフトモード）を指定した場合，com.cosminexus.manager.messagelog.time が有効になります。
- com.cosminexus.manager.messagelog.time
  統合メッセージログファイルの出力先を切り替える時刻を指定します。
- com.cosminexus.manager.tracelog.size
  統合トレースログファイルの 1 ファイル当たりの最大サイズを指定します。
- com.cosminexus.manager.tracelog.fnum
  統合トレースログファイルの面数を指定します。
- com.cosminexus.manager.tracelog.style
  統合トレースログファイルの出力先の切り替え方法を指定します。SHIFT（シフトモード）を指定した場合，com.cosminexus.manager.tracelog.time が有効になります。
- com.cosminexus.manager.tracelog.time
  統合トレースログファイルの出力先を切り替える時刻を指定します。
- com.cosminexus.manager.cmdtracelog.size

コマンド保守ログファイルの 1 ファイル当たりの最大サイズを指定します。

- com.cosminexus.manager.cmdtracelog.fnum
  コマンド保守ログファイルの面数を指定します。
- com.cosminexus.manager.log.compatible
  運用管理エージェント，および Management Server のログを個別で出力するかどうかを指定します。デフォルトの設定では，統合ログと同時に個別のログも出力されます。個別のログを出力しない場合には，false を指定してください。

3.3.11　リソースアダプタのログ取得の設定

ここでは，リソースアダプタログを取得するための設定について説明します。リソースアダプタのログを取得するために必要な設定は次の二つです。

- **ログ出力の有無の設定**
  サーバ管理コマンドで，リソースアダプタ単位でのログ出力の有無を設定します。cjgetrarprop コマンドで Connector 属性ファイルを取得し，<property>タグで LogEnabled に true を指定します。ファイル編集後に，cjsetrarprop コマンドで編集内容を反映させます。なお，リソースアダプタをデプロイする前にプロパティを定義する場合は，cjgetresprop コマンドと cjsetresprop コマンドを使用してください。
  Connector 属性ファイルについては，マニュアル「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)」の「4.1　Connector 属性ファイル」を参照してください。コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。サーバ管理コマンドの操作については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「3.　サーバ管理コマンドの基本操作」を参照してください。
- **ログのサイズ，面数，ログレベルの設定**
  簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，次のパラメタで，リソースアダプタのログのサイズ，面数を設定します。
  - ejbserver.connector.logwriter.filesize
    リソースアダプタのログファイル 1 面当たりの最大サイズ（単位：バイト）を指定します。
  - ejbserver.connector.logwriter.filenum
    リソースアダプタのログファイルの面数を指定します。

  また，J2EE サーバのログの出力レベル（ログレベル）がリソースアダプタのログレベルになります。J2EE サーバのログレベルの設定については，「3.3.6(3)　ログレベルの変更」を参照してください。

注意事項

- CJMSP リソースアダプタの場合，ログサイズ，面数，ログレベルの設定方法が，ほかのリソースアダプタと異なります。詳細は，「3.3.13　CJMS プロバイダのログ取得の設定」を参照してください。

- リソースアダプタのログ出力先ディレクトリのパスの長さと，リソースアダプタ表示名の長さを加えた長さが OS のパス長の制限を超えた場合，リソースアダプタのログの初期化に失敗し，KDJE90002-E メッセージを出力後サーバが停止します。これは，ログファイルが別のプロセスでロックされている場合などに発生します。

  次のどちらかの手段で，リソースアダプタのログが最大パス長を超えないようにしてください。詳細は，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

  ・リソースアダプタ表示名の長さを変更

  ・ログ出力先ディレクトリ，または作業ディレクトリを変更

  また，ログファイルを別のプロセスでロックしないでください。
- リソースアダプタのログの初期化に失敗した場合，KDJE90002-E メッセージを出力したあと，J2EE サーバが停止します。これは，リソースアダプタのログ出力先ディレクトリのパスの長さと，リソースアダプタ表示名の長さを加えた長さが OS のパス長の制限を超えた場合や，ログファイルがほかのプロセスでロックされている場合などに発生します。

  リソースアダプタ表示名の長さを変更するか，またはログ出力先ディレクトリもしくは作業ディレクトリを変更して，リソースアダプタのログが最大パス長を超えないようにしてください。また，ほかのプロセスでログファイルをロックしないようにしてください。

3.3.12 TPBroker のログ取得の設定

ここでは，TPBroker のトレースファイルの出力先，ファイル数，エントリ数の変更について説明します。

TPBroker のトレースファイルのデフォルトの出力先は次のとおりです。

J2EE サーバのトレース情報の場合

- Windows の場合
  <作業ディレクトリ>¥ejb¥<サーバ名称>¥logs¥TPB¥logj
- UNIX の場合
  <作業ディレクトリ>/ejb/<サーバ名称>/logs/TPB/logj

サーバ管理コマンドのトレース情報の場合

- Windows の場合
  <製品のインストールディレクトリ>¥CC¥admin¥logs¥TPB¥logj
- UNIX の場合
  /opt/Cosminexus/CC/admin/logs/TPB/logj

EJB クライアントアプリケーションのトレース情報の場合

- Windows の場合
  <EJB クライアントログ出力ディレクトリ>※¥system¥TPB¥logj

- UNIX の場合

  <EJB クライアントログ出力ディレクトリ>※/system/TPB/logj

注※　EJB クライアントログ出力ディレクトリの出力先については，「4.5.2　EJB クライアントアプリケーションのシステムログの出力先」を参照してください。

CTM のトレース情報の場合

- Windows の場合

  <製品のインストールディレクトリ>¥TPB¥log

  <製品のインストールディレクトリ>¥TPB¥logj
- UNIX の場合

  /opt/Cosminexus/TPB/log

  /opt/Cosminexus/TPB/logj

バッチアプリケーションで使用するコマンド

- Windows の場合

  <製品のインストールディレクトリ>¥TPB¥log
- UNIX の場合

  /opt/Cosminexus/TPB/log

変更できる項目と，項目に対応する簡易構築定義ファイルのパラメタまたはユーザ定義ファイルとキーを次の表に示します。

表 3-15　TPBroker のログ取得の設定項目

項目	分類	対応する簡易構築定義ファイルのパラメタまたはユーザ定義ファイルとキー
トレースファイルの出力先	J2EE サーバ	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejb.server.log.directory または 論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の vbroker.orb.htc.tracePath
	サーバ管理コマンド	usrconf.bat（Windows の場合），または usrconf（UNIX の場合）の USRCONF_JVM_ARGS キーの-Dejbserver.log.directory オプション または サーバ管理コマンド用の usrconf.properties の vbroker.orb.htc.tracePath キー※
	EJB クライアントアプリケーション	• cjclstartap コマンドの場合，Java アプリケーション用の usrconf.properties の vbroker.orb.htc.tracePath キー • vbj コマンドの場合，vbj コマンドで指定する JavaVM のシステムプロパティの vbroker.orb.htc.tracePath キー

項目	分類	対応する簡易構築定義ファイルのパラメタまたはユーザ定義ファイルとキー
	CTM	• グローバル CORBA ネーミングサービスは，adminagent.xml（運用管理エージェント設定ファイル）の論理サーバ種類がネーミングサービスの個所で，環境変数 HVI_TRACEPATH を設定。 • CTM ドメインマネジャは，論理 CTM ドメインマネジャ（ctm-domain-manager）の<configuration>タグ内の user.env.variable で，環境変数 HVI_TRACEPATH を設定。 • CTM は，論理 CTM（component-transaction-monitor）の<configuration>タグ内の user.env.variable で，環境変数 HVI_TRACEPATH を設定。
	バッチアプリケーションで使用するコマンド	環境変数 HVI_TRACEPATH
トレースファイルの面数	J2EE サーバ	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の vbroker.orb.htc.comt.fileCount
	EJB クライアントアプリケーション	• cjclstartap コマンドの場合，Java アプリケーション用の usrconf.propertiesvbroker.orb.htc.comt.fileCount キー • vbj コマンドの場合，vbj コマンドで指定する JavaVM のシステムプロパティの vbroker.orb.htc.comt.fileCount キー
	CTM	• グローバル CORBA ネーミングサービスは，adminagent.xml（運用管理エージェント設定ファイル）の論理サーバ種類がネーミングサービスの個所で，環境変数 HVI_COMTFILECOUNT を設定。 • CTM ドメインマネジャは，論理 CTM ドメインマネジャ（ctm-domain-manager）の<configuration>タグ内の user.env.variable で，環境変数 HVI_COMTFILECOUNT を設定。 • CTM は，論理 CTM（component-transaction-monitor）の<configuration>タグ内の user.env.variable で，環境変数 HVI_COMTFILECOUNT を設定。
トレースファイルのエントリ数	J2EE サーバ	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の vbroker.orb.htc.comt.entryCount
	EJB クライアントアプリケーション	• cjclstartap コマンドの場合，Java アプリケーション用の usrconf.propertiesvbroker.orb.htc.comt.entryCount キー • vbj コマンドの場合，vbj コマンドで指定する JavaVM のシステムプロパティの vbroker.orb.htc.comt.entryCount キー
	CTM	• グローバル CORBA ネーミングサービスは，adminagent.xml（運用管理エージェント設定ファイル）の論理サーバ種類がネーミングサービスの個所で，環境変数 HVI_COMTENTRYCOUNT を設定。

項目	分類	対応する簡易構築定義ファイルのパラメタまたはユーザ定義ファイルとキー
		• CTM ドメインマネジャは，論理 CTM ドメインマネジャ（ctm-domain-manager）の<configuration>タグ内の user.env.variable で，環境変数 HVI_COMTENTRYCOUNT を設定。 • CTM は，論理 CTM（component-transaction-monitor）の<configuration>タグ内の user.env.variable で，環境変数 HVI_COMTENTRYCOUNT を設定。
	バッチアプリケーションで使用するコマンド	環境変数 HVI_COMTENTRYCOUNT

注※　サーバ管理コマンド用の usrconf.bat ファイル（Windows の場合）または usrconf ファイル（UNIX の場合），および usrconf.properties については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「3.3　サーバ管理コマンドの動作設定のカスタマイズ」を参照してください。サーバ管理コマンド用の各ファイルおよびキーの詳細については，次の個所を参照してください。

- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「5.1　サーバ管理コマンドで使用するファイルの一覧」
- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「5.2.1　usrconf（サーバ管理コマンド用オプション定義ファイル)」
- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「5.2.2　usrconf.bat（サーバ管理コマンド用オプション定義ファイル)」
- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「5.2.3　usrconf.properties（サーバ管理コマンド用システムプロパティファイル)」

簡易構築定義ファイルの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

Java アプリケーション用の usrconf.properties ファイルとキーの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「12.2.2　usrconf.properties（Java アプリケーション用ユーザプロパティファイル)」を参照してください。

vbj コマンドで指定する JavaVM のシステムプロパティの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「12.2.3　Java アプリケーションに指定するシステムプロパティ」

adminagent.xml（運用管理エージェント設定ファイル）の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.4　adminagent.xml（運用管理エージェント設定ファイル)」を参照してください。

通信トレースの詳細については，マニュアル「TPBroker 運用ガイド」を参照してください。

注意事項

- トレースファイルの出力先を変更する場合は，変更後のトレースファイル出力先ディレクトリのサブディレクトリとして comtrc と mdltrc をあらかじめ作成しておく必要があります。出力先を変更すると，変更後のログ出力先ディレクトリ下の comtrc と mdltrc にトレースファイルが出力されます。
- 簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内で，<param-name>に ejb.server.log.directory と，vbroker.orb.htc.tracePath の両方を設定した場合は，vbroker.orb.htc.tracePath の設定が優先されます。
- usrconf.bat（Windows の場合），または usrconf（UNIX の場合）の USRCONF_JVM_ARGS キーの-Dejbserver.log.directory オプションと，サーバ管理コマンド用の usrconf.properties の

vbroker.orb.htc.tracePath キーをどちらも設定した場合は，サーバ管理コマンド用の usrconf.properties の vbroker.orb.htc.tracePath キーの設定が優先されます。

- Management Server リモート管理機能から操作した場合，サーバ管理コマンドのログ出力先は変更できません。
- CTM ドメインマネジャと CTM でトレースファイルの面数やエントリ数を設定すると，CTM ドメインマネジャや CTM デーモンだけでなく，CTM レギュレータや ctmstart コマンドなどのプロセスにも有効となります。トレースファイルの面数やエントリ数を大きくする場合は，ディスク使用量の増加に注意してください。
- 通信トレースファイルのトレースファイルのエントリ数を大きくする場合は，メモリ使用量の増加に注意してください。

3.3.13 CJMS プロバイダのログ取得の設定

ここでは，CJMS プロバイダで使用する，CJMSP ブローカー，管理コマンド（cjmsicmd）および CJMSP リソースアダプタのログの出力レベル，面数，ログファイルサイズなどを変更するための設定について説明します。

CJMS プロバイダでは，次の 3 種類のログを出力します。

- CJMSP ブローカーのログ
- 管理コマンド（cjmsicmd）のログ
- CJMSP リソースアダプタのログ

それぞれのデフォルトの出力先を次の表に示します。

表 3-16 CJMS プロバイダのログのデフォルトの出力先

ログの種類	デフォルトの出力先
CJMSP ブローカーのログ	Windows の場合 <CJMSP_HOME>※1¥var¥instances¥<instanceName>¥log UNIX の場合 <CJMSP_HOME>※1/var/instances/<instanceName>/log
管理コマンド（cjmsicmd）のログ	Windows の場合 <CJMSP_HOME>※1¥var¥admin¥log UNIX の場合 <CJMSP_HOME>※1/var/admin/log
CJMSP リソースアダプタのログ	Windows の場合 <J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)> ※2¥cjms¥Cosminexus_JMS_Provider_RA

ログの種類	デフォルトの出力先
	UNIX の場合 <J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)> ※2/cjms/Cosminexus_JMS_Provider_RA

注※1

<CJMSP_HOME>は，次のディレクトリです。

Windows の場合

<製品のインストールディレクトリ>¥CC¥cjmsp

UNIX の場合

/opt/Cosminexus/CC/cjmsp

注※2

<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>は，J2EE サーバのオプション定義で指定したディレクトリです。デフォルトでは，次のディレクトリになります。

Windows の場合

<ejb.public.directory で指定したディレクトリ>¥ejb¥<J2EE サーバ名>¥logs

UNIX の場合

<ejb.public.directory で指定したディレクトリ>/ejb/<J2EE サーバ名>/logs

なお，デフォルトの出力先がない場合，ログ出力時にディレクトリが作成されます。

(1) CJMSP ブローカーのログ取得の設定

CJMSP ブローカーでは，ログ取得の設定のうち，ログの出力レベル，面数，ファイルサイズを変更できます。

変更方法について次の表に示します。

表 3-17 CJMSP ブローカーのログ取得の設定の変更方法

項目	変更方法
ログの出力レベル	commonconfig.properties または config.properties の次のプロパティで指定します。 • broker.logger.MessageLogFile.trace.level
面数	commonconfig.properties または config.properties の次のプロパティで指定します。 • broker.logger.MessageLogFile.filenum • broker.logger.ExceptionLogFile.filenum
ファイルサイズ	commonconfig.properties または config.properties の次のプロパティで指定します。 • broker.logger.MessageLogFile.filesize • broker.logger.ExceptionLogFile.filesize

注

CJMSP ブローカーでは，cjmsbroker コマンドの-varhome オプションでログの出力先を設定できます。ただし，<CJMSP_HOME>¥var ディレクトリ（Windows の場合）または<CJMSP_HOME>/var ディレクトリ（UNIX の場合）以外のディレクトリについては，ログの出力先の変更はできません。

なお，指定したディレクトリがない場合，デフォルトの設定が有効になります。

各プロパティの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「6.2.2　commonconfig.properties（CJMSP ブローカー共通プロパティファイル)」，およびマニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「6.2.3　config.properties（CJMSP ブローカー個別プロパティファイル)」を参照してください。

(2)　管理コマンド（cjmsicmd）のログ取得の設定

管理コマンド（cjmsicmd）では，ログ取得の設定のうち，ログの出力レベル，出力先，面数，ファイルサイズを変更できます。

変更方法について次の表に示します。

表 3-18　管理コマンド（cjmsicmd）のログ取得の設定の変更方法

項目	変更方法
ログの出力レベル	admin.properties の次のプロパティで指定します。 • admin.logger.MessageLogFile.trace.level
出力先	admin.properties の次のプロパティで指定します。 • admin.logger.MessageLogFile.filepath • admin.logger.ExceptionLogFile.filepath
面数	admin.properties の次のプロパティで指定します。 • admin.logger.MessageLogFile.filenum • admin.logger.ExceptionLogFile.filenum
ファイルサイズ	admin.properties の次のプロパティで指定します。 • admin.logger.MessageLogFile.filesize • admin.logger.ExceptionLogFile.filesize

なお，指定したディレクトリがない場合，デフォルトの設定が有効になります。

各プロパティの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「6.2.1　admin.properties（管理コマンドプロパティファイル)」を参照してください。

(3)　CJMSP リソースアダプタのログ取得の設定

CJMSP リソースアダプタでは，ログ取得の設定のうち，ログの出力レベル，面数，ファイルサイズを変更できます。

変更方法について次の表に示します。

表 3-19　CJMSP リソースアダプタのログ取得の設定の変更方法

項目	変更方法
ログの出力レベル	Connector 属性ファイルの<resourceadapter>下の<config-property>で，次のプロパティを指定します。

項目	変更方法
	• MsgLogLevel
面数	Connector 属性ファイルの<resourceadapter>下の<config-property>で，次のプロパティを指定します。 • MsgLogFileNum • ExpLogFileNum
ファイルサイズ	Connector 属性ファイルの<resourceadapter>下の<config-property>で，次のプロパティを指定します。 • MsgLogFileSize • ExpLogFileSize

Connector 属性ファイルの詳細については，マニュアル「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)」の「4.1　Connector 属性ファイル」を参照してください。

3.3.14　OS の統計情報取得の設定

ここでは，Windows の場合に，OS の統計情報を取得するための設定について説明します。

Windows のシステムモニタを使用して，システムリソースのパフォーマンスデータを取得できます。トラブル発生時またはトラブルの兆候が見られる場合には，Windows のシステムモニタでパフォーマンスデータの取得を開始し，トラブル発生後にパフォーマンスデータのログを保存します。システムモニタの操作の詳細については，OS 付属のマニュアルなどを参照してください。

次の表に示すシステムモニタのログを 60 秒間隔で採取します。なお，具体的な設定方法は，OS 付属のマニュアルなどを確認してください。

表 3-20　システムモニタの設定内容

パフォーマンスオブジェクト	インスタンス	項目名	説明
processor	−	%Processor Time	CPU 使用率 (非アイドル状態のスレッドを除く合計値)
		%Privileged Time	CPU 使用率(カーネルモード分)
		%User Time	CPU 使用率(ユーザモード分)
memory	−	Cache Bytes	ファイルシステム キャッシュが現在使用しているバイト数
		Cache Faults/sec	1 秒間当たりのメモリの別の場所からの取り出しか，ディスクから取り出しの回数
		Page Faults/sec	1 秒間当たりのページフォールトの数

パフォーマンスオブジェクト	インスタンス	項目名	説明
		Transition Faults/sec	1 秒間当たりのフォールトの数
process	_Total	Handle Count	現在オープンしているハンドルの総数
		Page Faults/sec	ページフォールトの発生率
		Private Bytes	メモリ使用量(バイト)
		Virtual Bytes	仮想メモリ使用量(バイト)
		Working Set Bytes	実メモリ使用量(バイト)
	cjstartsv	%Processor Time	CPU 使用率 (非アイドル状態のスレッドを除く合計値)
		%Privileged Time	CPU 使用率(カーネルモード分)
		%User Time	CPU 使用率(ユーザモード分)
		Page Faults/sec	ページフォールトの発生率
		Thread Count	スレッド数
		Private Bytes	メモリ使用量(バイト)
		Virtual Bytes	仮想メモリ使用量(バイト)
		Working Set Bytes	実メモリ使用量(バイト)

(凡例) －：該当しない

3.3.15 ユーザダンプ取得の設定

ここでは，Windows の場合に，ユーザダンプを取得するための設定について説明します。

(1) タスクマネージャまたは Windows のデバッグツールを使用する場合

製品がハングアップした場合，トラブルシューティングに必要な資料としてユーザダンプが必要となります。ユーザダンプを取得する場合は，タスクマネージャまたは Windows のデバッグツールを使用します。詳細は，Microsoft 社のホームページを参照してください。

なお，JavaVM の異常終了時にユーザダンプを即座に取得したい場合は，製品を起動する前に，レジストリで次の設定をしておいてください。レジストリの設定は，システム全体に影響を与えることがあるため，設定時には十分注意してください。ただし，cjstartsv.exe，cjstartweb.exe，cjclstartap.exe，および adminagent.exe については，製品のインストール時に自動的に設定されます。

- レジストリキー

  ¥¥HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥Windows Error Reporting¥LocalDumps

- レジストリ値
  - DumpCount：<保存するダンプの数>
  - DumpType：2

(2)　cjstopsv コマンドを使用する場合

cjstopsv コマンドの-fd オプションを使用してユーザダンプを取得する場合は，環境変数「CJMEMDUMP_PATH」にユーザダンプの出力先ディレクトリを指定しておいてください。ユーザダンプのファイル名は，cjmemdump.dmp です。

環境変数「CJMEMDUMP_PATH」の設定例を次に示します。

環境変数「CJMEMDUMP_PATH」の設定例

```
set CJMEMDUMP_PATH=C:¥temp
```

この例の場合，C:¥temp の下に cjmemdump.dmp が作成されます。

なお，環境変数「CJMEMDUMP_PATH」を指定する場合には，次の点に注意してください。

- 保存先のディスクに十分な空き容量があることを確認してください。ユーザダンプのファイルサイズは J2EE サーバの実メモリ所要量以上になります。
- 日本語などのマルチバイト文字を含むディレクトリを指定しないでください。ユーザダンプの出力に失敗する場合があります。
- ユーザダンプの出力先ディレクトリには，存在するディレクトリを指定してください。

(3)　論理サーバの強制停止時に取得する場合

Management Server を使用してシステムを構築する場合，論理サーバの強制停止時にユーザダンプを取得するときは，環境変数「CJMEMDUMP_PATH」にユーザダンプの出力先ディレクトリを指定しておいてください。この環境変数を指定しておくと，論理 J2EE サーバが強制停止されたときに，この環境変数で指定したディレクトリ下にユーザダンプを取得できます。ユーザダンプのファイル名は，cjmemdump.dmp です。詳細は，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「4.1.11　システムの環境変数を設定する」を参照してください。

3.3.16　core ダンプ取得の設定

ここでは，UNIX の場合に，core ダンプを取得するための設定について説明します。

注意事項

Linux の仕様によって，core ファイル内のサイズ情報が不正となる場合があります。

(1) core ファイルのサイズの上限値の設定

システムの動作環境によっては，core ファイルのサイズの上限値が 0 になっている場合があります。この場合，プロセスの core ダンプを取得できないため，あらかじめ core ファイルのサイズの上限値を無制限に設定しておく必要があります。core ファイルのサイズの上限値を無制限にするためには，簡易構築定義ファイルまたはオプション定義ファイルで JavaVM 起動パラメタにオプションを指定するか，またはシェルコマンドを実行します。

なお，簡易構築定義ファイルまたはオプション定義ファイルで JavaVM 起動パラメタに指定するメモリプールサイズが大きいほど，core ファイルのサイズも大きくなるため，十分空きディスク容量を確保してください。

- **簡易構築定義ファイルで JavaVM 起動パラメタにオプションを指定する場合**

  簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，JavaVM 起動パラメタを定義します。JavaVM 起動パラメタの指定内容を次に示します。

  <param-name>タグ

  　add.jvm.arg

  <param-value>タグ

  　-XX:+HitachiFullCore

  指定例（物理ティアの定義の場合）

```
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>add.jvm.arg</param-name>
    <param-value>-XX:+HitachiFullCore</param-value>
  </param>
   :
</configuration>
```

- **オプション定義ファイルで JavaVM 起動パラメタにオプションを指定する場合**

  オプション定義ファイルに JavaVM 起動パラメタを定義します。

  指定例（Management Server 用オプション定義ファイルの場合）

```
add.jvm.arg=-XX:+HitachiFullCore
```

- **シェルコマンドを実行する場合**

  シェルコマンドを実行して，core ファイルの上限値を無制限にしてください。

  csh（C シェル）の場合の実行例

```
limit coredumpsize unlimit
```

  sh（標準シェル）の場合の実行例

```
ulimit -c unlimited
```

また，これらの設定に加え，シェルコマンドを実行して，一つのファイルサイズの上限値を無制限にしておくことをお勧めします。

- csh（C シェル）の場合の実行例

```
limit filesize unlimit
```

- sh（標準シェル）の場合の実行例

```
ulimit -f unlimited
```

参考

core ファイルのサイズの見積もり式

JavaVM がプロセスダウンした場合に生成される core ファイルサイズは，仮想メモリの使用量と同じになります。仮想メモリの使用量の計算式については，次に示すマニュアルを参照してください。core ファイル生成先である JavaVM プロセスのカレントディレクトリがあるディスクには，常に，この core ファイルサイズ以上の空き領域が必要です。

- J2EE アプリケーション実行基盤の場合

  マニュアル「アプリケーションサーバ システム設計ガイド」の「5.3　プロセスごとに使用するメモリの見積もり」

- バッチアプリケーション実行基盤の場合

  マニュアル「アプリケーションサーバ システム設計ガイド」の「6.3　仮想メモリの使用量の見積もり」

- Management Server と運用管理エージェントの場合

  マニュアル「アプリケーションサーバ システム設計ガイド」の「5.3　プロセスごとに使用するメモリの見積もり」およびリリースノートの Cosminexus Component Container が使用するメモリ所要量の「標準」の所要量

(2)　core ファイルのファイル数の上限値の設定

- 論理 J2EE サーバの core ファイルのファイル数の上限値は，簡易構築定義ファイルの「j2ee-server」の<configuration>タグ内に，ejb.server.corefilenum パラメタで定義できます。
  ejb.server.corefilenum パラメタは，J2EE サーバの拡張パラメタで定義します。

  cjstartsv プロセスの再起動時に「<作業ディレクトリ>/ejb/<サーバ名称>/」に出力される core ダンプファイルの合計が上限値を超えた場合，出力日時が古い順に削除されます。

- Management Server の core ファイルのファイル数の上限値は，Management Server 用オプション定義ファイルに，ejb.server.corefilenum パラメタで定義できます。

  Management Server プロセスの再起動時に「<Application Server のインストールディレクトリ>/manager/containers/m/ejb/<Management Server のサーバ名>/」に出力される core ダンプファイルの合計が上限値を超えた場合，出力日時が古い順に削除されます。

3.3.17　JavaVM の資料取得の設定

ここでは，次に示す JavaVM の資料を取得するための設定について説明します。

- JavaVM のスレッドダンプ
- JavaVM ログ（JavaVM ログファイル）
- 明示管理ヒープ機能のイベントログ

JavaVM 起動オプションについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.　JavaVM 起動オプション」を参照してください。

なお，JavaVM のスレッドダンプは，デフォルトでは次のディレクトリ下に出力されます。

- Windows の場合
  <作業ディレクトリ>¥ejb¥<サーバ名称>
- UNIX の場合
  <作業ディレクトリ>/ejb/<サーバ名称>

JavaVM ログは，デフォルトでは次のディレクトリ下に出力されます。

- Windows の場合
  <作業ディレクトリ>¥ejb¥<サーバ名称>¥logs
- UNIX の場合
  <作業ディレクトリ>/ejb/<サーバ名称>/logs

出力先を変更したい場合は，JavaVM または J2EE サーバ起動前に，簡易構築定義ファイルで論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.server.log.directory パラメタで，ログ出力ディレクトリを変更しておく必要があります。

次に，JavaVM のスレッドダンプ，JavaVM ログ（JavaVM ログファイル）および明示管理ヒープ機能のイベントログについて，それぞれの取得の設定を説明します。

(1)　JavaVM のスレッドダンプ取得の設定

JavaVM のスレッドダンプを取得するための設定について説明します。

JavaVM のスレッドダンプに出力される内容は，簡易構築定義ファイルで JavaVM 起動パラメタに指定している JavaVM 起動オプションによって異なります。JavaVM 起動オプションは，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，JavaVM 起動パラメタで定義します。JavaVM 起動パラメタの指定内容を次に示します。

<param-name>タグ
　add.jvm.arg

<param-value>タグ
　<JavaVM 起動オプション>

JavaVM 起動オプションに指定するオプションを次の表に示します。なお，オプションの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14. JavaVM 起動オプション」を参照してください。

表 3-21　JavaVM 起動オプションに指定するオプション（JavaVM のスレッドダンプ取得の設定）

オプション	説明
-XX:+HitachiThreadDump	拡張スレッドダンプを出力します。デフォルトの状態では，出力するように設定されています。
-XX:+HitachiThreadDumpToStdout	拡張スレッドダンプを標準出力に出力します。デフォルトの状態では，出力するように設定されています。
-XX:+HitachiThreadDumpWithHashCode	スレッド情報にスレッドのハッシュコードを出力します。デフォルトの状態では，出力するように設定されています。
-XX:+HitachiThreadDumpWithCpuTime	スレッド情報にスレッドを開始してからのユーザ CPU 時間，およびカーネル CPU 時間を出力します。デフォルトの状態では，出力するように設定されています。
-XX:+HitachiThreadDumpWithBlockCount	スレッド情報にスレッドが処理をブロックした回数，および処理が待ち状態になった回数を出力します。デフォルトの状態では，出力するように設定されています。
-XX:+HitachiOutOfMemoryAbort -XX:+HitachiOutOfMemoryAbortThreadDump	どちらも設定されている場合は，OutOfMemoryError によって強制終了した時に，スレッドダンプを出力します。ただし，J2SE クラスライブラリで C ヒープ不足が発生した場合，および JavaVM の処理中に C ヒープ不足が発生した場合は，スレッドダンプを出力しません。
-XX:+HitachiOutOfMemoryAbortThreadDumpWithJHeapProf	OutOfMemoryError によって強制終了した時に出力されるスレッドダンプに，クラス別統計情報を出力します。デフォルトでは出力されません。

スレッドダンプファイルの出力先を変更する場合は，JAVACOREDIR 環境変数に出力先を指定してください。JAVACOREDIR 環境変数の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.7　JavaVM で使用する環境変数の詳細」を参照してください。

(2)　JavaVM ログ取得の設定

JavaVM ログを取得するための設定について説明します。

JavaVM ログとは，製品が標準の JavaVM に追加した拡張オプションを使用して取得できるログです。標準の JavaVM よりも，多くのトラブルシュート情報が取得できます。このログファイルを，**JavaVM ログファイル**といいます。このファイルには，JavaVM の GC のログも出力されます。

JavaVM ログファイルを取得するためには，簡易構築定義ファイルで JavaVM 起動パラメタに JavaVM 起動オプションを指定します。JavaVM 起動オプションは，論理 J2EE サーバ（j2ee-server）の

<configuration>タグ内に，JavaVM 起動パラメタで定義します。JavaVM 起動パラメタの指定内容を次に示します。

<param-name>タグ

add.jvm.arg

<param-value>タグ

<JavaVM 起動オプション>

JavaVM 起動オプションに指定するオプションを次の表に示します。なお，オプションの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14. JavaVM 起動オプション」を参照してください。

表 3-22 JavaVM 起動オプションに指定するオプション（JavaVM ログ取得の設定）

オプション	説明
-XX:+HitachiOutOfMemoryStackTrace -XX:+HitachiVerboseGC -XX:+HitachiOutOfMemoryHandling -XX:+HitachiJavaClassLibTrace -XX:+JITCompilerContinuation	次のオプションのうちどれかを指定している場合に，JavaVM ログファイルを出力します。 • -XX:+HitachiOutOfMemoryStackTrace を指定している場合は，例外情報とスタックトレースを JavaVM ログファイルに出力します。なお，このオプションを指定すると，-XX:+HitachiOutOfMemorySize および-XX:+HitachiOutOfMemoryCause も同時に指定されます。 • -XX:+HitachiVerboseGC を指定している場合は，GC が発生したときに，拡張 verbosegc 情報を JavaVM ログファイルに出力します。 • -XX:+HitachiOutOfMemoryHandling を指定している場合は，Java ヒープ不足または Metaspace 領域不足が原因の OutOfMemory 発生時に，OutOfMemory の発生頻度に関する情報を JavaVM ログファイルに出力します。 • -XX:+HitachiJavaClassLibTrace を指定している場合は，クラスライブラリのスタックトレースを JavaVM ログファイルに出力します。 • -XX:+JITCompilerContinuation を指定している場合は，JIT コンパイラ稼働継続機能が有効になるため，JIT コンパイルがアプリケーションを構成するメソッドの論理矛盾で失敗すると，JIT コンパイラ稼働継続機能のログを JavaVM ログファイルに出力します。

必要に応じて，次の拡張オプションを指定して，JavaVM ログファイルの出力方法や出力内容を設定してください。

- JavaVM ログファイルのファイルサイズやファイル数
- 拡張 verbosegc 機能オプション
- OutOfMemoryError 発生時の拡張機能オプション
- クラスライブラリトレース機能オプション

- ローカル変数情報出力機能オプション
- ログファイルの非同期出力機能オプション　など

次に，JavaVM の保守情報（Java ヒープの情報）および GC のログに拡張 verbosegc 情報を出力するための設定について説明します。次の表に示すオプションで，拡張 verbosegc 情報の出力を有効にして，拡張 verbosegc 情報の出力形式を指定してください。

表 3-23　拡張 verbosegc 情報を出力するために指定するオプション

オプション	説明
-XX:+HitachiVerboseGC	拡張 verbosegc 情報を JavaVM ログファイルに出力します。
-XX:+HitachiVerboseGCPrintDate	拡張 verbosegc 情報の各行に日付を出力します。
-XX:+HitachiVerboseGCCpuTime	GC の開始から終了までの間で，GC を実行したスレッドの CPU 利用時間を出力します。CPU 利用時間は，ユーザモードで費やした CPU 時間とカーネルモードで費やした CPU 時間に分けて出力されます。
-XX:HitachiVerboseGCIntervalTime=<時間間隔(秒)>	拡張 verbosegc 情報の出力間隔を指定します。
-XX:+HitachiVerboseGCPrintCause	拡張 verbosegc 情報に，GC が発生した要因を出力します。
-XX:+HitachiOutputMilliTime	拡張 verbosegc 情報の各行に日時（ミリ秒まで）を出力します。
-XX:+HitachiCommaVerboseGC	拡張 verbosegc 情報を CSV 形式で出力します。
-XX:+HitachiVerboseGCPrintTenuringDistribution	Survivor 領域の年齢分布情報を出力します。出力形式や出力情報については，「9.11　Survivor 領域の年齢分布情報出力機能」を参照してください。
-XX:+HitachiVerboseGCPrintJVMInternalMemory	JavaVM 内部で管理しているヒープ情報を JavaVM ログファイルに出力します。
-XX:+HitachiVerboseGCPrintThreadCount	Java スレッドの数を監視するために，Java スレッドの数を JavaVM ログファイルに出力します。
-XX:+HitachiVerboseGCPrintDeleteOnExit	java.io.File.deleteOnExit()を呼び出したことによって JavaVM が確保した累積のヒープサイズとメソッドの呼び出し回数を，JavaVM ログファイルに出力します。

拡張 verbosegc 情報から，そのサーバで必要とする Java ヒープ領域サイズ，Metaspace 領域サイズなどを見積もるための情報を取得できます。

(3)　明示管理ヒープ機能のイベントログ取得の設定

ここでは，**明示管理ヒープ機能のイベントログ**を取得するための設定について説明します。また，JavaVM ログファイルとの関連についても説明します。

明示管理ヒープ機能のイベントログを取得するために必要な設定を次に示します。

- ログ出力レベルの設定

  目的に応じたログを出力するために，ログ出力レベルを設定します。「none」「normal」「verbose」「debug」の４段階で指定できます。J2EE サーバのデフォルトは「normal」です。「none」＜「normal」＜「vervose」＜「debug」の順でログが詳細になり，出力量も多くなります。

  例えば，通常は「normal」を指定して運用し，障害が発生した場合には「verbose」を指定して詳細なログを取得する，といった運用ができます。

- ログを出力するファイルについての設定

  明示管理ヒープ機能のログは，ほかの JavaVM のログとは別のファイルに出力されます。想定する運用に応じて，出力先のファイル，ファイルのサイズ，およびファイルの個数を設定してください。

J2EE サーバまたはバッチサーバの場合，これらの設定は，簡易構築定義ファイルで JavaVM 起動パラメタに指定している JavaVM 起動オプションによって設定できます。JavaVM 起動オプションは，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，JavaVM 起動パラメタで定義します。JavaVM 起動パラメタの指定内容を次に示します。

<param-name>タグ

　add.jvm.arg

<param-value>タグ

　<JavaVM 起動オプション>

JavaVM 起動オプションに指定するオプションを次の表に示します。なお，オプションの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.　JavaVM 起動オプション」を参照してください。

表 3-24　JavaVM 起動オプションに指定するオプション（明示管理ヒープ機能のイベントログ取得の設定）

設定内容	オプション	説明
出力レベルの設定	-XX:HitachiExplicitMemoryLogLevel:＜文字列＞	ログ出力レベルを指定します。 • none 明示管理ヒープ機能のイベントログが出力されません。 • normal 通常運用の場合に指定します。Explicit ヒープのサイズが大きく変化するイベントが発生した場合や，GC 発生時の Explicit ヒープの状態が出力されます。 • verbose 障害が発生した場合など，詳細なログが必要なときに指定します。 • debug verbose よりもさらに詳細なログが必要な場合に指定します。システムのパフォーマンスは低下します。

設定内容	オプション	説明
出力するファイルについての設定	-XX:HitachiExplicitMemoryJavaLog:＜文字列＞	イベントログを出力するファイルのファイル名を指定します。
	-XX:HitachiExplicitMemoryJavaLogFileSize=＜自然数＞	1 ファイル当たりの最大ファイルサイズを指定します。
	-XX:HitachiExplicitMemoryJavaLogNumberOfFile=<自然数>	作成するログファイルの最大数を指定します。指定値を超えると，ラップアラウンドされ，ログは最初に作成したファイルに出力されます。

なお，明示管理ヒープ機能のイベントログは，JavaVM ログファイルとは異なるファイルに出力されます。ただし，一部のオプションについては，JavaVM ログファイルに設定した値が引き継がれます。

JavaVM ログファイルの設定を引き継ぐ項目を次の表に示します。

表 3–25　JavaVM ログファイルの設定を引き継ぐ項目

オプション名	意味	デフォルト値
-XX:+HitachiJavaLogNoMoreOutput	ログファイルの出力に失敗した場合の挙動を指定するオプションです。	有効
-XX:+HitachiOutputMilliTime	ログファイルに出力する時間の単位をミリ秒にするかどうかを指定するオプションです。	無効

JavaVM 拡張オプションの詳細は，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.2　JavaVM 拡張オプションの詳細」を参照してください。

3.3.18　WebSocket コンテナのログ取得の設定

ここでは，WebSocket コンテナのログ取得で設定できる項目について説明します。

WebSocket コンテナのログ取得で変更できる設定項目と，項目に対応する簡易構築定義ファイルのパラメタを次の表に示します。

表 3–26　WebSocket コンテナのログ取得の設定項目

ログおよびトレース	項目	対応する簡易構築定義ファイルのパラメタ
アクセスログ	アクセスログの出力の有無	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.access_log.websocket.enabled （デフォルトではアクセスログが出力されません）
	アクセスログ出力時のフォーマット	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.access_log.websocket.format※
	アクセスログのファイルサイズ	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の

ログおよびトレース	項目	対応する簡易構築定義ファイルのパラメタ
		ejbserver.logger.channels.define.WebSocketAccessLogFile.filesize
	アクセスログのファイル面数	論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の ejbserver.logger.channels.define.WebSocketAccessLogFile.filenum
性能解析トレース	－	ほかの性能解析トレースと同様に，日常的なシステム運用の作業で，cprfed コマンドを実行するときに取得条件などを指定します。性能解析トレースファイルの取得については，「7. 性能解析トレースを使用した性能解析」を参照してください。

（凡例）－：該当しない

注※　アクセスログでは，このキーでフォーマットを定義することで，ログの出力形式をカスタマイズできます。WebSocket 通信を行う NIO HTTP サーバのアクセスログのカスタマイズについては，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「7.11.2　NIO HTTP サーバのアクセスログのカスタマイズ」を参照してください。

4 トラブルシューティングで必要な資料の出力先と出力方法

トラブルシューティングで使用する資料は，資料ごとに任意のタイミングで取得できます。この章では，ログやスレッドダンプなどの資料を個別に出力する方法や，これらのデフォルトの出力先について説明します。また，トラブルシューティングで使用するディレクトリについても説明します。

4.1　この章の構成

トラブルシューティングに関する説明のうち，トラブルシューティングの資料の出力先と出力方法について説明します。

トラブルシューティングで使用する資料には，それぞれデフォルトの出力先があります。また，トラブルシューティングに必要な資料は，snapshot ログを使用しないで，それぞれの情報を個別に取得できます。

この章の構成を次の表に示します。

表 4-1　この章の構成（トラブルシューティングで使用する資料の出力先と出力方法）

分類	タイトル	参照先
解説	トラブルシューティングで使用する資料の種類（snapshot ログを使用しない場合）	4.2
	アプリケーションサーバのログ（J2EE アプリケーションを実行するシステム）	4.3
	アプリケーションサーバのログ（バッチアプリケーションを実行するシステム）	4.4
	EJB クライアントアプリケーションのシステムログ	4.5
	性能解析トレース	4.6
	JavaVM のスレッドダンプ	4.7
	JavaVM の GC ログ	4.8
	メモリダンプ	4.9
	JavaVM ログ（JavaVM ログファイル）	4.10
	JavaVM 出力メッセージログ（標準出力またはエラーリポートファイル）	4.11
	OS の状態情報と OS のログ	4.12
	OS の統計情報	4.13
	アプリケーションサーバの定義情報	4.14
	J2EE サーバまたはバッチサーバの作業ディレクトリの内容	4.15
	アプリケーションサーバのリソース設定情報	4.16
	Web サーバログ	4.17
	JavaVM のスタックトレース情報	4.18
	明示管理ヒープ機能のイベントログ	4.19
	Component Container 管理者のセットアップコマンドの実行情報（UNIX の場合）	4.20

なお，トラブルシューティングの概要，資料を自動で出力する方法，資料の取得や出力に関する設定，および資料の出力内容については，それぞれ次の個所を参照してください。

- トラブルシューティングの概要と，資料を自動で出力する方法
  「2.　トラブルシューティング」
- 資料の取得や出力に関する設定
  「3.　トラブルシューティングのための準備」
- 資料に出力される内容
  「5.　トラブルの分析」

4.2　トラブルシューティングで使用する資料の種類（snapshot ログを使用しない場合）

アプリケーションサーバの構成ソフトウェアが出力するログなどの資料を，snapshot ログを使用しないで個別に取得することもできます。この節では，トラブル発生時に必要な資料を個別に取得する場合の方法について説明します。snapshot ログを使用した資料の取得については，「2.3.3　snapshot ログの収集」を参照してください。

snapshot ログを使用しない場合に，個別に取得する必要がある資料と参照先を次の表に示します。

表 4-2　取得が必要な資料（snapshot ログを使用しない場合）

取得する資料	参照先
アプリケーションサーバのログ（J2EE アプリケーションを実行するシステムの場合）	4.3
アプリケーションサーバのログ（バッチアプリケーションを実行するシステムの場合）	4.4
EJB クライアントアプリケーションのシステムログ※	4.5
性能解析トレース	4.6
JavaVM のスレッドダンプ	4.7
JavaVM の GC ログ	4.8
メモリダンプ	4.9
JavaVM ログ（JavaVM ログファイル）	4.10
JavaVM 出力メッセージログ（標準出力またはエラーリポートファイル）	4.11
OS の状態情報と OS のログ	4.12
OS の統計情報	4.13
アプリケーションサーバの定義情報	4.14
J2EE サーバまたはバッチサーバの作業ディレクトリの内容	4.15
アプリケーションサーバのリソース設定情報	4.16
Web サーバログ	4.17
JavaVM のスタックトレース情報	4.18
明示管理ヒープ機能のイベントログ	4.19
Component Container 管理者のセットアップコマンドの実行情報（UNIX の場合）	4.20

注※

EJB クライアントアプリケーションを実行するシステムの場合に取得してください。

4.3　アプリケーションサーバのログ（J2EE アプリケーションを実行するシステム）

この節では，J2EE アプリケーションを実行するシステムで，アプリケーションサーバの構成ソフトウェアが出力するログを手動で取得する方法について説明します。なお，snapshot ログとしてアプリケーションサーバのログをすでに収集している場合は，ここで説明する操作は不要です。

ここでは，次のログの取得方法について説明します。

- Component Container のログ
- Performance Tracer のログ
- Component Transaction Monitor のログ
- 監査ログで出力するログ
- アプリケーションのユーザログ

> **参考**
>
> HTTP Server のログの取得方法については，「4.17　Web サーバログ」で説明します。

4.3.1　Component Container のログの取得

Component Container のログの種類とログの出力先について説明します。Component Container のログには，次のログがあります。

- J2EE サーバ・サーバ管理コマンドのログ
- 運用管理エージェント・運用監視エージェント・Management Server のログ
- 仮想サーバマネージャの内部構築ツールおよびサーバ通信エージェントのログ
- 統合ユーザ管理のログ
- CJMS プロバイダのログ

それぞれのログの出力先について説明します。

(1)　J2EE サーバ・サーバ管理コマンドのログの取得

J2EE サーバ・サーバ管理コマンドのログの取得方法について説明します。

また，Component Container では，これらのログに加えて，移行コマンドのログが出力されます。リソース枯渇監視機能を使用している場合はリソース枯渇監視ログが出力されます。

- J2EE サーバのログには，メッセージログ，ユーザログ，例外ログ，アクセスログ，および保守用ログの 5 種類があります。なお，J2EE サーバでは，これらのログに加えて，起動，停止および異常終了時にイベントログまたは syslog を出力します。
- サーバ管理コマンドのログには，メッセージログ，例外ログ，および保守用ログの 3 種類があります。
- リソースアダプタのバージョンアップコマンド（cjrarupdate）のログには，メッセージログ，例外ログ，および保守用ログの 3 種類があります。
- 移行コマンドのログには，メッセージログ，例外ログ，および保守用ログの 3 種類があります。

次にそれぞれのログについて説明します。

メッセージログ

J2EE サーバ，サーバ管理コマンド，移行コマンドなどの稼働状態が出力されます。各種サーバおよびコマンドの稼働監視の情報として使用します。

ユーザログ

アプリケーション中で出力される標準出力および標準エラー出力の情報が出力されます。アプリケーションの開発時の動作確認用に使用します。なお，java.security.debug プロパティを指定してサーバを起動した場合，標準出力および標準エラー出力の情報はユーザログに出力されません。JavaVM のメモリ関連ログも含みます。

例外ログ

システムでトラブルが発生したときの Component Container の例外情報が出力されます。なお，例外ログは日常的な運用で監視する必要はありません。ログにメッセージが出力された場合に，例外情報を参照するときにご利用ください。

アクセスログ

Web アプリケーションへのリクエストの処理結果や，WebSocket の通信履歴が出力されます。

保守用ログ

システムでトラブルが発生したときの Component Container の障害保守情報が出力されます。保守員が Component Container の障害解析用に使用します。

イベントログ（Windows の場合）

J2EE サーバが起動，停止または異常終了したことを示す情報が出力されます。出力先は Windows のイベントログの設定によって異なります。

なお，イベントログは，J2EE サーバの停止のしかたによっては，出力されません。次の場合は，正しくログが出力されないことがあります。

- J2EE サーバが動作している JavaVM 自体に問題が発生した場合
- J2EE サーバのプロセスを TerminateProcess によって外部から停止した場合
- JavaVM の起動オプションとして-XX:+HitachiOutOfMemoryAbort オプションを指定している場合にメモリ不足によって J2EE サーバが異常終了したとき

  なお，-XX:+HitachiOutOfMemoryAbort オプションは，デフォルトで設定されているオプションです。

syslog（UNIX の場合）

J2EE サーバが起動，停止または異常終了したことを示す情報が出力されます。出力先は UNIX の syslog の設定によって異なります。

なお，syslog は，J2EE サーバの停止のしかたによっては，出力されません。次の場合は，正しくログが出力されないことがあります。

- J2EE サーバが動作している JavaVM 自体に問題が発生した場合
- J2EE サーバのプロセスを SIGKILL シグナル（kill -9 など）によって外部から停止した場合
- JavaVM の起動オプションとして-XX:+HitachiOutOfMemoryAbort オプションを指定している場合にメモリ不足によって J2EE サーバが異常終了したとき

  -XX:+HitachiOutOfMemoryAbort オプションは，デフォルトで設定されているオプションです。

リソース枯渇監視ログ

リソース枯渇監視機能を使用している場合に，監視対象のリソースについてのリソース枯渇監視情報が出力されます。リソースの使用率または使用数がしきい値を超えた場合の原因調査に使用します。

ログは，若い面番号の付いたログファイルから順に記録されます。一つのログファイルのサイズが 1 面当たりの最大サイズに達すると，ログは次の面番号の付いたログファイルに記録されます。最後のログファイル（面数の番号が付いたログファイル）のサイズが 1 面当たりの最大サイズに達すると，面の番号 1 のログファイルを空にし，そこへログを記録していきます。以降，ログファイルを空にしながら，面番号の順にログファイルへログを記録していきます。

ログの出力先のデフォルトを次の表に示します。Component Container のログは，サーバ単位またはコマンド単位に取得できます。

ログの出力先に示す<作業ディレクトリ>は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.public.directory パラメタで指定したディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥CC¥server¥public（Windows の場合），または/opt/Cosminexus/CC/server/public（UNIX の場合）です。

(a) J2EE サーバのログ

表 4-3 J2EE サーバのログの出力先（デフォルト）

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
メッセージログ	稼働ログ	• Windows の場合 <ejb.server.log.directory>※2¥cjmessage[n].log • UNIX の場合 <ejb.server.log.directory>※2/cjmessage[n].log	1MB×2	MessageLogFile
	ログ稼働ログ※3	• Windows の場合	1MB×2	−

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
		<ejb.server.log.directory>※2¥cjlogger.log • UNIX の場合 <ejb.server.log.directory>※2/cjlogger.log		
	J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの稼働ログ※4	• Windows の場合 (Connector 1.0 仕様のリソースアダプタ) <ejb.server.log.directory>※2¥connectors¥<リソースアダプタの表示名>[n].log (Connector 1.5 仕様のリソースアダプタ) <ejb.server.log.directory>※2¥connectors¥<リソースアダプタの表示名>_<コネクション定義の並び順>_[n].log • UNIX の場合 (Connector 1.0 仕様のリソースアダプタ) <ejb.server.log.directory>※2/connectors/<リソースアダプタの表示名>[n].log (Connector 1.5 仕様のリソースアダプタ) <ejb.server.log.directory>※2/connectors/<リソースアダプタの表示名>_<コネクション定義の並び順>_[n].log	2MB×4	−
	J2EE アプリケーションに含めて使用するリソースアダプタの稼働ログ※4	• Windows の場合 (通常モード／Connector 1.0 仕様のリソースアダプタ) <ejb.server.log.directory>※2¥connectors¥<J2EE アプリケーション名>¥<リソースアダプタの表示名>[n].log (テストモード／Connector 1.0 仕様のリソースアダプタ) <ejb.server.log.directory>※2¥connectors¥test#<J2EE アプリケーション名>¥<リソースアダプタの表示名>[n].log	2MB×4	−

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
		(通常モード／Connector 1.5 仕様のリソースアダプタ) <ejb.server.log.directory>※2¥connectors¥<J2EE アプリケーション名>¥<リソースアダプタの表示名>_<コネクション定義の並び順>_[n].log (テストモード／Connector 1.5 仕様のリソースアダプタ) <ejb.server.log.directory※2>¥connectors¥test#<J2EE アプリケーション名>¥<リソースアダプタの表示名>_<コネクション定義の並び順>_[n].log • UNIX の場合 (通常モード／Connector 1.0 仕様のリソースアダプタ) <ejb.server.log.directory>※2/connectors/<J2EE アプリケーション名>/<リソースアダプタの表示名>[n].log (テストモード／Connector 1.0 仕様のリソースアダプタ) <ejb.server.log.directory>※2/connectors/test#<J2EE アプリケーション名>/<リソースアダプタの表示名>[n].log (通常モード／Connector 1.5 仕様のリソースアダプタ) <ejb.server.log.directory>※2/connectors/<J2EE アプリケーション名>/<リソースアダプタの表示名>_<コネクション定義の並び順>_[n].log (テストモード／Connector 1.5 仕様のリソースアダプタ) <ejb.server.log.directory>※2/connectors/test#<J2EE アプリケーション名>/<リソースアダプタの表示名>_<コネクション定義の並び順>_[n].log		
ユーザログ	Web サーブレットログ※5	• Windows の場合 <ejb.server.log.directory>※2¥web_servlet[n].log • UNIX の場合	4MB×4	WebServletLogFile

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
		<ejb.server.log.directory>※2/web_servlet[n].log		
	ユーザ出力ログ	• Windows の場合 <ejb.server.log.directory>※2¥user_out[n].log • UNIX の場合 <ejb.server.log.directory>※2/user_out[n].log	1MB×2	UserOutLogFile
	ユーザエラーログ	• Windows の場合 <ejb.server.log.directory>※2¥user_err[n].log • UNIX の場合 <ejb.server.log.directory>※2/user_err[n].log	1MB×2	UserErrLogFile
	JavaVM の保守情報および GC のログ	• Windows の場合 <ejb.server.log.directory>※2¥javalog[nn].log • UNIX の場合 <ejb.server.log.directory>※2/javalog[nn].log	4MB×4	−
	明示管理ヒープ機能のイベントログ	• Windows の場合 <ejb.server.log.directory>※2¥ehjavalog[nn].log • UNIX の場合 <ejb.server.log.directory>※2/ehjavalog[nn].log	4MB×4	−
例外ログ	障害発生時の例外情報	• Windows の場合 <ejb.server.log.directory>※2¥cjexception[n].log • UNIX の場合 <ejb.server.log.directory>※2/cjexception[n].log	1MB×2	ExceptionLogFile
保守用ログ	保守情報	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjmaintenance[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjmaintenance[n].log	16MB×4	MaintenanceLogFile

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
	コンソールメッセージ	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjconsole[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjconsole[n].log	1MB×2	ConsoleLogFile
	EJB コンテナの保守情報	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjejbcontainer[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjejbcontainer[n].log	1MB×2	EJBContainerLogFile
	Web コンテナの保守情報	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjwebcontainer[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjwebcontainer[n].log	1MB×2	WebContainerLogFile
	起動プロセス標準出力情報※6	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjstdout.log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjstdout.log	−	−
	起動プロセス標準エラー情報※6	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjstderr.log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjstderr.log	−	−
	終了プロセス情報	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cj_shutdown[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cj_shutdown[n].log	4KB×2※7	−

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
	J2EE サーバの RMI 通信ログ	• Windows の場合 <ejb.server.log.directory>※2¥CC¥rmi¥cjrmi[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/rmi/cjrmi[n].log	1MB×4	−
イベントログ	J2EE サーバの起動，停止または異常終了を示すログ	Windows のイベントビューアのアプリケーションログ※8	−	−
syslog	J2EE サーバの起動，停止または異常終了を示すログ	UNIX の syslog の設定に依存します。※9	−	−
デバッグ用ログ	開発調査ログ	• Windows の場合 <ejb.server.log.directory>※2¥cjdevelopment[n].log • UNIX の場合 <ejb.server.log.directory>※2/cjdevelopment[n].log	1MB×4	DevelopmentLogFile
アクセスログ	HTTP 通信の処理結果	• Windows の場合 <ejb.server.log.directory>※2¥cj_access_niohttp[n].log • UNIX の場合 <ejb.server.log.directory>※2/cj_access_niohttp[n].log	4MB×16	NIOHTTPAccessLogFile
	WebSocket 通信の処理結果	• Windows の場合 <ejb.server.log.directory>※2¥cj_access_websocket[n].log • UNIX の場合 <ejb.server.log.directory>※2/cj_access_websocket[n].log	4MB×16	WebSocketAccessLogFile

（凡例） −：該当しない

注

チャネル名は，ログの出力先を識別する名称です。ログの属性（サイズ，面数）を変更する場合のキー値として使用します。

注※1

ログファイル名の[n]の部分には，面の番号（1 から面数（最大 16 まで））が付きます。

ただし，EJB クライアントアプリケーションのサブディレクトリ共有モードを使用する場合は，最大面数が 64 になります。

また，[nn]の部分には，01～99 の通し番号が付きます。

注※2

<ejb.server.log.directory>は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.server.log.directory パラメタで指定したディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs です。

簡易構築定義ファイルの ejb.server.log.directory パラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.11.3　J2EE サーバ用オプション定義を設定するパラメタ」およびマニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.2　usrconf.cfg（J2EE サーバ用オプション定義ファイル）」を参照してください。

注※3

ファイル出力時にファイル容量をチェックします。チェック時に最大容量を超えた場合，cjlogger.log ファイルの名称をバックアップファイルの名称（cjlogger_save.log）に変更します。

注※4

リソースアダプタのログを取得するかどうかは，サーバ管理コマンドで指定されている内容に従います。また，リソースアダプタのログは，簡易構築定義ファイルでサイズおよび面数を変更できます。リソースアダプタのログ取得の設定については，「3.3.11　リソースアダプタのログ取得の設定」を参照してください。

注※5

サーブレット，JSP で発生した例外のスタックトレースについても出力されます。

注※6

起動プロセス情報だけを取得するログです。主に J2EE サーバの起動または終了時に出力されるため，オンライン中に出力されることはほとんどありません。ファイルのサイズが上限に達したときは，<作業ディレクトリ>¥ejb¥<サーバ名称>¥logs 下（Windows の場合），または<作業ディレクトリ>/ejb/<サーバ名称>/logs 下（UNIX の場合）の cjstdout_save.log または cjstderr_save.log に退避されます。すでに cjstdout_save.log または cjstderr_save.log があったときは，上書きされます。

注※7

サイズおよび面数は，変更できません。

注※8

ログファイルの出力先は Windows のイベントログの設定によって異なります。

注※9

J2EE サーバの起動，停止および異常終了のメッセージを syslog に出力するためには，syslog の設定で，facility「daemon」に対する priority を「info」または「debug」に設定する必要があります。syslog の設定に関する詳細については，OS 付属のマニュアルを参照してください。

参考

セッションフェイルオーバ機能を使用している場合，セッションフェイルオーバについてのログは，セッションフェイルオーバ機能を使用している J2EE サーバのログとして出力されます。

(b)　サーバ管理コマンドのログ

表 4-4　サーバ管理コマンドのログの出力先（デフォルト）

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
メッセージログ	稼働ログ※2，※3	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥cjmessage[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/cjmessage[n].log	1024KB×3	MessageLogFile

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
	ログ稼働ログ※2	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥cjlogger.log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/cjlogger.log	1024KB×2	−
例外ログ	障害発生時の例外情報※2，※3	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥cjexception[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/cjexception[n].log	1024KB×6	ExceptionLogFile
保守用ログ	保守情報※2	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥CC¥maintenance¥cjmaintenance[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjmaintenance[n].log	1024KB×3	MaintenanceLogFile
	コンソールメッセージ※2	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥CC¥maintenance¥cjconsole[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjconsole[n].log	32KB×3	ConsoleLogFile
	サーバ管理コマンドの保守情報※2	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥CC¥maintenance¥cjserveradmin[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjserveradmin[n].log	32KB×3	ServerAdminLogFile

（凡例）　－：該当しない

注

チャネル名は，ログの出力先を識別する名称です。

注※1

ログファイル名の[n]の部分には，面の番号（1 から各ログの最大面数まで）が付きます。

注※2

トレース共通ライブラリ形式の出力メッセージ（アプリケーション識別名）には，コマンド名が表示されます。トレース共通ライブラリ形式のログについては，「5.2　アプリケーションサーバのログ」を参照してください。

注※3

互換モードの場合，稼働ログと障害発生時の例外情報の出力先は標準モードと異なります。互換モードの場合の出力先とデフォルトのサイズ・面数は次のようになります。

表 4-5　サーバ管理コマンドのログの出力先（互換モード）

内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
稼働ログ	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥<コマンド名称>message[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/<コマンド名称>message[n].log	128KB×2
障害発生時の例外情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥<コマンド名称>exception[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/<コマンド名称>exception[n].log	256KB×2

注※

ログファイル名の[n]の部分には，面の番号（1 から各ログの最大面数まで）が付きます。

サーバ管理コマンドのメッセージログに出力されるメッセージには，メッセージ ID フィールドが空白で，メッセージテキストフィールドにメッセージ ID（KDJEnnnnn-Y など）が含まれる場合があります。それらはサーバ側で発生したメッセージで，前後に出力されるメッセージの付加情報となります。

(c)　リソースアダプタバージョンアップコマンド（cjrarupdate）のログ

表 4-6　リソースアダプタバージョンアップコマンド（cjrarupdate）のログの出力先

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
メッセージログ	稼働ログ	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjrarupdatemessage[n].log※1 • UNIX の場合	1MB×2

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
		/opt/Cosminexus/CC/logs/cjrarupdatemessage[n].log※1	
例外ログ	障害発生時の例外情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjrarupdateexception[n].log※1 • UNIX の場合 /opt/Cosminexus/CC/logs/cjrarupdateexception[n].log※1	1MB×2
保守用ログ	保守情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjrarupdatemaintenance[n].log※2 • UNIX の場合 /opt/Cosminexus/CC/logs/cjrarupdatemaintenance[n].log※2	16MB×4

注※1

ログファイル名の[n]の部分には，面の番号（1 または 2）が付きます。

注※2

ログファイル名の[n]の部分には，面の番号（1 から 4 まで）が付きます。

(d) 移行コマンド（cjenvupdate）のログ

表 4-7 移行コマンド（cjenvupdate）のログの出力先

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
メッセージログ	cjenvupdate コマンドの稼働ログ	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjenvupdatemessage[n].log • UNIX の場合 /opt/Cosminexus/CC/logs/cjenvupdatemessage[n].log	4MB×4
例外ログ	cjenvupdate コマンドの例外情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjenvupdateexception[n].log • UNIX の場合 /opt/Cosminexus/CC/logs/cjenvupdateexception[n].log	4MB×4
保守用ログ	cjenvupdate コマンドの保守情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjenvupdatemaintenance[n].log • UNIX の場合	4MB×4

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
		/opt/Cosminexus/CC/logs/cjenvupdatemaintenance[n].log	

注※

[n]には，面の番号（1 から 4 まで）が付きます。

(e) リソース枯渇監視のログ

表 4-8 リソース枯渇監視のログの出力先

監視対象リソース	ログ取得場所およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
メモリ	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjmemorywatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjmemorywatch[n].log	1MB×2	MemoryWatchLogFile
ファイルディスクリプタ	• UNIX の場合※3 <ejb.server.log.directory>※2/watch/cjfiledescriptorwatch[n].log	1MB×2	FileDescriptorWatchLogFile
スレッド	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjthreadwatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjthreadwatch[n].log	1MB×2	ThreadWatchLogFile
スレッドダンプ	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjthreaddumpwatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjthreaddumpwatch[n].log	1MB×2	ThreaddumpWatchLogFile
HTTP リクエスト実行待ちキュー	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjrequestqueuewatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjrequestqueuewatch[n].log	1MB×2	RequestQueueWatchLogFile
HTTP セッション数	• Windows の場合	1MB×2	HttpSessionWatchLogFile

監視対象リソース	ログ取得場所およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
	<ejb.server.log.directory>※2¥watch¥cjhttpsessionwatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjhttpsessionwatch[n].log		
コネクションプール	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjconnectionpoolwatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjconnectionpoolwatch[n].log	1MB×2	ConnectionPoolWatchLogFile

注

チャネル名は，ログの出力先を識別する名称です。ログの属性（サイズ，面数）を変更する場合のキー値として使用します。

注※1

[n]には，面の番号（1 から面数（最大 16 まで））が付きます。

注※2

<ejb.server.log.directory>は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.server.log.directory パラメタで指定したディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs です。

簡易構築定義ファイルの ejb.server.log.directory パラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.11.3　J2EE サーバ用オプション定義を設定するパラメタ」およびマニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.2　usrconf.cfg（J2EE サーバ用オプション定義ファイル）」を参照してください。

注※3

Windows の場合，または AIX の場合は，ファイルディスクリプタを監視できません。

リソース枯渇監視ログファイルに出力される情報やログファイルの出力形式については，マニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「4.3　リソース枯渇監視機能とリソース枯渇監視情報の出力」を参照してください。

(f)　ログの出力先を設定するユーザ定義ファイル

J2EE サーバおよびサーバ管理コマンドのログの出力先を変更している場合は，次の表に示す，ログの出力先を設定するユーザ定義ファイルを参照して出力先を確認してください。なお，ログの出力先を変更した場合，変更後の出力先は snapshot ログの一括収集時に収集対象外となります。必要に応じて snapshot ログの収集先を変更してください。

表 4-9　ログの出力先を設定するユーザ定義ファイル

分類	ユーザ定義ファイル
J2EE サーバ	簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に指定する，ejb.server.log.directory パラメタ

分類	ユーザ定義ファイル
	デフォルトは，「<Application Server のインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs」（Windows の場合），または「/opt/Cosminexus/CC/server/public/ejb/<サーバ名称>/logs」（UNIX の場合）です。
サーバ管理コマンド	サーバ管理コマンドの usrconf.bat（Windows の場合），または usrconf（UNIX の場合）の ejbserver.log.directory キー キーのデフォルトは，「<Application Server のインストールディレクトリ>¥CC¥admin¥logs」（Windows の場合），または「/opt/Cosminexus/CC/admin/logs」（UNIX の場合）です。 Management Server リモート管理機能から操作した場合は，サーバ管理コマンドのログ出力先は変更できません。

ログの出力先の変更方法など，トラブルシューティングの資料取得の設定については，「3. トラブルシューティングのための準備」を参照してください。

(2) 運用管理エージェント・運用監視エージェント・Management Server のログの取得

ここでは，運用管理エージェント，運用監視エージェント，および Management Server のログの出力先について説明します。

運用管理エージェント・運用監視エージェント・Management Server のログには，個別に取得する以外に，**統合ログ**としてまとめて取得できるものがあります。統合ログには，次の種類があります。

- 統合メッセージログ
  Manager のメッセージログが統合されて出力されます。
- 統合トレースログ
  Manager のトレースログが統合されて出力されます。
- コマンド保守ログ※
  運用管理コマンド，Smart Composer 機能で使用するコマンド，および snapshotlog コマンドのトレースログが統合されて出力されます。

注※　Smart Composer 機能で使用するコマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「8. Smart Composer 機能で使用するコマンド」を参照してください。

統合ログの出力先を次の表に示します。

表 4-10　統合ログの出力先（Windows の場合）

ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数
mngmessage[n]. ※1log	統合メッセージログ	<Manager のログ出力ディレクトリ>¥message※2	256KB×4

ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数
mngtrace[n] ※1.log	統合トレースログ	<Manager のログ出力ディレクトリ>¥trace※2	1MB×4
mngcmd[n] ※1.log	コマンド保守ログ	<Manager のログ出力ディレクトリ>¥maintenance※2	16MB×4

注※1

ログファイル名の[n]の部分には，面の番号（1 から面数（最大 64 まで））が付きます。

注※2

<Manager のログ出力ディレクトリ>は，manager.cfg（Manager ログ設定ファイル）で指定されたディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥manager¥log です。manager.cfg の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.9　manager.cfg（Manager 設定ファイル)」を参照してください。

表 4–11　統合ログの出力先（UNIX の場合）

ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数
mngmessage[n] ※1.log	統合メッセージログ	<Manager のログ出力ディレクトリ>/message※2	256KB×4
mngtrace[n] ※1.log	統合トレースログ	<Manager のログ出力ディレクトリ>/trace※2	1MB×4
mngcmd[n] ※1.log	コマンド保守ログ	<Manager のログ出力ディレクトリ>/maintenance※2	16MB×4

注※1

ログファイル名の[n]の部分には，面の番号（1 から面数（最大 64 まで））が付きます。

注※2

<Manager のログ出力ディレクトリ>は，manager.cfg（Manager ログ設定ファイル）で指定されたディレクトリを指します。デフォルト値は，/opt/Cosminexus/manager/log です。manager.cfg の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.9　manager.cfg（Manager 設定ファイル)」を参照してください。

なお，統合ログに出力したログは，デフォルトでは個別のログとしても出力されます。

また，統合ログに出力した場合には個別に出力しない設定にできるログもあります。統合ログの出力に関する設定については，「3.3.10　Manager のログ取得の設定」を参照してください。

個別に取得する場合の運用管理エージェント，運用監視エージェント，および Management Server のログの出力先，および統合メッセージログ・統合トレースログへの出力の有無を次の表に示します。

表 4-12　運用管理エージェント，運用監視エージェント，および Management Server のログを個別に取得する場合の出力先（Windows の場合）

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
運用管理エージェント	adminagent.err.[1-16].log※1	運用管理エージェントの標準エラー出力	<Manager のログ出力ディレクトリ>	64KB×4	×
	adminagent.out.[1-16].log※1	運用管理エージェントの標準出力		64KB×4	×
	adminagent.err	運用管理エージェントの標準エラー出力コマンドライン		－	×
	adminagent[1-16].log	運用管理エージェントのログ		64KB×4	○
	adminagentctl.exe.[1-2].log	運用管理エージェントの起動・停止コマンドのログ		64KB×2	×
	adminagent[n]※2.log	運用管理エージェントの保守ログ	<Manager のログ出力ディレクトリ>¥maintenance	16MB×4	×
	mngrmi[n] ※2.log	運用管理エージェントが行う RMI 処理での保守ログ		16MB×8	×
	processConsole[n]※2.log	コンソールログ	<Manager のログ出力ディレクトリ>	64KB×4	×
	adminagentsv.exe.[1-16]log※1	運用管理エージェントサービスのログ		64KB×2	×
	adminagentsv.exe.out※1	運用管理エージェントサービスの標準出力		－	×
	adminagentsv.exe.err※1	運用管理エージェントサービスの標準エラー出力		－	×
	adminagent.javalog[01-04].log	運用管理エージェントの JavaVM ログファイル		256KB×4	×
運用監視エージェント	mngagent-<ドメイン名>-<Agent名>.[n]※2.log※3	• 運用監視エージェントのログ・トレース • J2EE サーバ用システム JP1 イベントおよび J2EE サーバ用ユーザ JP1 イベントのログ※4		64KB×4	×

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
		• Management イベント発行ログ※5			
Management Server	mngsvr.exe.[1-2].log	Management Server サービスのログ		64KB×2	×
	mngsvr.exe.err※1	Management Server サービスの標準エラー出力		−	×
	mngsvr.exe.out※1	Management Server サービスの標準出力		−	×
	mngsvrctl.exe.[1-2].log	Management Server サービス起動・停止コマンドのログ		64KB×2	×
	mngsvr[n]※2.log	• Management Server のログ • 運用管理サーバ※6 のシステム JP1 イベントのログ※4		64KB×4	○
	mngsvr[n]※2.log	Management Server の保守ログ	<Manager のログ出力ディレクトリ>¥maintenance	16MB×2	×
	cjmessage[n].log	稼働ログ	<Manager のログ出力ディレクトリ>	1MB×2	×
	cjexception[n].log	障害発生時の例外情報	<Manager のログ出力ディレクトリ>¥maintenance	1MB×2	×
	cjmaintenance[n].log	保守情報		16MB×4	×
	cjconsole[n].log	コンソールメッセージ		1MB×2	×
	cjejbcontainer[n].log	EJB コンテナの保守情報		1MB×2	×
	web_servlet[n].log	Web サーブレットログ		4MB×4	×
	user_out[n].log	ユーザ出力ログ		1MB×2	×
	user_err[n].log	ユーザエラーログ		1MB×2	×
	cjlogger.log	ログ稼働ログ		1MB×2	×
	javalog[nn].log	JavaVM の保守情報および GC のログ		4MB×4	×

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
	ehjavalog[nn].log	明示管理ヒープ機能のイベントログ		4MB×4	×
	cjwebcontainer[n].log	Web コンテナの保守情報		1MB×2	×
	cjstdout.log	起動プロセス標準出力情報	<Manager のログ出力ディレクトリ>¥maintenance/CC/maintenance	−	×
	cjstderr.log	起動プロセス標準エラー情報		−	×
	cj_shutdown[n].log	終了プロセス情報		4KB×4	×
	cjrmi[n].log	Management Server が行う RMI 処理での J2EE サーバの RMI 通信ログ	<Manager のログ出力ディレクトリ>¥maintenance¥CC¥rmi	1MB×4	×
	cjhttp_thr.<時間情報>.inprocess_http.mm	スレッドトレースの情報	<Manager のログ出力ディレクトリ>¥maintenance¥http¥maintenance¥thr	約 3.2MB×16	×
	cjhttp_comm.<時間情報>.inprocess_http.mm	通信トレースの情報	<Manager のログ出力ディレクトリ>¥maintenance¥http¥maintenance¥comm	約 16.6MB×16	×
	cjmemorywatch[n].log	リソース枯渇監視のログ（メモリ）	<Manager のログ出力ディレクトリ>¥maintenance¥watch	1MB×2	×
	cjthreadwatch[n].log	リソース枯渇監視のログ（スレッド）		1MB×2	×
	cjthreaddumpwatch[n].log	リソース枯渇監視のログ（スレッドダンプ）		1MB×2	×
	cjrequestqueuewatch[n].log	リソース枯渇監視のログ（HTTP リクエスト実行待ちキュー）		1MB×2	×
	cjhttpsessionwatch[n].log	リソース枯渇監視のログ（HTTP セッション数）		1MB×2	×

（凡例）

○：統合ログに出力される

×：統合ログには出力されない

[1-n]：1～n のログの面数の通し番号が付くことを示す

－：該当しない

注

<Manager のログ出力ディレクトリ>は，manager.cfg（Manager ログ設定ファイル）で指定されたディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥manager¥log（Windows の場合），または/opt/Cosminexus/manager/log（UNIX の場合）です。manager.cfg の詳細については，マニュアル「アプリケーションサーバリファレンス 定義編(サーバ定義)」の「8.2.9　manager.cfg（Manager 設定ファイル）」を参照してください。

注※1

トレース共通ライブラリ形式以外の形式でログが出力されます。トレース共通ライブラリ形式のログについては，「5.2　アプリケーションサーバのログ」を参照してください。

注※2

ファイル名の[n]の部分には，1～指定したログの面数の通し番号が付きます。

注※3

運用監視エージェントのログ・トレースの出力先は変更できます。運用監視エージェントのログ・トレースの出力先を変更している場合は，mngagent.properties ファイル（運用監視エージェントプロパティファイル）の mngagent.log.filename キーの値を参照してください。

注※4

JP1 と連携してアプリケーションサーバで構築したシステムを運用する場合に出力されます。

注※5

Management イベントを使用している場合に出力されます。

注※6

アプリケーションサーバの運用管理サーバのことです。

表 4-13　運用管理エージェント，運用監視エージェント，および Management Server のログを個別に取得する場合の出力先（UNIX の場合）

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
運用管理エージェント	adminagent.err.[1-16].log※1	運用管理エージェントの標準エラー出力	<Manager のログ出力ディレクトリ>	64KB×4	×
	adminagent.out.[1-16].log※1	運用管理エージェントの標準出力		64KB×4	×
	adminagent.err	運用管理エージェントの標準エラー出力コマンドライン		－	×
	adminagent[1-16].log	運用管理エージェントのログ		64KB×4	○
	adminagentctl.[1-16].log	運用管理エージェントの起動・停止コマンドのログ		64KB×2	×
	adminagent[n]※2.log	運用管理エージェントの保守ログ	<Manager のログ出力ディレクトリ>/maintenance	16MB×4	×

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
	mngrmi[n] ※2.log	運用管理エージェントが行う RMI 処理での保守ログ		16MB×8	×
	processConsole[n]※2.log	コンソールログ	<Manager のログ出力ディレクトリ>	64KB×4	×
	adminagent.javalog[01-04].log	運用管理エージェントの JavaVM ログファイル		256KB×4	×
運用監視エージェント	mngagent-<ドメイン名>-<Agent名>.[n]※2.log※3	• 運用監視エージェントのログ・トレース • J2EE サーバ用システム JP1 イベントおよび J2EE サーバ用ユーザ JP1 イベントのログ※4 • Management イベント発行ログ※5		64KB×4	×
Management Server	mngsvrctlstart.[1-2].log	Management Server 起動コマンド		64KB×2	×
	mngsvrctlstop.[1-2].log	Management Server 停止コマンド		64KB×2	×
	mngsvrctlsetup.[1-2].log	Management Server セットアップコマンド		64KB×2	×
	mngsvr[n].log	• Management Server のログ • 運用管理サーバ※6 のシステム JP1 イベントのログ※4		64KB×4	○
	mngenvsetup.[1-2].log	mngenvsetup コマンドの実行ログ	<Manager のログ出力ディレクトリ>/maintenance	512KB×2	×
	mngsvr[n]※2.log	Management Server の保守ログ	<Manager のログ出力ディレクトリ>/maintenance	16MB×2	×
	cjmessage[n].log	稼働ログ	<Manager のログ出力ディレクトリ>	1MB×2	×
	cjexception[n].log	障害発生時の例外情報	<Manager のログ出力ディレクトリ>/maintenance	1MB×2	×

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
	cjmaintenance[n].log	保守情報		16MB×4	×
	cjconsole[n].log	コンソールメッセージ		1MB×2	×
	cjejbcontainer[n].log	EJB コンテナの保守情報		1MB×2	×
	web_servlet[n].log	Web サーブレットログ		4MB×4	×
	user_out[n].log	ユーザ出力ログ		1MB×2	×
	user_err[n].log	ユーザエラーログ		1MB×2	×
	cjlogger.log	ログ稼働ログ		1MB×2	×
	javalog[nn].log	JavaVM の保守情報および GC のログ		4MB×4	×
	ehjavalog[nn].log	明示管理ヒープ機能のイベントログ		4MB×4	×
	cjwebcontainer[n].log	Web コンテナの保守情報		1MB×2	×
	cjstdout.log	起動プロセス標準出力情報	<Manager のログ出力ディレクトリ>/maintenance/CC/maintenance	−	×
	cjstderr.log	起動プロセス標準エラー情報		−	×
	cj_shutdown[n].log	終了プロセス情報		4KB×4	×
	cjrmi[n].log	Management Server が行う RMI 処理での J2EE サーバの RMI 通信ログ	<Manager のログ出力ディレクトリ>/maintenance/CC/rmi	1MB×4	×
	cjhttp_thr.<時間情報>.inprocess_http.mm	スレッドトレースの情報	<Manager のログ出力ディレクトリ>/maintenance/http/maintenance/thr	約 3.2MB×16	×
	cjhttp_comm.<時間情報>.inprocess_http.mm	通信トレースの情報	<Manager のログ出力ディレクトリ>/maintenance/http/maintenance/comm	約 16.6MB×16	×
	cjmemorywatch[n].log	リソース枯渇監視のログ（メモリ）	<Manager のログ出力ディレクトリ>/maintenance/watch	1MB×2	×

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
	cjfiledescriptorwatch[n].log	リソース枯渇監視のログ（ファイルディスクリプタ）		1MB×2	×
	cjthreadwatch[n].log	リソース枯渇監視のログ（スレッド）		1MB×2	×
	cjthreaddumpwatch[n].log	リソース枯渇監視のログ（スレッドダンプ）		1MB×2	×
	cjrequestqueuewatch[n].log	リソース枯渇監視のログ（HTTP リクエスト実行待ちキュー）		1MB×2	×
	cjhttpsessionwatch[n].log	リソース枯渇監視のログ（HTTP セッション数）		1MB×2	×

（凡例）

○：統合ログに出力される

×：統合ログには出力されない

[1-n]：1～n のログの面数の通し番号が付くことを示す

－：該当しない

注※1

トレース共通ライブラリ形式以外の形式でログが出力されます。トレース共通ライブラリ形式のログについては，「5.2　アプリケーションサーバのログ」を参照してください。

注※2

ファイル名の[n]の部分には，1～指定したログの面数の通し番号が付きます。

注※3

運用監視エージェントのログ・トレースの出力先は変更できます。運用監視エージェントのログ・トレースの出力先を変更している場合は，mngagent.properties ファイル（運用監視エージェントプロパティファイル）の mngagent.log.filename キーの値を参照してください。

注※4

JP1 と連携してアプリケーションサーバで構築したシステムを運用する場合に出力されます。

注※5

Management イベントを使用している場合に出力されます。

注※6

アプリケーションサーバの運用管理サーバのことです。

注意事項

コンソールログには，運用管理エージェントが起動したサーバプロセスの標準出力，および標準エラー出力が出力されます。コンソールログについての注意事項を次に示します。

- Windows の場合，次のプロセスに対しては，コンソールログは出力されません。

論理パフォーマンストレーサ

論理 Web サーバ

論理 CTM ドメインマネジャ

論理 CTM

間接起動の論理ユーザサーバ

論理ユーザサーバの起動種別については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.19　論理ユーザサーバ定義ファイル」を参照してください。

- コンソールログに出力する情報が一度に複数行出力された場合，コンソールログには 1 行にまとめて表示されることがあります。
- コンソール情報に出力する情報の文字数が 2039 文字を超えてしまう場合，2039 文字以降の情報は次の行に分割されて出力されます。

(3)　仮想サーバマネージャの内部構築ツールおよびサーバ通信エージェントのログの取得

ここでは，仮想サーバマネージャの内部構築ツールおよびサーバ通信エージェントのログの出力先について説明します。

仮想サーバマネージャの内部構築ツールおよびサーバ通信エージェントのログの出力先，および統合メッセージログ・統合トレースログへの出力の有無を次の表に示します。

表 4-14　仮想サーバマネージャの内部構築ツールおよびサーバ通信エージェントのログの出力先

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
仮想サーバマネージャの内部構築ツール	rasetup[n]※2.log	仮想サーバマネージャの内部構築ツールのログ	Windows の場合 <Application Server のインストールディレクトリ>¥manager¥setup¥log UNIX の場合 /opt/Cosminexus/manager/setup/log	262144 バイト×4	×
	rasetup[n]※2.log	仮想サーバマネージャの内部構築ツールの保守ログ	Windows の場合 <Application Server のインストールディレクトリ>¥manager¥setup¥log¥maintenance UNIX の場合 /opt/Cosminexus/manager/setup/log/maintenance	16777216 バイト×4	×

分類	ファイル名	内容	出力先ディレクトリ	デフォルトのサイズ×面数	統合メッセージログ／統合トレースログ
サーバ通信エージェント	sinaviagent[n]※2.log	サーバ通信エージェントのログ	<サーバ通信エージェントのログ出力ディレクトリ>※1	524288 バイト×4	×
	sinaviagentsv[n]※2.log	サーバ通信エージェントサービスのログ		65536 バイト×4	×
	snactl[n]※2.log	サーバ通信エージェントの起動・停止コマンドのログ		65536 バイト×4	×
	sinaviagent.err	サーバ通信エージェントの標準エラー出力		65536 バイト×1	×
	sinaviagent.out	サーバ通信エージェントの標準出力		65536 バイト×1	×
	processConsole[n]※2.log	コンソールログ		65536 バイト×4	×
	sinaviagent[n]※2.log	サーバ通信エージェントの保守ログ	Windows の場合 <Application Server のインストールディレクトリ>¥sinagent¥log¥maintenance UNIX の場合 /opt/Cosminexus/sinagent/log/maintenance	1048576 バイト×4	×
	sinaviagentsv[n]※2.log	サーバ通信エージェントサービスの保守ログ		65536 バイト×4	×
	snactl[n]※2.log	サーバ通信エージェントの起動・停止コマンドの保守ログ		65536 バイト×4	×
	sinaviagent.javalog[n].※2log	サーバ通信エージェントの JavaVM ログファイル		256KB×4	×

（凡例）×：統合ログには出力されない　[n]：1～n のログの面数の通し番号が付くことを示す

注※1

<サーバ通信エージェントのログ出力ディレクトリ>は，sinaviagent.cfg（サーバ通信エージェント用オプション定義ファイル）で指定されたディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥sinagent¥log（Windows の場合），または/opt/Cosminexus/sinagent/log（UNIX の場合）です。

注※2

ファイル名の[n]の部分には，1 から順に指定したログの面数の通し番号が付きます。

(4) 統合ユーザ管理のログの取得

統合ユーザ管理のトレースファイルは，ua.conf ファイル（統合ユーザ管理のコンフィグレーションファイル）の com.cosminexus.admin.auth.trace.prefix オプションの設定に応じて出力されます。ua.conf ファイルの詳細については，マニュアル「アプリケーションサーバ 機能解説 セキュリティ管理機能編」の「14.2.2 ua.conf（統合ユーザ管理のコンフィグレーションファイル)」を参照してください。

(5) CJMS プロバイダのログの取得

CJMS プロバイダのログの取得について説明します。CJMS プロバイダで取得できるログの種類には，CJMSP ブローカーのログ，管理コマンド（cjmsicmd）のログ，および CJMSP リソースアダプタのログがあります。ログの出力先のデフォルトを次の表に示します。

表 4-15 CJMS プロバイダのログの出力先（デフォルト）

ログの種類	分類	デフォルトの出力先	デフォルトのサイズ×面数
CJMSP ブローカーのログ	メッセージログ	• Windows の場合 <CJMSP_HOME>※1¥var¥instances¥<instanceName>¥log¥cjmsbroker_msg[n].log • UNIX の場合 <CJMSP_HOME>※1/var/instances/<instanceName>/log/cjmsbroker_msg[n].log	1MB×2
	エラーログ	• Windows の場合 <CJMSP_HOME>※1¥var¥instances¥<instanceName>¥log¥cjmsbroker_err[n].log • UNIX の場合 <CJMSP_HOME>※1/var/instances/<instanceName>/log/cjmsbroker_err[n].log	1MB×2
管理コマンド（cjmsicmd）のログ※2	メッセージログ	• Windows の場合 <CJMSP_HOME>※1¥var¥admin¥log¥cjmsadmin_msg[n].log • UNIX の場合 <CJMSP_HOME>※1/var/admin/log/cjmsadmin_msg[n].log	1MB×2
	エラーログ	• Windows の場合 <CJMSP_HOME>※1¥var¥admin¥log¥cjmsadmin_err[n].log • UNIX の場合 <CJMSP_HOME>※1/var/admin/log/cjmsadmin_err[n].log	1MB×2

ログの種類	分類	デフォルトの出力先	デフォルトのサイズ×面数
CJMSP リソースアダプタのログ	メッセージログ	• Windows の場合 <J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>※3¥cjms¥Cosminexus_JMS_Provider_RA¥cjmsra_msg[n].log • UNIX の場合 <J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>※3/cjms/Cosminexus_JMS_Provider_RA/cjmsra_msg[n].log	1MB×2
	エラーログ	• Windows の場合 <J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>※3¥cjms¥Cosminexus_JMS_Provider_RA¥cjmsra_err[n].log • UNIX の場合 <J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>※3/cjms/Cosminexus_JMS_Provider_RA/cjmsra_err[n].log	1MB×2

注

[n]には，面の番号（1 から面数（最大 16 まで））が付きます。

注※1

<CJMSP_HOME>は，次のディレクトリです。

Windows の場合

<Application Server のインストールディレクトリ>¥CC¥cjmsp

UNIX の場合

/opt/Cosminexus/CC/cjmsp

注※2

ログ出力の結果，指定した最大ファイルサイズになると，ログの出力先が次のファイルに切り替わります。ファイル数が指定した面数に達した場合，ラップアラウンドによって出力先が最初のファイルに切り替わり，元の情報は上書きされます。なお，ファイルは作成時にだけ初期化され，ラップアラウンドの際には初期化されません。ファイルサイズを拡張した場合は，拡張部分に対してだけ初期化されます。初期化とは，空白文字（0x20）を EOF（ファイルの最後）から指定したファイルサイズまでの領域に書き込むことです。既存のデータに影響はありません。

注※3

<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>は，J2EE サーバのオプション定義で指定したディレクトリです。デフォルトでは，次のディレクトリになります。

Windows の場合

<ejb.public.directory で指定したディレクトリ>¥ejb¥<J2EE サーバ名>¥logs

UNIX の場合

<ejb.public.directory で指定したディレクトリ>/ejb/<J2EE サーバ名>/logs

(6) ログ以外に取得が必要な情報

ここでは，ログ以外に取得する必要がある情報について説明します。

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

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，インプロセストランザクションサービスのステータスファイルを取得する必要があります。なお，ステータスファイルを二重化している場合は，予備のステータスファイルも取得してください。

ステータスファイルは，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejbserver.distributedtx.ots.status.directory1 パラメタ，および ejbserver.distributedtx.ots.status.directory2 パラメタ（二重化している場合）で指定したパスに格納されています。

4.3.2 Performance Tracer のログの取得

Performance Tracer のログの種類とログの出力先について説明します。

(1) Performance Tracer のログの種類

Performance Tracer では，PRF 識別子ごとに PRF デーモン，PRF コマンドのログを出力します。また，システムでトラブルが発生したときの保守員による障害解析用のログ（各種保守情報）を環境変数 PRFSPOOL の設定ディレクトリに出力します。

日常的な運用で監視する場合はイベントログ，および syslog を監視します。

(2) Performance Tracer のログの出力先

Performance Tracer のログの出力先を次に示します。

表 4-16 Performance Tracer のログの出力先

内容	出力先ディレクトリ※1
PRF デーモンおよび PRF コマンドのログ	• Windows の場合 <環境変数 PRFSPOOL の設定ディレクトリ>¥log¥<PRF 識別子>¥ctmlog[n] およびイベントログ※2 • UNIX の場合 $PRFSPOOL/log/<PRF 識別子>/ctmlog[n] および syslog※3
モジュールトレース	• Windows の場合 <環境変数 PRFSPOOL の設定ディレクトリ>¥utt¥umt • UNIX の場合 $PRFSPOOL/utt/umt

内容	出力先ディレクトリ※1
構造化例外発生ログ （Windows の場合）	<環境変数 PRFSPOOL の設定ディレクトリ>¥oslttrc
保守情報	• Windows の場合 <環境変数 PRFSPOOL の設定ディレクトリ> • UNIX の場合 $PRFSPOOL/

注※1

[n]には「01」または「02」が表示されます。

注※2

運用に必要なメッセージが出力されます。ログファイルの出力先は Windows のイベントログの設定によって異なります。

注※3

運用に必要なメッセージが出力されます。syslog の設定に関する詳細については，OS 付属のマニュアルを参照してください。

4.3.3 Component Transaction Monitor のログの取得

ここでは，Component Transaction Monitor のログの出力先について説明します。

(1) Component Transaction Monitor のログの種類

Component Transaction Monitor では，CTM 識別子ごとに CTM デーモン，CTM コマンドのログを出力します。また，システムでトラブルが発生したときの保守員による障害解析用に，各種保守情報を環境変数 CTMSPOOL の設定ディレクトリに出力します。ctmlogcat コマンドを実行することで出力メッセージを確認できます。

日常的な運用で監視する場合はイベントログ，および syslog を監視します。

(2) Component Transaction Monitor のログの出力先

Component Transaction Monitor のログの出力先を次に示します。

表 4-17 Component Transaction Monitor のログの出力先

内容	出力先ディレクトリ※1
CTM デーモンおよび CTM コマンドのログ	• Windows の場合 <環境変数 CTMSPOOL の設定ディレクトリ>¥log¥<CTM デーモン識別子>¥ctmlog[n] およびイベントログ※2 • UNIX の場合 $CTMSPOOL/log/<CTM デーモン識別子>/ctmlog[n] および syslog※3

内容	出力先ディレクトリ※1
CTM ドメインデーモンのログ	• Windows の場合 <環境変数 CTMSPOOL の設定ディレクトリ>¥log¥ctmdmlog[n] • UNIX の場合 $CTMSPOOL/log/ctmdmlog[n]
モジュールトレース	• Windows の場合 <環境変数 CTMSPOOL の設定ディレクトリ>¥utt¥umt • UNIX の場合 $CTMSPOOL/utt/umt
構造化例外発生ログ （Windows の場合）	<環境変数 PRFSPOOL の設定ディレクトリ>¥oslttrc
保守情報	• Windows の場合 <環境変数 CTMSPOOL の設定ディレクトリ> • UNIX の場合 $CTMSPOOL

注※1

[n]には「01」または「02」が表示されます。

注※2

運用に必要なメッセージが出力されます。ログファイルの出力先は Windows のイベントログの設定によって異なります。

注※3

運用に必要なメッセージが出力されます。syslog の設定に関する詳細については，OS 付属のマニュアルを参照してください。

4.3.4　監査ログで出力するログの取得

ここでは，監査ログの機能を使用した場合に出力されるログファイルについて説明します。

監査ログで出力するログの種類と出力先を次の表に示します。

表 4–18　監査ログで出力するのログの出力先（デフォルト）

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
メッセージログ	監査ログのメッセージログ	• Windows の場合 <Application Server のインストールディレクトリ>¥auditlog¥rasmessage[n].log • UNIX の場合 /opt/Cosminexus/auditlog/rasmessage[n].log	1MB×4
例外ログ	監査ログの例外情報	• Windows の場合 <Application Server のインストールディレクトリ>¥auditlog¥rasexception[n].log • UNIX の場合	1MB×8

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
		/opt/Cosminexus/auditlog/rasexception[n].log	

注※

ログファイル名の[n]の部分には，面の番号（1 から面数（最大 64 まで））が付きます。

4.3.5　アプリケーションのユーザログの取得

アプリケーションのユーザログは，トレース共通ライブラリ形式で出力するように設定している場合に取得できます。ユーザログには，次の 2 種類があります。

- J2EE サーバ上で動作する J2EE アプリケーション（J2EE コンポーネント）が出力するユーザログ
- EJB クライアントアプリケーションが出力するユーザログ

なお，トレース共通ライブラリ形式のログについては，「5.2　アプリケーションサーバのログ」を参照してください。

(1)　J2EE アプリケーションのユーザログの取得

J2EE アプリケーションのログの出力先については，次のディレクトリに，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に ejbserver.application.userlog.CJLogHandler.<ハンドラ名>.path パラメタで指定したプリフィックスを持つファイル名で出力されます。なお，<ハンドラ名>には，キーの値を区別するためのハンドラ名が指定されています。

- Windows の場合

  <ログ出力先ルート（ejb.server.log.directory の値）>¥user¥（デフォルトは<J2EE サーバの作業ディレクトリ>¥ejb¥<J2EE サーバ名>¥logs¥user）
- UNIX の場合

  <ログ出力先ルート（ejb.server.log.directory の値）>/user/（デフォルトは<J2EE サーバの作業ディレクトリ>/ejb/<J2EE サーバ名>/logs/user）

なお，J2EE アプリケーションのユーザログ出力の設定については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「8.8　J2EE アプリケーションのユーザログ出力の設定」を参照してください。

(2)　EJB クライアントアプリケーションのユーザログの取得

EJB クライアントアプリケーションのユーザログの出力先については，次のディレクトリに EJB クライアントアプリケーションのシステムプロパティの ejbserver.application.userlog.CJLogHandler.<ハンドラ名>.path キーの値で指定したプリフィックスを持つファイル名で出力されます。なお，<ハンドラ名>には，キーの値を区別するためのハンドラ名が指定されています。

- Windows の場合

  <ログ出力先ルート（ejb.server.log.directory の値）>¥user¥（デフォルトは<J2EE サーバの作業ディレクトリ>¥ejb¥<J2EE サーバ名>¥logs¥user）

- UNIX の場合

  <ログ出力先ルート（ejb.server.log.directory の値）>/user/（デフォルトは<J2EE サーバの作業ディレクトリ>/ejb/<J2EE サーバ名>/logs/user）

EJB クライアントアプリケーションでユーザログを出力するためのシステムプロパティの設定方法については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「8.10　EJB クライアントアプリケーションのユーザログ出力の設定（cjclstartap コマンドを使用する場合)」，またはマニュアル「アプリケーションサーバ 機能解説 拡張編」の「8.11　EJB クライアントアプリケーションのユーザログ出力の実装と設定（vbj コマンドを使用する場合)」を参照してください。

4.4 アプリケーションサーバのログ（バッチアプリケーションを実行するシステム）

この節では，バッチアプリケーションを実行するシステムで，アプリケーションサーバの構成ソフトウェアが出力するログを手動で取得する方法について説明します。なお，snapshot ログとしてアプリケーションサーバのログをすでに収集している場合は，ここで説明する操作は不要です。

ここでは，次のログの取得方法について説明します。

- Component Container のログ
- アプリケーションのユーザログ

また，バッチアプリケーションを実行するシステムでは，アプリケーションサーバの構成ソフトウェアが出力するログを手動で取得する場合，次のログも取得してください。

- Performance Tracer のログ
- 監査ログで出力するログ

これらのログの取得方法については，「4.3　アプリケーションサーバのログ（J2EE アプリケーションを実行するシステム）」を参照してください。それぞれの参照先の詳細を次に示します。

- **Performance Tracer のログ**
  「4.3.2　Performance Tracer のログの取得」
- **監査ログで出力するログ**
  「4.3.4　監査ログで出力するログの取得」

4.4.1 Component Container のログの取得（バッチアプリケーションを実行するシステム）

Component Container のログの種類とログの出力先について説明します。Component Container のログには，次の 2 種類のログがあります。

- バッチサーバ・サーバ管理コマンド・バッチアプリケーションのログ
- 運用管理エージェント・運用監視エージェント・Management Server のログ

それぞれのログの出力先について説明します。

(1) バッチサーバ・サーバ管理コマンド・バッチアプリケーションのログの取得

バッチサーバ，サーバ管理コマンド，およびバッチアプリケーションのログの取得方法について説明します。

また，Component Container では，これらのログに加えて，移行コマンドのログが出力されます。また，リソース枯渇監視機能を使用している場合はリソース枯渇監視ログが出力されます。

- バッチサーバのログには，メッセージログ，ユーザログ，例外ログ，および保守用ログの 4 種類があります。なお，バッチサーバでは，これらのログに加えて，起動，停止および異常終了時にイベントログまたは syslog を出力します。
- サーバ管理コマンドのログには，メッセージログ，例外ログ，および保守用ログの 3 種類があります。
- バッチアプリケーションのログは，メッセージログの 1 種類です。
- リソースアダプタのバージョンアップコマンド（cjrarupdate）のログには，メッセージログ，例外ログ，および保守用ログの 3 種類があります。
- 移行コマンドのログには，メッセージログ，例外ログ，および保守用ログの 3 種類があります。

次にそれぞれのログについて説明します。

メッセージログ

バッチサーバ，サーバ管理コマンド，移行コマンドなどの稼働状態が出力されます。各種サーバおよびコマンドの稼働監視の情報として使用します。

ユーザログ

アプリケーション中で出力される標準出力および標準エラー出力の情報が出力されます。アプリケーションの開発時の動作確認用に使用します。なお，java.security.debug プロパティを指定してサーバを起動した場合，標準出力および標準エラー出力の情報はユーザログに出力されません。JavaVM のメモリ関連ログも含みます。

例外ログ

システムでトラブルが発生したときの Component Container の例外情報が出力されます。なお，例外ログは日常的な運用で監視する必要はありません。ログにメッセージが出力された場合に，例外情報を参照するときにご利用ください。

保守用ログ

システムでトラブルが発生したときの Component Container の障害保守情報が出力されます。保守員が Component Container の障害解析用に使用します。

イベントログ（Windows の場合）

バッチサーバが起動，停止または異常終了したことを示す情報が出力されます。出力先は Windows のイベントログの設定によって異なります。

なお，イベントログは，バッチサーバの停止のしかたによっては，出力されません。次の場合は，正しくログが出力されないことがあります。

- バッチサーバが動作している JavaVM 自体に問題が発生した場合
- バッチサーバのプロセスを TerminateProcess によって外部から停止した場合
- JavaVM の起動オプションとして-XX:+HitachiOutOfMemoryAbort オプションを指定している場合にメモリ不足によってバッチサーバが異常終了したとき

なお，-XX:+HitachiOutOfMemoryAbort オプションは，デフォルトで設定されているオプションです。

syslog（UNIX の場合）

バッチサーバが起動，停止または異常終了したことを示す情報が出力されます。出力先は UNIX の syslog の設定によって異なります。

なお，syslog は，バッチサーバの停止のしかたによっては，出力されません。次の場合は，正しくログが出力されないことがあります。

- バッチサーバが動作している JavaVM 自体に問題が発生した場合
- バッチサーバのプロセスを SIGKILL シグナル（kill -9 など）によって外部から停止した場合
- JavaVM の起動オプションとして-XX:+HitachiOutOfMemoryAbort オプションを指定している場合にメモリ不足によってバッチサーバが異常終了したとき

  -XX:+HitachiOutOfMemoryAbort オプションは，デフォルトで設定されているオプションです。

リソース枯渇監視ログ

リソース枯渇監視機能を使用している場合に，監視対象のリソースについてのリソース枯渇監視情報が出力されます。リソースの使用率または使用数がしきい値を超えた場合の原因調査に使用します。

ログは，若い面番号の付いたログファイルから順に記録されます。一つのログファイルのサイズが 1 面当たりの最大サイズに達すると，ログは次の面番号の付いたログファイルに記録されます。最後のログファイル（面数の番号が付いたログファイル）のサイズが 1 面当たりの最大サイズに達すると，面の番号 1 のログファイルを空にし，そこへログを記録していきます。以降，ログファイルを空にしながら，面番号の順にログファイルへログを記録していきます。

ログの出力先のデフォルトを次の表に示します。Component Container のログは，サーバ単位またはコマンド単位に取得できます。

ログの出力先に示す<作業ディレクトリ>は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.public.directory パラメタで指定したディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥CC¥server¥public（Windows の場合），または/opt/Cosminexus/CC/server/public（UNIX の場合）です。

(a) バッチサーバのログ

表 4-19 バッチサーバのログの出力先（デフォルト）

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
メッセージログ	稼働ログ	• Windows の場合 <ejb.server.log.directory>※2¥cjmessage[n].log • UNIX の場合	1MB×2	Message LogFile

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
		<ejb.server.log.directory>※2/cjmessage[n].log		
	ログ稼働ログ※3	• Windows の場合 <ejb.server.log.directory>※2¥cjlogger.log • UNIX の場合 <ejb.server.log.directory>※2/cjlogger.log	1MB×2	−
	バッチサーバにデプロイして使用するリソースアダプタの稼働ログ※4	• Windows の場合 <ejb.server.log.directory>※2¥connectors¥<リソースアダプタの表示名>[n].log • UNIX の場合 <ejb.server.log.directory>※2/connectors/<リソースアダプタの表示名>[n].log	1MB×2	−
ユーザログ	ユーザ出力ログ	• Windows の場合 <ejb.server.log.directory>※2¥user_out[n].log • UNIX の場合 <ejb.server.log.directory>※2/user_out[n].log	1MB×2	UserOutLogFile
	ユーザエラーログ	• Windows の場合 <ejb.server.log.directory>※2¥user_err[n].log • UNIX の場合 <ejb.server.log.directory>※2/user_err[n].log	1MB×2	UserErrLogFile
	JavaVM の保守情報および GC のログ	• Windows の場合 <ejb.server.log.directory>※2¥javalog[nn].log • UNIX の場合 <ejb.server.log.directory>※2/javalog[nn].log	4MB×4	−
	明示管理ヒープ機能のイベントログ	• Windows の場合 <ejb.server.log.directory>※2¥ehjavalog[nn].log • UNIX の場合	4MB×4	−

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
		<ejb.server.log.directory>※2/ehjavalog[nn].log		
例外ログ	障害発生時の例外情報	• Windows の場合 <ejb.server.log.directory>※2¥cjexception[n].log • UNIX の場合 <ejb.server.log.directory>※2/cjexception[n].log	1MB×2	ExceptionLogFile
保守用ログ	保守情報	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjmaintenance[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjmaintenance[n].log	16MB×4	MaintenanceLogFile
	コンソールメッセージ	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjconsole[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjconsole[n].log	1MB×2	ConsoleLogFile
	EJB コンテナの保守情報	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjejbcontainer[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjejbcontainer[n].log	1MB×2	EJBContainerLogFile
	起動プロセス標準出力情報※5	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjstdout.log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cjstdout.log	−	−
	起動プロセス標準エラー情報※5	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cjstderr.log • UNIX の場合	−	−

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
		<ejb.server.log.directory>※2/CC/maintenance/cjstderr.log		
	終了プロセス情報	• Windows の場合 <ejb.server.log.directory>※2¥CC¥maintenance¥cj_shutdown[n].log • UNIX の場合 <ejb.server.log.directory>※2/CC/maintenance/cj_shutdown[n].log	4KB×2※6	−
イベントログ	バッチサーバの起動，停止または異常終了を示すログ	Windows のイベントビューアのアプリケーションログ※7	−	−
syslog	バッチサーバの起動，停止または異常終了を示すログ	UNIX の syslog の設定に依存します。※8	−	−

（凡例） −：該当しない

注

チャネル名は，ログの出力先を識別する名称です。

注※1

ログファイル名の[n]の部分には，面の番号（1 から各ログの最大面数まで）が付きます。

また，[nn]の部分には，01～99 の通し番号が付きます。

注※2

<ejb.server.log.directory>は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.server.log.directory パラメタで指定したディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs です。

ejb.server.log.directory パラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「3.2.1　usrconf.cfg（バッチサーバ用オプション定義ファイル）」を参照してください。

注※3

ファイル出力時にファイル容量をチェックします。チェック時に最大容量を超えた場合，cjlogger.log ファイルの名称をバックアップファイルの名称（cjlogger_save.log）に変更します。

注※4

リソースアダプタのログを取得するかどうかは，サーバ管理コマンドで指定されている内容に従います。また，リソースアダプタのログは，簡易構築定義ファイルでサイズおよび面数を変更できます。リソースアダプタのログ取得の設定については，「3.3.11　リソースアダプタのログ取得の設定」を参照してください。

注※5

起動プロセス情報だけを取得するログです。主にバッチサーバの起動または終了時に出力されるため，オンライン中に出力されることはほとんどありません。ファイルのサイズが上限に達したときは，<作業ディレクトリ>¥ejb¥<サーバ名称>¥logs 下（Windows の場合），または<作業ディレクトリ>/ejb/<サーバ名称>/logs 下（UNIX の場合）の cjstdout_save.log または cjstderr_save.log に退避されます。すでに cjstdout_save.log または cjstderr_save.log があったときは，上書きされます。

注※6

サイズおよび面数は，変更できません。

注※7

ログファイルの出力先は Windows のイベントログの設定によって異なります。

注※8

バッチサーバの起動，停止および異常終了のメッセージを syslog に出力するためには，syslog の設定で，facility「daemon」に対する priority を「info」または「debug」に設定する必要があります。syslog の設定に関する詳細については，OS 付属のマニュアルを参照してください。

(b) サーバ管理コマンドのログ

表 4-20 サーバ管理コマンドのログの出力先（デフォルト）

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
メッセージログ	稼働ログ※2，※3	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥cjmessage[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/cjmessage[n].log	1024KB×3	MessageLogFile
	ログ稼働ログ※2	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥cjlogger.log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/cjlogger.log	1024KB×2	−
例外ログ	障害発生時の例外情報※2，※3	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥cjexception[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/cjexception[n].log	1024KB×6	ExceptionLogFile
保守用ログ	保守情報※2	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥CC¥maintenance¥cjmaintenance[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjmaintenance[n].log	1024KB×3	MaintenanceLogFile

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
	コンソールメッセージ※2	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥CC¥maintenance¥cjconsole[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjconsole[n].log	32KB×3	ConsoleLogFile
	サーバ管理コマンドの保守情報※2	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥CC¥maintenance¥cjserveradmin[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/CC/maintenance/cjserveradmin[n].log	32KB×3	ServerAdminLogFile

（凡例） －：該当しない

注

チャネル名は，ログの出力先を識別する名称です。ログの属性（サイズ，面数）を変更する場合のキー値として使用します。

注※1

ログファイル名の[n]の部分には，面の番号（1 から各ログの最大面数まで）が付きます。

注※2

トレース共通ライブラリ形式の出力メッセージ（アプリケーション識別名）には，コマンド名が表示されます。トレース共通ライブラリ形式のログについては，「5.2　アプリケーションサーバのログ」を参照してください。

注※3

互換モードの場合，稼働ログと障害発生時の例外情報の出力先は標準モードと異なります。互換モードの場合の出力先とデフォルトのサイズ・面数は次のようになります。

表 4-21　サーバ管理コマンドのログの出力先（互換モード）

内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
稼働ログ	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥admin¥logs¥<コマンド名称>message[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/<コマンド名称>message[n].log	128KB×2
障害発生時の例外情報	• Windows の場合	256KB×2

内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
	<Application Server のインストールディレクトリ>¥CC¥admin¥logs¥<コマンド名称>exception[n].log • UNIX の場合 /opt/Cosminexus/CC/admin/logs/<コマンド名称>exception[n].log	

注※

ログファイル名の[n]の部分には，面の番号（1 から面数（最大 16 まで））が付きます。

サーバ管理コマンドのメッセージログに出力されるメッセージには，メッセージ ID フィールドが空白で，メッセージテキストフィールドにメッセージ ID（KDJEnnnnn-Y など）が含まれる場合があります。それらはサーバ側で発生したメッセージで，前後に出力されるメッセージの付加情報となります。

(c) バッチアプリケーションのログ

表 4-22 バッチアプリケーションのログの出力先（デフォルト）

分類	内容	ログ出力先およびログファイル名※1	デフォルトのサイズ×面数
メッセージログ	cjexecjob コマンド，cjkilljob コマンドおよび cjlistjob コマンドの稼働ログ	• Windows の場合 <batch.log.directory>※2¥cjmessage[n].log • UNIX の場合 <batch.log.directory>※2/cjmessage[n].log	1MB×2

注※1

ログファイル名の[n]の部分には，面の番号（1 から面数（最大 16 まで））が付きます。

注※2

<batch.log.directory>は，バッチアプリケーション用オプション定義ファイル（usrconf.cfg ファイル）の batch.log.directory で指定されたディレクトリを指します。デフォルト値は，次のとおりです。

Windows の場合

<Application Server のインストールディレクトリ>¥CC¥batch¥logs

UNIX の場合

/opt/Cosminexus/CC/batch/logs

(d) リソースアダプタバージョンアップコマンド（cjrarupdate）のログ

表 4-23 リソースアダプタバージョンアップコマンド（cjrarupdate）のログの出力先

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
メッセージログ	稼働ログ	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjrarupdatemessage[n].log • UNIX の場合	1MB×2

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
		/opt/Cosminexus/CC/logs/cjrarupdatemessage[n].log	
例外ログ	障害発生時の例外情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjrarupdateexception[n].log • UNIX の場合 /opt/Cosminexus/CC/logs/cjrarupdateexception[n].log	1MB×2
保守用ログ	保守情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjrarupdatemaintenance[n].log • UNIX の場合 /opt/Cosminexus/CC/logs/cjrarupdatemaintenance[n].log	1MB×2

注※

ログファイル名の[n]の部分には，面の番号（1 から面数（最大 16 まで））が付きます。

(e) 移行コマンド（cjenvupdate）のログ

表 4-24 移行コマンド（cjenvupdate）のログの出力先

分類	内容	ログ出力先およびログファイル名※	デフォルトのサイズ×面数
メッセージログ	cjenvupdate コマンドの稼働ログ	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjenvupdatemessage[n].log • UNIX の場合 /opt/Cosminexus/CC/logs/cjenvupdatemessage[n].log	4MB×4
例外ログ	cjenvupdate コマンドの例外情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjenvupdateexception[n].log • UNIX の場合 /opt/Cosminexus/CC/logs/cjenvupdateexception[n].log	4MB×4
保守用ログ	cjenvupdate コマンドの保守情報	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥logs¥cjenvupdatemaintenance[n].log • UNIX の場合 /opt/Cosminexus/CC/logs/cjenvupdatemaintenance[n].log	4MB×4

注※

[n]には，面の番号（1 から面数（最大 16 まで））が付きます。

(f)　リソース枯渇監視のログ

表 4-25　リソース枯渇監視のログの出力先

監視対象リソース	ログ取得場所およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
メモリ	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjmemorywatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjmemorywatch[n].log	1MB×2	MemoryWatchLogFile
ファイルディスクリプタ	• UNIX の場合※3 <ejb.server.log.directory>※2/watch/cjfiledescriptorwatch[n].log	1MB×2	FileDescriptorWatchLogFile
スレッド	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjthreadwatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjthreadwatch[n].log	1MB×2	ThreadWatchLogFile
スレッドダンプ	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjthreaddumpwatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjthreaddumpwatch[n].log	1MB×2	ThreaddumpWatchLogFile
HTTP リクエスト実行待ちキュー	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjrequestqueuewatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjrequestqueuewatch[n].log	1MB×2	RequestQueueWatchLogFile
HTTP セッション数	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjhttpsessionwatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjhttpsessionwatch[n].log	1MB×2	HttpSessionWatchLogFile

監視対象リソース	ログ取得場所およびログファイル名※1	デフォルトのサイズ×面数	チャネル名
コネクションプール	• Windows の場合 <ejb.server.log.directory>※2¥watch¥cjconnectionpoolwatch[n].log • UNIX の場合 <ejb.server.log.directory>※2/watch/cjconnectionpoolwatch[n].log	1MB×2	ConnectionPoolWatchLogFile

注

チャネル名は，ログの出力先を識別する名称です。ログの属性（サイズ，面数）を変更する場合のキー値として使用します。

注※1

[n]には，面の番号（1 から面数（最大 16 まで））が付きます。

注※2

<ejb.server.log.directory>は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.server.log.directory パラメタで指定したディレクトリを指します。デフォルト値は，<Application Server のインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs です。

ejb.server.log.directory パラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「3.2.1　usrconf.cfg（バッチサーバ用オプション定義ファイル）」を参照してください。

注※3

Windows の場合，または AIX の場合は，ファイルディスクリプタを監視できません。

リソース枯渇監視ログファイルに出力される情報やログファイルの出力形式については，マニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「4.3　リソース枯渇監視機能とリソース枯渇監視情報の出力」を参照してください。

(g)　ログの出力先を設定するユーザ定義ファイル

バッチサーバおよびサーバ管理コマンドのログの出力先を変更している場合は，次の表に示す，ログの出力先を設定するユーザ定義ファイルを参照して出力先を確認してください。なお，ログの出力先を変更した場合，変更後の出力先は snapshot ログの一括収集時に収集対象外となります。必要に応じて snapshot ログの収集先を変更してください。

表 4-26　ログの出力先を設定するユーザ定義ファイル

分類	ユーザ定義ファイルの指定個所
バッチサーバ	簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に指定する，ejb.server.log.directory パラメタ。 デフォルトは，「<Application Server のインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs」（Windows の場合），または「/opt/Cosminexus/CC/server/public/ejb/<サーバ名称>/logs」（UNIX の場合）です。
サーバ管理コマンド	サーバ管理コマンドの usrconf.bat（Windows の場合），または usrconf（UNIX の場合）の ejbserver.log.directory キー。 キーのデフォルトは，「<Application Server のインストールディレクトリ>¥CC¥admin¥logs」（Windows の場合），または「/opt/Cosminexus/CC/admin/logs」（UNIX の場合）です。

分類	ユーザ定義ファイルの指定個所
	Management Server リモート管理機能から操作した場合は，サーバ管理コマンドのログ出力先は変更できません。

ログの出力先の変更方法など，トラブルシューティングの資料取得の設定については，「3.　トラブルシューティングのための準備」を参照してください。

(2)　運用管理エージェント・運用監視エージェント・Management Server のログの取得

ここでは，運用管理エージェント，運用監視エージェント，および Management Server のログの出力先について説明します。

運用管理エージェント・運用監視エージェント・Management Server のログには，個別に取得する以外に，**統合ログ**としてまとめて取得できるものがあります。

運用管理エージェント・運用監視エージェント・Management Server のログを統合ログとしてまとめて取得する場合，および個別に取得する場合の出力先については，「4.3.1(2)　運用管理エージェント・運用監視エージェント・Management Server のログの取得」を参照してください。

(3)　ログ以外に取得が必要な情報

ログ以外に取得する必要がある情報については，「4.3.1(6)　ログ以外に取得が必要な情報」を参照してください。

4.4.2　アプリケーションのユーザログの取得（バッチアプリケーションを実行するシステム）

バッチアプリケーションのユーザログは，トレース共通ライブラリ形式で出力するように設定している場合に取得できます。トレース共通ライブラリ形式のログについては，「5.2　アプリケーションサーバのログ」を参照してください。

バッチアプリケーションのユーザログの出力先については，次のディレクトリに，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に ejbserver.application.userlog.CJLogHandler.<ハンドラ名>.path パラメタで指定したプリフィックスを持つファイル名で出力されます。なお，<ハンドラ名>には，キーの値を区別するためのハンドラ名が指定されています。

- Windows の場合

  <ログ出力先ルート（ejb.server.log.directory の値）>¥user¥（デフォルトは<バッチサーバの作業ディレクトリ>¥ejb¥<バッチサーバ名>¥logs¥user）
- UNIX の場合

<ログ出力先ルート（ejb.server.log.directory の値）>/user/（デフォルトは<バッチサーバの作業ディレクトリ>/ejb/<バッチサーバ名>/logs/user）

なお，バッチアプリケーションのユーザログ出力の設定については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「2.3.5　バッチアプリケーションのログ出力」を参照してください。

4.5　EJB クライアントアプリケーションのシステムログ

EJB クライアントアプリケーションのシステムログの種類と出力先について説明します。

なお，EJB クライアントアプリケーションのシステムログを運用するときに留意することについては，「2.6.1　EJB クライアントアプリケーションのシステムログに関する留意事項」を参照してください。

注意事項

Client を使用する場合は，EJB クライアントアプリケーションのログの格納ディレクトリを示す「<Application Server のインストールディレクトリ>¥CC」を，「<Application Server のインストールディレクトリ>¥CCL」と読み替えてください。

4.5.1　EJB クライアントアプリケーションのシステムログの種類

EJB クライアントアプリケーションでは，EJB クライアントアプリケーションのプロセス単位にシステムログを出力します。EJB クライアントアプリケーションのシステムログの種類を次に示します。

表 4-27　EJB クライアントアプリケーションのシステムログの種類

種類	ログの内容	ファイル名	デフォルトのサイズ×面数	チャネル名
メッセージログ	稼働ログ	cjclmessage[n].log※1	1MB×2	ClientMessageLogFile
	cjclstartap コマンドの稼働ログ	cjclstartap[n].log※2	1MB×2	−
	cjcldellog コマンドの稼働ログ	cjcldellog.log	1MB×2※3	−
ユーザログ	ユーザ出力ログ	user_out[n].log※1	1MB×2	UserOutLogFile
	ユーザエラーログ	user_err[n].log※1	1MB×2	UserErrLogFile
Java ログ	JavaVM の保守情報，GC のログ	javalog[n].log※1	256KB×4	−
例外ログ	障害発生時の例外情報	cjclexception[n].log※1	1MB×2	ClientExceptionLogFile
保守用ログ※4	保守情報	cjclmaintenance[n].log※1	1MB×2	ClientMaintenanceLogFile
	EJB コンテナの保守情報	cjejbcontainer[n].log※1	1MB×2	EJBContainerLogFile
	起動プロセス標準出力情報	cjstdout[n].log※2	1MB×2	−
	起動プロセス標準エラー情報	cjstderr[n].log※2	1MB×2	−

種類	ログの内容	ファイル名	デフォルトのサイズ×面数	チャネル名
	ログの稼働情報	cjlogger.log	1MB×2※3	－

（凡例） －：該当しない

注※1

ファイル名の[n]の部分には，1～指定したログの面数の通し番号が付きます。

注※2

ファイル名の[n]の部分には指定したログの面数の番号（1 または 2）が付きます。

注※3

cjcldellog.log のサイズが 1MB を超えた場合，cjcldellog_save.log というバックアップファイル名称のログファイルになります。

注※4

このログは，保守員へ送付する場合に必要に応じて収集してください。

4.5.2 EJB クライアントアプリケーションのシステムログの出力先

EJB クライアントアプリケーションのシステムログは，次のキーに指定したディレクトリに出力されます。

- Java アプリケーション用オプション定義ファイル（usrconf.cfg）の ejb.client.log.directory キー，および ejb.client.ejb.log キーに指定したディレクトリ
- システムプロパティの ejbserver.client.log.directory キーおよび ejbserver.client.ejb.log キーに指定したディレクトリ

Java アプリケーション用オプション定義ファイル（usrconf.cfg）とシステムプロパティの両方に異なる値が指定された場合，システムプロパティの設定が有効になります。

なお，システムプロパティで出力先を設定できるのは，次のログだけです。

- 稼働ログ（cjclmessage[n].log）
- 障害発生時の例外情報（cjclexception[n].log）
- 保守情報（cjclmaintenance[n].log）
- ログの稼働情報（cjlogger.log）

cjcldellog コマンドの稼働ログ（cjcldellog.log）および cjclstartap コマンドの稼働ログ（cjclstartap[n].log）は，「<Application Server のインストールディレクトリ>¥CC¥client¥logs」（Windows の場合），または「/opt/Cosminexus/CC/client/logs」（UNIX の場合）の直下に出力されます。

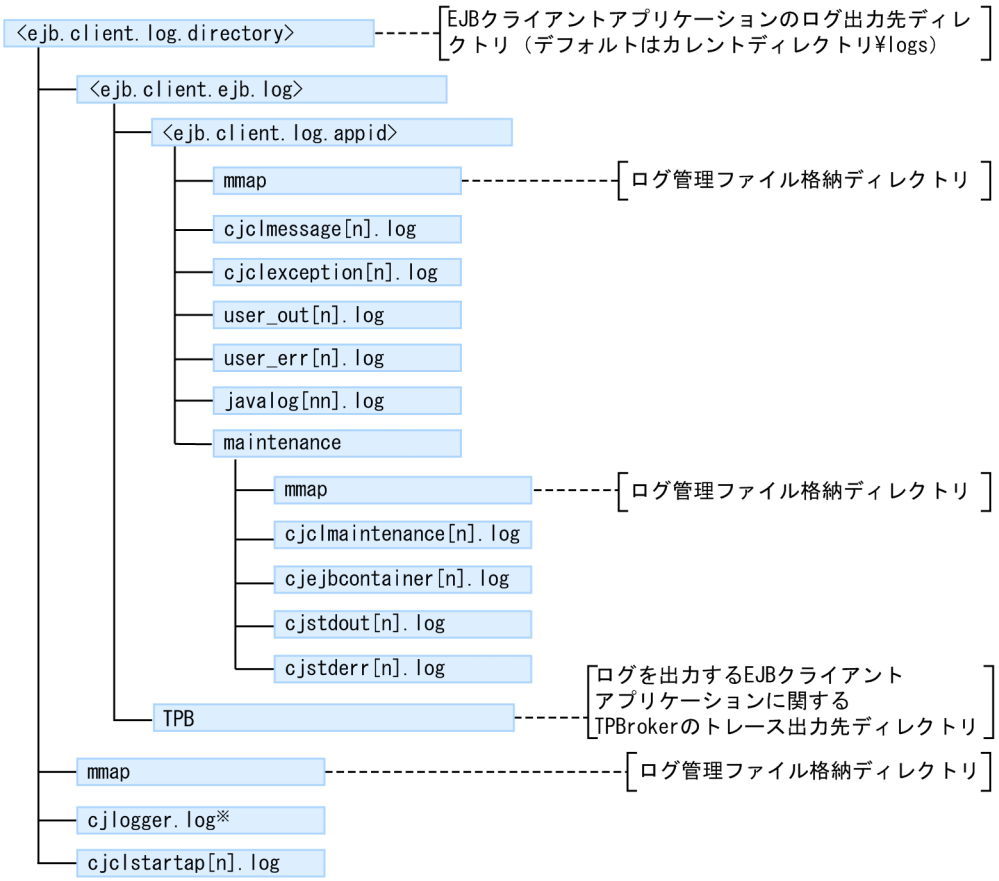
注意事項

- EJB クライアントアプリケーションの標準出力，および標準エラーの内容は，ログファイルには出力されません。ログファイルに出力する場合は，ユーザログ機能を使用するか，リダイレクトしてください。

複数のプロセスのログは，一つのディレクトリ（デフォルトは ejbcl ディレクトリ）下に出力します。なお，プロセス単位（アプリケーション単位）でログの出力先を別々にしたい場合，usrconf.cfg の ejb.client.log.directory キーにそれぞれの出力先ディレクトリを指定してください。

EJB クライアントアプリケーションのシステムログの出力先を次の図に示します。

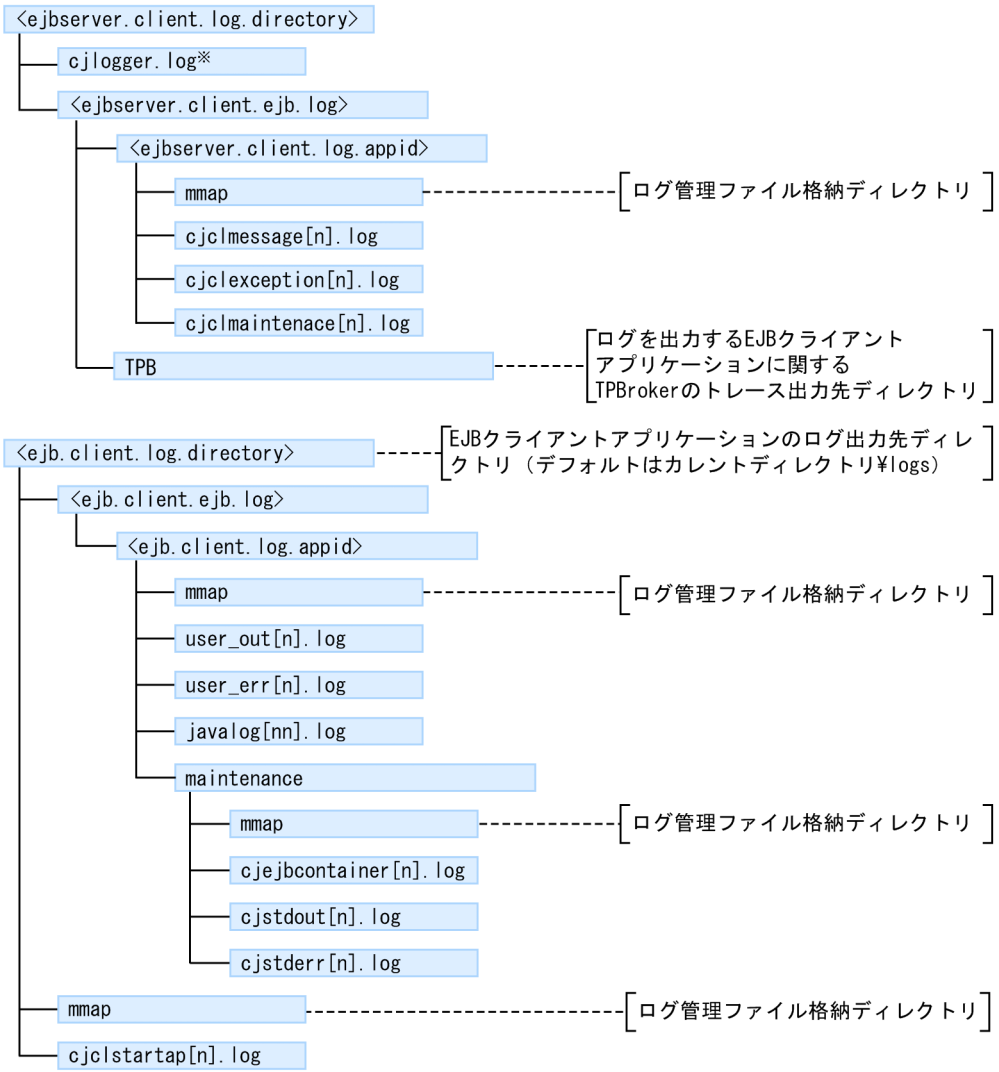
図 4-1　EJB クライアントアプリケーションのシステムログの出力先（Java アプリケーション用オプション定義ファイルで指定した場合）



注
　ファイル名の[n]は，1～（指定したログの面数）の通し番号になります。

注※
　cjlogger.logは，EJBクライアントアプリケーションのプロパティの設定などにエラーがある場合だけ出力されます。

図 4-2　EJB クライアントアプリケーションのシステムログの出力先（システムプロパティで指定した場合）



注
　ファイル名の[n]は，1～（指定したログの面数）の通し番号になります。

注※
　cjlogger.logは，EJBクライアントアプリケーションのプロパティの設定などにエラーがある場合だけ出力されます。

4.6　性能解析トレース

性能解析トレースの格納場所は次のとおりです。

- Windows の場合
  <環境変数 PRFSPOOL の設定ディレクトリ>¥utt¥prf¥<PRF 識別子>
- UNIX の場合
  $PRFSPOOL/utt/prf/<PRF 識別子>

性能解析トレースの収集方法については，「7.　性能解析トレースを使用した性能解析」を参照してください。性能解析トレースには，セッショントレースも出力されている場合があります。

4.7　JavaVM のスレッドダンプ

ここでは，JavaVM のスレッドダンプを取得する方法について説明します。

JavaVM のスレッドダンプは，次の方法で取得できます。

- 運用管理コマンド（mngsvrutil）を使用して，J2EE サーバ，CORBA ネーミングサービス，および CTM のスレッドダンプを取得する。
- 個別のコマンドを使用して，J2EE サーバ，CORBA ネーミングサービス，および EJB クライアントアプリケーションのスレッドダンプを取得する。
- JavaVM のコマンドを使用して，クラス別統計情報または Explicit ヒープ詳細情報を拡張スレッドダンプに取得する。

注意事項

スレッドダンプを取得する場合は，前回のスレッドダンプの出力が終了してから取得してください。

それぞれの方法について次に説明します。

4.7.1　運用管理コマンドを使用する場合

運用管理コマンド（mngsvrutil）を利用した JavaVM のスレッドダンプの取得は，mngsvrutil コマンドのサブコマンド「dump」の引数に server を指定して実行します。

次のスレッドダンプを取得できます。

- J2EE サーバ（クラスタを含む）
- CORBA ネーミングサービスおよび CTM

mngsvrutil コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「mngsvrutil（Management Server の運用管理コマンド）」を参照してください。mngsvrutil コマンドのサブコマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「7.3 mngsvrutil コマンドのサブコマンドの詳細」を参照してください。

実行形式，実行例，および出力先を次に示します。

実行形式

```
mngsvrutil -m <Management Serverのホスト名> [:<ポート番号>] -u <管理ユーザID> -p <管理パスワード> -t <論理サーバ名> dump server
```

実行例

```
mngsvrutil -m mnghost -u user01 -p pw1 -t myserver dump server
```

出力先

J2EE サーバを対象にする場合

- Windows の場合

  <作業ディレクトリ>※¥ejb¥<サーバ名称>¥javacore*.txt

- UNIX の場合

  <作業ディレクトリ>※/ejb/<サーバ名称>/javacore*.txt

注※　<作業ディレクトリ>は，J2EE サーバのユーザ定義（usrconf.cfg ファイル中の ejb.public.directory）で指定されたディレクトリを指します。デフォルト値は，次のとおりです。

- Windows の場合

  <Application Server のインストールディレクトリ>¥CC¥server¥public

- UNIX の場合

  /opt/Cosminexus/CC/server/public

CORBA ネーミングサービスおよび CTM を対象にする場合

- Windows の場合

  <Application Server のインストールディレクトリ>¥TPB¥logj¥javacore*.txt

- UNIX の場合

  /opt/Cosminexus/TPB/logj/javacore*.txt

4.7.2　個別のコマンドを使用する場合

スレッドダンプの出力のされ方は，指定している J2EE サーバを実行する JavaVM の起動オプションによって異なります。

- -XX:+HitachiThreadDump が設定されている場合，**拡張スレッドダンプ**を取得できます。このオプションはデフォルトで設定されています。
- -XX:+HitachiThreadDumpToStdout が設定されている場合，スレッドダンプは標準出力にも出力されます。このオプションはデフォルトでは設定されていません。必要に応じて設定してください。
- 次に示す起動オプションが設定されている場合に，-XX:+HitachiOutOfMemoryAbortThreadDump が設定されていると，OutOfMemoryError によって強制終了した時に，スレッドダンプが出力されます。
  - -XX:+HitachiOutOfMemoryAbort
  - -XX:+HitachiThreadDump

  ただし，次の場合は除きます。
  - J2SE クラスライブラリで C ヒープ不足が発生した場合
  - JavaVM の処理中に C ヒープ不足が発生した場合

- 次に示す起動オプションが設定されている場合に，-XX:+HitachiOutOfMemoryAbortThreadDumpWithJHeapProf が設定されていると，OutOfMemoryError によって強制終了した時に，スレッドダンプにクラス別統計情報が出力されます。クラス別統計情報については，「9.3　クラス別統計機能」を参照してください。このオプション設定時に出力されるクラス別統計情報は，jheapprof コマンド実行時に取得する情報と同じです。
  - -XX:+HitachiOutOfMemoryAbort
  - -XX:+HitachiOutOfMemoryAbortThreadDump
  - -XX:+HitachiThreadDump

(1)　J2EE サーバのスレッドダンプの取得

J2EE サーバプロセス(cjstartsv)が存在する場合，J2EE サーバのスレッドダンプは，cjdumpsv コマンドを実行して取得します。cjdumpsv コマンドの実行例を次に示します。時間の経過に応じた各スレッドの状態遷移を確認するため，複数回 cjdumpsv コマンドを実行します。目安として 3 秒おきに 10 回程度実行します。

- Windows の場合

```
<Application Serverのインストールディレクトリ>¥CC¥server¥bin¥cjdumpsv <J2EEサーバ名>
```

- UNIX の場合

```
/opt/Cosminexus/CC/server/bin/cjdumpsv <J2EEサーバ名>
```

cjdumpsv コマンドを実行すると，次に示すファイルに JavaVM のスレッドダンプが出力されます。

- サーバ標準出力ログ
- <作業ディレクトリ>¥ejb¥<サーバ名称>¥javacore<プロセス番号>.<コマンド実行日時>.txt（Windows の場合）
- <作業ディレクトリ>/ejb/<サーバ名称>/javacore<プロセス番号>.<コマンド実行日時>.txt（UNIX の場合）

サーバ標準出力ログのデフォルトの出力先は「<ejb.server.log.directory>¥CC¥maintenance¥cjstdout.log」（Windows の場合），または「<ejb.server.log.directory>/CC/maintenance/cjstdout.log」（UNIX の場合）です。出力先を変更している場合は，「4.3　アプリケーションサーバのログ（J2EE アプリケーションを実行するシステム）」または「4.4　アプリケーションサーバのログ（バッチアプリケーションを実行するシステム）」を参照してください。なお，作業ディレクトリのデフォルトのディレクトリパスは「<Application Server のインストールディレクトリ>¥CC¥server¥public」（Windows の場合），または「/opt/Cosminexus/CC/server/public」（UNIX の場合）です。

また，javacore<プロセス番号>.<コマンド実行日時>.txt ファイルの出力先は，環境変数 JAVACOREDIR で変更できます。ただし，指定したディレクトリへの書き込みに失敗した場合は，デフォルトの出力先に出力されます。このディレクトリにも出力できなかった場合は，標準エラー出力にだけ出力されます。

cjdumpsv コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjdumpsv（J2EE サーバのスレッドダンプの取得)」を参照してください。

参考

スレッドダンプが出力されると，標準出力に，次のメッセージが出力され，java プログラムの実行が継続されます。このメッセージは，-XX:+HitachiThreadDumpToStdout の設定とは関係なく出力されます。

```
Writing Java core to <ファイル名（フルパス）>...OK
```

(2) CORBA ネーミングサービスのスレッドダンプの取得

Windows の場合，CORBA ネーミングサービスのプロセス（nameserv）が存在するときは，CORBA ネーミングサービスを起動したコマンドプロンプトに対して Ctrl+Break キーを押します。時間の経過に応じた各スレッドの状態遷移を確認するため，複数回実行します。目安として 3 秒おきに 10 回程度実行します。なお，CORBA ネーミングサービスを Management Server から監視している場合はスレッドダンプを取得できません。

UNIX の場合，CORBA ネーミングサービスのプロセス（java）が存在するときは，kill コマンドを実行して CORBA ネーミングサービスのスレッドダンプを取得します。なお，CORBA ネーミングサービスを Management Server から監視している場合はスレッドダンプを取得できません。

UNIX での CORBA ネーミングサービスのスレッドダンプの取得手順を次に示します。

1. CORBA ネーミングサービスのプロセス ID を取得します。

   CORBA ネーミングサービスのプロセス ID の取得方法は，次の場合によって異なります。

   java プロセスがほかに起動していない場合

   ```
   ps -ef | grep java
   ```

   java プロセスが複数起動している場合

   CORBA ネーミングサービスの起動用シェルスクリプトを使用すると，カレントワーキングディレクトリに生成される namesv_pid ファイルに CORBA ネーミングサービスのプロセス ID を出力できます。

   CORBA ネーミングサービスの起動用シェルスクリプトの例を次に示します。

   ```
   #!/bin/sh
   export VBROKER_ADM=/opt/Cosminexus/TPB/adm
   export SHLIB_PATH="${SHLIB_PATH}:/opt/Cosminexus/TPB/lib"

   # start name server process
   exec /opt/Cosminexus/TPB/bin/nameserv ¥
   -J-Dvbroker.agent.enableLocator=false ¥
   -J-Djava.security.policy==/opt/Cosminexus/CC/server/sysconf/cli.policy ¥
   -J-Dvbroker.se.iiop_tp.scm.iiop_tp.listener.port=900 &
   ```

```
# save background java process pid
echo $! > ./namesv_pid
```

2. 取得したプロセス ID を指定して，kill コマンドを実行します。

```
kill -3 `cat namesv_pid`
```

(3) EJB クライアントアプリケーションのスレッドダンプの取得

EJB クライアントアプリケーションのスレッドダンプは，cjcldumpap コマンドを実行して取得します。

cjcldumpap コマンドを実行すると，cjclstartap コマンドを実行して起動した EJB クライアントアプリケーションのスレッドダンプが出力されます。また，特定のプロセスのスレッドダンプを出力することもできます。cjcldumpap コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjcldumpap（Java アプリケーションのスレッドダンプの取得)」を参照してください。

cjcldumpap コマンドの実行形式，実行例，およびスレッドダンプの出力先を次に示します。

実行形式

cjclstartap コマンドを使用して起動した EJB クライアントアプリケーションのスレッドダンプを出力する場合

```
cjcldumpap
```

特定のプロセスのスレッドダンプを出力する場合

```
cjcldumpap ＜プロセスID＞
```

実行例

cjclstartap コマンドを使用して起動した EJB クライアントアプリケーションのスレッドダンプを出力する場合

```
cjcldumpap
```

特定のプロセスのスレッドダンプを出力する場合

```
cjcldumpap 3264
```

出力先

cjclstartap コマンドを実行しているカレントディレクトリ

4.7.3 JavaVM のコマンドを使用する場合

JavaVM のコマンドを使用すると，拡張スレッドダンプにクラス別統計情報や Explicit ヒープ詳細情報を取得できます。

(1)　クラス別統計情報の拡張スレッドダンプへの取得

クラス別統計情報を含む拡張スレッドダンプは，jheapprof コマンドを実行して取得します。クラス別統計情報は，クラス別統計機能で出力する情報で，GC による Java オブジェクトの変化や，Java オブジェクトの参照関係などを調査する場合に使用します。クラス別統計機能については，「9.3　クラス別統計機能」を，クラス別統計情報の出力方法については，「9.3.3　クラス別統計情報の出力」を参照してください。jheapprof コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「jheapprof（クラス別統計情報付き拡張スレッドダンプの出力)」を参照してください。

(2)　Explicit ヒープ詳細情報の拡張スレッドダンプへの取得

Explicit ヒープ詳細情報のうち，次の情報は明示管理ヒープ機能が有効な場合に，拡張スレッドダンプに必ず取得されます。

- Explicit ヒープ情報
- Explicit メモリブロック情報

Explicit ヒープ詳細情報のオブジェクト統計情報，および Explicit メモリブロックの解放率情報は，eheapprof コマンドを実行して拡張スレッドダンプに取得します。オブジェクト統計情報は，Explicit メモリブロック内の詳細情報です。解放率情報は，Explicit メモリブロックの自動解放処理で解放されたオブジェクトの割合です。明示管理ヒープ機能を使用するシステムのデバッグや障害解析に使用します。

明示管理ヒープ機能については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7.　明示管理ヒープ機能を使用した FullGC の抑止」を参照してください。eheapprof コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「eheapprof（Explicit ヒープ詳細情報付き拡張スレッドダンプの出力)」を参照してください。

実行形式

Windows の場合

```
eheapprof [-f|-i] [-freeratio] -p <プロセスID>
```

UNIX の場合

```
eheapprof [-f|-i] [-freeratio] [-force] -p <プロセスID>
```

実行例

ここでは，プロセス ID が 2463 の Java プロセスのクラス別統計情報を出力します。

1. -p オプションに，クラス別統計情報を出力したい Java プロセスのプロセス ID を指定して，eheapprof コマンドを実行します。

```
% eheapprof -p 2463
```

eheapprof コマンドで-f オプションを省略している場合，次の確認メッセージが表示されます。

Windows の場合

クラス別統計情報付き拡張スレッドダンプを出力するかどうかを確認するメッセージが次の形式で表示されます。

```
Force VM to output ExplicitHeapProf: ? (y/n)
```

UNIX の場合

プロセス ID を確認するメッセージが次の形式で表示されます。

```
send SIGQUIT to 2463: ? (y/n)
```

2. y を入力します。

クラス別統計情報付き拡張スレッドダンプを出力すると，実行中の java プログラムでは次のメッセージが出力されます。

```
Writing Java core to javacore2463.030806215140.txt... OK
```

実行中の java プログラムは，カレントディレクトリにクラス別統計情報付き拡張スレッドダンプ（javacore<プロセス ID>.<日時>.txt）を作成し，プログラムを継続します。

4.7.4 スレッドダンプにクラス別統計情報を出力する場合の注意事項

JavaVM の起動オプションに-XX:+HitachiOutOfMemoryAbortThreadDumpWithJHeapProf が設定されていると，OutOfMemoryError によって強制終了した時に，スレッドダンプにクラス別統計情報が出力されます。

大量のヒープ（Java ヒープ，Explicit ヒープまたは Metaspace 領域）を使用している場合に，スレッドダンプにクラス別統計情報が出力されると時間が掛かります。このため，OutOfMemoryError 発生時に JavaVM を即時に強制終了して回復させることができないことがあります。

注意事項

次の JavaVM の起動オプションが設定されていない場合，スレッドダンプにクラス別統計情報は出力されません。

- -XX:+HitachiOutOfMemoryAbort
- -XX:+HitachiOutOfMemoryAbortThreadDump
- -XX:+HitachiThreadDump

4.8 JavaVM の GC ログ

JavaVM の GC ログは，JavaVM または J2EE サーバ起動前にログ出力先が設定されている場合にだけ取得できます。

GC ログの出力先は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.server.log.directory パラメタで指定します。

また，実行中の Java プロセスで FullGC を発生させたい場合は，**javagc コマンド**を実行してください。javagc コマンドは，トラブル発生時の要因調査のほか，1 トランザクション当たりのメモリ使用量の測定，メモリリークの調査，アプリケーションのデバッグなどの目的でも使用できます。

実行中の Java プロセスを指定して FullGC を発生させる場合の javagc コマンドの実行形式を次に示します。なお，これ以外に指定できるオプションについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「javagc（GC の強制発生）」を参照してください。

実行形式

```
javagc -p ＜プロセスID＞
```

実行結果として，次のログが出力されます。なお，これは，オプションとして，-XX:+HitachiVerboseGCPrintCause が指定されている場合の例です。

```
[VGC]<Wed Mar 17 00:42:30 2004>(Skip Full:0,Copy:0)[ Full GC 149K->149K(1984K),0.0786038 secs] [ DefNew::Eden:264K->0K(512K)] [ DefNew::Survivor:0K->63K(64K)] [ Tenured: 85K->149K(1408K)] [Metaspace: 3634K(4492K, 4492K)->3634K(4492K, 4492K)][class space: 356K(388K, 388K)->356K(388K, 388K)] [ cause:JavaGC Command]
```

4.9　メモリダンプ

J2EE サーバまたは CORBA ネーミングサービスを再起動する場合は，メモリダンプとして次のファイルを取得します。

- ユーザダンプ（Windows の場合）
- J2EE サーバのメモリダンプ
- CORBA ネーミングサービスのメモリダンプ
- Management Server のメモリダンプ
- 運用管理エージェントのメモリダンプ

これらのファイルは，システムでトラブルが発生したときに，保守員が障害を解析するために使用します。

4.9.1　ユーザダンプの取得（Windows の場合）

ユーザダンプの取得について説明します。

タスクマネージャまたは Windows のデバッグツールを使用して，ユーザダンプを取得します。詳細は，Microsoft 社のホームページを参照してください。

または cjstopsv コマンドの-fd オプションを使用して次のファイルを取得します。

<ユーザダンプの出力先ディレクトリ>¥cjmemdump.dmp

<ユーザダンプの出力先ディレクトリ>は，環境変数 CJMEMDUMP_PATH で指定しておいてください。

4.9.2　J2EE サーバのメモリダンプの取得

J2EE サーバのメモリダンプの取得について，OS ごとに説明します。

(1)　Windows の場合

J2EE サーバが稼働している場合（cjstartsv プロセスが存在する場合），タスクマネージャ※からメモリダンプを採取します。

J2EE サーバが稼働していて，かつ，強制停止してメモリダンプを取得する場合，cjstopsv コマンドに-fd オプションを指定して実行し，メモリダンプを取得します。

J2EE サーバがダウンしている場合，Windows のデバッグツール※からメモリダンプを採取します。

注※

詳細は，Microsoft 社のホームページを参照してください。

ダウンしている場合のメモリダンプを取得するためには，事前に設定を行う必要があります。設定方法については，「3.3.15　ユーザダンプ取得の設定」を参照してください。

(2)　UNIX の場合

cjstartsv プロセスがダウンした場合，「<作業ディレクトリ>/ejb/<サーバ名称>」に出力された core ダンプを取得します。

core ダンプのファイル名は，cjstartsv プロセスの再起動時に「core.<出力日時※>」（AIX の場合）または「core.<プロセス ID>.<出力日時※>」（Linux の場合）にリネームされます。cjstartsv プロセスの再起動時に core ダンプが上書き保存されないため，障害発生時の core ダンプを保存できます。

注※

出力日時は「YYMMDDhhmmss」の形式で出力されます。

YY：西暦（下 2 けた）　MM：月（2 けた）　DD：日（2 けた）

hh：時（24 時間表記で 2 けた）　mm：分（2 けた）　ss：秒（2 けた）

なお，保存する core ダンプは上限値を設定できます。Windows の場合，cjstartsv プロセスの再起動時，および javacore コマンド実行時に，出力日時が古い順に削除されます。UNIX の場合，「<作業ディレクトリ>/ejb/<サーバ名称>」に出力される core ダンプファイルの合計が上限値を超えたときに，出力日時が古い順に削除されます。上限値は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の <configuration>タグ内に，J2EE サーバの拡張パラメタの ejb.server.corefilenum で指定します。なお，ファイルの削除は cjstartsv プロセスの再起動時に実行されます。core ファイルのファイル数の上限値の設定については，「3.3.16　core ダンプ取得の設定」を参照してください。

core ダンプを取得したあとに，core ダンプからスタックトレース情報だけを取得する場合は，javatrace コマンドを実行してください。スタックトレース情報は，JavaVM の異常終了の原因を究明するために必要な情報です。スタックトレース情報の取得方法については，「4.18　JavaVM のスタックトレース情報」を参照してください。

core ダンプは次のような場合にも取得できます。それぞれの場合の取得方法について説明します。

- **cjstartsv プロセスが稼働している状態で core ダンプを取得する場合**

  cjstartsv プロセス（J2EE サーバ）が稼働している状態で core ダンプを取得する場合，cjstartsv プロセスのプロセス ID を確認して，kill コマンドを実行します。kill コマンドは次の形式で実行してください。なお，kill コマンドを実行するとプロセスが終了するため，再起動する前に kill コマンドを実行することを推奨します。

  ```
  kill -6 <cjstartsvのプロセスID>
  ```

- **実行中の Java プロセスで core ダンプとスレッドダンプを同時に取得する場合**

  実行中の Java プロセスで core ダンプとスレッドダンプを同時に取得する場合は，javacore コマンドを実行してください。javacore コマンドの実行形式を次に示します。なお，指定できるオプションについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「javacore（core ファイルとスレッドダンプの取得／UNIX の場合）」を参照してください。

```
javacore <-プロセスID>
```

上記の形式でコマンドを実行すると，次のメッセージが出力されます。

```
send SIGQUIT to 8662: ? (y/n)
```

「y」を入力すると，実行中の Java プログラムのカレントディレクトリに，"javacore<プロセス ID>.<出力日時>.core"（core ダンプ），および"javacore<プロセス ID>.<出力日時>.txt"（スレッドダンプ）が出力されます。「n」を入力すると，core ダンプおよびスレッドダンプを取得しないでコマンドの実行を終了します。

core ダンプおよびスレッドダンプを取得すると，実行中の Java プログラムには，次のメッセージが出力されます。なお，斜体の部分は実際には表示されません。

```
Now generating core file (javacore8662.030806215140.core)...
done

(coreダンプおよびスレッドダンプの出力終了)

Writing Java core to javacore8662.030806215140.txt... OK
```

4.9.3　CORBA ネーミングサービスのメモリダンプの取得

CORBA ネーミングサービスのメモリダンプの取得について OS ごとに説明します。

(1)　Windows の場合

CORBA ネーミングサービスが稼働している場合（CORBA ネーミングサービスのプロセスが存在する場合），タスクマネージャ※からメモリダンプを採取します。

CORBA ネーミングサービスがダウンしている場合，Windows のデバッグツール※からメモリダンプを採取します。

注※

詳細は，Microsoft 社のホームページを参照してください。

ダウンしている場合のメモリダンプを取得するためには，事前に設定を行う必要があります。設定方法については，「3.3.15　ユーザダンプ取得の設定」を参照してください。

(2)　UNIX の場合

CORBA ネーミングサービスが稼働している場合（CORBA ネーミングサービスのプロセスが存在する場合），CORBA ネーミングサービスのプロセス ID を確認して，kill コマンドを実行します。kill コマンドは次の形式で実行してください。なお，kill コマンドを実行するとプロセスが終了するため，再起動する前に kill コマンドを実行することを推奨します。

```
ps -ef | grep java
kill -6 <CORBAネーミングサービスのプロセスID>
```

4.9.4　Management Server のメモリダンプの取得

Management Server のメモリダンプの取得について，OS ごとに説明します。

(1)　Windows の場合

Management Server が稼働している場合（cjstartsv.exe プロセスが存在する場合），タスクマネージャ※からメモリダンプを採取します。

Management Server がダウンしている場合，Windows のデバッグツール※からメモリダンプを採取します。

注※

詳細は，Microsoft 社のホームページを参照してください。

Management Server がダウンしている場合のメモリダンプを取得するためには，事前に設定をする必要があります。設定方法については，「3.3.15　ユーザダンプ取得の設定」を参照してください。

(2)　UNIX の場合

Management Server（cjstartsv プロセス）がダウンした場合，「<Application Server のインストールディレクトリ>/manager/containers/m/ejb/<Management Server のサーバ名>」に出力された core ダンプを取得します。

core ダンプのファイル名は，Management Server の再起動時に「core.<出力日時※>」（AIX の場合）または「core.<プロセス ID>.<出力日時※>」（Linux の場合）にリネームされます。Management Server の再起動時に core ダンプが上書き保存されないため，障害発生時の core ダンプを保存できます。

注※

出力日時は「YYMMDDhhmmss」の形式で出力されます。

YY：西暦（下 2 けた）　MM：月（2 けた）　DD：日（2 けた）

hh：時（24 時間表記で 2 けた）　mm：分（2 けた）　ss：秒（2 けた）

core ダンプを取得したあとに，core ダンプからスタックトレース情報だけを取得する場合は，javatrace コマンドを実行してください。スタックトレース情報は，JavaVM の異常終了の原因を究明するために必要な情報です。スタックトレース情報の取得方法については，「4.18　JavaVM のスタックトレース情報」を参照してください。

core ダンプは次のような場合にも取得できます。それぞれの場合の取得方法について説明します。

- **Management Server が稼働している状態で core ダンプを取得する場合**

  Management Server が稼働している状態で core ダンプを取得する場合，cjstartsv プロセスのプロセス ID を確認して，kill コマンドを実行します。kill コマンドは次の形式で実行してください。なお，kill コマンドを実行するとプロセスが終了するため，Management Server が再起動する前に kill コマンドを実行することを推奨します。

```
kill -6 <Management Server（cjstartsv）のプロセスID>
```

- **実行中の Java プロセスで core ダンプとスレッドダンプを同時に取得する場合**

  実行中の Java プロセスで core ダンプとスレッドダンプを同時に取得する場合は，javacore コマンドを実行してください。javacore コマンドの実行形式を次に示します。なお，指定できるオプションについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「javacore（core ファイルとスレッドダンプの取得／UNIX の場合）」を参照してください。

```
javacore -p <プロセスID>
```

  上記の形式でコマンドを実行すると，次のメッセージが出力されます。

```
send SIGQUIT to 8662: ? (y/n)
```

  「y」を入力すると，実行中の Java プログラムのカレントディレクトリに，"javacore<プロセス ID>.<出力日時>.core"（core ダンプ），および"javacore<プロセス ID>.<出力日時>.txt"（スレッドダンプ）が出力されます。「n」を入力すると，core ダンプおよびスレッドダンプを取得しないでコマンドの実行を終了します。

  core ダンプおよびスレッドダンプを取得すると，実行中の Java プログラムには，次のメッセージが出力されます。なお，斜体の部分は実際には表示されません。

```
Now generating core file (javacore8662.030806215140.core)...
  done

(coreダンプおよびスレッドダンプの出力終了)

Writing Java core to javacore8662.030806215140.txt... OK
```

4.9.5　運用管理エージェントのメモリダンプの取得

運用管理エージェントのメモリダンプの取得について，OS ごとに説明します。

(1)　Windows の場合

運用管理エージェントが稼働している場合（adminagent プロセスが存在する場合），タスクマネージャ※からメモリダンプを採取します。

運用管理エージェントがダウンしている場合，Windows のデバッグツール※からメモリダンプを採取します。

注※

詳細は，Microsoft 社のホームページを参照してください。

運用管理エージェントがダウンしている場合のメモリダンプを取得するためには，事前に設定をする必要があります。設定方法については，「3.3.15　ユーザダンプ取得の設定」を参照してください。

(2)　UNIX の場合

運用管理エージェント（adminagent プロセス）がダウンした場合，「<Application Server のインストールディレクトリ>/manager/bin」に出力された core ダンプを取得します。

core ダンプを取得したあとに，core ダンプからスタックトレース情報だけを取得する場合は，javatrace コマンドを実行してください。スタックトレース情報は，JavaVM の異常終了の原因を究明するために必要な情報です。スタックトレース情報の取得方法については，「4.18　JavaVM のスタックトレース情報」を参照してください。

core ダンプは次のような場合にも取得できます。それぞれの場合の取得方法について説明します。

- **運用管理エージェントが稼働している状態で core ダンプを取得する場合**

  運用管理エージェントが稼働している状態で core ダンプを取得する場合，adminagent プロセスのプロセス ID を確認して，kill コマンドを実行します。kill コマンドは次の形式で実行してください。なお，kill コマンドを実行するとプロセスが終了するため，運用管理エージェントが再起動する前に kill コマンドを実行することを推奨します。

  ```
  kill -6 〈運用管理エージェント（adminagent）のプロセスID〉
  ```

- **実行中の Java プロセスで core ダンプとスレッドダンプを同時に取得する場合**

  実行中の Java プロセスで core ダンプとスレッドダンプを同時に取得する場合は，javacore コマンドを実行してください。javacore コマンドの実行形式を次に示します。なお，指定できるオプションについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「javacore（core ファイルとスレッドダンプの取得／UNIX の場合）」を参照してください。

  ```
  javacore -p 〈プロセスID〉
  ```

  上記の形式でコマンドを実行すると，次のメッセージが出力されます。

  ```
  send SIGQUIT to 8662: ? (y/n)
  ```

  「y」を入力すると，実行中の Java プログラムのカレントディレクトリに，"javacore<プロセス ID>.<出力日時>.core"（core ダンプ），および"javacore<プロセス ID>.<出力日時>.txt"（スレッドダンプ）が出力されます。「n」を入力すると，core ダンプおよびスレッドダンプを取得しないでコマンドの実行を終了します。

  core ダンプおよびスレッドダンプを取得すると，実行中の Java プログラムには，次のメッセージが出力されます。なお，斜体の部分は実際には表示されません。

  ```
  Now generating core file (javacore8662.030806215140.core)...
    done
  ```

（coreダンプおよびスレッドダンプの出力終了）

```
Writing Java core to javacore8662.030806215140.txt... OK
```

4.9.6　メモリダンプ取得時の注意事項

(1)　Windows の場合

- 次の場合，ユーザダンプが出力されません。
  - -XX:+HitachiOutOfMemoryAbort オプションの指定による強制停止の場合
  - JavaVM 処理中の C ヒープ不足発生による強制終了の場合

4.10　JavaVM ログ（JavaVM ログファイル）

JavaVM ログとは，製品が標準の JavaVM に追加した拡張オプションを使用して取得できるログです。標準の JavaVM よりも，多くのトラブルシュート情報が取得できます。JavaVM ログは，次のどれかのオプションを指定した場合に，ログファイルに出力されます。なお，このログファイルを，JavaVM ログファイルといいます。

- -XX:+HitachiOutOfMemoryStackTrace
  なお，このオプションを指定した場合に同時に指定される，-XX:+HitachiOutOfMemorySize および-XX:+HitachiOutOfMemoryCause が指定された場合も，JavaVM ログファイルが出力されます。
- -XX:+HitachiVerboseGC
- -XX:+HitachiOutOfMemoryHandling
- -XX:+HitachiJavaClassLibTrace
- -XX:+JITCompilerContinuation

このほか，出力内容や出力方法についても，オプションに指定しておく必要があります。JavaVM の資料取得の設定については，「3.3.17　JavaVM の資料取得の設定」を参照してください。

JavaVM ログファイルは，-XX:HitachiJavaLog:＜パスおよびファイル名称＞オプションに指定した出力先に，指定したファイル名称で出力されます。指定を省略した場合は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.server.log.directory パラメタで指定したディレクトリに，「javalogxx.log」という名称で出力されます。xx は 01 から始まる 2 けたの通番です。

参考

JavaVM ログファイルの出力が設定されている場合に，JavaVM 起動時にファイルを作成します。このため，JavaVM 終了時までに JavaVM ログが何も出力されない場合には，何も情報がない JavaVM ログファイルが残ります。

4.11　JavaVM 出力メッセージログ（標準出力またはエラーリポートファイル）

ここでは，JavaVM が出力するメッセージログの取得について説明します。

JavaVM でクラッシュが発生した場合，JavaVM はデバッグ情報を標準出力とエラーリポートファイルに出力します。

エラーリポートファイルは，次の場合に出力されます。

- JNI の中でシグナルが発生したとき
- JavaVM で C ヒープ不足が発生したとき
- JavaVM で予期しないシグナルが発生したとき
- JavaVM で Internal Error（内部論理エラー）が発生したとき

ただし，エラーリポートファイルの作成時にシグナルやメモリ不足が発生した場合，このファイルは作成されません。

4.11.1　Windows の場合

エラーリポートファイルの出力先および出力ファイル名は次のとおりです。

<作業ディレクトリ>¥ejb¥<サーバ名>¥hs_err_pid<サーバプロセスのプロセス ID>.log

参考

C ヒープが不足した場合，次の順序でメッセージ出力およびダンプ出力が実行されます。必要な情報を取得してください。

1. C ヒープ不足を示すメッセージログが，エラーリポートファイルおよび標準出力に出力されます。
2. 1.の実行中にメモリ不足が発生した場合，簡易メッセージが標準出力に出力されます。

4.11.2　UNIX の場合

エラーリポートファイルの出力先および出力ファイル名は次のとおりです。

<作業ディレクトリ>/ejb/<サーバ名>/hs_err_pid<サーバプロセスのプロセス ID>.log

参考

C ヒープが不足した場合，次の順序でメッセージ出力および core ダンプの生成が実行されます。必要な情報を取得してください。

1. C ヒープ不足を示すメッセージログが，エラーリポートファイルおよび標準出力に出力されます。
2. 1.の実行中にメモリ不足が発生した場合，簡易メッセージが標準出力に出力されます。
3. 簡易メッセージの出力処理中にさらにメモリ不足が発生した場合，メッセージおよびエラーログファイルの出力処理を中止して，core ダンプを生成します。

4.12　OS の状態情報と OS のログ

トラブルシュート情報として必要な OS のログ情報を次に示します。

4.12.1　OS の状態情報の取得

トラブルシュート情報として必要な OS の状態情報の取得について，OS ごとに説明します。

(1)　Windows の場合

OS の状態情報は，cjgetsysinfo コマンドを使用して取得できます。-f オプションを指定すると，OS 状態出力ファイルに出力できます。

コマンドは，次の形式で実行します。

```
cjgetsysinfo -f <OS状態出力ファイルパス>
```

このコマンドによって，OS の次のコマンドを実行した場合と同じ情報が取得できます。

```
netstat -e
netstat -s
netstat -an
set
```

なお，cjgetsysinfo コマンドを実行しない場合に，トラブルシュート情報として取得する必要がある OS の状態情報を次に示します。各情報の取得はあらかじめディレクトリを作成し，そこにファイルを生成します。作成するディレクトリのパスは任意です。

表 4–28　トラブルシュート情報として必要な OS の状態情報

情報の種類	デフォルトのファイル名
ネットワークの情報	プロトコルの統計情報と現在の TCP/IP ネットワーク接続の情報。次のコマンドで順次取得します。 netstat -e > netstat_e.txt netstat -s > netstat_s.txt netstat -an > netstat_an.txt
環境変数	現在設定されている環境変数。次のコマンドで取得します。 set >set.txt

(2)　UNIX の場合

OS の状態情報は，cjgetsysinfo コマンドを使用して取得できます。-f オプションを指定すると，OS 状態出力ファイルに出力できます。

コマンドは，次の形式で実行します。

```
cjgetsysinfo -f <OS状態出力ファイルパス>
```

このコマンドによって，OS の次の表に示すコマンドを実行した場合と同じ情報が取得できます。

表 4-29　cjgetsysinfo コマンドの実行によって実行される OS のコマンド

AIX の場合	Linux の場合
• df -k • ps -elf • ps -A -m -o THREAD • vmstat -t 1 1 • vmstat -s • lsps -s • netstat -i • netstat -m • netstat -an • iostat • svmon -P • svmon -G • sar -A 1 • instfix -i • lslpp -hac • uname -a • env • set • ipcs -a	• df • ps -eflm • vmstat • netstat -s • netstat -an • iostat※ • top -b -n 1 • sysctl -a • sar -A 1 1※ • rpm -qa • rpm -qai • uname -a • env • set • ipcs • ipcs -t • ipcs -p • ipcs -c • ipcs -u • ipcs -l

注※　sar コマンドおよび iostat コマンドを実行するためには，Linux に含まれている sysstat パッケージをインストールする必要があります。

なお，cjgetsysinfo コマンドを実行しない場合に，トラブルシュート情報として取得する必要がある OS の状態を示す情報の取得方法（コマンド）を次に示します。

AIX の場合

```
df -k > df_k`date +"%y%m%d%H%M%S"`.txt
ps -efl > ps_efl`date +"%y%m%d%H%M%S"`.txt
ps -A -m -o THREAD > ps_AmoTHREAD`date +"%y%m%d%H%M%S"`.txt
vmstat -t 1 5 > vmstat`date +"%y%m%d%H%M%S"`.txt
vmstat -s > vmstat_s`date +"%y%m%d%H%M%S"`.txt
lsps -s > lsps_s`date +"%y%m%d%H%M%S"`.txt
netstat -i > netstat_i`date +"%y%m%d%H%M%S"`.txt
netstat -m > netstat_m`date +"%y%m%d%H%M%S"`.txt
netstat -an > netstat_an`date +"%y%m%d%H%M%S"`.txt
iostat 1 5 > iostat`date +"%y%m%d%H%M%S"`.txt
```

```
svmon -P > svmon_P`date +"%y%m%d%H%M%S"`.txt※1
svmon -G -i 1 5 > svmon_G`date +"%y%m%d%H%M%S"`.txt※1
sar -A 1 5 > sar_A`date +"%y%m%d%H%M%S"`.txt※1
/usr/samples/kernel/vmtune > vmtune`date +"%y%m%d%H%M%S"`.txt
instfix -i > instfix_i`date +"%y%m%d%H%M%S"`.txt
lslpp -hac >  lslpp_hac`date +"%y%m%d%H%M%S"`.txt
uname -a > uname_a`date +"%y%m%d%H%M%S"`.txt
env > env`date +"%y%m%d%H%M%S"`.txt
set > set`date +"%y%m%d%H%M%S"`.txt
ipcs -a > ipcs_a`date +"%y%m%d%H%M%S"`.txt
```

Linux の場合

```
df > df`date +"%y%m%d%H%M%S"`.txt
ps -eflm > ps`date +"%y%m%d%H%M%S"`.txt
vmstat 1 5 > vmstat`date +"%y%m%d%H%M%S"`.txt
netstat -s > netstat_s`date +"%y%m%d%H%M%S"`.txt
netstat -an > netstat_an`date +"%y%m%d%H%M%S"`.txt
iostat 1 5 > iostat`date +"%y%m%d%H%M%S"`.txt※2
top n 5 > top`date +"%y%m%d%H%M%S"`.txt
sar -A 1 5 > sar`date +"%y%m%d%H%M%S"`.txt※2
sysctl -a > sysctl`date +"%y%m%d%H%M%S"`.txt
rpm -qa > rpm_qa`date +"%y%m%d%H%M%S"`.txt
rpm -qai > rpm_qai`date +"%y%m%d%H%M%S"`.txt
uname -a > uname_a`date +"%y%m%d%H%M%S"`.txt
env > env`date +"%y%m%d%H%M%S"`.txt
set > set`date +"%y%m%d%H%M%S"`.txt
ipcs > ipcs`date +"%y%m%d%H%M%S"`.txt
ipcs -t > ipcs_t`date +"%y%m%d%H%M%S"`.txt
ipcs -p > ipcs_p`date +"%y%m%d%H%M%S"`.txt
ipcs -c > ipcs_c`date +"%y%m%d%H%M%S"`.txt
ipcs -u > ipcs_u`date +"%y%m%d%H%M%S"`.txt
ipcs -l > ipcs_l`date +"%y%m%d%H%M%S"`.txt
```

注※1

コマンドの実行には root 権限が必要です。

注※2

sar コマンドおよび iostat コマンドを実行するためには，Linux に含まれている sysstat パッケージをインストールする必要があります。

4.12.2　OS のログの取得

トラブルシュート情報として必要な OS のログの取得について，OS ごとに説明します。

(1)　Windows の場合

トラブルシュート情報として必要な OS のログを次に示します。

表 4-30　トラブルシュート情報として必要な OS のログ情報

情報の種類	デフォルトのファイル名
イベントログ	イベントビューアを開き，アプリケーションとシステムのログを保存します。

(2)　UNIX の場合

トラブルシュート情報として必要な OS のログ（syslog）の格納場所を次に示します。

AIX の場合

/var/adm/ras 下すべて

Linux の場合

/var/log 下すべて

4.13　OS の統計情報

OS の統計情報の取得について，OS ごとに説明します。

4.13.1　Windows の場合

障害発生後のパフォーマンスログを保存しておきます。「パフォーマンス」の操作の詳細については，OS 付属のマニュアルなどを参照してください。

ポイント

OS の統計情報は，あらかじめ，OS 付属の「パフォーマンス」でパフォーマンスログの取得を開始している場合にだけ取得できます。

J2EE サーバ実行中に次のシステムモニタのログを 60 秒間隔で採取します。具体的な設定方法は，OS 付属のマニュアルなどを確認してください。

表 4-31　システムモニタの設定

パフォーマンスオブジェクト	インスタンス	項目名	説明
processor	−	%Processor Time	CPU 使用率（非アイドル状態のスレッドを除く合計値）。
		%Privileged Time	CPU 使用率（カーネルモード分）。
		%User Time	CPU 使用率（ユーザモード分）。
memory	−	Cache Bytes	ファイルシステム キャッシュが現在使用しているバイト数。
		Cache Faults/sec	1 秒間当たりのメモリの別の場所からの取り出しか，ディスクから取り出しの回数。
		Page Faults/sec	1 秒間当たりのページフォールトの数。
		Transition Faults/sec	1 秒間当たりのフォールトの数。
process	_Total	Handle Count	現在オープンしているハンドルの総数。
		Page Faults/sec	ページフォールトの発生率。
		Private Bytes	メモリ使用量（バイト）。
		Virtual Bytes	仮想メモリ使用量（バイト）。
		Working Set Bytes	実メモリ使用量（バイト）。
	cjstartsv	%Processor Time	CPU 使用率（非アイドル状態のスレッドを除く合計値）。
		%Privileged Time	CPU 使用率（カーネルモード分）。

パフォーマンスオブジェクト	インスタンス	項目名	説明
		%User Time	CPU 使用率（ユーザモード分）。
		Page Faults/sec	ページフォールトの発生率。
		Thread Count	スレッド数。
		Private Bytes	メモリ使用量（バイト）。
		Virtual Bytes	仮想メモリ使用量（バイト）。
		Working Set Bytes	実メモリ使用量（バイト）。

（凡例） －：該当しない

4.13.2 UNIX の場合

アプリケーションサーバ起動中に次に示すコマンドを実行して，OS の統計情報を取得します。60 秒間隔で取得することを推奨しますが，ディスク容量に応じて取得間隔を決めてください。なお，取得間隔を長くすると，OS の統計情報の取得による性能劣化を少なくできますが，OS の統計情報の精度が低下することがあります。

AIX の場合

```
ps -efl
ps -A -m -o THREAD
vmstat -t
vmstat -s
lsps -s
svmon -P <cjstartsv，cjstartwebのプロセスID>※1
svmon -G ※1
sar -A 1※1
```

Linux の場合

```
ps -eflm
top n 1
vmstat
sar -A 1 1※2
```

注※1

コマンドの実行には root 権限が必要です。

注※2

sar コマンドを実行するためには，Linux に含まれている sysstat パッケージをインストールする必要があります。

4.14　アプリケーションサーバの定義情報

アプリケーションサーバの定義を取得します。この情報を使用して，障害が発生したときに設定されていた定義内容を確認します。

Component Container に関する定義情報

次のディレクトリ下に格納されているファイル一式を取得します。

Windows の場合

- <Application Server のインストールディレクトリ>¥CC¥server¥usrconf¥ejb¥<サーバ名称>

UNIX の場合

- /opt/Cosminexus/CC/server/usrconf/ejb/<サーバ名称>

トラブルシューティング用に保存するファイルからは，パスワードなどの公開できない情報を削除することをお勧めします。

4.15　J2EE サーバまたはバッチサーバの作業ディレクトリの内容

システムでトラブルが発生した場合に，保守員による原因究明のため作業ディレクトリを調査に使用する場合があります。**作業ディレクトリ**は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に，ejb.public.directory パラメタで指定したディレクトリです。なお，作業ディレクトリのデフォルトのディレクトリパスは「<Application Server のインストールディレクトリ>¥CC¥server¥public」（Windows の場合），または「/opt/Cosminexus/CC/server/public」（UNIX の場合）です。

設定を確認した上で，トラブルが発生した場合に保存しておいてください。

4.16　アプリケーションサーバのリソース設定情報

コネクションプールの設定などリソースの設定を確認するために，リソースの設定情報を取得します。設定情報の格納場所は次のとおりです。

- Windows の場合
  - <作業ディレクトリ>¥ejb¥<サーバ名称>¥import
  - <作業ディレクトリ>¥ejb¥<サーバ名称>¥rars
- UNIX の場合
  - <作業ディレクトリ>/ejb/<サーバ名称>/import
  - <作業ディレクトリ>/ejb/<サーバ名称>/rars

4.17 Web サーバログ

システムで使用している Web サーバ（HTTP Server または Microsoft IIS）のログを取得します。

- HTTP Server の場合

  ログの格納場所を次に示します。

  - Windows の場合

    <Application Server のインストールディレクトリ>¥httpsd¥logs

  - UNIX の場合

    /opt/hitachi/httpsd/logs（デフォルト）

- Microsoft IIS の場合

  ログの格納場所を次に示します。

  C:¥inetpub¥logs（C:の部分にはシステムドライブを指定します）

4.18　JavaVM のスタックトレース情報

UNIX では，JavaVM が異常終了して core ダンプが出力された場合，異常終了した原因の究明に必要な情報（**スタックトレース情報**）を，javatrace コマンドで取得できます。javatrace コマンドでは，出力された core ダンプから，スタックトレース情報を取得します。javatrace コマンドは，「/opt/Cosminexus/jdk/bin」にインストールされています。

javatrace コマンドの実行形式を次に示します。

```
javatrace coreダンプのファイル名称 coreダンプを生成した実行ファイルの名称
```

JavaVM が異常終了して，「core」というファイル名称で core ダンプが作成されているときにこのコマンドを実行すると，実行結果として，カレントディレクトリ下に「javatrace.log」というファイルが出力されます。このファイルは保守員に送付してください。

実行例

JavaVM が異常終了して core ダンプが作成されたときに出力されるメッセージの例を，次に示します。

```
   :
# You can get further information from javatrace.log file generated
# by using javatrace command.
# usage: javatrace core-file-name loadmodule-name [out-file-name] [-l(library-name)...]
# Please use javatrace command as follows and submit a bug report
# to Hitachi with javatrace.log file:
# [/opt/Cosminexus/jdk/bin/javatrace core /opt/Cosminexus/CC/server/bin/cjstartsv]
#
```

メッセージ内に表示される javatrace コマンドの文字列を実行します。この例の場合は，「/opt/Cosminexus/jdk/bin/javatrace core /opt/Cosminexus/CC/server/bin/cjstartsv」を実行してください。実行結果として，カレントディレクトリに「javatrace.log」というファイルが出力されます。

なお，OS によっては実際に出力される core ダンプのファイル名が「core.プロセス ID」になる場合があります。その場合は，実際に出力された core ダンプのファイル名を javatrace の引数に指定してください。

4.19　明示管理ヒープ機能のイベントログ

明示管理ヒープ機能が出力するイベントログは，JavaVM 起動オプションである-XX:HitachiExplicitMemoryJavaLog オプションで指定したファイルに出力されます。また，出力される内容は，-XX:HitachiExplicitMemoryLogLevel オプションに指定したログ出力レベルによって異なります。

これらのオプションの設定方法については，「3.3.17(3)　明示管理ヒープ機能のイベントログ取得の設定」を参照してください。

JavaVM 起動オプションについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.　JavaVM 起動オプション」を参照してください。

4.20 Component Container 管理者のセットアップコマンドの実行情報（UNIX の場合）

Component Container 管理者を設定している場合，Component Container 管理者のセットアップ（cjenvsetup コマンド）の実行情報はメッセージログおよびテキストファイルに出力されます。過去の実行情報が存在する場合，種別ごとに 4 世代前まで保存され最新の実行情報を含め最大 5 ファイルまで保存されます。

また，上書きインストールで，上書きインストール前に Component Container 管理者のセットアップを行っていた場合は，Component Container 管理者のセットアップ（cjenvsetup コマンド）は，インストール時に自動的に実行されます。

実行情報の出力先を次に示します。

表 4-32 Component Container 管理者のセットアップ（cjenvsetup コマンド）の実行情報の出力先

分類	内容	ログ出力先およびログファイル名
メッセージログ	cjenvsetup コマンドの稼働ログ	/opt/Cosminexus/CC/logs/cjenvsetupmessage[n]※.log
テキストファイル	変更前ファイル（ディレクトリ）情報	/opt/Cosminexus/CC/logs/before_cjenvsetup_files[n]※.txt
テキストファイル	変更後ファイル（ディレクトリ）情報	/opt/Cosminexus/CC/logs/after_cjenvsetup_files[n]※.txt

注※　[n]には，世代数（1 から 4 まで）が付きます。最新の実行情報には世代数は付きません。ログファイルは最大 5 ファイルが保存されます。

出力される内容の詳細については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「4.1.4　Component Container 管理者を設定するときの注意事項（UNIX の場合）」を参照してください。

5 トラブルの分析

この章では，トラブルシューティングで使用するログやトレース情報について説明します。

5.1　この章の構成

トラブルシューティングに関する説明のうち，トラブルシューティングの資料の出力内容について説明します。

取得した資料に出力される内容を基に，トラブルの原因を判別してください。なお，「2.3　資料の取得」で取得した資料のうち，OS の統計情報については，ご使用の OS 付属のマニュアルなどを参照してください。また，メモリダンプは，保守員が確認する情報なので，ここでは説明しません。

注意事項

- Windows の場合，大きいサイズのログファイル（3 メガバイト以上）をテキストエディタなどで開くと，マシンに負担が掛かり，システムに影響を与えるおそれがあります。ログファイルを参照する際には，十分に注意してください。
- Windows を使用している場合，次の点に注意してください。

  ログや PRF トレースに出力する内容に Unicode の補助文字が含まれている場合，その文字は正しく出力されません。ただし，それ以外の出力内容や，アプリケーションの動作に問題はありません。Unicode の補助文字は，Microsoft Edge などのクライアントから送信されたリクエストに含まれることがあります。

この章の構成を次の表に示します。

表 5-1　この章の構成（トラブルシューティングで使用する資料の出力内容）

分類	タイトル	参照先
解説	アプリケーションサーバのログ	5.2
	EJB クライアントアプリケーションのログの内容	5.3
	性能解析トレースの内容	5.4
	JavaVM のスレッドダンプの内容	5.5
	JavaVM の GC ログの内容	5.6
	JavaVM ログ（JavaVM ログファイル）	5.7
	JavaVM が出力するメッセージログの内容（標準出力およびエラーリポートファイル）	5.8
	OS の状態情報および OS のログの内容	5.9
	JavaVM スタックトレース情報の内容	5.10
	明示管理ヒープ機能のイベントログの内容	5.11

なお，トラブルシューティングの概要，資料の出力先と出力方法，および資料の取得や出力に関する設定については，それぞれ次の個所を参照してください。

- トラブルシューティングの概要と，資料を自動で出力する方法
  「2.　トラブルシューティング」
- 資料の取得や出力に関する設定
  「3.　トラブルシューティングのための準備」
- 資料のデフォルトでの出力先と，資料を個別に出力する方法
  「4.　トラブルシューティングで必要な資料の出力先と出力方法」

5.2　アプリケーションサーバのログ

この節では，アプリケーションサーバのログの調査方法について説明します。

アプリケーションサーバのログのメッセージログ・ユーザログを調査することでエラーの発生原因を調べることができます。また，プロセス障害などの場合も，処理の進行具合や障害の兆候を確認できます。

アプリケーションサーバのログのうち，EJB クライアントアプリケーションのシステムログを参照する場合の留意事項については，「2.6.1　EJB クライアントアプリケーションのシステムログに関する留意事項」を参照してください。

アプリケーションサーバのログには，次の三つの種別があります。

- トレース共通ライブラリ形式（シングルプロセス）
- トレース共通ライブラリ形式（マルチプロセス）
- 独自形式

トレース共通ライブラリ形式とは，トレース共通ライブラリを使用して出力するログです。トレース共通ライブラリ形式については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「8.2.2　ユーザログ出力の仕組み」を参照してください。

独自形式は，トレース共通ライブラリ形式以外の形式で出力されるログです。

ここでは，アプリケーションサーバのログについて，種別ごとに分類して説明します。説明するアプリケーションサーバのログは次のログです。

- トレース共通ライブラリ形式のログ
- イベントログ
- syslog
- 監査ログで出力するログ
- CJMS プロバイダで出力するログ

なお，トレース共通ライブラリ形式のログには，時刻による出力先の切り替えおよびシフトモードでのファイル名付与が有効になるものがあります。詳細は，「3.2.1　設定できる内容」を参照してください。

出力されるログと対応するログの種別，ならびに時刻指定切り替えおよびシフトモードに対応しているかどうかを，ログの取得対象ごとに示します。

表 5-2　J2EE サーバのログ種別

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
メッセージログ	稼働ログ	トレース共通ライブラリ形式（シングルプロセス）	○
	ログ稼働ログ	独自形式	−
	J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの稼働ログ	トレース共通ライブラリ形式（シングルプロセス）	○
	J2EE アプリケーションに含めて使用するリソースアダプタの稼働ログ	トレース共通ライブラリ形式（シングルプロセス）	○
	Web サーブレットログ	トレース共通ライブラリ形式（シングルプロセス）	○
ユーザログ	ユーザ出力ログ	トレース共通ライブラリ形式（シングルプロセス）	○
	ユーザエラーログ	トレース共通ライブラリ形式（シングルプロセス）	○
	JavaVM の保守情報および GC のログ	独自形式	−
	明示管理ヒープ機能のイベントログ	独自形式	−
例外ログ	障害発生時の例外情報	トレース共通ライブラリ形式（シングルプロセス）	○
保守用ログ	保守情報	トレース共通ライブラリ形式（シングルプロセス）	○
	コンソールメッセージ	トレース共通ライブラリ形式（シングルプロセス）	○
	EJB コンテナの保守情報	トレース共通ライブラリ形式（シングルプロセス）	○
	Web コンテナの保守情報	トレース共通ライブラリ形式（シングルプロセス）	○
	起動プロセス標準出力情報	独自形式	−
	起動プロセス標準エラー情報	独自形式	−
	終了プロセス情報	トレース共通ライブラリ形式（シングルプロセス）	−

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
イベントログ	J2EE サーバの起動，停止または異常終了を示すログ	トレース共通ライブラリ形式（シングルプロセス）	－
syslog	J2EE サーバの起動，停止または異常終了を示すログ	トレース共通ライブラリ形式（シングルプロセス）	－
アクセスログ	HTTP 通信の処理結果および WebSocket 通信の処理結果	アクセスログ形式	○

（凡例）○：対応している　－：対応していない，または該当しない

表 5-3　サーバ管理コマンドのログ

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
メッセージログ	稼働ログ	トレース共通ライブラリ形式（マルチプロセス）	－
	ログ稼働ログ	独自形式	－
例外ログ	障害発生時の例外情報	トレース共通ライブラリ形式（マルチプロセス）	－
保守用ログ	保守情報	トレース共通ライブラリ形式（マルチプロセス）	－
	コンソールメッセージ	トレース共通ライブラリ形式（マルチプロセス）	－
	サーバ管理コマンドの保守情報	トレース共通ライブラリ形式（マルチプロセス）	－

（凡例）－：対応していない，または該当しない

表 5-4　リソースアダプタバージョンアップコマンド（cjrarupdate）のログ

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
メッセージログ	稼働ログ	トレース共通ライブラリ形式（マルチプロセス）	－
例外ログ	障害発生時の例外情報	トレース共通ライブラリ形式（マルチプロセス）	－
保守用ログ	保守情報	トレース共通ライブラリ形式（マルチプロセス）	－

（凡例）－：対応していない，または該当しない

表 5-5　移行コマンド（cjenvupdate）のログ

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
メッセージログ	cjenvupdate コマンドの稼働ログ	トレース共通ライブラリ形式（マルチプロセス）	－
例外ログ	cjenvupdate コマンドの例外情報	トレース共通ライブラリ形式（マルチプロセス）	－
保守用ログ	cjenvupdate コマンドの保守情報	トレース共通ライブラリ形式（マルチプロセス）	－

（凡例）－：対応していない，または該当しない

表 5-6　リソース枯渇監視のログ

監視対象リソース	種別	時刻指定切り替えおよびシフトモードへの対応
メモリ	トレース共通ライブラリ形式（シングルプロセス）	○
ファイルディスクリプタ	トレース共通ライブラリ形式（シングルプロセス）	○
スレッド	トレース共通ライブラリ形式（シングルプロセス）	○
スレッドダンプ	トレース共通ライブラリ形式（シングルプロセス）	○
HTTP リクエスト実行待ちキュー	トレース共通ライブラリ形式（シングルプロセス）	○
HTTP セッション数	トレース共通ライブラリ形式（シングルプロセス）	○

監視対象 リソース	種別	時刻指定切り替えおよびシフトモードへの対応
コネクションプール	トレース共通ライブラリ形式（シングルプロセス）	○

（凡例）○：対応している

表 5-7　運用管理エージェント・運用監視エージェント・Management Server のログ

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
統合ログ	統合メッセージログ	トレース共通ライブラリ形式（マルチプロセス）	○
	統合トレースログ	トレース共通ライブラリ形式（マルチプロセス）	○
	コマンド保守ログ	トレース共通ライブラリ形式（マルチプロセス）	－
運用管理エージェント	運用管理エージェントの標準エラー出力	トレース共通ライブラリ形式（シングルプロセス）	－
	運用管理エージェントの標準出力	トレース共通ライブラリ形式（シングルプロセス）	－
	運用管理エージェントの標準エラー出力コマンドライン	独自形式	－
	運用管理エージェントのログ	トレース共通ライブラリ形式（シングルプロセス）	－
	運用管理エージェントの起動・停止コマンドのログ	トレース共通ライブラリ形式（シングルプロセス）	－
	運用管理エージェントの保守ログ	トレース共通ライブラリ形式（シングルプロセス）	－
	コンソールログ	トレース共通ライブラリ形式（シングルプロセス）	○
	運用管理エージェントサービスのログ	トレース共通ライブラリ形式（シングルプロセス）	－
	運用管理エージェントサービスの標準出力	独自形式	－

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
	運用管理エージェントサービスの標準エラー出力	独自形式	−
運用監視エージェント	運用監視エージェントのログ・トレース J2EE サーバ用システム JP1 イベントおよび J2EE サーバ用ユーザ JP1 イベントのログ Management イベント発行ログ	トレース共通ライブラリ形式（シングルプロセス）	−
Management Server	Management Server サービスのログ	トレース共通ライブラリ形式（シングルプロセス）	−
	Management Server サービスの標準エラー出力	独自形式	−
	Management Server サービスの標準出力	独自形式	−
	Management Server サービス起動・停止コマンド	トレース共通ライブラリ形式（シングルプロセス）	−
	Management Server のログ 運用管理サーバ※のシステム JP1 イベントのログ	トレース共通ライブラリ形式（シングルプロセス）	−
	mngenvsetup コマンドの実行ログ	トレース共通ライブラリ形式（シングルプロセス）	−
	Management Server の保守ログ	トレース共通ライブラリ形式（シングルプロセス）	−

（凡例）○：対応している　−：対応していない，または該当しない

注※　アプリケーションサーバの運用管理サーバのことです。

表 5-8　仮想サーバマネージャの内部構築ツールおよびサーバ通信エージェントのログ

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
仮想サーバマネージャの内部構築ツール	仮想サーバマネージャの内部構築ツールのログ	トレース共通ライブラリ形式（シングルプロセス）	○

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
	仮想サーバマネージャの内部構築ツールの保守ログ	トレース共通ライブラリ形式（シングルプロセス）	－
サーバ通信エージェント	サーバ通信エージェントのログ	トレース共通ライブラリ形式（シングルプロセス）	○
	サーバ通信エージェントサービスのログ	トレース共通ライブラリ形式（シングルプロセス）	－
	サーバ通信エージェントの起動・停止コマンドのログ	トレース共通ライブラリ形式（シングルプロセス）	－
	サーバ通信エージェントの標準エラー出力	トレース共通ライブラリ形式（シングルプロセス）	－
	サーバ通信エージェントの標準出力	トレース共通ライブラリ形式（シングルプロセス）	－
	コンソールログ	トレース共通ライブラリ形式（シングルプロセス）	○
	サーバ通信エージェントの保守ログ	トレース共通ライブラリ形式（シングルプロセス）	－
	サーバ通信エージェントサービスの保守ログ	トレース共通ライブラリ形式（シングルプロセス）	－
	サーバ通信エージェントの起動・停止コマンドの保守ログ	トレース共通ライブラリ形式（シングルプロセス）	－
	サーバ通信エージェントの JavaVM ログファイル	トレース共通ライブラリ形式（シングルプロセス）	－

（凡例）○：対応している　－：対応していない，または該当しない

表 5-9　Performance Tracer のログ

内容	種別
PRF デーモンおよび PRF コマンドのログ	独自形式
モジュールトレース	独自形式
構造化例外発生ログ	独自形式
保守情報	独自形式

表 5-10　Component Transaction Monitor のログ

内容	種別
CTM デーモンおよび CTM コマンドのログ	独自形式
保守情報	独自形式

表 5-11　監査ログで出力するログ

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
メッセージログ	監査ログのメッセージログ	トレース共通ライブラリ形式（マルチプロセス）	－
例外ログ	監査ログの例外情報	トレース共通ライブラリ形式（マルチプロセス）	－

（凡例）－：対応していない，または該当しない

表 5-12　CJMS プロバイダで出力するログ

分類	内容	種別	時刻指定切り替えおよびシフトモードへの対応
メッセージログ	CJMSP ブローカーのメッセージログ	トレース共通ライブラリ形式（シングルプロセス）	－
	管理コマンド（cjmsicmd コマンド）のメッセージログ	トレース共通ライブラリ形式（マルチプロセス）	－
	CJMSP リソースアダプタのメッセージログ	トレース共通ライブラリ形式（シングルプロセス）	○
例外ログ	CJMSP ブローカーの例外ログ	トレース共通ライブラリ形式（シングルプロセス）	－
	管理コマンド（cjmsicmd コマンド）の例外ログ	トレース共通ライブラリ形式（マルチプロセス）	－
	CJMSP リソースアダプタの例外ログ	トレース共通ライブラリ形式（シングルプロセス）	○

（凡例）○：対応している　－：対応していない，または該当しない

5.2.1　トレース共通ライブラリ形式のログの出力形式と出力項目

トレース共通ライブラリ形式のログの出力形式と出力項目について説明します。

アプリケーションサーバのログは，**トレース共通ライブラリ形式**で出力されます。

(1) 出力形式

トレース共通ライブラリ形式のログの出力形式を次に示します。

```
番号　日付　時刻　AP名　pid tid メッセージID　種別　メッセージテキスト　CRLF
```

(2) 出力項目

トレース共通ライブラリ形式のログの出力項目を次に示します。

表 5-13　トレース共通ライブラリ形式のログの出力項目

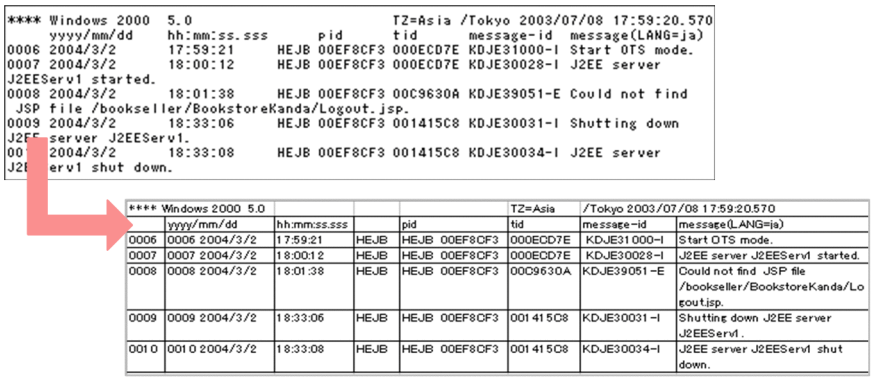
項目名	説明
番号	トレースレコードの通番を示す 4 けたの番号が出力されます。
日付	トレースの取得日付が「yyyy/mm/dd」の形式で出力されます。
時刻	トレースの取得時刻が「hh:mm:ss.sss」の形式で出力されます。
AP 名	プログラムを示す文字列が出力されます。
pid	プロセス ID が出力されます。
tid	スレッド ID が出力されます。
メッセージ ID	メッセージ ID が「XXXXnnnnn-Y」の形式で出力されます。
種別	トレース出力の契機となったイベント種別が出力されます。
メッセージテキスト	メッセージテキストが出力されます（最大 4,095 バイト）。4,095 バイトを超える場合は切り捨てられます。また，付加情報が出力されることもあります。
CRLF	レコードの終端符号（0x0D，0x0A）が出力されます。

ログファイルの編集表示について

トレース共通ライブラリ形式のログを Microsoft Excel で表示して，フィルタリングや並べ替えの機能を利用すると，トラブルの発生要因の調査を効率良く進められるようになります。

Microsoft Excel を利用したトレース共通ライブラリ形式のログの表示例を次に示します。

図 5-1　Microsoft Excel を利用したトレース共通ライブラリ形式のログの表示例



**** Windows 2000 5.0					TZ=Asia	/Tokyo 2003/07/08 17:59:20.570	
	yyyy/mm/dd	hh:mm:ss.sss		pid	tid	message-id	message(LANG=ja)
0006	0006 2004/3/2	17:59:21	HEJB	HEJB 00EF8CF3	000ECD7E	KDJE31000-I	Start OTS mode.
0007	0007 2004/3/2	18:00:12	HEJB	HEJB 00EF8CF3	000ECD7E	KDJE30028-I	J2EE server J2EEServ1 started.
0008	0008 2004/3/2	18:01:38	HEJB	HEJB 00EF8CF3	00C9630A	KDJE39051-E	Could not find JSP file /bookseller/BookstoreKanda/Logout.jsp.
0009	0009 2004/3/2	18:33:06	HEJB	HEJB 00EF8CF3	001415C8	KDJE30031-I	Shutting down J2EE server J2EEServ1.
0010	0010 2004/3/2	18:33:08	HEJB	HEJB 00EF8CF3	001415C8	KDJE30034-I	J2EE server J2EEServ1 shut down.

参考

- 区切るデータの形式として，「スペースによって右または左にそろえられた固定長フィールドデータ」を選択してください。
- ［データのプレビュー］ボックスで不要な矢印を削除してください。
- 各列の［表示形式］に［文字列］を指定してください。

5.2.2　トレース共通ライブラリ形式のログを参照する場合の注意

トレース共通ライブラリ形式のログを使用する場合の注意事項について説明します。

(1)　マルチプロセス・シングルプロセス共通の注意事項

マルチプロセス・シングルプロセス共通の注意事項について説明します。

- 出力されたログファイルを編集しないでください。
- テキストエディタの機能などを使用して，ファイルロックをしないでください。
- 出力されたログファイルにアクセス権限を手動で設定する場合は，適切なアクセス権を与えてください。
- 出力されたログファイルの更新時刻を変更しないでください。
- トレース出力中に，ログファイルを削除したり，ファイル名を変更したりしないでください。ログファイルの削除，またはファイル名の変更は，すべてのトレース出力プロセスを停止してから実行してください。
- ログファイルとともに，管理ファイル「xxxxxx.conf」が出力されます。トレース出力プロセスの稼働中，このファイルを変更，削除しないでください。
- ログファイルのサイズ，面数などの設定変更は，トレース出力プロセスを停止してから実施してください。その際，ログ出力ディレクトリ配下(サブディレクトリ含む)のすべてのログファイルと管理ファイル「xxxxxx.conf」を，退避または削除してください。

- ログファイルの所有者や所有グループを変更する場合は，管理ファイル「xxxxxx.conf」も同じ所有者および所有グループに変更してください。

(2) マルチプロセスのトレース共通ライブラリ参照時の注意事項

マルチプロセスに対応したトレース共通ライブラリ形式のログファイルを参照する場合の注意事項を次に示します。

- メッセージ末尾の改行コードは，使用している OS に関係なく CRLF です。
- ログファイルのファイルサイズ，面数，およびモードオプションの変更を有効にするために，ログファイルを削除する必要がある場合があります。ログファイルの削除は，すべてのトレース出力プロセスを停止してから実行してください。
- トレースが出力されても，ログファイルの更新時刻が更新されない場合があります。そのため，ファイルの更新時刻を基にトレースが出力されたかどうかを判断することはできません。

(3) マルチプロセスに対応したトレース共通ライブラリの旧バージョンからの変更点

マルチプロセスに対応したトレース共通ライブラリのログファイルを旧バージョンから変更しています。変更点を次に示します。

表 5-14 マルチプロセスに対応したトレース共通ライブラリの旧バージョンとの差異

項目	アプリケーションサーバ V9	アプリケーションサーバ V9 より前
ファイルサイズ	可変（指定サイズを超える場合があります）	固定（指定サイズ）
ファイル切り替えのタイミング	切り替えのタイミングを次に示します。 • ファイルサイズが指定サイズを超えた場合 • 時刻指定ローテーションの場合 • プロセス起動中に指定した時刻に達した場合	ファイルサイズ＋出力しようとしているサイズが指定サイズを超えた場合
ファイル切り替え時のサイズ	0（全行削除します）	固定（上書き保存）
ファイル切り替え動作（ラップアラウンド方式）	ファイルの先頭からトレースデータを上書き出力（ラップアラウンド前の情報はそのまま残ります）	ファイルの先頭からトレースデータを上書き出力（ラップアラウンド前の情報はそのまま残ります）
ファイル切り替え動作（シフト方式）	全行削除し，ファイルの先頭からトレースデータを出力	−
ログファイル名通番規則（ラップアラウンド方式）	xxxxxx1.log, xxxxxx2.log, xxxxxx3.log, …	xxxxxx1.log, xxxxxx2.log, xxxxxx3.log, …
ログファイル名通番規則（シフト方式）	xxxxxx.log, xxxxxx1.log,	−

項目	アプリケーションサーバ V9	アプリケーションサーバ V9 より前
	xxxxxx2.log, …	
終端識別子	なし（ファイルの終端）	あり※
管理ファイル	作成する	作成する
プロセス再起動時の出力ファイル判定	タイムスタンプ	管理ファイル，またはタイムスタンプ

（凡例）

－：該当なし。

注※

旧バージョンのマルチプロセスに対応したトレース共通ライブラリでの終端識別子を次に示します。

```
EOF CRLF CRLF CRLF CRLF----------< End of Data >----------CRLF CRLF
```

EOF はトレースデータの終端を表す文字（0x1A）です。CRLF は，改行（0x0D，0x0A）を表します。

表 5-15　マルチプロセスに対応したトレース共通ライブラリ形式の変更ファイル一覧

ログ分類	ログ種別	ログファイルと標準のファイル出力先
共通ログ	監査ログ	<製品のインストールディレクトリ>/auditlog/audit?.log
		<製品のインストールディレクトリ>/auditlog/rasexception?.log
		<製品のインストールディレクトリ>/auditlog/rasmessage?.log
機能固有の公開ログ	性能，障害解析トレース	<製品の作業ディレクトリ>/ejb/<J2EE サーバ名称>/logs/RM/maintenance/mtd_<RM の表示名>_??.log
		<製品の作業ディレクトリ>/ejb/<J2EE サーバ名称>/logs/RM/maintenance/shq_<RM の表示名>_??.log
		<製品の作業ディレクトリ>/ejb/<J2EE サーバ名称>/logs/RM/maintenance/lin_<RM の表示名>_??.log
		<製品のインストールディレクトリ>/CC/server/public/ejb/<J2EE サーバ名称>csmxsec_trace?.log
	メッセージログ	<製品の作業ディレクトリ>/ejb/<J2EE サーバ名称>/logs/CJW/cjwmessage?.log
		<製品の作業ディレクトリ>/ejb/<J2EE サーバ名称>/logs/CJR/cjrmessage?.log
		<製品の作業ディレクトリ>/ejb/<J2EE サーバ名称>/logs/WS/<log_file_prefix>-j2ee-<J2EE サーバのサーバ名>-<log_file_num>.log
		<製品の作業ディレクトリ>/ejb/<J2EE サーバ名称>/logs/WS/ c4webcl-default-?.log

ログ分類	ログ種別	ログファイルと標準のファイル出力先
Manager のログ	統合メッセージログ	<製品のインストールディレクトリ>/manager/log/message/mngmessage?.log
	統合トレースログ	<製品のインストールディレクトリ>/manager/log/trace/mngtrace?.log
	コンソールログ	<製品のインストールディレクトリ>/manager/log/processConsole?.log
	仮想化機能の内部構築ツールのログ	<製品のインストールディレクトリ>/manager/setup/log/rasetup?.log
	サーバ通信エージェントのログ	<製品のインストールディレクトリ>/sinagent/log/sinaviagent?.log
	サーバ通信エージェントのコンソールログ	<製品のインストールディレクトリ>/sinagent/log /processConsole?.log

5.2.3 NIO HTTP サーバのアクセスログの出力形式と出力項目

NIO HTTP サーバのアクセスログの出力形式と出力項目について説明します。

アクセスログには，NIO HTTP サーバのリクエストの処理結果が出力されます。NIO HTTP サーバで扱う，HTTP 通信と WebSocket 通信に分けて，それぞれの出力形式と出力項目について説明します。

(1) HTTP 通信のアクセスログ

出力内容を次の表に示します。

表 5-16 HTTP 通信の出力内容

フォーマット引数	出力内容	出力例
%a	Web クライアントの IP アドレス。	10.20.30.40
%A	J2EE サーバの IP アドレス。	10.20.30.100
%b	レスポンスボディの送信バイト数。 0 バイトのときは「-」になる。	2048
%B	レスポンスボディの送信バイト数。 0 バイトのときは「0」になる。	1024
%h	Web クライアントのホスト名または IP アドレス。 ホスト名が得られない場合は IP アドレスになる。	10.20.30.40
%H	リクエストプロトコル。	HTTP/1.1
%l	リモートログ名。 常に「-」になる※1。	-

フォーマット引数	出力内容	出力例
%m	リクエストメソッド	GET
%p	Web クライアントからのリクエストを受け付けたポート番号。	80
%q	クエリ文字列。 「?」から始まる。 クエリ文字列がない場合は空文字になる。	?id=100&page=15
%r	リクエストライン。	GET /index.html HTTP/1.0
%s	最終ステータスコード。	200
%S※2	クッキー名 JSESSIONID の値を出力する。 クッキー名 JSESSIONID の値がない場合は「-」になる。	00455AFE4DA4E7B7789F247B8FE5D605
%t	Web クライアントのリクエスト処理を開始した時刻を秒精度で表示。 [dd/MMM/YYYY:HH:mm:ss Z]	[18/Jan/2005:13:06:10 +0900]
%T	Web クライアントのリクエストの処理に要した時間（秒単位）。	2
%d	Web クライアントのリクエスト処理を開始した時刻をミリ秒精度で表示。 [dd/MMM/YYYY:HH:mm:ss.nnn Z]（nnn はミリ秒）	[18/Jan/2005:13:06:10.152 +0900]
%D	Web クライアントのリクエストの処理に要した時間（ミリ秒単位）。	38
%u	ベーシック認証ユーザ名，フォーム認証ユーザ名。 認証ユーザ名がない場合は「-」になる。	user
%U	リクエストファイルパス。	/index.html
%v	J2EE サーバのローカルホスト名。	server
%{foo}i※3	リクエストヘッダ foo の内容。 foo ヘッダが存在しない場合は「-」になる。	%{Host}i の場合 www.example.com:8888
%{foo}c	Web クライアントが送信した Cookie 情報で Cookie の名前が foo の内容を表示する。 Cookie の名前に foo がない場合は「-」になる。	%{MYSESSIONID}c の場合 00455AFE4DA4E7B7789F247B8FE5D605
%{foo}o※3	レスポンスヘッダ foo の内容。 foo ヘッダが存在しない場合は「-」になる。	%{Server}o の場合 CosminexusComponentContainer
%rootap	ルートアプリケーション情報。	10.100.10.100/1234/0x0000000000000001

フォーマット引数	出力内容	出力例
%clport	Web クライアントからのリクエストを送信したポート番号。	888

注

- 上記の表以外の%から始まる文字（%G など）を指定した場合は，メッセージ KDJE39401-W を出力し，デフォルトフォーマットを使用します。また，%{foo}i，%{foo}c，%{foo}o で指定するヘッダの内容，または Cookie 名で 0 文字（%{}i など）を指定した場合は，メッセージ KDJE39401-W を出力し，デフォルトフォーマットを使用します。
- フォーマット形式に使用できる文字列長は 1024 文字までです。1024 文字を超えた場合は，メッセージ KDJE39400-W を出力しデフォルト値を使用します。
- フォーマット形式に使用できる文字は，ASCII コードの 32（10 進数）から 127（10 進数）未満の文字です。
- 文字列が何も指定されていない場合は，メッセージ KDJE39009-W を出力し，デフォルト値を使用します。
- 範囲外の文字を指定した場合は，メッセージ KDJE39401-W を出力しデフォルト値を使用します。

注※1

リモートログ名は，RFC1413 で規定されている Identification プロトコルによって得られる Web クライアント側のユーザ名です。

注※2

%S で表示される値は，標準で HTTP セッション ID として使用するクッキー名 JSESSIONID の値です。Servlet3.0 以降でクッキー名 JSESSIONID の名前を変更した場合は，%{foo}c を使用します。

注※3

一度の HTTP リクエストまたは HTTP レスポンスで同じヘッダ名を複数回送信する場合があります。この場合，すべてのヘッダの内容を「,」（コンマ）区切りで出力します。

フォーマット引数で記述した表記を次に示します。

```
%h %{X-Forwarded-For}i %l %u %d %rootap ”%r” %s %b %D %S
```

出力形式を次に示します。

```
Webクライアントのホスト名またはIPアドレス△X-Forwarded-Forヘッダ△リモートログ名△認証ユーザ名△Webクライアントのリクエスト処理を開始した時刻△ルートアプリケーション情報△”リクエストライン”△最終ステータスコード△HTTPヘッダを除く送信バイト数△Webクライアントのリクエストの処理に要した時間△HTTPセッションID
```

注

HTTP セッション ID のあとに改行されます。

（凡例）

△：半角スペース

出力例を次に示します。

```
10.20.30.40 50.60.70.80 - user [18/Jan/2005:13:06:10.152 +0900] 10.100.10.100/1234/0x0000000
000000001 "GET /index.html HTTP/1.0" 200 1024 38 00455AFE4DA4E7B7789F247B8FE5D605
```

(2) WebSocket 通信のアクセスログ

出力内容を次の表に示します。

表 5-17 WebSocket 通信の出力内容

フォーマット引数	出力内容	出力例
%TS	WebSocket フレームの送受信時刻。	2001/01/01 01:01:01.111 +0900
%IO	WebSocket フレームの送受信方向。IN または OUT が出力される。 IN： サーバインスタンスが WebSocket フレームを受信したことを示す。 OUT： サーバインスタンスが WebSocket フレームを送信したことを示す。	IN
%OPCODE	WebSocket フレームの種別。Text，Binary，Ping，Pong，または Close が出力される。	Text
%URI	リクエスト URI。	/websocket_server/test001
%FIN	WebSocket フレームの終端を示す識別子。CONT または FINAL が出力される。 CONT： WebSocket フレームの継続。 FINAL： WebSocket フレームの終端。	CONT

フォーマット引数	出力内容	出力例
%PAYLOADDATALEN	ペイロードデータ長。	100
%ROOTAP	ルートアプリケーション情報。	10.100.10.100/1234/0x0000000000000001
%CLIENTAP	クライアントアプリケーション情報。	10.100.10.100/1234/0x0000000000000001
%CLOSEREASON	WebSocket コネクションが切断された理由。	NORMAL_CLOSURE:closereason specified by WebSocketClient001
%CLIENTADDR	Web クライアントの IP アドレスとポート番号。	10.20.30.40:55555
%SERVERADDR	J2EE サーバの IP アドレスとポート番号。	10.20.30.100:44444
%SESSIONID	WebSocket セッション ID。	11111111-2222-3333-4444-555555555555
%MASK	WebSocket フレームの情報に設定された MASK を表す。 MASK： MASK されている。 NOMASK： MASK されていない。	MASK
%MASKKEY	WebSocket フレームの情報を MASK するためのキー。 MASK しない場合は「-」になる。	EEEEEEEE
%ISEXTENDED	エンドポイントがメッセージの送受信に設定されたフレームを表す。 BASE： 基本フレームを使用している。 EXTENDED： 拡張フレームを使用している。	BASE
%RSV	拡張のネゴシエーション中にクライアントエンドポイントによって設定された予約ビットを表す。	RSV-000
%FRAMEMAINTYPE	WebSocket フレームがデータフレームかコント	Data

フォーマット引数	出力内容	出力例
	ロールフレームかを指定する。 Data： データフレームを使用する。 Control： コントロールフレームを使用する。	
%PAYLOADDATA	ペイロードデータ。	aaaaaa
%PAYLOADDATA(n)	メッセージの最初と最後を示す一定数の文字だけを表示する場合のペイロードデータ。 バイナリの場合は常に「-」になる。	aaaaaaaaaa....aaaaaaaaaa

注

- フォーマット形式に使用できる文字列長は 1024 文字までです。1024 文字を超えた場合は，メッセージ KDJE39400-W を出力しデフォルト値を使用します。
- 文字列が何も指定されていない場合は，メッセージ KDJE39009-W を出力し，デフォルト値を使用します。
- 表内で定義されているフォーマット引数を半角スペースでつなげたものだけ指定できます。その際，次の点に注意してください。

  ・半角スペースの個数は保持されません。

  ・先頭，末尾のスペースは保持されません。

  ・フォーマット引数の順序に制限はありません。

  ・出現回数（同じものを複数回使用するなど）に制限があり，同じフォーマット引数を繰り返し定義した場合，初めに定義したフォーマット引数だけが有効になります。二つ目以降で定義したフォーマット引数は有効になりません。

フォーマット引数で記述した表記を次に示します。

```
%TS %IO %OPCODE %ROOTAP %URI %FIN %PAYLOADDATALEN %CLIENTAP %CLOSEREASON
```

出力形式を次に示します。

```
WebSocketフレームの送受信時刻△WebSocketフレームの送受信方向△WebSocketフレームの種別△ルートアプリケーション情報△リクエストURI△WebSocketフレームの終端を示す識別子△ペイロードデータ長△クライアントアプリケーション情報△WebSocketコネクションが切断された理由
```

注

ペイロードデータ長のあとに改行されます。

（凡例）

△：半角スペース

出力例を次に示します。

```
2001/01/01 01:01:01.111 +0900 IN Text10.100.10.100/1234/0x0000000000000001  /websocket_serve
r/test001 CONT 100 10.100.10.100/1234/0x0000000000000003 -
```

5.2.4　イベントログの出力形式と出力項目（Windows の場合）

イベントログの出力形式と出力項目について説明します。

イベントログは，J2EE サーバの起動，停止および異常終了時に出力されるログです。出力形式は次のとおりです。

```
ID文字列　[pid]:　メッセージテキスト
```

出力項目を次に示します。

表 5-18　イベントログの出力項目

項目名	説明
ID 文字列	アプリケーションを示す文字列として，「HEJB」が出力されます。
pid	プロセス ID が出力されます。
メッセージテキスト	メッセージテキストが出力されます。

5.2.5　syslog の出力形式と出力項目（UNIX の場合）

syslog の出力形式と出力項目について説明します。

syslog は，J2EE サーバの起動，停止および異常終了時に出力されるログです。出力形式は次のとおりです。

```
日付　時刻　ホスト名　ID文字列　[pid]:メッセージテキスト
```

出力項目を次に示します。

表 5-19　syslog の出力項目

項目名	説明
日付	メッセージを出力した日付です。
時刻	メッセージを出力した時刻です。
ホスト名	ホスト名を示す文字列が出力されます。
pid	プロセス ID が出力されます。
ID 文字列	アプリケーションを示す文字列として，「HEJB」が出力されます。
メッセージテキスト	メッセージテキストが出力されます。

5.3　EJB クライアントアプリケーションのログ

ここでは，EJB クライアントアプリケーションのログの種別について説明します。ログの種別の詳細，それぞれのログの出力形式と出力項目，および参照する場合の注意事項については，「5.2　アプリケーションサーバのログ」を参照してください。

EJB クライアントアプリケーションのログの種別を次に示します。

表 5-20　EJB クライアントアプリケーションのログの種別

種類	内容	種別
メッセージログ	稼働ログ	トレース共通ライブラリ形式（マルチプロセス）
	cjclstartap コマンドの稼働ログ	トレース共通ライブラリ形式（マルチプロセス）
	cjcldellog コマンドの稼働ログ	トレース共通ライブラリ形式（マルチプロセス）
ユーザログ	ユーザ出力ログ	トレース共通ライブラリ形式（マルチプロセス）
	ユーザエラーログ	トレース共通ライブラリ形式（マルチプロセス）
Java ログ	JavaVM の保守情報，GC のログ	独自形式
例外ログ	障害発生時の例外情報	トレース共通ライブラリ形式（マルチプロセス）

なお，EJB クライアントアプリケーションのログは，ファイルサイズごとに出力先を切り替えるラップアラウンドモードだけに対応しています。

5.4　性能解析トレース

性能解析トレースを調査することで，リクエストごとに処理進ちょくやボトルネックになっている処理を解析できます。また，セッショントレースを調査すると，セッションのライフサイクルが確認できます。

これらを基にして，トラブルが発生した個所を特定したり，ボトルネックになっている個所に適切に対処したりできます。必要に応じて確認してください。

性能解析トレースの出力内容については，「7.　性能解析トレースを使用した性能解析」を参照してください。

5.5　JavaVM のスレッドダンプ

JavaVM のスレッドダンプを調査すると，システムのデッドロックなどの java プログラムレベルでのトラブル要因の調査が容易になります。

出力される情報の種類は，J2EE サーバの起動時に指定しているオプションによって異なります。JavaVM の資料取得の設定については，「3.3.17　JavaVM の資料取得の設定」を参照してください。

5.5.1　スレッドダンプ情報の構成

JavaVM のスレッドダンプ情報の構成を次の表に示します。

表 5-21　スレッドダンプ情報の構成

出力情報	内容
ヘッダ	日付，JavaVM バージョン情報，起動コマンドラインを出力します。
システム設定	次の情報を出力します。 • JDK の実行環境のインストール場所を表す Java ホームパス • JDK を構成するライブラリのインストールディレクトリを表す Java DLL パス • システムクラスパス • Java コマンドオプション
動作環境	次の情報を出力します。 • ホスト名 • OS バージョン • CPU 情報 • リソース情報（UNIX の場合）
メモリ情報（Windows の場合）	現在のメモリ使用量および各未使用サイズ情報を出力します。
Java ヒープ情報	Java ヒープの各世代のメモリ使用状況を出力します。
JavaVM 内部メモリマップ情報	JavaVM 自身の確保しているメモリの領域情報を出力します。
JavaVM 内部メモリサイズ情報	JavaVM 自身の確保しているメモリのサイズ情報を出力します。
アプリケーション環境	次の情報を出力します。 • シグナルハンドラ • 環境変数
ライブラリ情報	ローディングされているライブラリの情報を出力します。
スレッド情報 <スレッド 1> : <スレッド n>	スレッドごとにスレッド情報を出力します。
Java モニタダンプ※	Java モニタオブジェクトの一覧を表示します。

出力情報	内容
JNI グローバル参照情報	JavaVM が保持している JNI のグローバル参照の数を出力します。
Explicit ヒープ詳細情報	明示管理ヒープ機能使用時には，Java プロセスのクラスごとに次の情報を出力します。 • Explicit ヒープ全体の利用状況 • Explicit メモリブロックごとの利用状況 また，明示管理ヒープ機能使用時に，eheapprof コマンドを実行すると，Explicit メモリブロック内のオブジェクトの統計情報，および Explicit メモリブロックの解放率情報を出力します。
クラス別統計情報	jheapprof コマンドで指定した Java プロセスのクラスごとに次の情報を出力します。 • インスタンスがメンバとして持つインスタンスの合計サイズ，およびインスタンスの参照関係 • static メンバが持つインスタンスの合計サイズ • Tenured 領域の増加原因となるオブジェクトのクラス，およびインスタンスの合計サイズ
フッタ	スレッドダンプが終了した時刻を表示します。

注※　UNIX の場合，notify 待ちの一覧が表示されないことがあります。

JavaVM のスレッドダンプ情報の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]HitachiThreadDump（拡張スレッドダンプ情報出力オプション）」を参照してください。なお，クラス別統計情報については，「9.3　クラス別統計機能」を参照してください。Explicit ヒープ詳細情報については，「5.5.3　Explicit ヒープ詳細情報の出力内容」を参照してください。

5.5.2　スレッドダンプと性能解析トレースファイルとの対応

J2EE アプリケーションやバッチアプリケーションでスローダウンやハングアップが発生した時に，スレッドダンプと性能解析トレースを対応づけることで，問題が発生した個所を調査できます。

スレッドダンプと性能解析トレースファイルの対応づけには，スレッドダンプに出力されたスレッド情報の nativeID（OS レベルのスレッド ID）と性能解析トレースファイルに出力されたスレッド ID を使用します。スレッドダンプから対応する性能解析トレースファイルを特定する手順について説明します。

1. スレッドダンプ，および性能解析トレースファイルを収集します。

   スレッドダンプの収集方法については，「4.7　JavaVM のスレッドダンプ」を参照してください。性能解析トレースファイルの収集方法については，「7.3.1　性能解析トレースファイルの収集方法」を参照してください。

2. 使用するスレッドダンプ，および性能解析トレースファイルを選びます。

   スレッドダンプと性能解析トレースファイルが出力された時刻を基に，調査に使用するスレッドダンプ，および性能解析トレースファイルを選びます。出力された時刻は，次の情報を参考にしてください。

スレッドダンプ

ファイル名，およびファイルの末尾に出力される時刻

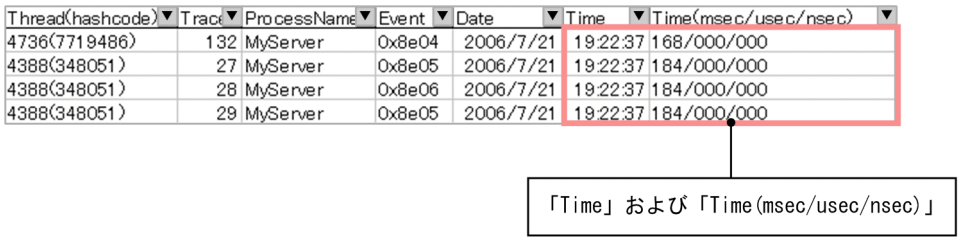
ファイルの末尾に出力される時刻の例を次に示します。

```
   :
   :
Full thread dump completed.    Fri Jul 21 19:22:47 2006
```

性能解析トレースファイル

「Time」および「Time(msec/usec/nsec)」

性能解析トレースファイルの「Time」および「Time(msec/usec/nsec)」の例を次の図に示します。



3. スレッドダンプの「nid」（16 進数）または「jid」（16 進数）を 10 進数に変換します。

- Windows，AIX の場合

  スレッドダンプの「nid」（16 進数）を 10 進数に変換します。

```
   :
   :
"VBJ ThreadPool Worker" daemon prio=5 jid=0x00054f93 tid=0x04cef380 nid=0x1124 in Obje
ct.wait() [0x0632f000..0x0632fd18]
  stack=[0x06330000..0x062f5000..0x062f1000..0x062f0000]
  [user cpu time=0ms, kernel cpu time=15ms] [blocked count=1, waited count=29]
at java.lang.Object.wait(Native Method)
   :
   :
```

  1124（16 進数）＝ 4388（10 進数）

- Linux の場合

  スレッドダンプの「jid」（16 進数）を 10 進数に変換します。

```
   :
   :
"main" prio=1 jid=0x00006d75 tid=0x00201d70 nid=0x1e51 waiting on condition [0x0000000
0..0xbfe80488]  stack=[0xbfe87000..0xbfc8c000..0xbfc88000..0xbfc87000]  [user cpu tim
e=1320ms, kernel cpu time=4280ms] [blocked count=5, waited count=4]
   :
   :
```

  6d75（16 進数）＝ 28021（10 進数）

4. 性能解析トレースファイルの「Thread（hashcode)」の値（10 進数）が手順 3.で 10 進数に変換した値と一致する行を探して，トレース情報を特定します。

- Windows，AIX の場合

  Thread（スレッド ID）の値が，10 進数に変換した値と一致する行を探します。

Thread(hashcode)	Trace	ProcessName	Event	Date	Time	Time(msec/usec/nsec)
4736(7719486)	132	MyServer	0x8e04	2006/7/21	19:22:37	168/000/000
4388(348051)	27	MyServer	0x8e05	2006/7/21	19:22:37	184/000/000
4388(348051)	28	MyServer	0x8e06	2006/7/21	19:22:37	184/000/000
4388(348051)	29	MyServer	0x8e05	2006/7/21	19:22:37	184/000/000

- Linux の場合

  hashcode（ハッシュコード）の値が，10 進数に変換した値と一致する行を探します。

Thread(hashcode)	Trace	ProcessName	Event	Date	Time	Time(msec/usec/nsec)
4736(7719486)	132	MyServer	0x8604	2008/10/9	9:55:48	168/000/000
3086362848(28021)	27	MyServer	0x8605	2008/10/9	9:55:48	673/992/000
3086362848(28021)	28	MyServer	0x8606	2008/10/9	9:55:48	673/992/000
3086362848(28021)	29	MyServer	0x8605	2008/10/9	9:55:48	673/992/000

5.5.3　Explicit ヒープ詳細情報の出力内容

Explicit ヒープ詳細情報には，Explicit ヒープ情報および Explicit メモリブロック情報が出力されます。Explicit メモリブロック情報は，Explicit メモリブロックが一つ以上ある場合に，その個数分出力されます。Explicit メモリブロック情報には，Explicit メモリブロック内のオブジェクト統計情報，および Explicit メモリブロックの解放率情報も出力されます。

Explicit ヒープ詳細情報の出力形式，出力項目および出力例を次に示します。

出力形式

eheapprof コマンドを実行しているかどうかによって出力形式が異なります。

- eheapprof コマンドを実行している場合

```
Explicit Heap Status
--------------------
 max <EH_MAX>, total <EH_TOTAL>, used <EH_USED>, garbage <EH_GARB> (<EH_PER1> used/max
, <EH_PER2> ¥
used/total, <EH_PER3> garbage/used), <EM_NUMS> spaces exist

 Explicit Memories(<EM_MGR_PTR>)
...
  "<EM_NAME>" eid=<EID>(<EM_PTR>)/<EM_TYPE>, total <EM_TOTAL>, used <EM_USED>, garbage
 <EM_GARB> ¥
(<EM_PER1> used/total, <EM_PER2> garbage/used, <FL_BLOCKS> blocks) <EM_STAT>
    deployed objects
    ________________Size__Instances__FreeRatio__Class_________________
                <ISIZE>     <INUM>   <FRATIO> <CNAME>
                ...
               <AISIZE>    <AINUM> total
...
```

注　出力形式で使用している記号については，「5.11.2(3)　イベントログの出力形式の説明で使用する記号」を参照してください。

- eheapprof コマンドを実行していない場合

```
Explicit Heap Status
--------------------
 max <EH_MAX>, total <EH_TOTAL>, used <EH_USED>, garbage <EH_GARB> (<EH_PER1> used/max
, <EH_PER2> ¥
used/total, <EH_PER3> garbage/used), <EM_NUMS> spaces exist

 Explicit Memories(<EM_MGR_PTR>)
...
  "<EM_NAME>" eid=<EID>(<EM_PTR>)/<EM_TYPE>, total <EM_TOTAL>, used <EM_USED>, garbage
 <EM_GARB> ¥
(<EM_PER1> used/total, <EM_PER2> garbage/used, <FL_BLOCKS> blocks) <EM_STAT>
...
```

注　出力形式で使用している記号については，「5.11.2(3)　イベントログの出力形式の説明で使用する記号」を参照してください。

出力項目

出力形式で示した各項目について説明します。

表 5-22　出力項目（Explicit ヒープ詳細情報）

分類	出力項目	出力内容	意味
Explicit ヒープ情報	<EH_MAX>	<const>K	Explicit ヒープの最大サイズが出力されます。単位はキロバイトです。
	<EH_TOTAL>	<const>K	確保済み Explicit ヒープサイズが出力されます。単位はキロバイトです。
	<EH_USED>	<const>K	利用済み Explicit ヒープサイズが出力されます。単位はキロバイトです。
	<EH_GARB>	<const>K	Explicit ヒープの内部状態が出力されます。
	<EH_PER1>	<decimal>%	Explicit ヒープ利用率（<EH_USED>/<EH_MAX>）が%表記で出力されます。
	<EH_PER2>	<decimal>%	Explicit ヒープ利用率（<EH_USED>/<EH_TOTAL>）が%表記で出力されます。
	<EH_PER3>	<decimal>%	Explicit ヒープの内部状態が出力されます。
	<EM_NUMS>	<const>	有効な Explicit メモリブロックの数が出力されます。
	<EM_MGR_PTR>	<ptr>	Explicit ヒープ制御のための内部情報があるメモリアドレスが出力されます。障害調査時などに利用します。
Explicit メモリブロック情報	<EM_NAME>	<letters>	Explicit メモリブロックの名称が出力されます。 Explicit メモリブロックの名称に多バイト文字が含まれている場合，出力内容は不定です（通常は文字化けして出力されます）。 Explicit メモリブロックの初期化とほぼ同時に出力された場合や，JavaVM が内部で生成した Explicit メモリブロックの場合は，"NULL"が出力されることがあります。

分類	出力項目	出力内容	意味
	<EID>	<const>	Explicit メモリブロックの ID が出力されます。
	<EM_PTR>	<ptr>	Explicit メモリブロック内部構造があるメモリアドレスが出力されます。障害調査時などに利用します。
	<EM_TYPE>	R|B|A	Explicit の種別が出力されます。R は，アプリケーションサーバの内部で利用されている Explicit メモリブロックを示します。B は，アプリケーションが利用している Explicit メモリブロックを示します。A は，自動配置設定ファイルを使って指定した Explicit メモリブロックを示します。
	<EM_TOTAL>	<const>K	Explicit メモリブロックのメモリ確保済みサイズが出力されます。単位はキロバイトです。
	<EM_USED>	<const>K	Explicit メモリブロックの利用済みサイズが出力されます。単位はキロバイトです。
	<EM_GARB>	<const>K	Explicit メモリブロックの内部状態が出力されます。単位はキロバイトです。
	<EM_PER1>	<decimal>%	Explicit メモリブロック利用率（<EM_USED>/<EM_TOTAL>）が%表記で出力されます。
	<EM_PER2>	<decimal>%	Explicit メモリブロックの内部状態が出力されます。
	<FL_BLOCKS>	<const>	常に 0 が出力されます。
	<EM_STAT>	Enable|Disable	Explicit メモリブロックのサブ状態が出力されます。
オブジェクト統計情報※1	<ISIZE>	<const>	あるクラスをインスタンス化したオブジェクトの Explicit メモリブロック内のサイズが出力されます。
	<INUM>	<const>	あるクラスをインスタンス化したオブジェクトの Explicit メモリブロック内の個数が出力されます。
	<CNAME>	<letters>	<ISIZE>および<INUM>が示すクラスの完全クラス名が出力されます。
	<AISIZE>	<const>	Explicit メモリブロック内の全オブジェクトの合計サイズが出力されます。
	<AINUM>	<const>	Explicit メモリブロック内の全オブジェクトの個数が出力されます。
オブジェクト解放率情報※2	<FRATIO>	<decimal>%	Explicit メモリブロックの自動解放処理で解放されたオブジェクトの割合（オブジェクト解放率）が%表記で出力されます。 オブジェクト解放率＝ （自動解放処理前のクラスのオブジェクト数－自動解放処理後のクラスのオブジェクト数）／自動解放処理前のクラスのオブジェクト数×100 なお，オブジェクト解放率情報出力時に，自動解放処理の対象とならなかった Explicit メモリブロックには，「－」が出力されます。

注　出力内容で使用している記号については，「5.11.2(3)　イベントログの出力形式の説明で使用する記号」を参照してください。

注※1　オブジェクト統計情報は，eheapprof コマンドを実行している場合に出力されます。なお，オブジェクト統計情報には，実際に作成したサイズ，個数よりも多く，int 型配列を示す"［I"が出力されることがあります。この場合，"［I"は，Explicit メモリブロック内で使用されていないオブジェクトを示します。Explicit メモリブロック内で使用されていないオブジェクトは，JavaVM の内部処理で int 型配列化されます。

注※2　オブジェクト解放率情報は，-freeratio オプションを指定した eheapprof コマンドを実行している場合に出力されます。

出力例

eheapprof コマンドを実行しているかどうかによって出力形式が異なります。

- eheapprof コマンドを実行している場合

```
Explicit Heap Status
--------------------
 max 31415926K, total 162816K, used 150528K, garbage 10004K (0.0% used/max, 91.1% used
/total, 6.6% garbage/used), 3 spaces exist

 Explicit Memories(0x12345678)

  "EJBMgrData" eid=1(0x02f25610)/R, total 54272K, used 50176K, garbage 0K (91.2% used/
total, 0.0% garbage/used, 0 blocks)
    deployed objects
    ________________Size__Instances__FreeRatio__Class________________
                35234568      10648          - java.util.HashMap
                 5678900      10668          - [Ljava.util.HashMap$Entry;
                 4456788       7436          - java.util.HashMap$Entry
                 4321000        200          - java.util.WeakHashMap
                 1234568        190          - [Ljava.util.WeakHashMap$Entry;
                  454400          4          - java.util.WeakHashMap$Entry
                51380224      29146 total

  "VJBStored" eid=3(0x02f25910)/B, total 54272K, used 50176K, garbage 10004K (90.7% us
ed/total, 19.9% garbage/used, 5 blocks)
    deployed objects
    ________________Size__Instances__FreeRatio__Class________________
                35234568      10648         49 java.util.HashMap
                 5678900      10668         43 [Ljava.util.HashMap$Entry;
                 4456788       7436         50 java.util.HashMap$Entry
                 4321000        200         32 java.util.WeakHashMap
                 1234568        190         45 [Ljava.util.WeakHashMap$Entry;
                  454400          4         22 java.util.WeakHashMap$Entry
                51380224      29146 total

  "ExplicitMemory-2" eid=2(0x02f25700)/B, total 54272K, used 50176K,  garbage 0K (91.1
% used/total, 0.0% garbage/used, 0 blocks)
    deployed objects
    ________________Size__Instances__FreeRatio__Class________________
                35234568      10648          - java.util.HashMap
                 5678900      10668          - [Ljava.util.HashMap$Entry;
                 4456788       7436          - java.util.HashMap$Entry
                 4321000        200          - java.util.WeakHashMap
                 1234568        190          - [Ljava.util.WeakHashMap$Entry;
                  454400          4          - java.util.WeakHashMap$Entry
                51380224      29146 total
```

- eheapprof コマンドを実行していない場合

```
Explicit Heap Status
--------------------
 max 31415926K, total 213971K, used 205369K, garbage 1234K (1.1% used/max, 96.2% used/
total, 0.0% garbage/used), 3 spaces exist

 Explicit Memories(0x12345678)

  "EJBMgrData" eid=1(0x02f25610)/R, total 154272K, used 150176K, garbage 1234K (97.0%
used/total, 1.2% garbage/used, 0 blocks) Enable

  "VJBStored" eid=3(0x02f25910)/B, total 54272K, used 50176K, garbage 0K (90.9% used/t
otal, 0.0% garbage/used, 2 blocks) Enable

  "ExplicitMemory-2" eid=2(0x02f25700)/R, total 5427K, used 5017K, garbage 0K (92.1% u
sed/total, 0.0% garbage/used, 0 blocks) Enable
```

5.6　JavaVM の GC ログ

GC のログは，JavaVM ログファイルに出力されます。

詳細については，「5.7　JavaVM ログ（JavaVM ログファイル）」を参照してください。

5.7　JavaVM ログ（JavaVM ログファイル）

JavaVM ログファイルは，製品が標準の JavaVM に追加した拡張オプションを使用して出力されます。このログを取得するためには，対象の J2EE サーバを起動するときに必要なオプションを指定しておく必要があります。

5.7.1　JavaVM ログファイルを出力するオプション

JavaVM ログファイルを出力するオプションを次に示します。

- -XX:+HitachiOutOfMemoryStackTrace

  OutOfMemoryError 発生時にスタックトレースを出力するオプションです。なお，このオプションを指定した場合に同時に指定される，-XX:+HitachiOutOfMemorySize および-XX:+HitachiOutOfMemoryCause が指定された場合も，JavaVM ログファイルが出力されます。

- -XX:+HitachiVerboseGC

  GC が発生した時の拡張 verbosegc 情報を出力するオプションです。拡張 verbosegc 情報の取得については，「5.7.2　拡張 verbosegc 情報の取得」を参照してください。

- -XX:+HitachiJavaClassLibTrace

  System.gc()，System.exit()，System.runFinalizersOnExit()，Runtime.exit()，Runtime.halt()，または Runtime.runFinalizersOnExit()のどれかの API を実行したときに，これらの API の呼び出しトレースを出力するオプションです。

  なお，-XX:HitachiJavaClassLibTraceLineSize オプションを指定している場合，出力されるトレースは，指定した文字数（バイト数）以内で出力されます。1 行の文字数が指定した値を超える場合は，「at」以降の文字列の前半部分が削除されて，指定した文字数分出力されます。

- -XX:+JITCompilerContinuation

  JIT コンパイラ稼働継続機能を有効にするオプションです。JIT コンパイルがアプリケーションを構成するメソッドの論理矛盾で失敗すると，JIT コンパイラ稼働継続機能のログが JavaVM ログファイルに出力されます。

それぞれのオプションを指定した場合の出力内容の詳細については，次の個所を参照してください。

- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]HitachiOutOfMemoryStackTrace（スタックトレース出力オプション）」
- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]HitachiVerboseGC（拡張 verbosegc 情報出力オプション）」
- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]HitachiJavaClassLibTrace（クラスライブラリのスタックトレース出力オプション）」
- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]JITCompilerContinuation（JIT コンパイラ稼働継続機能オプション）」

5.7.2　拡張 verbosegc 情報の取得

J2EE サーバの usrconf.cfg ファイルに，次の表に示すオプションを指定すると，拡張 verbosegc 情報を取得できます。拡張 verbosegc 情報からは，そのサーバで必要とする Java ヒープ領域サイズ，Metaspace 領域サイズなどを見積もるための情報が取得できます。なお，Java ログには，OutOfMemoryError 発生時のスタックトレースも出力されます。

表 5–23　拡張 verbosegc 情報の取得を指定するオプション

オプション	意味
-XX:+HitachiVerboseGC	拡張 verbosegc 情報を出力するかどうかを指定します。GC の内部領域である Eden，Survivor，Tenured，Metaspace 領域種別ごとに情報を出力します。デフォルトでは出力されません。-XX:+HitachiVerboseGC と指定すると拡張 verbosegc 情報が出力され，-XX:-HitachiVerboseGC と指定すると拡張 verbosegc 情報は出力されません。
-XX:+HitachiVerboseGCPrintDate	拡張 verbosegc 情報を出力するログの各行の先頭に，ログを出力した日付を表示するかどうかを指定します。
-XX:+HitachiVerboseGCCpuTime	GC の開始から終了までの間で，GC の実行スレッドのユーザモードおよびカーネルモードに費やされた時間だけを表示するか，GC の開始から終了までの実時間を表示するかを指定します。
-XX:HitachiVerboseGCIntervalTime=<時間間隔>	-XX:+HitachiVerboseGC に対する出力時間の間隔を数値（単位：秒）で指定します。時間間隔のデフォルト値は 0（GC 発生のたびに出力）です。なお，時間間隔を指定すると，その時間間隔の間に発生した GC 回数も表示されます。
-XX:+HitachiVerboseGCPrintCause	拡張 verbosegc 情報を出力するログに，GC が発生した原因を表示するかどうかを指定します。
-XX:+HitachiCommaVerboseGC	拡張 verbosegc 情報を出力するログを CSV 形式で出力するかどうかを指定します。CSV 形式で出力する場合，拡張 verbosegc 情報の括弧「( )」「[ ]」「< >」や，区切り「:」はすべて省略され，数値または文字列が「,」で区切られて出力されます。
-XX:+HitachiVerboseGCPrintTenuringDistribution	Survivor 領域の年齢分布情報を出力するかどうかを指定します。デフォルトでは出力されません。出力形式や出力情報については，「9.11　Survivor 領域の年齢分布情報出力機能」を参照してください。
-XX:+HitachiVerboseGCPrintJVMInternalMemory	JavaVM 内部で管理しているヒープ情報を JavaVM ログファイルに出力するかどうかを指定します。
-XX:+HitachiVerboseGCPrintThreadCount	Java スレッドの数を監視するために，Java スレッドの数を JavaVM ログファイルに出力するかどうかを指定します。
-XX:+HitachiVerboseGCPrintDeleteOnExit	java.io.File.deleteOnExit()を呼び出したことによって JavaVM が確保した累積のヒープサイズとメソッドの呼び出し回数を，JavaVM ログファイルに出力するかどうかを指定します。
-XX:+PrintCodeCacheInfo	コードキャッシュ領域の使用量を出力するかどうか，また，使用量がしきい値に達したことを知らせるメッセージを出力するかどうかを指定します。

ログファイルの出力形式と出力例を次に示します。

出力形式

```
[id] <date> (Skip Full:full_count, Copy:copy_count) [gc_kind gc_info, gc_time secs][Eden:
 eden_info][Survivor: survivor_info][Tenured: tenured_info][Metaspace: metaspace_info][cl
ass space: class_space_info]  [cause:cause_info] [User: user_cpu secs] [Sys: system_cpu s
ecs][IM: jvm_alloc_size, mmap_total_size, malloc_total_size][TC: thread_count][DOE: doe_a
lloc_size, called_count][CCI: cc_used_sizeK, cc_max_sizeK, cc_infoK]
```

説明

- id：JavaVM ログファイル識別子

  JavaVM ログファイル識別子を次の表に示します。この識別子は，ログをログの内容（機能）ごとにフィルタリングして調査する場合に利用できます。

表 5-24　JavaVM ログファイル識別子

識別子	ログの内容
CCI	コードキャッシュ領域情報
CLT	クラスライブラリのスタックトレース
JCC	JIT コンパイル失敗情報
JMS	JIT コンパイルを抑制した JIT コンパイラスレッドに関する情報
OMH	OutOfMemory の発生頻度に関する情報
OOM	OutOfMemoryError 発生時の例外情報とスタックトレース
PTD	Survivor 領域の年齢分布情報
VGC	拡張 verbosegc 情報

- date：日時
- full_count：FullGC をスキップした回数（-XX:HitachiVerboseGCIntervalTime を指定した場合だけ）
- copy_count ：CopyGC をスキップした回数（-XX:HitachiVerboseGCIntervalTime を指定した場合だけ）
- gc_kind：GC 種別（Full GC または GC）
- gc_info：GC 情報（GC 前の領域長 -> GC 後の領域長(領域サイズ)）

  (例)　264K->0K(512K)
- gc_time：GC 経過時間（単位：秒）
- eden_info：Eden 情報
- survivor_info：Survivor 情報
- tenured_info：Tenured 情報
- metaspace_info：Metaspace 領域情報
- class_space_info：CompressedClassSpace 情報

- cause_info：GC の原因
- user_cpu secs：GC スレッドがユーザモードに費やした CPU 時間（単位：秒）
- system_cpu secs：GC スレッドがカーネルモードに費やした CPU 時間（単位：秒）
- jvm_alloc_size：JavaVM 内部で管理している領域のうち，現在使用中の領域のサイズ（mmap_total_size と malloc_total_size の合計サイズのうち，現在使用中の領域のサイズ）（-XX:+HitachiVerboseGCPrintJVMInternalMemory を指定した場合だけ）
- mmap_total_size：JavaVM 内部で管理している領域のうち，mmap（Windows の場合は VirtualAlloc）で割り当てた C ヒープの総サイズ（-XX:+HitachiVerboseGCPrintJVMInternalMemory を指定した場合だけ）
- malloc_total_size：JavaVM 内部で管理している領域のうち，malloc で割り当てた C ヒープの総サイズ（-XX:+HitachiVerboseGCPrintJVMInternalMemory を指定した場合だけ）
- thread_count：Java スレッドの数（-XX:+HitachiVerboseGCPrintThreadCount を指定した場合だけ）
- doe_alloc_size：java.io.File.deleteOnExit()を呼び出して確保した累積のヒープサイズ（-XX:+HitachiVerboseGCPrintDeleteOnExit を指定した場合だけ）
- called_count：java.io.File.deleteOnExit()の呼び出し回数（-XX:+HitachiVerboseGCPrintDeleteOnExit を指定した場合だけ）
- cc_used_size：GC 時のコードキャッシュ領域の使用サイズ（単位：キロバイト）（-XX:+PrintCodeCacheInfo を指定した場合だけ）
- cc_max_size：コードキャッシュ領域の最大サイズ（単位：キロバイト）（-XX:+PrintCodeCacheInfo を指定した場合だけ）
- cc_info：保守情報（-XX:+PrintCodeCacheInfo を指定した場合だけ）

出力例

-XX:+HitachiCommaVerboseGC オプションを指定した場合の出力例を次に示します。

```
VGC,Fri Jan 23 21:37:50 2004,11,41,0,GC,16886,16886,65088,0.0559806,
4094,0,4096,447,447,448,12345,16439,60544,1116,1116,4096,0,0.0312500, 0.0156250,729,928,0
,509,2167,2054,2301,49152,2304
VGC,Fri Jan 23 21:37:55 2004,6,24,0,Full GC,65082,65082,65088,0.4294532,
4094,4094,4096,447,447,448,60541,60541,60544,1116,1116,4096,0,0.0156250, 0.0312500,729,92
8,0,509,16,170,2301,49152,2304
```

5.7.3 コードキャッシュ領域に関するログの内容

JavaVM は，呼び出し回数やループ回数が多い Java メソッドを JIT コンパイルして実行することで，処理の高速化を行います。JIT コンパイルによって生成された JIT コンパイルコードは，コードキャッシュ領域に配置されます。

通常，コードキャッシュ領域のサイズはデフォルト値で問題ありません。しかし，実行環境やJavaアプリケーションの規模によっては，コードキャッシュ領域のサイズがデフォルト値では枯渇する場合があります。

コードキャッシュ領域が枯渇すると，JavaVMはJITコンパイルを実行できなくなり，Javaアプリケーションの実行で十分な性能が得られないおそれがあります。この場合は，「-XX:[+|-]PrintCodeCacheInfo(コードキャッシュ領域情報出力オプション)」や「-XX:[+|-]PrintCodeCacheFullMessage（コードキャッシュ領域枯渇メッセージ出力オプション)」を有効にして，コードキャッシュ領域の使用量や出力されるメッセージを監視するようにしてください。

これらのオプションについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]PrintCodeCacheInfo（コードキャッシュ領域情報出力オプション)」および「-XX:[+|-]PrintCodeCacheFullMessage（コードキャッシュ領域枯渇メッセージ出力オプション)」を参照してください。

(1) コードキャッシュ領域の使用量がしきい値に達したことを知らせるメッセージの出力内容

コードキャッシュ領域の使用量がしきい値に達したことを知らせるメッセージの出力形式を次に示します。

```
[cc_id]<cc_date>CodeCache usage has exceeded the threshold.[cc_used_sizeK, cc_max_sizeK, cc_infoK]
```

出力項目について次に示します。

出力項目	説明
cc_id	CCI（JavaVMログファイル識別子）が出力されます。
cc_date	JITコンパイルを実行した日時が出力されます。
cc_used_size	JITコンパイル後のコードキャッシュ領域の使用サイズが出力されます（単位：キロバイト)。
cc_max_size	コードキャッシュ領域の最大サイズが出力されます（単位：キロバイト)。
cc_info	保守情報が出力されます。

(2) コードキャッシュ領域が枯渇したことを知らせるメッセージの出力内容

コードキャッシュ領域が枯渇したことを知らせるメッセージの出力形式を次に示します。

```
[cc_id]<cc_date>CodeCache is full. Compiler has been disabled.[cc_used_sizeK, cc_max_sizeK, cc_infoK]
```

出力項目について次に示します。

出力項目	説明
cc_id	CCI（JavaVMログファイル識別子）が出力されます。

出力項目	説明
cc_date	Java メソッドが JIT コンパイルの対象になった日時が出力されます。
cc_used_size	Java メソッドが JIT コンパイルの対象になったときのコードキャッシュ領域の使用サイズが出力されます（単位：キロバイト）。
cc_max_size	コードキャッシュ領域の最大サイズが出力されます（単位：キロバイト）。
cc_info	保守情報が出力されます。

5.8 JavaVM が出力するメッセージログ（標準出力およびエラーリポートファイル）

JavaVM でクラッシュが発生した場合，JavaVM によってデバッグ情報が標準出力とエラーリポートファイルに出力されます。

エラーリポートファイルに出力されるのは，次の場合です。

- JNI の中でのシグナルが発生したとき
- JavaVM で C ヒープ不足が発生したとき
- JavaVM で予期しないシグナルが発生したとき
- JavaVM で Internal Error（内部論理エラー）が発生したとき

ここでは，次の場合のメッセージログの出力内容について説明します。

表 5-25　JavaVM が出力するメッセージログ

メッセージの種類	出力先
JNI の中でのシグナルが発生した場合，または JavaVM でのメッセージ※	標準出力 エラーリポートファイル
C ヒープが不足した場合のメッセージ※	標準出力 エラーリポートファイル
Internal Error が発生した場合のメッセージ※	標準出力 エラーリポートファイル
スレッド作成に失敗した場合のメッセージ※	標準出力

注※　JavaVM 独自の出力先または出力内容があります。

なお，スレッド作成に失敗した場合のメッセージは，標準出力だけに出力されます。

5.8.1 シグナルが発生した場合

シグナルが発生した場合，次に示す項目がログ出力されます。出力内容には，JavaVM として拡張された内容が含まれます。

- 異常終了位置とシグナル種別※
- カレントスレッド情報
- シグナル情報の格納先アドレス※
- シグナル情報
- siginfo 情報※（UNIX の場合）

- レジスタ情報
- スタックの先頭から格納されている情報
- 命令コード情報
- スタックトレース
- スレッド情報
- VM の状態
- メモリ情報※
- Java ヒープの使用状況※
- Card table マップアドレスの表示
- Polling page アドレスの表示
- Large pages 確保失敗情報
- CodeCache 情報
- Event 情報
- ライブラリ
- コマンドおよび VM パラメタ※
- 環境変数
- 登録済みシグナルハンドラ
- マシン情報※
- システム名，CPU，実メモリ，および VM 情報
- 時間情報※
- javatrace 起動コマンドのコマンドライン※（UNIX の場合）

注※　製品で拡張された出力内容です。

それぞれの出力内容について説明します。

(1)　異常終了位置とシグナル種別

異常終了時の状態に応じて，次のどちらかの内容が出力されます。この内容は，JavaVM で拡張された出力内容です。

(a)　シグナルを検出した場合

次のメッセージが出力されます。

```
#
# A fatal error has been detected by the Java Runtime Environment:
```

次の内容が出力されます。

シグナルを検出した場合の出力内容

```
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  <発生したシグナル名> (<シグナル番号>) at pc=<PCアドレス>, pid=<プロセスID>, tid=<スレッドID>
#
# JRE version: <jreバージョン情報>
# Java VM: Java HotSpot(TM) <VM種別> (<Sunバージョン情報>-<バージョン情報>-<ビルド年月日> mixed mode<OS名>-<CPU種別>)
# Problematic frame:
# <種別コード> [<シグナルが発生したライブラリ名>+<オフセット>]
#
```

注

シグナルが発生した関数名が取り出せた場合，「<シグナルが発生したライブラリ名>＋<オフセット>」に続いて，その関数名とオフセットが表示されることがあります。

(b) 内部論理エラーが発生した場合

次の内容が出力されます。

内部論理エラーが発生した場合の出力内容

```
#
#  Internal Error (<ファイル名>:<行数> または，Internal Errorのコード), pid=<プロセスID>, tid=<スレッドID>
#  <内部論理エラー種別>: 内部論理エラーメッセージ
#
#  JRE version: <jreバージョン情報>
#  Java VM: Java HotSpot(TM) <VM種別> (<Sunバージョン情報>-<バージョン情報>-<ビルド年月日> mixed mode<OS名>-<CPU種別> <圧縮OOP>)
#
#  <coreファイル情報>
```

注

Internal Error には，ファイル名および行数の組み合わせ，または Internal Error のコードが出力されます。<内部論理エラー種別>には，発生した内部論理エラーの種類によって，"fatal error"，"guarantee(<論理式>) failed"，または"Error"のどれかが出力されます。

(2) カレントスレッド情報

スレッドの種類に応じて次の 3 種類の情報が出力されます。

```
Current thread (<アドレス>):  <スレッド名> "<スレッド名称>" [_<状態>, id=<スレッドID>, stack(<開始アドレス>,<終了アドレス>)]
```

```
または
Current thread (<アドレス>):  <スレッド名> [_id=<スレッドID>, stack(<開始アドレス>,<終了アドレス>)]

または
Current thread is native thread
```

(3) シグナル情報の格納先アドレス

次の内容が出力されます。この内容は，JavaVM で拡張された出力内容です。

```
siginfo address: <アドレス>, context address: <アドレス>
```

(4) シグナル情報

次の内容が出力されます。

Windows の場合

EXCEPTION_ACCESS_VIOLATION(読み込み違反)

```
siginfo: ExceptionCode=<シグナル番号>, reading address <アドレス>
```

EXCEPTION_ACCESS_VIOLATION(書き込み違反)

```
siginfo: ExceptionCode=<シグナル番号>, writing address <アドレス>
```

EXCEPTION_ACCESS_VIOLATION(その他)

```
siginfo: ExceptionCode=<シグナル番号>, ExceptionInformation=<付加情報>
```

EXCEPTION_ACCESS_VIOLATION 以外

```
siginfo: ExceptionCode=<シグナル番号>, ExceptionInformation=<付加情報1> <付加情報2>
 ・・・
```

UNIX の場合

```
siginfo:si_signo=<発生したシグナル番号> (<発生したシグナル名>), si_errno:_<番号>, si_code:_<番号>(<シグナル理由種別>), si_addr:_<アドレス>
```

(5) siginfo 情報（UNIX の場合）

次の内容が出力されます。この内容は，JavaVM で拡張された出力内容です。

```
siginfo structure dump (location: <siginfoのアドレス>)
<siginfoのアドレス> <siginfoのアドレス> <siginfoのアドレス> <siginfoのアドレス>
 :
<siginfoのアドレス> <siginfoのアドレス> <siginfoのアドレス> <siginfoのアドレス>
```

注

<siginfo のアドレス>は 16 進数で出力されます。

(6) レジスタ情報

次の内容が出力されます。ただし，内部論理エラーの場合は出力されません。

```
Registers:<レジスタ情報>
 :
```

注

UNIX の場合，BSP レジスタの値がデバッガ（gdb）の値と異なって出力されます。これは，デバッガでは BSP の指すバッキングストア領域の内容を出力しており，BSP の指す位置を修正しているためです。

(7) スタックの先頭から格納されている情報

次の内容が出力されます。ただし，内部論理エラーの場合は出力されません。

```
Top of Stack: (sp=<スタックポインタのアドレス>)
<アドレス>:   <格納されている内容>
 :
```

注

<格納されている内容>は 16 進数で出力されます。

(8) スタックトレース

次の内容が出力されます。ただし，Current thread が JavaThread 以外の場合は出力されません。

```
Java frames: (J=compiled Java code, j=interpreted, Vv=VM code)
 <スタックトレース>
 :
```

(9) スレッド情報

次の内容が出力されます。

```
Java Threads: ( => current thread )
  <アドレス>  JavaThread  “<スレッド名称>“  [<状態>, id=<スレッドID>, stack(<開始アドレス>,
<終了アドレス>)]
     :
=><アドレス>  JavaThread  “<スレッド名称>“  [<状態>, id=<スレッドID>, stack(<開始アドレス>,
<終了アドレス>)]
Other  Threads:
  <アドレス>  <スレッド名> [stack(<開始アドレス>,<終了アドレス>)] [id=<スレッドID>]
     :
```

(10) VM の状態

次の内容が出力されます。

```
VM state:<現在の状態>

VM Mutex/Monitor currently owned by a thread: <mutexs/moniter>
```

注

この情報に続いて，ロック情報が出力される場合があります。

(11) メモリ情報

次の内容が出力されます。この内容は，JavaVM で拡張された出力内容です。

```
Memory :
<メモリ確保関数>:address<開始アドレス> - <終了アドレス>(size:<サイズ>)
        :

        Heap Size:<確保しているメモリサイズ>
        Alloc Size:<使用中のメモリサイズ>
        Free Size:<未使用のメモリサイズ>
```

<メモリ確保関数>は，mmap()または malloc()のどちらかです。アドレスは 16 進数で表示されます。

各種メモリサイズの単位はバイトです。

(12) Java ヒープの使用状況

次の内容が出力されます。この内容は，JavaVM で拡張された出力内容です。

```
Heap※
<Javaヒープ情報>
```

注※

見出し部分は拡張スレッドダンプとエラーリポートファイルで次のように異なる。

拡張スレッドダンプの場合	Heap Status ----------
エラーリポートファイルの場合	Heap

(13) Card table マップアドレスの表示

次の内容が出力されます。ただし，ZGC を使用している場合は出力されません。

```
Card table byte_map: [<アドレス>,<アドレス>] byte_map_base: <アドレス>
```

(14) Polling page アドレスの表示

次の内容が出力されます。

```
Polling page: <アドレス>
```

(15) Large pages 確保失敗情報

mmap()関数でのメモリ確保に失敗した場合に，失敗回数が出力されます。

```
Large page allocation failures have occurred  <回数> times
```

(16) CodeCache 情報

次の内容が出力されます。

```
CodeCache: size=<全体サイズ> used=<使用済みサイズ> max_used=<最大サイズ> free=<空きサイズ>
 bounds [<bottom>, <commit addr>, <reserve addr>]
 total_blobs=<CodeBlobの総数> nmethods=<nmethodsの総数> adapters=<adapterの総数>
 compilation: <enabled/disabled>
```

(17) Event 情報

イベントバッファの内容がすべて出力されます。

```
<イベント種別名> (<イベント数> events):
<イベントレコード>
    :
```

イベント情報はリングバッファで管理していて，イベント種別ごとに保持するイベント数の最大値は 10 です。イベント数が 0 の場合は<イベントレコード>には"No events"が出力されます。

出力できる<イベント種別>の出力例を次に示します。

- Compilation events

  JIT コンパイル情報

  ```
  Compilation events (10 events):
  Event: 0.923 Thread 0x00002aaab2f01800  389    b        java.io.FileOutputStream::write (
  12 bytes)
  Event: 0.923 Thread 0x00002aaab2f01800 nmethod 389 0x00002aaaac3ea490 code [0x00002aaaac3
  ea5e0, 0x00002aaaac3ea668]
      :
  ```

- GC Heap History

  before GC，after GC 情報

```
GC Heap History (4 events):
Event: 23.719 GC heap before
{Heap before GC invocations=0 (full 0):
 def new generation   max 154880K, total 9664K, used 345K (0.2% used/max, 3.6% used/total
)
     :
```

- Deoptimization events

  Deopt 情報

```
Deoptimization events (10 events):
Event: 0.818 Thread 0x00002aaaaba7e000 Uncommon trap 24 fr.pc 0x00002aaaac3d1eec
Event: 0.818 Thread 0x00002aaaaba7e000 Uncommon trap 54 fr.pc 0x00002aaaac3d0dd8
     :
```

- Internal exceptions

  internal exception 情報

```
Internal exceptions (2 events):
Event: 0.025 Thread 0x00002aaaaba7e000 Threw 0x00000000db606140 at /hotspot/src/share/vm/
prims/jni.cpp:4008
Event: 0.061 Thread 0x00002aaaaba7e000 Threw 0x00000000db649980 at /hotspot/src/share/vm/
prims/jvm.cpp:1167
     :
```

- Events

  クラスローダ情報など

```
Events (10 events):
Event: 0.080 loading class 0x00002aaab302e990
Event: 0.080 loading class 0x00002aaab302e990 done
Event: 4.286 Executing VM operation: EnableBiasedLocking
Event: 4.286 Executing VM operation: EnableBiasedLocking done
     :
```

- Classes unloaded（JDK17 以降の場合）

  アンロードしたクラス情報

```
Classes unloaded (20 events):
Event: 18.331 Thread 0x0000027bd4519370 Unloading class 0x0000000800c00400 'Test'
Event: 18.337 Thread 0x0000027bd4519370 Unloading class 0x0000000800c00400 'Test'
Event: 18.342 Thread 0x0000027bd4519370 Unloading class 0x0000000800c00400 'Test'
     :
```

(18) ライブラリ

次の内容に続いて，ローディングされているライブラリの一覧が出力されます。

```
Dynamic libraries:
<ライブラリ>
 :
```

(19) コマンドおよびVMパラメタ

次の内容が出力されます。この内容は，JavaVMで拡張された出力内容です。

```
Command :〈コマンドライン〉

Java Home Dir   :〈JDK実行環境インストールディレクトリ〉
Java DLL Dir    :〈JDKのライブラリインストールディレクトリ〉
Sys Classpath   :〈システムクラスパス〉
User Args       :
〈コマンドオプション1〉
〈コマンドオプション2〉
 :
```

(20) 環境変数

次の内容が出力されます。

```
Environment Variables:
〈環境変数=値〉
 :
```

(21) 登録済みシグナルハンドラ

次の内容が出力されます。

```
Signal Handlers:
〈シグナル種別〉:
 [〈シグナルハンドラアドレス〉], sa_mask[0]=〈シグナルマスク〉, sa_flags=〈特殊フラグ〉
 :
Changed Signal Handlers -
〈シグナル種別〉: [〈シグナルハンドラアドレス〉], sa_mask[0]=〈シグナルマスク〉, sa_flags=〈特殊フラグ〉
 :
```

出力内容の意味は次のとおりです。

- <シグナル種別>：/usr/include/sys/signal.hに定義されているシグナル名です。
- <シグナルハンドラアドレス>：シグナルハンドラのアドレスが16進で出力された値です。「ライブラリ名＋オフセット」という形式で表示されることもあります。
- <シグナルマスク>：sigaction()で取り出せる構造のsa_maskフィールド値が16進で出力された値です。
- <特殊フラグ>：sigaction()で取り出せる構造のsa_flagsフィールド値が16進で出力された値です。

(22) マシン情報

次の内容が出力されます。この内容は，JavaVMで拡張された出力内容です。

```
Host:〈ホスト名〉:〈IPアドレス〉
```

注

〈IP アドレス〉には複数の IP アドレスが表示されることがあります。

(23) システム名，CPU，実メモリ，および VM 情報

次の内容が出力されます。

Windows の場合

```
OS:〈OSバージョン〉

CPU:〈利用可能CPU数〉,〈CPU種別〉

Memory:〈実メモリ情報〉

vm_info:〈VM情報〉
```

UNIX かつ JDK11 以前の場合

```
OS:〈OSバージョン〉

[uname:〈uname出力〉]
[libc:〈libcのバージョン番号〉]
[rlimit:〈リミット値〉]
[load average:〈ロードアベレージ〉]
[/proc/meminfo:〈/proc/meminfoの内容〉]

CPU:〈利用可能CPU数〉[,〈CPU種別〉]

Memory:〈実メモリ情報〉

vm_info:〈VM情報〉
```

UNIX かつ JDK17 以降の場合

```
OS:
〈OSバージョン〉
uname:〈uname出力〉
OS uptime:〈システム起動時からの経過時間〉
libc:〈libcのバージョン番号〉
rlimit (soft/hard):〈リミット値〉
load average:〈ロードアベレージ〉

/proc/meminfo:
〈/proc/meminfoの内容〉

/sys/kernel/mm/transparent_hugepage/enabled:〈/sys/kernel/mm/transparent_hugepage/enabledの
内容〉
/sys/kernel/mm/transparent_hugepage/defrag (defrag/compaction efforts parameter):〈/sys/kern
el/mm/transparent_hugepage/defragの内容〉
```

```
Process Memory:
<プロセスのメモリ情報>

/proc/sys/kernel/threads-max (system-wide limit on the number of threads): </proc/sys/kernel
/threads-maxの内容>
/proc/sys/vm/max_map_count (maximum number of memory map areas a process may have): </proc/s
ys/vm/max_map_countの内容>
/proc/sys/kernel/pid_max (system-wide limit on number of process identifiers): </proc/sys/ke
rnel/pid_maxの内容>

container (cgroup) information:
<コンテナ情報>

CPU:total <CPU数> <CPU概要>
CPU Model and flags from /proc/cpuinfo:
<model nameの内容>
<flagsの内容>

<CPUに関する追加情報>

Memory: <1ページのバイト数>k page, physical <物理メモリ数>k(<空きメモリ数>k free), swap <ス
ワップメモリ数>k(<空きスワップのメモリ数>k free)

vm_info: Java HotSpot(TM) <VM種別> (<Oracleバージョン情報>-<日立バージョン情報>-<ビルド年月
日>) for linux-amd64 JRE (<Oracleバージョン情報>-<日立バージョン情報>-<ビルド年月日>), built
 on <MMM dd yyyy HH:mm:ss> by "Java" with gcc 8.3.1 20190507 (Red Hat 8.3.1-4)
```

(24) 時間情報

次の内容が出力されます。

```
time: <実行日付>

elapsed time: <実行時間> seconds <(実行時間 形式出力)>
```

注

実行日付の例を次に示します。

例：「Wed Aug 25 14:55:04 2004」

秒単位だけでは実行時間が把握しづらいため，<(実行時間 形式出力)>ではユーザが読みやすいように，(DAYS HOURS MINS SECS)フォーマットで出力します。

(例) elapsed time: 900 seconds (0d 0h 15m 0s)

(25) javatrace 起動コマンドのコマンドライン（UNIX の場合）

次の内容が出力されます。この内容は，JavaVM で拡張された出力内容です。

```
# You can get further information from javatrace.log file generated
# by using javatrace command.
# usage: javatrace core-file-name loadmodule-name [out-file-name] [-l(library-name)...]
# Please use javatrace command as follows and submit a bug report
```

```
# to Hitachi with javatrace.log file:
#[<インストールディレクトリ>/bin/javatrace <coreファイル> <ロードモジュール>]
```

(26) ZGC ページ情報

ZGC を使用している場合，次の内容が出力されます。

```
ZGC Page Table:
<ページの種類>l   <開始アドレス><先頭アドレス><終了アドレス> <AllocatingまたはRelocatable
>

ZbarrierSet
```

5.8.2 C ヒープが不足した場合

C ヒープが不足した場合，次の順序でメッセージ出力およびダンプ出力，または core ダンプの生成が実行されます。

1. C ヒープ不足を示すメッセージログが，エラーリポートファイルおよび標準出力に出力されます。
2. 1.の実行中にメモリ不足が発生した場合，簡易メッセージが標準出力に出力されます。
3. UNIX の場合，簡易メッセージの出力処理中にさらにメモリ不足が発生したときに，メッセージおよびエラーログファイルの出力処理を中止して，core ダンプを生成します。

それぞれの出力形式を次に示します。

(1) C ヒープ不足を示すメッセージログの出力内容

C ヒープ不足を示すメッセージログの出力形式を次に示します。この形式は，エラーリポートファイルと標準出力で共通です。

- Windows の場合

```
Exception in thread <ThreadName> java.lang.OutOfMemoryError:requested <n> bytes [ for <me
ssage>].

Memory Status
-------------
       Memory in use   :使用率%
       Physical memory :空きメモリサイズ /総メモリサイズ free
       Virtual memory  :空きメモリサイズ/総メモリサイズ free
       Paging file     :空き容量/総容量 free

Heap Status
-----------
<Javaヒープ情報>
-----------
Stack Trace
```

```
-----------
〈スタックトレース〉
JVM Internal Memory Status
-------------------------
〈独自メモリ管理機能で管理している領域情報〉
Insufficient memory for malloc. JVM generates core file.
-----------
```

- AIX，またはLinuxの場合

```
Exception in thread 〈ThreadName〉 java.lang.OutOfMemoryError: requested 〈n〉 bytes [for 〈message〉].

Memory Status
-------------
  maximum size of data segment
      soft(current) limit :getrlimit(RLIMIT_DATA)で取得したソフトリミット値 kbytes (16進数)
      hard limit          :getrlimit(RLIMIT_DATA)で取得したハードリミット値 kbytes (16進数)
  current end of the heap :sbrk(0)によって取得した値
  JVM allocation size by malloc :JavaVMが割り当てたメモリサイズ kbytes (16進数)
  malloc information
      total space in arena            :mallinfo.arenaの値
      number of ordinary blocks       :mallinfo.ordblksの値
      number of small blocks          :mallinfo.smblksの値
      number of holding blocks        :mallinfo.hblksの値
      space in holding block headers  :mallinfo.hblkhdの値
      space in small blocks in use    :mallinfo.usmblksの値
      space in free small blocks      :mallinfo.fsmblksの値
      space in ordinary blocks in use :mallinfo.uordblksの値
      space in free ordinary blocks   :mallinfo.fordblksの値
      cost of enabling keep option    :mallinfo.keepcostの値

Heap Status
-----------
〈Javaヒープ情報〉
-----------
Stack Trace
-----------
〈スタックトレース〉
-----------

JVM Internal Memory Status
-------------------------
〈独自メモリ管理機能で管理している領域情報〉
```

このようなメッセージが出力された場合は，Cヒープを減らすなど，適切な対策をしてください。

出力項目について次に示します。

表5-26　Cヒープが不足した場合のメッセージログの出力項目

出力項目	説明
ThreadName	Thread#getName()メソッドで取り出せるスレッド名称が出力されます。

出力項目	説明
n	メモリ確保要求サイズが出力されます。
message	保守員による調査に必要な内部メッセージが出力されます。出力されない場合もあります。
Java ヒープ情報	Java ヒープの使用状況が出力されます。
スタックトレース	メモリ不足の発生したスレッドが Java コードを実行しているスレッドである場合に，スタックトレースが出力されます。 コンパイル処理のような JavaVM の内部処理を実行するスレッドでメモリ不足が発生した場合には出力されません。

(2)　メモリ不足を示すメッセージの出力内容

C ヒープ不足を示すメッセージログが出力されている間にさらにメモリ不足が発生した場合は，処理の続行ができません。この場合は，次の形式の簡易メッセージが標準出力に出力されます。

```
java.lang.OutOfMemoryError:requested <n> bytes for <message>
```

出力項目について次に示します。

表 5-27　メモリ不足が発生した場合の簡易メッセージの出力項目

出力項目	説明
n	メモリ確保要求サイズが出力されます。
message	保守員による調査に必要な内部メッセージが出力されます。

(3)　core ダンプの生成を示すメッセージの出力内容

簡易メッセージの出力処理中にさらにメモリ不足が発生した場合，メッセージおよびエラーログファイルの出力処理を中止して，core ダンプを生成します。core ダンプが生成されると，次の形式のメッセージが標準出力に出力されます。

```
Can't create logs because of memory shortage.
Insufficient memory for malloc. JVM generates core file
```

5.8.3　Internal Error が発生した場合

JavaVM 内部の論理エラーである Internal Error が発生した場合，次の情報が出力されます。

- 異常終了位置とシグナル種別
- カレントスレッド情報
- シグナル情報の格納先アドレス※

- スレッド情報
- VM の状態
- メモリ情報※
- ヒープ情報※
- Card table マップアドレス
- Polling page アドレス
- Large pages 確保失敗情報
- CodeCache 情報
- Event 情報
- ライブラリ
- コマンド・VM パラメタ※
- 環境変数
- 登録済みシグナルハンドラ
- マシン情報※
- システム名，CPU，実メモリ，および VM 情報
- 時間情報
- javatrace 起動コマンドのコマンドライン※（UNIX の場合）

注※　製品で拡張された出力内容です。

それぞれの情報の出力形式については，「5.8.1　シグナルが発生した場合」を参照してください。

5.8.4　スレッド作成に失敗した場合

メモリ不足（OutOfMemoryError）などが発生して新しくスレッドを作成できなかった場合，その時点でのスレッド数が，標準出力に出力されます。ここには，作成に失敗したスレッド数も含まれます。

メモリ不足などによってスレッドの作成に失敗した場合の出力例を次に示します。

```
java.lang.OutOfMemoryError:unable to create new native thread.1200 threads exist.
…
```

JavaVM 起動時にスレッド作成に失敗した場合の出力例を次に示します。

```
Error occurred during initialization of VM
Could not create thread for VM:VM Thread.5 threads exist.
```

5.9　OS の状態情報および OS のログ

取得した OS の状態・ログ情報で異常な傾向の有無を確認します。具体的な資料の見方については OS 付属のマニュアルなどを参照してください。

5.10　JavaVM スタックトレース情報

ここでは，スタックトレースに出力される情報のうち，JavaVM で拡張された内容について説明します。

サーバやアプリケーションでトラブルが発生した場合，トラブル発生までのスタックトレースの内容を確認することで，トラブル発生の要因を調査できます。

スタックトレースは，次のどれかのタイミングで出力されます。

- J2EE サーバまたは J2EE アプリケーションで例外が発生した場合
- バッチサーバまたはバッチアプリケーションで例外が発生した場合
- JavaVM のスレッドダンプが出力された場合

JavaVM では，サーバを起動するときのオプションの指定によって，スタックトレースに Java メソッドのローカル変数についての情報を出力できます。例外発生時に Java メソッドに定義されていたローカル変数の情報は，トラブルの要因を分析するために有効です。

なお，ここで対象にする**ローカル変数**とは，メソッドに渡される引数およびインスタンスメソッドでの呼び出しの対象になるオブジェクト（this）のことです。**ローカル変数情報**には，これらのローカル変数の変数名，型名，およびローカル変数の値が出力されます。なお，型名は，基本形名，クラス名（インタフェース名を含みます）または配列形名のことです。

スタックトレースにローカル変数情報を出力するためのオプションを，次の表に示します。JavaVM の資料取得の設定については，「3.3.17　JavaVM の資料取得の設定」を参照してください。

表 5-28　スタックトレースにローカル変数情報を出力するためのオプション

起動オプション	スタックトレースを出力するタイミング	同時に指定できるオプション
-XX:+HitachiLocalsInThrowable	サーバまたはアプリケーション内で例外が発生したとき※	• -XX:+HitachiLocalsSimpleFormat • -XX:+HitachiTrueTypeInLocals • -XX:HitachiCallToString=<適用範囲指定>
-XX:+HitachiLocalsInStackTrace	JavaVM のスレッドダンプを出力したとき	• -XX:+HitachiLocalsSimpleFormat • -XX:+HitachiTrueTypeInLocals

注※　ただし，発生した例外が「java.lang.StackOverflowError」または「java.lang.OutOfMemoryError」の場合，スタックトレースにローカル変数は出力されません。

次に，それぞれのオプションを指定した場合に出力される内容について，例を基に説明します。それぞれのオプションを指定した時に出力される項目の詳細については，次の個所を参照してください。

- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]HitachiLocalsInThrowable（例外発生時のローカル変数情報収集オプション）」

- マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:[+|-]HitachiLocalsInStackTrace（スレッドダンプ出力時のローカル変数出力オプション）」

5.10.1 -XX:+HitachiLocalsInThrowable オプションが指定されている場合

java.lang.Throwable.printStackTrace メソッドで出力されるスタックトレース情報の 1 スタックフレーム情報ごとに，そのスタックフレームに対応するメソッド内のローカル変数情報が挿入されて出力されます。

(1) 標準の形式および簡易出力フォーマットでの出力例

ローカル変数を出力するための機能として，-XX:+HitachiLocalsInThrowable オプションだけを指定している場合の出力例です。

Java プログラムの例と，それに対するスタックトレース内のローカル変数情報の出力例を示します。

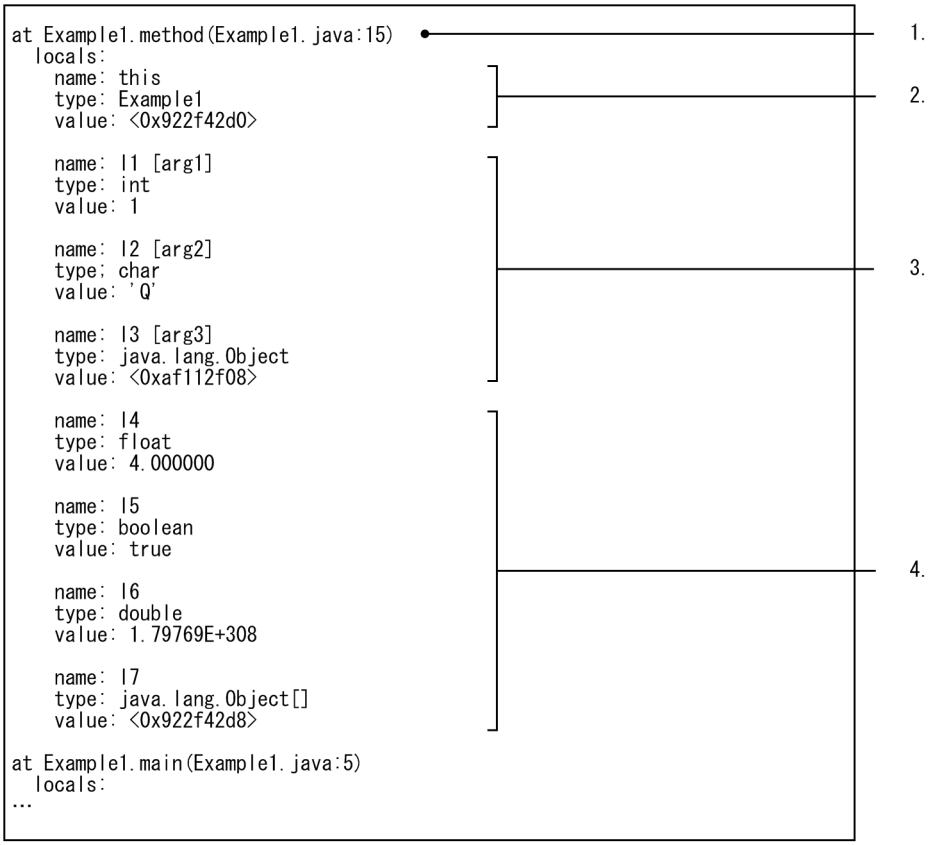
Java プログラムの例 1

```
class Example1 {
    public static void main(String[] args) {
        Example1 e1 = new Example1();
        Object obj = new Object();
        e1.method(1, 'Q', obj); // e1.methodメソッドを実行します（5行目）。
    }

    void method(int l1, char l2, Object l3) {
        float l4 = 4.0f;
        boolean l5 = true;
        double l6 = Double.MAX_VALUE;
        Object[] l7 = new Object[10];

        try {
          <例外発生!> // methodメソッドの処理内で例外が発生した場合の処理です（15行目）。
        } catch (Exception e) {
          e.printStackTrace(); // スタックトレース情報を出力します（17行目）。
        }
    }
}
```

出力例を次に示します。

この例は，Java プログラムの例 1 の 5 行目で実行した e1.method メソッドの処理内で例外が発生したときに，17 行目の e.printStackTrace メソッドによって出力されるスタックトレース情報です。

図 5-2　Java プログラムの例 1 に対するローカル変数情報の出力例（-g オプションまたは-g:vars オプションを指定して作成された class ファイルの場合）



出力内容について説明します。

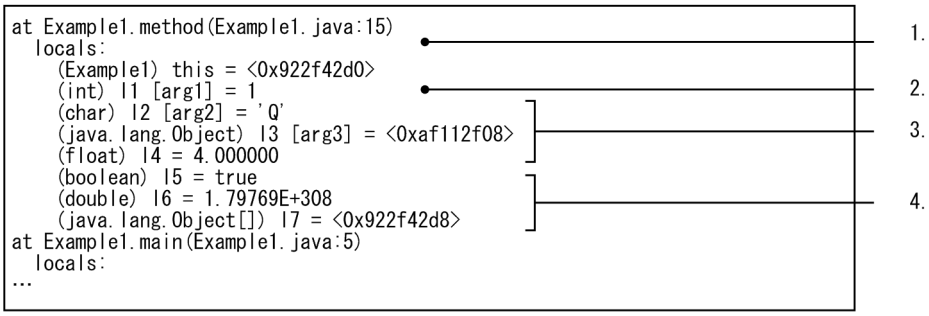
1. スタックトレースを出力する処理を実行したメソッドの情報が出力されます。この例では，java プログラム 1 の 15 行目で例外が発生したときのスタックトレース情報が出力されることを示しています。
2. ローカル変数情報として，インスタンスメソッドの呼び出しの対象になるオブジェクトについての情報が出力されます。この例の場合は，Java プログラムの例 1 の 3 行目で作成した Example1 クラスのオブジェクトのクラス名とアドレスが出力されます。
3. ローカル変数情報として，method メソッドの引数として指定されたローカル変数の値についての情報が出力されます。変数名の後ろの[arg*]は，何番目のメソッド引数かを示す情報です。Java プログラムの例 1 の 5 行目で e1.method メソッドを実行したときに指定された値が出力されます。

   なお，l1 および l2 は基本型（int 型および char 型）の変数なので実際の値が出力されます。l3 は java.lang.Object クラス型の変数なので，アドレスで出力されます。
4. ローカル変数情報として，method メソッド内のローカル変数のうち，メソッド引数として指定した以外のローカル変数の値についての情報が出力されます。

   なお，l4～l6 は基本型（float 型，boolean 型および double 型）の変数なので実際の値が出力されます。l7 は java.lang.Object クラス型の変数なので，アドレスで出力されます。

次に，-XX:+HitachiLocalsSimpleFormat オプションを指定した場合の，簡易出力フォーマットでの出力例を次に示します。なお，出力内容の説明は，標準の形式と同じです。

図 5-3　-XX:+HitachiLocalsSimpleFormat オプションを指定した場合の出力例



また，javac コマンド実行時に，-g オプションまたは-g:vars オプションを指定しなかった場合，ローカル変数情報がないため，出力内容が次のように制限されます。これらは，native メソッドを実行した場合も同様です。

- 出力できるローカル変数が，メソッドに渡される引数とインスタンスメソッド呼び出し対象オブジェクト（this）だけになります。
- メソッドに渡される引数は，変数名が出力されません。引数番号だけが出力されます。
- native メソッドの場合，ローカル変数の値として native メソッドを呼び出した時点での値が出力されます。native メソッドの実行結果を反映した出力結果にはなりません。

ローカル変数情報がない場合の，Java プログラムの例 1 に対する出力例を次に示します。ここでは，簡易出力フォーマットでの出力例を示します。

図 5-4　ローカル変数情報がない場合の出力例（簡易出力フォーマット）

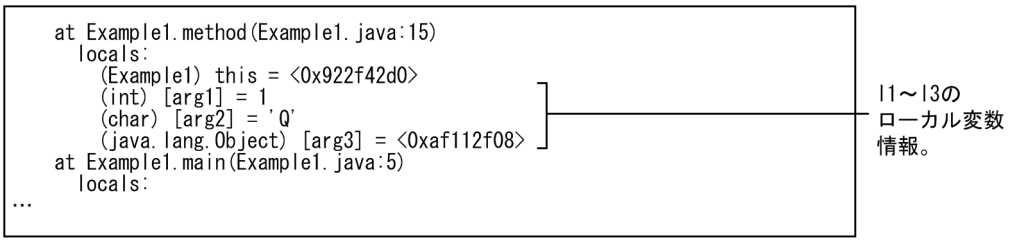


図 5-3 に比べて，次の違いがあります。

- メソッドに渡される引数（l1～l3）の変数名が出力されません。
- メソッド呼び出し時点の値が出力されるため，メソッド内で設定されたローカル変数についての情報（l4～l7）が出力されません。

(2)　クラスまたは配列型の変数を文字列として出力する場合の出力例

出力するローカル変数がクラスまたは配列型の場合，アドレスだけの値表現ではトラブルシューティングに必要な情報が取得できない場合があります。このとき，-XX:HitachiCallToString オプションの指定をしておくと，クラスまたは配列型の変数の値を文字列で取得できます。オプションには，適用範囲として，minimal または full が指定できます。

-XX:HitachiCallToString=minimal オプションが指定されている場合は，java.lang パッケージ内のクラスのうち，String，StringBuffer，Boolean，Byte，Character，Short，Integer，Long，Float，または Double が対象になります。-XX:HitachiCallToString=full オプションが指定されている場合は，すべてのクラスが対象になります。

次に，Java プログラムの例と，-XX:HitachiCallToString オプションが指定されている場合の出力例を示します。なお，ここでは簡易出力フォーマットで示します。

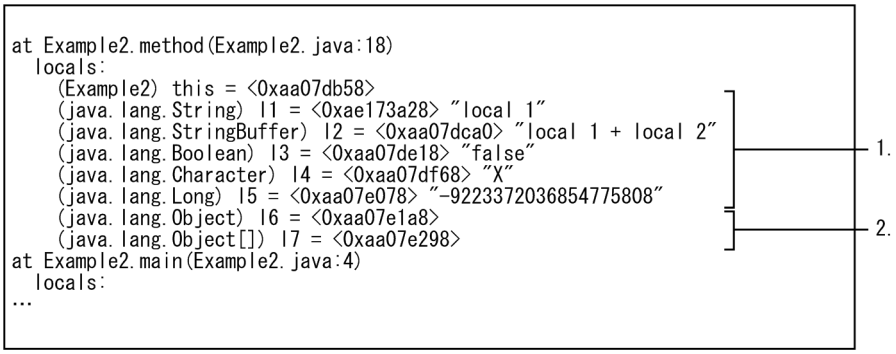
Java プログラムの例 2

```
class Example2 {
    public static void main(String[] args) {
        Example2 e2 = new Example2();
        e2.method();// e2.methodメソッドを実行します（4行目）。
    }

    void method() {
        String l1 = "local 1";
        StringBuffer l2 = new StringBuffer(l1);
        l2.append(" + local 2");
        Boolean l3 = new Boolean(false);
        Character l4 = new Character('X');
        Long l5 = new Long(Long.MIN_VALUE);
        Object l6 = new Thread();
        Object[] l7 = new Thread[10];

        try {
          <例外発生!> // methodメソッドの処理内で例外が発生した場合の処理です（18行目）。
        } catch (Exception e) {
          e.printStackTrace(); // スタックトレース情報を出力します（20行目）。
        }
    }

    public String toString() {
        return "I am an Example2 instance.";
    }
}
```

-XX:HitachiCallToString=minimal オプションを指定した場合の出力例を次に示します。

この例は，Java プログラムの例 1 の 4 行目で実行した e2.method メソッドの処理内で例外が発生したときに，20 行目の e.printStackTrace メソッドによって出力されるスタックトレース情報です。

図 5-5　-XX:HitachiCallToString=minimal オプションを指定した場合の出力例（簡易出力フォーマット）



出力内容について説明します。

1. クラス型のローカル変数のうち，文字列を出力する適用対象のクラスの型のローカル変数情報です。アドレスに続いて，文字列に変換した値が出力されます。
2. クラス型のローカル変数のうち，文字列を出力する適用対象外のクラスの型のローカル変数情報です。アドレスだけが出力されます。

次に，-XX:HitachiCallToString=full オプションを指定した場合の出力例を次に示します。

図 5-6　-XX:HitachiCallToString=full オプションを指定した場合の出力例（簡易出力フォーマット）

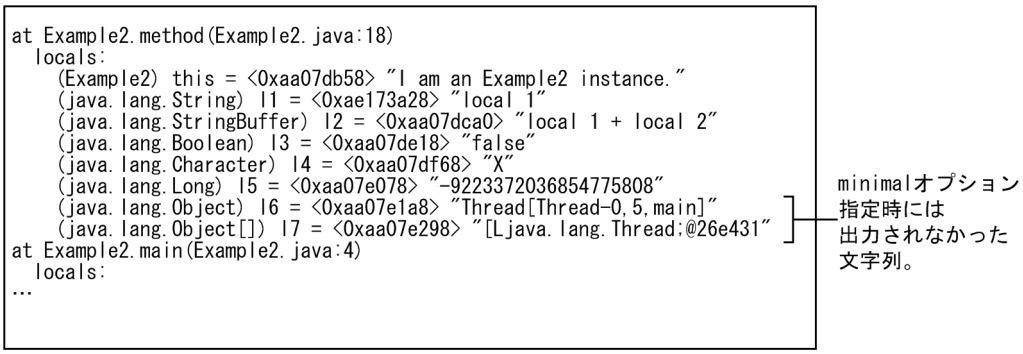


図 5-5 に比べて，次の違いがあります。

- minimal を指定した場合に適用対象外のクラスだった java.lang.Object クラスおよび java.lang.Object クラスの配列のローカル変数（l6 および l7）に対しても，文字列が出力されます。

ただし，次の場合は，文字列は出力されないで，アドレスだけが出力されます。

- ローカル変数の値が null だった場合
- 文字列で出力するときに再度例外が発生して，正常に値が得られなかった場合

注意事項

ローカル変数情報で出力するオブジェクトは，ほかのスレッドで並行して操作していることがあります。このため，このオプションを指定して出力した文字列は，実際に例外が発生したときのオブジェクトに対応する情報とは異なっているおそれがあります。

(3) クラスまたは配列型の変数の実際の型名を出力する場合の出力例

出力するローカル変数がクラスまたは配列型の場合，変数の型名と実際に代入されている値の型名が異なる場合があります。例えば，クラスの継承関係やインタフェースの実装関係によっては，異なる型名の値が変数に代入されます。

この場合に，-XX:+HitachiTrueTypeInLocals オプションの指定によって，クラスまたは配列型の変数に実際に代入されている値の型名を追記して取得できます。

取得した文字列は，値表現の末尾に半角スペース 1 文字分が追記され，( )で囲まれて出力されます。このとき，出力文字列長に制限はありません。

次に，-XX:+HitachiTrueTypeInLocals オプションが指定されている場合の出力例を示します。ここでは簡易出力フォーマットで示します。実行するプログラムは，Java プログラムの例 2 です。また，この例は，-XX:HitachiCallToString=minimal オプションが指定されている場合の例です。

図 5-7 -XX:+HitachiTrueTypeInLocals オプションを指定した場合の出力例（簡易出力フォーマット）

```
at Example2.method(Example2.java:18)
  locals:
   (Example2) this = <0xaa07db58> (Example2)
   (java.lang.String) l1 = <0xae173a28> "local 1" (java.lang.String)
   (java.lang.StringBuffer) l2 = <0xaa07dca0> "local 1 + local 2" (java.lang.StringBuffer)
   (java.lang.Boolean) l3 = <0xaa07de18> "false" (java.lang.Boolean)
   (java.lang.Character) l4 = <0xaa07df68> "X" (java.lang.Character)
   (java.lang.Long) l5 = <0xaa07e078> "-9223372036854775808" (java.lang.Long)
   (java.lang.Object) l6 = <0xaa07e1a8> (java.lang.Thread)                    ┐
   (java.lang.Object[]) l7 = <0xaa07e298> (java.lang.Thread[])                ┘ 1.
at Example2.main(Example2.java:4)
  locals:
...
```

出力内容について説明します。

1. 変数の型名は java.lang.Object クラスまたは java.lang.Object クラスの配列ですが，実際に代入されている値の型名は java.lang.Thread クラスおよび java.lang.Thread クラスの配列であることが出力されています。

なお，-XX:+HitachiTrueTypeInLocals オプションは，-XX:HitachiCallToString=full オプションとも同時に指定できます。

5.10.2　-XX:+HitachiLocalsInStackTrace オプションが指定されている場合

スレッドダンプ内のスタックトレース情報の 1 スタックフレーム情報ごとに，そのスタックフレームに対応するメソッド内のローカル変数情報が挿入されて，出力されます。

出力形式および出力内容は，-XX:+HitachiLocalsInThrowable が指定されている場合に標準出力に出力される内容と同じです。

ただし，-XX:+HitachiLocalsInStackTrace オプションでは，次のオプションの指定は無効になります。

- -XX:HitachiCallToString オプション

Java プログラムの例と，それに対するスタックトレース内のローカル変数情報の出力例を示します。

Java プログラムの例 3

```
class Example3 {
    public static void main(String[] args) {
        Example3 e3 = new Example3();
        e3.method();
    }

    synchronized void method() {
        int l1 = 1;
        float l2 = 2.0f;
        String l3 = "local 3";
        Character l4 = new Character('X');
        Object l5 = new Thread();
        Object[] l6 = new Thread[10];

        <ここでスレッドダンプ出力!>
    }
}
```

出力例を次に示します。この例は，次の場合の例です。

- -g オプションまたは-g:vars オプションを指定して作成された class ファイルである
- -XX:+HitachiLocalSimpleFormat オプションを指定している
- -XX:+HitachiTrueTypeInLocals オプションを指定している

図 5-8　Java プログラムの例 3 に対するローカル変数情報の出力例

```
...
"main" prio=1 tid=0xb6e88d20 nid=0xb7492080 runnable [bfffb000..bfffb474]
        at Example3.method(Example3.java:15)
        - locked <0xab040550> (a Example3)
          locals:
            (Example3) this = <0xab040550> (Example3)
            (int) l1 = 1
            (float) l2 = 2.000000
            (java.lang.String) l3 = <0xaf112cc0> (java.lang.String)
            (java.lang.Character) l4 = <0xab040698> (java.lang.Character)
            (java.lang.Object) l5 = <0xab0407c8> (java.lang.Thread)
            (java.lang.Object[]) l6 = <0xab0408b8> (java.lang.Thread[])
        at Example3.main(Example3.java:4)
          locals:
            (java.lang.String[]) args [arg1] = <0xab040540> (java.lang.String[])
            (Example3) e3 = <0xab040550> (Example3)
...
```

5.11　明示管理ヒープ機能のイベントログ

この節では，明示管理ヒープ機能のイベントログの内容について説明します。

明示管理ヒープ機能のイベントログは，GC 発生，Explicit メモリブロックの初期化・解放・拡張，Explicit メモリブロックへのオブジェクトの移動など，明示管理ヒープ機能に関連するイベントが発生したタイミングで出力されます。出力される内容は，JavaVM 起動オプションで指定したログ出力レベルの設定によって異なります。

明示管理ヒープ機能のイベントログは，次の用途で使用できます。

- Explicit ヒープのチューニング
- 明示管理ヒープ機能を実装したアプリケーションのデバッグ
- 稼働情報の収集

イベントログを参照すると，Explicit ヒープのサイズの変化，Explicit メモリブロックの数，Java ヒープの各領域サイズの変化などがわかります。なお，イベントログは，テキストファイルとして出力されます。

5.11.1　明示管理ヒープ機能のイベントログの出力契機

ここでは，明示管理ヒープ機能のイベントログの出力契機について説明します。設定しているログ出力レベルに応じて，契機となるイベントは異なります。

ログ出力レベルには，normal，verbose，debug の 3 種類があります。なお，verbose を指定した場合，normal で出力される内容に加えて，verbose に対応する内容が出力されます。debug を指定した場合，verbose で出力される内容に加えて，debug に対応する内容が出力されます。

ログ出力レベルと出力契機となるイベントの対応を次の表に示します。イベントに対応して出力される接頭語についてもあわせて示します。各イベントで出力される内容については，参照先を参照してください。

表 5-29　ログ出力レベルと出力契機となるイベントの対応

ログ出力レベル	出力契機となるイベント	接頭語※	参照先
normal	GC 発生（Explicit ヒープの利用状況出力）	ENS	5.11.3(1)
	Explicit メモリブロックの明示解放処理	ENS	5.11.3(2)
	Explicit メモリブロックの明示解放処理時の Java ヒープあふれ	ENS	5.11.3(3)
	Explicit メモリブロックの自動解放処理	ENS	5.11.3(4)
	Explicit メモリブロックの自動解放処理時の Java ヒープあふれ	ENS	5.11.3(5)
	明示管理ヒープ自動配置設定ファイルオープンエラー	ENA	5.11.3(6)
	明示管理ヒープ自動配置設定ファイルパースエラー	ENA	5.11.3(7)

ログ出力レベル	出力契機となるイベント	接頭語※	参照先
	明示管理ヒープ自動配置エラー	ENA	5.11.3(8)
	明示管理ヒープ機能適用除外クラス指定機能の設定ファイルオープンエラー	ENO	5.11.3(9)
	明示管理ヒープ機能適用除外クラス指定機能の設定ファイルパースエラー	ENO	5.11.3(10)
verbose	Explicit メモリブロックの初期化	EVO	5.11.4(1)
	Explicit メモリブロックの初期化失敗	EVO	5.11.4(2)
	Explicit メモリブロックのサブ状態の Disable 化	EVO	5.11.4(3)
	Explicit メモリブロックへのオブジェクト生成	EVS	5.11.4(4)
	Explicit メモリブロックへの移動（詳細情報出力）	EVS	5.11.4(5)
	Explicit メモリブロックの明示解放処理（詳細情報出力）	EVS	5.11.4(6)
	ファイナライザによる Explicit メモリブロックの解放予約	EVO	5.11.4(7)
	明示管理ヒープへの自動配置	EVA	5.11.4(8)
debug	Explicit メモリブロックの明示解放による Java ヒープへのオブジェクトの移動	EDO	5.11.5(1)
	Explicit メモリブロックの初期化（詳細情報出力）	EDO	5.11.5(2)
	Explicit メモリブロックの自動解放処理の詳細（詳細情報出力）	EDO	5.11.5(3)

注※

接頭語は，ログ行の最初に出力する「[」と「]」で囲まれたアルファベット（3 文字）です。

各イベントの詳細は，マニュアル「アプリケーションサーバ 機能解説 拡張編」を参照してください。

5.11.2　明示管理ヒープ機能のイベントログの確認方法

ここでは，明示管理ヒープ機能が出力するイベントログの参照方法について説明します。

(1)　出力契機の確認方法

ログ出力の契機となったイベントは，[cause:<CAUSE>]の形式で出力されます。この項目を確認することで，契機となったイベントを確認できます。

<CAUSE>に出力される文字列の意味を次の表に示します。

表 5-30　<CAUSE>に出力される文字列の意味

<CAUSE>	意味
GC	CopyGC 発生

<CAUSE>	意味
Full GC	FullGC 発生
Reclaim	Explicit メモリブロックの明示解放処理
Reclaiming	Explicit メモリブロックの明示解放処理中の Java ヒープあふれ
New	Explicit メモリブロックへのオブジェクト生成
Migrate	Explicit メモリブロックの自動解放処理
Migrating	Explicit メモリブロックの自動解放処理中の Java ヒープあふれ

ただし，一部のログでは，<CAUSE>が出力されません。

(2) ログの接頭語を使用した調査方法

[ENS]，[EVO]などのログの接頭語は，ログをフィルタリングして調査するために利用できます。

明示管理ヒープ機能の接頭語の意味は次のとおりです。

- 1 文字目の"E"は，明示管理ヒープ機能のログであることを示します。
- 2 文字目は，ログ出力レベルを示します。normal レベルの場合は「N」，verbose レベルの場合は「V」，debug レベルの場合は「D」が出力されます。
- 3 文字目は，Explicit ヒープまたは Java ヒープのメモリサイズが変化するかどうかを示します。「S」の場合はメモリサイズが変化します。「O」の場合は変化しません。「O」の場合，ログには発生したイベントの名称も出力されます。また，「A」の場合は明示管理ヒープ自動配置設定ファイルのログであることを示します。

例えば，各ヒープサイズの変化を調査したい場合に「S」でフィルタリングしたり，発生したイベントを調査したい場合に「O」でフィルタリングしたりできます。

(3) イベントログの出力形式の説明で使用する記号

出力形式の説明で使用する記号を次の表に示します。

表 5-31 出力形式の説明で使用する記号

記号	使用例	意味
*	X*	左辺を 0 回以上繰り返します。 使用例の場合，X を 0 回以上繰り返すことを意味します。
?	X?	左辺を 1 回以上繰り返します。 使用例の場合，X を 1 回以上繰り返すことを意味します。
{n,m}	X{1,5}	左辺を n 回以上 m 回以下繰り返します。 使用例の場合，X を 1 回以上 5 回以下繰り返すことを意味します。
{	{ABC}*	"{"と"}"で囲まれた範囲を"*"，"?"，"{n,m}"の左辺の参照単位にします。

記号	使用例	意味
}		使用例の場合，"ABC"を 0 回以上繰り返すことを意味します。
¥	なし	マニュアル上では見やすさのために改行した個所を示します。 実際の出力内容に改行は入りません。
|	X|Y	左辺または右辺を表します。 使用例の場合，X または Y を意味します。
.	なし	任意の文字を示します。
…	X|Y…	この行の前方直近の同一インデントの行から，この行前の行までを 1 回以上繰り返します。 使用例の場合，X または Y の行を 1 回以上繰り返すことを意味します。

また，出力形式の説明では，数値などを表 5-32 で示した記号に表 5-33 で示す記号を組み合わせて表します。

数値などを表すときに使用する記号を次の表に示します。

表 5-32　数値などを表すときに使用する記号

記号	定義	意味
<digit>	0|1|2|3|4|5|6|7|8|9	0～9 を表します。
<hex>	<digit>|a|b|c|d|e|f	16 進数を表します。
<const>	<const><digit>|<digit>	正の自然数を表します。
<decimal>	<const>.<digit>	正の整数（小数点第 1 位まで）を表します。
<ptr64>	0x<hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex><hex>	64 ビットポインタ値を表します。
<ptr32>	0x<hex><hex><hex><hex><hex><hex><hex><hex>	32 ビットポインタ値を表します。
<ptr>	<ptr64>|<ptr32>	ポインタ値を表します。
<letters>	.?	任意の文字または任意の文字列を表します。

注　この表で示す記号は，"*"，"?"または"|"の左辺または右辺の参照単位になります。例えば，「<digit>|<hex>」と表記されている場合は，0～9 の数値，または 16 進数であることを意味します。

5.11.3　出力レベルが normal の場合に出力される内容

ここでは，ログ出力レベルに normal を指定した場合に出力される内容をイベントごとに説明します。

normal は，通常運用時に指定するログ出力レベルです。

(1) GC 発生（Explicit ヒープ利用状況出力）

GC 発生時に，Explicit ヒープの利用状況が出力されます。

このログは，Java ヒープから Explicit ヒープに移動するオブジェクトがない場合にも出力されます。Explicit ヒープに移動するオブジェクトがない場合，<EH_USED_BF>と<EH_USED_AF>は同じ値になります。

(a) 出力の契機

GC の終了です。

(b) 出力形式

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>)][E/F/D: <AC_NUM>/<FL_NUM>/<DA_NUM>][cause:<CAUSE>][CF: <CF_CNT>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-33 出力項目（GC 発生（Explicit ヒープ利用状況出力））

出力項目	出力内容	意味
<ctime>	<letters>	GC の発生日時を示します。拡張 verbosegc 情報と同じ形式で出力されます。 -XX:+HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EH_USED_BF>	<const>K	GC が実行される前の Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_USED_AF>	<const>K	GC が実行されたあとの Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_TOTAL>	<const>K	GC が実行されたあとの Explicit ヒープの確保済みメモリサイズが出力されます。単位はキロバイトです。
<EH_MAX>	<const>K	Explicit ヒープ最大サイズが出力されます。単位はキロバイトです。
<AC_NUM>	<const>	GC 実行後にサブ状態が Enable である Explicit メモリブロックの数が出力されます。
<FL_NUM>	<const>	常に 0 が出力されます。
<DA_NUM>	<const>	GC 実行後にサブ状態が Disable である Explicit メモリブロックの数が出力されます。
<CAUSE>	GC|Full GC	契機となった GC の種類が出力されます。 "GC"は CopyGC，"Full GC"は FullGC を示します。

出力項目	出力内容	意味
<CF_CNT>	<const>	前回の GC が発生してから今回の GC が発生するまでの間に Explicit メモリブロックの初期化に失敗した回数が出力されます。

(d) 出力例

出力例を示します。

```
[ENS]<Thu Oct 21 14:55:50 2007>[EH: 150528K->162816K(162816K/1048576K)][E/F/D: 200/0/0][caus
e:GC][CF: 0]
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 10 月 21 日(木)14 時 55 分 50 秒に発生した CopyGC です。
- GC で Explicit ヒープへの移動が発生して，Explicit ヒープの利用済みサイズが 150,528K から 162,816K に変化しました。
- GC 発生後の Explicit ヒープの確保済みサイズは 162,816K です。Explicit ヒープの最大サイズは 1,048,576K です。
- GC 発生後，サブ状態が Enable である Explicit メモリブロックは 200 個あります。

(2) Explicit メモリブロックの明示解放処理

Explicit メモリブロックの明示解放処理終了後に，Explicit ヒープおよび Java ヒープの利用状況が出力されます。

(a) 出力の契機

Explicit メモリブロックの明示解放処理です。

Explicit メモリブロックの明示解放は，GC 直後に発生します。このため，ここで示すログは，「(1) GC 発生（Explicit ヒープ利用状況出力)」で示したログのあとに出力されます。

(b) 出力形式

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>), <ELAPSED> secs][E/F/D: <AC
_NUM>/<FL_NUM>/<DA_NUM>]¥
[DefNew::Eden: <ED_USED_BF>-><ED_USED_AF>(<ED_TOTAL>)][DefNew::Survivor: <SV_USED_BF>-><SV_U
SED_AF>(<SV_TOTAL>)]¥
[Tenured: <TN_USED_BF>-><TN_USED_AF>(<TN_TOTAL>)][User: <USERCPU> secs][Sys: <SYSCPU> secs][
cause:<CAUSE>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-34　出力項目（Explicit メモリブロックの明示解放処理）

出力項目	出力内容	意味
<ctime>	<letters>	Explicit メモリブロックの明示解放処理の発生日時を示します。拡張 verbosegc 情報と同じ形式で出力されます。 -XX:+HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EH_USED_BF>	<const>K	Explicit メモリブロックの明示解放処理前の Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_USED_AF>	<const>K	Explicit メモリブロックの明示解放処理後の Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_TOTAL>	<const>K	Explicit メモリブロックの明示解放処理後の Explicit ヒープの確保済みサイズが出力されます。単位はキロバイトです。
<EH_MAX>	<const>K	Explicit ヒープ最大サイズが出力されます。単位はキロバイトです。
<ELAPSED>	<time>	Explicit メモリブロックの明示解放処理に掛かった時間が出力されます。単位は秒です。
<AC_NUM>	<const>	Explicit メモリブロックの明示解放処理実行後にサブ状態が Enable である Explicit メモリブロックの数が出力されます。
<FL_NUM>	<const>	常に 0 が出力されます。
<DA_NUM>	<const>	Explicit メモリブロックの明示解放処理実行後にサブ状態が Disable である Explicit メモリブロックの数が出力されます。
<ED_USED_BF>	<const>K	Explicit メモリブロックの明示解放処理実行前の Eden 領域の利用済みサイズが出力されます。単位はキロバイトです。
<ED_USED_AF>	<const>K	Explicit メモリブロックの明示解放処理実行後の Eden 領域の利用済みサイズが出力されます。単位はキロバイトです。
<ED_TOTAL>	<const>K	Explicit メモリブロックの明示解放処理実行後の Eden 領域の確保済みサイズが出力されます。単位はキロバイトです。
<SV_USED_BF>	<const>K	Explicit メモリブロックの明示解放処理実行前の Survivor 領域の利用済みサイズが出力されます。単位はキロバイトです。
<SV_USED_AF>	<const>K	Explicit メモリブロックの明示解放処理実行後の Survivor 領域の利用済みサイズが出力されます。単位はキロバイトです。
<SV_TOTAL>	<const>K	Explicit メモリブロックの明示解放処理実行後の Survivor 領域の確保済みサイズが出力されます。単位はキロバイトです。
<TN_USED_BF>	<const>K	Explicit メモリブロックの明示解放処理実行前の Tenured 領域の利用済みサイズが出力されます。単位はキロバイトです。
<TN_USED_AF>	<const>K	Explicit メモリブロックの明示解放処理実行後の Tenured 領域の利用済みサイズが出力されます。単位はキロバイトです。
<TN_TOTAL>	<const>K	Explicit メモリブロックの明示解放処理実行後の Tenured 領域の確保済みサイズが出力されます。単位はキロバイトです。

出力項目	出力内容	意味
<USERCPU>	<time>	Explicit メモリブロックの明示解放処理に掛かったユーザ CPU 時間が出力されます。単位は秒です。
<SYSCPU>	<time>	Explicit メモリブロックの明示解放処理に掛かったシステム CPU 時間が出力されます。単位は秒です。
<CAUSE>	Reclaim	"Reclaim"と出力されます。Explicit メモリブロックの明示解放処理によって出力されたログであることを示します。

(d) 出力例

出力例を示します。

```
[ENS]<Tue Jul 24 01:23:51 2007>[EH: 150528K->149528K(162816K/1048576K), 0.1129602 secs][E/F/
D: 523/0/0]¥
[DefNew::Eden: 0K->0K(243600K)][DefNew::Survivor: 0K->0K(17400K)][Tenured: 103400K->103400K(
556800K)]¥
[User: 0.0900000 secs][Sys: 0.0200000 secs][cause:Reclaim]
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 7 月 24 日(火)1 時 23 分 51 秒に発生した Explicit メモリブロックの明示解放処理です。
- Explicit メモリブロックの明示解放処理によって，Explicit ヒープ利用済みサイズが 150,528K から 149,528K に減少しました。
- Explicit メモリブロックの明示解放処理後の Explicit ヒープの確保済みサイズは 162,816K です。Explicit ヒープの最大サイズは 1,048,576K です。
- Explicit メモリブロックの明示解放処理に，0.1129602 秒掛かりました。
- Explicit メモリブロックの明示解放処理後，サブ状態が Enable である Explicit メモリブロックは 523 個あります。
- Explicit メモリブロックの明示解放処理による Java ヒープの各領域の変化はありません。つまり，Java ヒープに移動したオブジェクトはありません。
- Explicit メモリブロックの明示解放処理に，ユーザ CPU 時間が 0.0900000 秒，システム CPU 時間が 0.0200000 秒掛かりました。

(3) Explicit メモリブロックの明示解放処理時の Java ヒープあふれ

Explicit メモリブロックの明示解放処理時に Java ヒープへのオブジェクトの移動が発生し，Java ヒープがあふれた場合に出力されます。あふれた時点での Explicit ヒープおよび Java ヒープの利用状況が出力されます。

(a) 出力の契機

Explicit メモリブロックの明示解放処理時に Explicit ヒープから Java ヒープへのオブジェクトの移動が発生し，Java ヒープがあふれた場合です。

(b) 出力形式

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EM_TOTAL>/<EH_MAX>), <ELAPSED> secs][E/F/D: <AC
_NUM>/<FL_NUM>/<DA_NUM>]¥
[DefNew::Eden: <ED_USED_BF>-><ED_USED_AF>(<ED_TOTAL>)][DefNew::Survivor: <SV_USED_BF>-><SV_U
SED_AF>(<SV_TOTAL>)]¥
[Tenured: <TN_USED_BF>-><TN_USED_AF>(<TN_TOTAL>)][User: <USERCPU> secs][Sys: <SYSCPU> secs][
cause:<CAUSE>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-35 出力項目（Explicit メモリブロックの明示解放処理時の Java ヒープあふれ）

出力項目	出力内容	意味
<ctime>	<letters>	Explicit メモリブロックの明示解放処理の発生日時を示します。拡張 verbosegc 情報と同じ形式で出力されます。 -XX:+HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EH_USED_BF>	<const>K	Explicit メモリブロックの明示解放処理前の Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_USED_AF>	<const>K	Java ヒープがあふれたあとの Explicit ヒープの利用済みサイズが出力されます。Java ヒープがあふれた場合は，Explicit メモリブロックの明示解放処理が実行されないため，必ず<EH_USED_BF>と同じ値になります。単位はキロバイトです。
<EH_TOTAL>	<const>K	Java ヒープがあふれたあとの Explicit ヒープの確保済みサイズが出力されます。単位はキロバイトです。
<EH_MAX>	<const>K	Explicit ヒープ最大サイズが出力されます。単位はキロバイトです。
<ELAPSED>	<time>	Explicit メモリブロックの明示解放処理開始から，Java ヒープがあふれるまでの時間が出力されます。単位は秒です。
<AC_NUM>	<const>	Java ヒープがあふれたあとでサブ状態が Enable である Explicit メモリブロックの数が出力されます。
<FL_NUM>	<const>	常に 0 が出力されます。
<DA_NUM>	<const>	Java ヒープがあふれたあとでサブ状態が Disable である Explicit メモリブロックの数が出力されます。
<ED_USED_BF>	<const>K	Explicit メモリブロックの明示解放処理実行前の Eden 領域の利用済みサイズが出力されます。単位はキロバイトです。

出力項目	出力内容	意味
<ED_USED_AF>	<const>K	Java ヒープがあふれたあとの Eden 領域の利用済みサイズが出力されます。単位はキロバイトです。
<ED_TOTAL>	<const>K	Java ヒープがあふれたあとの Eden 領域の確保済みサイズが出力されます。単位はキロバイトです。
<SV_USED_BF>	<const>K	Explicit メモリブロックの明示解放処理実行前の Survivor 領域の利用済みサイズが出力されます。単位はキロバイトです。
<SV_USED_AF>	<const>K	Java ヒープがあふれたあとの Survivor 領域の利用済みサイズが出力されます。単位はキロバイトです。
<SV_TOTAL>	<const>K	Java ヒープがあふれたあとの Survivor 領域の確保済みサイズが出力されます。単位はキロバイトです。
<TN_USED_BF>	<const>K	Explicit メモリブロックの明示解放処理実行前の Tenured 領域の利用済みサイズが出力されます。単位はキロバイトです。
<TN_USED_AF>	<const>K	Java ヒープがあふれたあとの Tenured 領域の利用済みサイズが出力されます。単位はキロバイトです。
<TN_TOTAL>	<const>K	Java ヒープがあふれたあとの Tenured 領域の確保済みサイズが出力されます。単位はキロバイトです。
<USERCPU>	<time>	Explicit メモリブロックの明示解放処理開始から，Java ヒープがあふれるまでのユーザ CPU 時間が出力されます。単位は秒です。
<SYSCPU>	<time>	Explicit メモリブロックの明示解放処理開始から，Java ヒープがあふれるまでのシステム CPU 時間が出力されます。単位は秒です。
<CAUSE>	Reclaiming	"Reclaiming"と出力されます。Explicit メモリブロックの明示解放処理時の Java ヒープあふれによって出力されたログであることを示します。

(d) 出力例

出力例を示します。

```
[ENS]<Tue Jul 24 01:23:51 2007>[EH: 706728K->706728K(706728K/1048576K), 0.1129602 secs][E/F/
D: 523/0/0]¥
[DefNew::Eden: 0K->243600K(243600K)][DefNew::Survivor: 0K->17400K(17400K)][Tenured: 278000K-
>556800K(556800K)]¥
[User: 0.0900000 secs][Sys: 0.0200000 secs][cause:Reclaiming]
[ENS]<Tue Jul 24 01:23:51 2007>[EH: 706728K->706728K(706728K/1048576K)][E/F/D: 523/0/0][caus
e:Full GC][CF: 0]
[ENS]<Tue Jul 24 01:23:53 2007>[EH: 706728K->148528K(148528K/1048576K), 0.0123405 secs][E/F/
D: 521/0/0]¥
[DefNew::Eden: 0K->0K(243600K)][DefNew::Survivor: 0K->0K(17400K)][Tenured: 551800K->552800K(
556800K)]¥
[User: 0.0090000 secs][Sys: 0.0020000 secs][cause:Reclaim]
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 7 月 24 日(火)1 時 23 分 51 秒に発生した Explicit メモリブロックの明示解放処理での Java ヒープあふれです。
- Explicit メモリブロックの明示解放処理開始から Java ヒープがあふれるまでに，0.1129602 秒掛かりました。
- Java ヒープがあふれたあとでサブ状態が Enable である Explicit メモリブロックは 523 個あります。
- Explicit メモリブロックの明示解放処理によって，5,398,00K が Java ヒープに移動したため，Java ヒープがあふれました。
- Explicit メモリブロックの明示解放処理開始から Java ヒープがあふれるまでに，ユーザ CPU 時間が 0.0900000 秒，システム CPU 時間が 0.020000 秒掛かりました。

また，出力例の 3 行目の[ENS]以降の出力内容は，Explicit メモリブロックの明示解放処理によって出力されたログです。Java ヒープあふれによって出力されたログのあとには，必ず Explicit メモリブロックの明示解放処理によってログが出力されます。この例では，次の内容が出力されています。

- 2007 年 7 月 24 日(火)1 時 23 分 53 秒に Explicit メモリブロックの明示解放処理を再開して，このログを出力しました。
- 再開した Explicit メモリブロックの明示解放処理によって，Explicit ヒープの利用済みサイズが 706,728K から 148,528K に減少しました。
- 再開した Explicit メモリブロックの明示解放処理を実行したあとでの Explicit ヒープの確保済みサイズは 148,528K です。Explicit ヒープの最大サイズは 1,048,576K です。
- 再開した Explicit メモリブロックの明示解放処理に，0.0123405 秒掛かりました。
- 再開した Explicit メモリブロックの明示解放処理後，サブ状態が Enable である Explicit メモリブロックは 521 個あります。
- 再開した Explicit メモリブロックの明示解放処理によって，Java ヒープの Tenured 領域の使用サイズが，551,800K から 552,800K に増加しました。
- 再開した Explicit メモリブロックの明示解放処理に，ユーザ CPU 時間が 0.0090000 秒，システム CPU 時間が 0.0020000 秒掛かりました。

(4) Explicit メモリブロックの自動解放処理

Explicit メモリブロックの自動解放自動予約または自動解放明示予約から，Explicit メモリブロックの自動解放処理までの，Explicit ヒープおよび Explicit メモリブロックの利用状況が出力されます。

(a) 出力の契機

Explicit メモリブロックの自動解放自動予約，自動解放明示予約，および Explicit メモリブロックの自動解放処理が起きた場合です。

(b) 出力形式

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>), <ELAPSED> secs][E/F/D: <AC
_NUM>/<FL_NUM>/<DA_NUM>]¥
[DefNew::Eden: <ED_USED_BF>-><ED_USED_AF>(<ED_TOTAL>)]
[DefNew::Survivor: <SV_USED_BF>-><SV_USED_AF>(<SV_TOTAL>)]¥
[Tenured: <TN_USED_BF>-><TN_USED_AF>(<TN_TOTAL>)][target:<EH_MIG_TRG>/<EH_MIG_DED>/<EH_MIG_L
IV>]¥
[User: <USERCPU> secs][Sys: <SYSCPU> secs][cause:<CAUSE>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-36 出力項目（Explicit メモリブロックの自動解放処理）

出力項目	出力内容	意味
<ctime>	<letters>	Explicit メモリブロックの自動解放自動予約が発生した日時を示します。拡張 verboseGC 機能で出力しているものと同一の時刻形式で出力されます。HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EH_USED_BF>	<const>K	Explicit メモリブロックの自動解放処理前の Explicit ヒープ利用済みサイズが出力されます。単位はキロバイトです。
<EH_USED_AF>	<const>K	Explicit メモリブロックの自動解放処理後の Explicit ヒープ利用済みサイズが出力されます。単位はキロバイトです。
<EH_TOTAL>	<const>K	Explicit メモリブロックの自動解放処理後の確保済み Explicit ヒープサイズが出力されます。単位はキロバイトです。
<EH_MAX>	<const>K	Explicit ヒープ最大サイズが出力されます。単位はキロバイトです。
<ELAPSED>	<time>	Explicit メモリブロックの自動解放自動予約の処理の開始から，自動解放処理終了までの時間が出力されます。単位は秒です。
<AC_NUM>	<const>	Explicit メモリブロックの自動解放処理後の，サブ状態が Enable である有効な Explicit メモリブロックの数が出力されます。
<FL_NUM>	<const>	常に 0 が出力されます。
<DA_NUM>	<const>	Explicit メモリブロックの自動解放処理後の，サブ状態が Disable である Explicit メモリブロックの数が出力されます。
<ED_USED_BF>	<const>K	Explicit メモリブロックの自動解放処理前の，Eden 領域利用済みサイズが出力されます。単位はキロバイトです。
<ED_USED_AF>	<const>K	Explicit メモリブロックの自動解放処理後の，Eden 領域利用済みサイズが出力されます。単位はキロバイトです。
<ED_TOTAL>	<const>K	Explicit メモリブロックの自動解放処理後の，Eden 領域確保済みサイズが出力されます。単位はキロバイトです。
<SV_USED_BF>	<const>K	Explicit メモリブロックの自動解放処理前の，Survivor 領域利用済みサイズが出力されます。単位はキロバイトです。

出力項目	出力内容	意味
<SV_USED_AF>	<const>K	Explicit メモリブロックの自動解放処理後の，Survivor 領域利用済みサイズが出力されます。単位はキロバイトです。
<SV_TOTAL>	<const>K	Explicit メモリブロックの自動解放処理後の，Survivor 領域確保済みサイズが出力されます。単位はキロバイトです。
<TN_USED_BF>	<const>K	Explicit メモリブロックの自動解放処理前の，Tenured 領域利用済みサイズが出力されます。単位はキロバイトです。
<TN_USED_AF>	<const>K	Explicit メモリブロックの自動解放処理後の，Tenured 領域利用済みサイズが出力されます。単位はキロバイトです。
<TN_TOTAL>	<const>K	Explicit メモリブロックの自動解放処理後の，Tenured 領域確保済みサイズが出力されます。単位はキロバイトです。
<EH_MIG_TRG>	<const>K	Explicit メモリブロックの自動解放処理をした Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_MIG_DED>	<const>K	Explicit メモリブロックの自動解放処理をしたことによって減少した，Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_MIG_LIV>	<const>K	Explicit メモリブロックの自動解放処理をしても減少しなかった，Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<USERCPU>	<time>	Explicit メモリブロックの自動解放自動予約の処理の開始から，自動解放処理終了までのユーザ CPU 時間が出力されます。単位は秒です。
<SYSCPU>	<time>	Explicit メモリブロックの自動解放自動予約の処理の開始から，自動解放処理終了までのシステム CPU 時間が出力されます。単位は秒です。
<CAUSE>	Migrate	"Migrate"が出力されます。Explicit メモリブロックの自動解放処理によって出力されたログであることを示します。

(d) 出力例

出力例を示します。

```
[ENS]<Tue Jul 14 02:31:22 2009>[EH: 256512K->256128K(256256K/1048576K), 0.1124626 secs][E/F/
D: 423/0/0]¥
[DefNew::Eden: 0K->0K(243600K)][DefNew::Survivor: 0K->0K(17400K)][Tenured: 103400K->103400K(
556800K)][target:584K/384K/200K]¥
[User: 0.0900000 secs][Sys: 0.0200000 secs][cause:Migrate]
```

この出力例では次の内容が確認できます。

- 出力契機は 2009 年 7 月 14 日（火）2 時 31 分 22 秒に発生した GC での Explicit メモリブロックの自動解放処理です。
- 自動解放処理によって Explicit ヒープの利用済みサイズが 256,512K から 256,128K に変化しました。
- 自動解放処理後の Explicit ヒープの確保済みサイズは 256,256K，最大サイズは 1,048,576K です。
- 自動解放処理に，0.1124626 秒掛かりました。

- 自動解放処理後の，サブ状態が Enable である Explicit メモリブロックは 423 個です。
- 自動解放処理によって，Explicit ヒープの利用済みサイズのうち 584K に対して自動解放が行われました。減少したサイズは 384K，減少しなかったサイズは 200K でした。
- Explicit メモリブロックの自動解放処理による，Java ヒープ各領域の変化はありませんでした。
- 自動解放処理に，ユーザ CPU 時間が 0.0900000 秒，システム CPU 時間が 0.0200000 秒掛かりました。

(5) Explicit メモリブロックの自動解放処理時の Java ヒープあふれ

Explicit メモリブロックの自動解放処理時には，Java ヒープへのオブジェクトの移動が発生します。このとき，Java ヒープがあふれた場合には，その時点での Explicit ヒープおよび Java ヒープの利用状況が出力されます。

Java ヒープがあふれた場合とは，Java ヒープへのオブジェクト移動時に Java ヒープに空き領域がなかった場合を指します。詳細については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7. 明示管理ヒープ機能を使用した FullGC の抑止」を参照してください。

(a) 出力の契機

Explicit メモリブロックの自動解放処理中に，Explicit ヒープ領域の空き領域が不足すると，Java ヒープへのオブジェクトの移動が発生します。このとき，Java ヒープがあふれた場合です。

(b) 出力形式

```
[ENS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>), <ELAPSED> secs][E/F/D: <AC
_NUM>/<FL_NUM>/<DA_NUM>]¥
[DefNew::Eden: <ED_USED_BF>-><ED_USED_AF>(<ED_TOTAL>)]
[DefNew::Survivor: <SV_USED_BF>-><SV_USED_AF>(<SV_TOTAL>)]¥
[Tenured: <TN_USED_BF>-><TN_USED_AF>(<TN_TOTAL>)][target:<EH_MIG_TRG>/<EH_MIG_DED>/<EH_MIG_L
IV>]¥
[User: <USERCPU> secs][Sys: <SYSCPU> secs][cause:<CAUSE>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-37 出力項目（Explicit メモリブロックの自動解放処理時の Java ヒープあふれ）

出力項目	出力内容	意味
<ctime>	<letters>	Explicit メモリブロックの自動解放処理が発生した日時を示します。拡張 verboseGC 機能で出力しているものと同一の時刻形式で出力されます。HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EH_USED_BF>	<const>K	Explicit メモリブロックの自動解放処理前の Explicit ヒープ利用済みサイズが出力されます。単位はキロバイトです。

出力項目	出力内容	意味
<EH_USED_AF>	<const>K	Java ヒープがあふれたあとの Explicit ヒープ利用済みサイズが出力されます。単位はキロバイトです。
<EH_TOTAL>	<const>K	Java ヒープがあふれたあとの確保済み Explicit ヒープサイズが出力されます。単位はキロバイトです。
<EH_MAX>	<const>K	Explicit ヒープ最大サイズが出力されます。単位はキロバイトです。
<ELAPSED>	<time>	Explicit メモリブロックの自動解放処理開始から，Java ヒープがあふれるまでの時間が出力されます。単位は秒です。
<AC_NUM>	<const>	Explicit メモリブロックの自動解放処理後の，サブ状態が Enable である Explicit メモリブロックの数が出力されます。
<FL_NUM>	<const>	常に 0 が出力されます。
<DA_NUM>	<const>	Java ヒープがあふれたあとの，サブ状態が Disable である Explicit メモリブロックの数が出力されます。
<ED_USED_BF>	<const>K	Explicit メモリブロックの自動解放処理前の，Eden 領域利用済みサイズが出力されます。単位はキロバイトです。
<ED_USED_AF>	<const>K	Java ヒープがあふれたあとの Eden 領域利用済みサイズが出力されます。単位はキロバイトです。
<ED_TOTAL>	<const>K	Java ヒープがあふれたあとの Eden 領域確保済みサイズが出力されます。単位はキロバイトです。
<SV_USED_BF>	<const>K	Explicit メモリブロックの自動解放処理前の Survivor 領域利用済みサイズが出力されます。単位はキロバイトです。
<SV_USED_AF>	<const>K	Java ヒープがあふれたあとの Survivor 領域利用済みサイズが出力されます。単位はキロバイトです。
<SV_TOTAL>	<const>K	Java ヒープがあふれたあとの Survivor 領域確保済みサイズが出力されます。単位はキロバイトです。
<TN_USED_BF>	<const>K	Explicit メモリブロックの自動解放処理前の Tenured 領域利用済みサイズが出力されます。単位はキロバイトです。
<TN_USED_AF>	<const>K	Java ヒープがあふれたあとの Tenured 領域利用済みサイズが出力されます。単位はキロバイトです。
<TN_TOTAL>	<const>K	Java ヒープがあふれたあとの Tenured 領域確保済みサイズが出力されます。単位はキロバイトです。
<EH_MIG_TRG>	<const>K	Explicit メモリブロックの自動解放処理をした Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_MIG_DED>	<const>K	Java ヒープがあふれる前までに Explicit メモリブロックの自動解放処理によって減少した，Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。常に OK が出力されます。

出力項目	出力内容	意味
<EH_MIG_LIV>	<const>K	Java ヒープがあふれる前までに Explicit メモリブロックの自動解放処理によって減少しなかった，Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。Java ヒープあふれの発生原因となったオブジェクトのサイズは含みません。
<USERCPU>	<time>	Explicit メモリブロックの自動解放処理開始から，Java ヒープがあふれるまでのユーザ CPU 時間が出力されます。単位は秒です。
<SYSCPU>	<time>	Explicit メモリブロックの自動解放処理開始から，Java ヒープがあふれるまでのシステム CPU 時間が出力されます。単位は秒です。
<CAUSE>	Migrating	"Migrating"が出力されます。Explicit メモリブロックの自動解放処理時の Java ヒープあふれによって出力されたログであることを示します。

(d) 出力例

出力例を示します。

```
[ENS]<Tue Jul 14 02:31:22 2009>[EH: 706728K->706728K(706728K/706728K), 0.1129602 secs][E/F/D
: 522/0/1]¥
[DefNew::Eden: 0K->243600K(243600K)][DefNew::Survivor: 0K->17400K(17400K)][Tenured: 278000K-
>556800K(556800K)]¥
[target:372000K/0K/339800K] [User: 0.0900000 secs][Sys: 0.0200000 secs][cause:Migrating]
```

この出力例では次の内容が確認できます。

- 出力契機は 2009 年 7 月 14 日(火)2 時 31 分 22 秒に発生した GC での Explicit メモリブロックの自動解放処理時の Java ヒープあふれです。
- 自動解放処理によって Explicit ヒープの利用済みサイズが 706,728K から変化ありません。
- 自動解放処理後の Explicit ヒープの確保済みサイズは 706,728K，最大サイズは 706,728K です。
- 自動解放処理に，0.1129602 秒掛かりました。
- 自動解放処理後の，サブ状態が Enable である Explicit メモリブロックは 522 個です。サブ状態が Disable である Explicit メモリブロックは 1 個です。
- 自動解放処理によって，Explicit ヒープの利用済みサイズのうち 372,000K に対して自動解放が実行されました。Java ヒープあふれの時点までに 339,800K が減少しませんでした。
- 自動解放処理時の Java ヒープあふれによって，Java ヒープの利用済みサイズが各領域の上限に到達しました。
- Explicit メモリブロックの自動解放処理開始から Java ヒープがあふれるまでに，ユーザ CPU 時間が 0.0900000 秒，システム CPU 時間が 0.0200000 秒掛かりました。

(6)　明示管理ヒープ自動配置設定ファイルオープンエラー

明示管理ヒープ自動配置設定ファイルのオープンや読み込みに失敗した場合，エラーメッセージが出力されます。

(a)　出力の契機

明示管理ヒープ自動配置設定ファイルのオープンや読み込みに失敗した場合です。例えば，ファイルが存在しない場合，ファイルの読み込み権限がない場合，ファイル読み込み中に予期しない IO エラーが発生した場合があります。

(b)　出力形式

```
[ENA]<ctime> failed to open file. [file=<FILENAME>]
```

(c)　出力項目

「(b)　出力形式」で示した各項目について説明します。

表 5-38　出力項目（明示管理ヒープ自動配置設定ファイルオープンエラー）

出力項目	出力内容	意味
<ctime>	<letters>	明示管理ヒープ自動配置設定ファイルのオープンに失敗した日時を示します。拡張 VerboseGC 機能で出力しているものと同一の時刻形式で出力されます。HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力します。
<FILENAME>	<letters>	ファイルのオープンに失敗した自動配置設定ファイルの名前が出力されます（ディレクトリ名を含みません）。

(d)　出力例

出力例を示します。

```
[ENA]<Tue Jul 24 01:23:51 2007> failed to open file. [file=usrexmem.cfg]
```

この出力例では次の内容が確認できます。

- 2007 年 7 月 24 日(火)1 時 23 分 51 秒に，明示管理ヒープ自動配置設定ファイルのオープンに失敗しました。

(7)　明示管理ヒープ自動配置設定ファイルパースエラー

明示管理ヒープ自動配置設定ファイルのパースに失敗した行がある場合，エラーメッセージが出力されます。

(a) 出力の契機

明示管理ヒープ自動配置設定ファイルのパースに失敗した行がある場合です。ファイルの複数行で記述フォーマットエラーがある場合，複数回ログ出力されます。

(b) 出力形式

```
[ENA]<ctime> parsed error line. [file=<FILENAME> line=<LINENO>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-39 出力項目（明示管理ヒープ自動配置設定ファイルパースエラー）

出力項目	出力内容	意味
<ctime>	<letters>	明示管理ヒープ機能自動配置設定ファイルのパースに失敗した日時を示します。拡張 verboseGC 機能で出力しているものと同一の時刻形式で出力されます。HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<FILENAME>	<letters>	ファイルのパースに失敗した自動配置設定ファイルの名前が出力されます（ディレクトリ名を含みません）。
<LINENO>	<const>	パースに失敗した行数が出力されます。

(d) 出力例

出力例を示します。

```
[ENA]<Tue Jul 24 01:23:51 2007> parsed error line. [file=usrexmem.cfg line=25]
```

この出力例では次の内容が確認できます。

- 2007 年 7 月 24 日(火)1 時 23 分 51 秒に，明示管理ヒープ機能自動配置設定ファイルのパースに 25 行目で失敗しました。

(8) 明示管理ヒープ自動配置エラー

明示管理ヒープ機能によって指定したクラスが，明示管理ヒープへの自動配置に失敗した場合，エラーメッセージが出力されます。

(a) 出力の契機

明示管理ヒープ機能によって指定したクラスが，明示管理ヒープへの自動配置に失敗した場合です。

(b) 出力形式

```
[ENA]<ctime> creation <CLASS_LIST> class's object in explicit memory is failed. [target=<CLA
SS_METHOD> ¥
detail=<MESSAGE>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-40 出力項目（明示管理ヒープ自動配置エラー）

出力項目	出力内容	意味
<ctime>	<letters>	明示管理ヒープ上への自動配置に失敗した日時を示します。拡張 VerboseGC 機能で出力しているものと同一の時刻形式で出力されます。HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<CLASS_LIST>	<letters>	明示管理ヒープ上への自動配置しようとしたオブジェクトの完全限定クラス名のリストが出力されます。リストが空の場合があります。
<CLASS_METHOD>	<letters>	明示管理ヒープ上への自動配置に失敗したクラスの完全限定名が出力されます。より詳細な失敗個所を示すメソッド名も出力されることがあります。
<MESSAGE>	<letters>	明示管理ヒープ上への自動配置に失敗した原因を示す詳細メッセージが出力されます。

(d) 出力例

出力例を示します。

```
[ENA]<Tue Jul 24 01:23:51 2007> creation java.util.HashMap, java.util.LinkedList ¥
class's object in explicit memory is failed. [target=com.sample.MainClass.main ¥
detail=Invalid class file format. (max_stack = 65536, max = 65535, min = 0)]
```

この出力例では次の内容が確認できます。

- 2007 年 7 月 24 日(火)1 時 23 分 51 秒に，明示管理ヒープ機能がクラス jp.co.sample.Main の明示管理ヒープ上への自動配置に失敗しました。

(9) 明示管理ヒープ機能適用除外クラス指定機能の設定ファイルオープンエラー

明示管理ヒープ機能適用除外クラス指定機能の設定ファイルのオープンや読み込みに失敗した場合，エラーメッセージが出力されます。

(a) 出力の契機

明示管理ヒープ機能適用除外クラス指定機能の設定ファイルのオープンや読み込みに失敗した場合です。例えば，ファイルが存在しない場合，ファイルの読み込み権限がない場合，ファイル読み込み中に予期しない IO エラーが発生した場合があります。

(b) 出力形式

```
[ENO]<ctime> failed to open file. [<TYPE>] [file=<FILENAME>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-41 出力項目（明示管理ヒープ機能適用除外クラス指定機能の設定ファイルオープンエラー）

出力項目	出力内容	意味
<ctime>	<letters>	設定ファイルのオープンに失敗した日時を示します。拡張 VerboseGC 機能で出力しているものと同一の時刻形式で出力されます。HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力します。
<TYPE>	SYS|USR|DEF	ファイルのオープンや読み込みに失敗した設定ファイルの種類が出力されます。 "SYS"はシステムで提供している設定ファイル，"USR"は JavaVM 起動オプションで指定したファイルパスの設定ファイル，"DEF"は JavaVM 起動オプションのデフォルトのファイルパスにある設定ファイルを示します。
<FILENAME>	<letters>	ファイルのオープンに失敗した設定ファイルの名前が出力されます（ディレクトリ名を含みません）。

(d) 出力例

出力例を示します。

```
[ENO]<Fri Aug 10 17:41:51 2012> failed to open file. [USR] [file=javamove.cfg]
```

この出力例では次の内容が確認できます。

- 2012 年 8 月 10 日(金)17 時 41 分 51 秒に，JavaVM 起動オプションで指定したファイルパスの設定ファイルのオープンに失敗しました。

(10) 明示管理ヒープ機能適用除外クラス指定機能の設定ファイルパースエラー

明示管理ヒープ機能適用除外クラス指定機能の設定ファイルのパースに失敗した行がある場合，エラーメッセージが出力されます。

(a) 出力の契機

明示管理ヒープ機能適用除外クラス指定機能の設定ファイルのパースに失敗した行がある場合です。ファイルの複数行で記述フォーマットエラーがある場合，複数回ログ出力されます。

(b) 出力形式

```
[ENO]<ctime> parsed error line. [TYPE] [file=<FILENAME> line=<LINENO>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-42 出力項目（明示管理ヒープ自動配置設定ファイルパースエラー）

出力項目	出力内容	意味
<ctime>	<letters>	明示管理ヒープ機能適用除外クラス指定機能の設定ファイルのパースに失敗した日時を示します。拡張 verboseGC 機能で出力しているものと同一の時刻形式で出力されます。HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<TYPE>	SYS|USR|DEF	ファイルのパースに失敗した設定ファイルの種類が出力されます。 "SYS"はシステムで提供している設定ファイル，"USR"は JavaVM 起動オプションで指定したファイルパスの設定ファイル，"DEF"は JavaVM 起動オプションのデフォルトのファイルパスにある設定ファイルを示します。
<FILENAME>	<letters>	ファイルのパースに失敗した設定ファイルの名前が出力されます（ディレクトリ名を含みません）。
<LINENO>	<const>	パースに失敗した行数が出力されます。

(d) 出力例

出力例を示します。

```
[ENO]<Fri Aug 10 17:41:51 2012> parsed error line. [USR] [file=javamove.cfg line=25]
```

この出力例では次の内容が確認できます。

- 2012 年 8 月 10 日(金)17 時 41 分 51 秒に，JavaVM 起動オプションでファイルパスを指定した設定ファイルのパースに 25 行目で失敗しました。

5.11.4　出力レベルが verbose の場合に出力される内容

ここでは，ログ出力レベルに verbose を指定した場合に出力される内容をイベントごとに説明します。verbose は，障害解析などに必要な詳細情報を出力するためのログ出力レベルです。

補足

verbose では，normal で出力する内容に加えて，normal では出力しない詳細なログを出力します。ログ出力のオーバーヘッドが掛かるため，通常運用時に指定するとスループットが低下するおそれがあります。

(1)　Explicit メモリブロックの初期化

新たに Explicit メモリブロックを初期化した場合に，初期化した Explicit メモリブロックの名称，ID，および Explicit メモリブロックの種類を出力します。

(a)　出力の契機

Explicit メモリブロックの初期化です。

(b)　出力形式

```
[EVO]<ctime>[Created]["<EM_NAME>" eid=<EID>(<EM_PTR>)/<EM_TYPE>]
```

(c)　出力項目

「(b)　出力形式」で示した各項目について説明します。

表 5-43　出力項目（Explicit メモリブロックの初期化）

出力項目	出力内容	意味
<ctime>	<letters>	Explicit メモリブロックを初期化した日時を示します。拡張 verbosegc 情報と同じ形式で出力されます。 -XX:+HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EM_NAME>	<letters>	初期化した Explicit メモリブロックの名称が出力されます。 Explicit メモリブロックの名称に多バイト文字が含まれている場合，出力内容は不定です（通常は文字化けして出力されます）。
<EID>	<const>	初期化した Explicit メモリブロックの ID が出力されます。
<EM_PTR>	<ptr>	Explicit メモリブロックの内部状態を示す値が出力されます。
<EM_TYPE>	R|B	初期化した Explicit メモリブロックの種類が出力されます。 R は，アプリケーションサーバの内部で利用されている Explicit メモリブロックを示します。B は，Explicit メモリブロックの JavaVM 内部での種別を示します。

(d) 出力例

出力例を示します。

```
[EVO]<Tue Jul 24 01:23:51 2007>[Created]["BasicExplicitMemory-2" eid=2(0x1234568)/B]
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 7 月 24 日(火)1 時 23 分 51 秒に実行された Explicit メモリブロックの初期化です。
- 初期化された Explicit メモリブロックの名称は「BasicExplicitMemory-2」です。

(2) Explicit メモリブロックの初期化失敗

Explicit メモリブロックの初期化時に，Explicit メモリブロックの初期化が失敗した場合に出力されます。失敗の原因は Explicit メモリブロックの最大数に達したことです。その時点の Explicit メモリブロックの数と，初期化に失敗した Java プログラム上のスタックトレースが出力されます。

なお，このログは複数行にわたって出力されます。また，Java プログラムの実行とは非同期に出力されます。このため，各行の間に別のログが出力されることがあります。ただし，1 行の中にほかのログが出力されることはありません。

(a) 出力の契機

Explicit メモリブロックの上限に達して，Explicit メモリブロックの初期化に失敗した場合です。

(b) 出力形式

```
[EVO]<ctime>[Creation failed][EH: <EH_USED>(<EH_GARB>)/<EH_TOTAL>/<EH_MAX>][E/F/D: <AC_NUM>/
<FL_NUM>/<DA_NUM>][Thread: <TH_PTR>]
[EVO][Thread: <TH_PTR>] at <FRAME><SOURCE>
…
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-44 出力項目（Explicit メモリブロックの初期化失敗）

出力項目	出力内容	意味
<ctime>	<letters>	Explicit メモリブロックを初期化が失敗した日時を示します。拡張 verbosegc 情報と同じ形式で出力されます。 -XX:+HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EH_USED>	<const>K	Explicit メモリブロック初期化失敗時の Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_GARB>	<const>K	Explicit ヒープの内部状態が出力されます。

出力項目	出力内容	意味
<EH_TOTAL>	<const>K	Explicit メモリブロック初期化失敗時の Explicit ヒープの確保済みサイズが出力されます。単位はキロバイトです。
<EH_MAX>	<const>K	Explicit ヒープ最大サイズが出力されます。単位はキロバイトです。
<AC_NUM>	<const>	Explicit メモリブロック初期化失敗時にサブ状態が Enable である Explicit メモリブロックの数が出力されます。
<FL_NUM>	<const>	常に 0 が出力されます。
<DA_NUM>	<const>	Explicit メモリブロック初期化失敗時にサブ状態が Disable である Explicit メモリブロックの数が出力されます。
<TH_PTR>	<ptr>	Explicit メモリブロックの初期化に失敗したスレッドのスレッド ID が出力されます。スレッドダンプに出力される tid と同一です。
<FRAME>	<letters>.<letters>	Explicit メモリブロック初期化失敗時のスタックトレース中の 1 フレームが出力されます。完全クラス名とメソッド名が"."で区切られて出力されます。
<SOURCE>	(<letters>:<const>)| (Native Method)| (Unknown Source)	<FRAME>で出力されたメソッドが記述されているソースファイル名と，スタックトレースと一致する行番号が出力されます。ファイル名と行番号は":"で区切られて出力されます。 ネイティブメソッドの場合は，"(Native Method)"と出力されます。ソースファイル名が取得できない場合は，"(Unknown Source)"と出力されます。

(d) 出力例

出力例を示します。

```
[EVO]<Tue Jul 24 01:23:51 2007>[Creation failed][EH: 12000K(0K)/15000K/30000K][E/F/D: 65535/
0/0][Thread: 0x00035a60]
[EVO][Thread: 0x00035a60] at ExplicitMemory.registerExplicitMemory(Native Method)
[EVO][Thread: 0x00035a60] at BasicExplicitMemory.<init>(Unknown Source)
[EVO][Thread: 0x00035a60] at AllocTest.test(AllocTest.java:64)
[EVO][Thread: 0x00035a60] at java.lang.Thread.run(Thread.java:2312)
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 7 月 24 日(火)1 時 23 分 51 秒に発生した Explicit メモリブロックの初期化失敗です。最大数である 65535 個の Explicit メモリブロックがすでに存在しているため，新たな Explicit メモリブロックの初期化に失敗しました。
- AllocTest.java の 64 行目で，BasicExplicitMemory クラスのコンストラクタを実行して，Explicit メモリブロックの初期化を試みています。

(3) Explicit メモリブロックのサブ状態の Disable 化

Explicit メモリブロックのサブ状態が Disable に変更になった場合に，その時点での Explicit ヒープの利用状況と，サブ状態が Disable に変更された Explicit メモリブロックの情報が出力されます。このログは，複数行にわたって，Java プログラムの実行と非同期に出力されます。このため，各行の間に別なログが出力されることがあります。ただし，1 行中にほかのログが出力されることはありません。

(a) 出力の契機

Explicit メモリブロックのサブ状態が Disable になり，その Explicit メモリブロックにオブジェクトを配置できなくなった場合です。

(b) 出力形式

```
[EVO]<ctime>[Alloc failed(Disable)][EH: <EH_USED>(<EH_GARB>)/<EH_TOTAL>/<EH_MAX>][E/F/D: <AC
_NUM>/<FL_NUM>/<DA_NUM>][cause:<CAUSE>]¥
["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_USED>(<EM_GARB>)/<EM_TOTAL>][Thread: <TH PTR>]
[EVO][Thread: <TH PTR>] at <FRAME><SOURCE>
…
```

注

下線部分は New の場合にだけ出力されます。

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-45 出力項目（Explicit メモリブロックのサブ状態の Disable 化）

出力項目	出力内容	意味
<ctime>	<letters>	<EID>が示す Explicit メモリブロックのサブ状態が Disable になった日時を示します。拡張 verbosegc 情報と同じ形式で出力されます。 -XX:+HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EH_USED>	<const>K	<EID>が示す Explicit メモリブロックのサブ状態が Disable になったときの Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_GARB>	<const>K	Explicit ヒープの内部状態を示す値が出力されます。
<EH_TOTAL>	<const>K	<EID>が示す Explicit メモリブロックのサブ状態が Disable になったときの Explicit ヒープの確保済みサイズが出力されます。単位はキロバイトです。
<EH_MAX>	<const>K	Explicit ヒープ最大サイズが出力されます。単位はキロバイトです。

出力項目	出力内容	意味
<AC_NUM>	<const>	<EID>が示す Explicit メモリブロックのサブ状態が Disable になったあとで，サブ状態が Enable である Explicit メモリブロックの数が出力されます。
<FL_NUM>	<const>	常に 0 が出力されます。
<DA_NUM>	<const>	<EID>が示す Explicit メモリブロックのサブ状態が Disable になったあとで，サブ状態が Disable である Explicit メモリブロックの数が出力されます。
<CAUSE>	New|GC|Full GC	サブ状態が Disable になった原因の処理が出力されます。 出力内容と原因の対応は次のとおりです。 • "New" newInstance()などによる Explicit ヒープへのオブジェクト直接生成が原因です。 • "GC" CopyGC による Explicit ヒープへのオブジェクト移動が原因です。 • "Full GC" FullGC による Explicit ヒープへのオブジェクト移動が原因です。
<EM_NAME>	<letters>	サブ状態が Disable になった Explicit メモリブロックの名称が出力されます。 Explicit メモリブロックの名称に多バイト文字が含まれている場合，出力内容は不定です（通常は文字化けして出力されます）。
<EID>	<const>	サブ状態が Disable になった Explicit メモリブロックの ID が出力されます。
<EM_TYPE>	R|B|A	サブ状態が Disable になった Explicit メモリブロックの種類が出力されます。 R は，アプリケーションサーバの内部で利用されている Explicit メモリブロックを示します。B は，Explicit メモリブロックの JavaVM 内部での種別を示します。A は，自動配置設定ファイルを使って指定した Explicit メモリブロックを示します。
<EM_USED>	<const>K	サブ状態が Disable になった Explicit メモリブロックの利用済みサイズが出力されます。単位はキロバイトです。
<EM_GARB>	<const>K	Explicit メモリブロックの内部状態を示す値が出力されます。
<EM_TOTAL>	<const>K	サブ状態が Disable になった Explicit メモリブロックの確保済みサイズが出力されます。単位はキロバイトです。
<TH_PTR>	<ptr>	サブ状態が Disable になった原因である，Explicit ヒープへのオブジェクト生成を実行したスレッドのスレッド ID が出力されます。スレッドダンプに出力される tid と同一です。 この項目は，<CAUSE>が"New"の場合だけに出力されます。

出力項目	出力内容	意味
<FRAME>	<letters>.<letters>	Explicit ヒープへのオブジェクト直接生成によってサブ状態が Disable になった場合に，オブジェクト直接生成で出力されたスタックトレース中の 1 フレームが出力されます。 完全クラス名とメソッド名が"."で区切られて出力されます。 この項目は，<CAUSE>が"New"の場合だけに出力されます。
<SOURCE>	(<letters>:<const>)| (Native Method)| (Unknown Source)	<FRAME>で出力されたメソッドが記述されているソースファイル名と，スタックトレースと一致する行番号が出力されます。ファイル名と行番号は":"で区切られて出力されます。 ネイティブメソッドの場合は，"(Native Method)"と出力されます。 ソースファイル名が取得できない場合は，"(Unknown Source)"と出力されます。 この項目は，<CAUSE>が"New"の場合だけに出力されます。

(d) 出力例

出力例を示します。

```
[EVO]<Tue Jul 24 01:23:51 2007>[Alloc failed(Disable)][EH: 12000K(1258K)/15000K/30000K][E/F/
D: 321/0/1][cause:GC]¥
[”ReferenceExplicitMemory-3” eid=3/R: 108K(20K)/108K]
```

この出力例では次の内容が確認できます。

- 出力契機は，2007 年 7 月 24 日(火)1 時 23 分 51 秒に Explicit メモリブロックのサブ状態が Disable になったことです。
- Explicit ヒープは，12000K が利用済み，15000K が確保済みです。
- Explicit ヒープの最大サイズは 30000K です。
- Explicit ヒープ全体で有効な Explicit メモリブロックは 322 個です。このうち，サブ状態が Enable のものは 321 個，Disable のものは 1 個あります。
- サブ状態が Disable になった原因の処理は，GC 発生時の Explicit メモリブロックへのオブジェクトの移動です。
- サブ状態が Disable になったのは，ID が"3"，名称が"ReferenceExplicitMemory-3"の Explicit メモリブロックです。
- "ReferenceExplicitMemory-3"では，108k のメモリを利用済みです。

(4) Explicit メモリブロックへのオブジェクト生成

ExplicitMemory.newInstance()などを使用して Explicit メモリブロックに直接オブジェクトを生成した場合に出力されます。

(a) 出力の契機

Explicit メモリブロックにオブジェクトが生成された時です。

(b) 出力形式

```
[EVS]<ctime>[EH: <EH_USED_BF>-><EH_USED_AF>(<EH_TOTAL>/<EH_MAX>)][E/F/D: <AC_NUM>/<FL_NUM>/<
DA_NUM>][cause:<CAUSE>]¥
[”<EM_NAME>” eid=<EID>/<EM_TYPE>: <EM_USED_BF>-><EM_USED_AF>(<EM_TOTAL>)]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-46 出力項目（Explicit メモリブロックへのオブジェクト生成）

出力項目	出力内容	意味
<ctime>	<letters>	Explicit メモリブロックにオブジェクトが生成された日時を示します。拡張 verbosegc 情報と同じ形式で出力されます。 -XX:+HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EH_USED_BF>	<const>K	オブジェクト生成前の Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_USED_AF>	<const>K	オブジェクト生成後の Explicit ヒープの利用済みサイズが出力されます。単位はキロバイトです。
<EH_TOTAL>	<const>K	オブジェクト生成後の Explicit ヒープの確保済みサイズが出力されます。単位はキロバイトです。
<EH_MAX>	<const>K	Explicit ヒープの最大サイズが出力されます。単位はキロバイトです。
<AC_NUM>	<const>	オブジェクト生成後にサブ状態が Enable である Explicit メモリブロックの数が出力されます。
<FL_NUM>	<const>	常に 0 が出力されます。
<DA_NUM>	<const>	オブジェクト生成後にサブ状態が Disable である Explicit メモリブロックの数が出力されます。
<CAUSE>	New	必ず"New"と出力されます。 Explicit メモリブロックへのオブジェクトへの移動が Java プログラムから実行されたことを示します。
<EM_NAME>	<letters>	オブジェクトを生成した Explicit メモリブロックの名称が出力されます。 Explicit メモリブロックの名称に多バイト文字が含まれている場合，出力内容は不定です（通常は文字化けして出力されます）。
<EID>	<const>	オブジェクトを生成した Explicit メモリブロックの ID が出力されます。

出力項目	出力内容	意味
<EM_TYPE>	R|B|A	オブジェクトを生成した Explicit メモリブロックの種類が出力されます。 R は，アプリケーションサーバの内部で利用されている Explicit メモリブロックを示します。B は，Explicit メモリブロックの JavaVM 内部での種別を示します。A は，自動配置設定ファイルを使って指定した Explicit メモリブロックを示します。
<EM_USED_BF>	<const>K	オブジェクト生成前の生成先 Explicit メモリブロックの利用済みサイズが出力されます。単位はキロバイトです。
<EM_USED_AF>	<const>K	オブジェクト生成後の生成先 Explicit メモリブロックの利用済みサイズが出力されます。単位はキロバイトです。
<EM_TOTAL>	<const>K	オブジェクト生成後の生成先 Explicit メモリブロックの確保済みサイズが出力されます。単位はキロバイトです。

(d) 出力例

出力例を示します。

```
[EVS]<Thu Oct 21 14:55:50 2007>[EH: 150528K->150529K(150532K/1048576K)][E/F/D: 200/0/0][cause:New]["BEM" eid=2/B: 30K->31K(32K)]
```

この出力例では次の内容が確認できます。

- 出力契機は，2007 年 10 月 21 日(木)14 時 55 分 50 秒に実行された，Explicit メモリブロックへのオブジェクトの生成です。
- Explicit ヒープにオブジェクトが生成されたため，Explicit ヒープの利用済みサイズが 150528K から 150529K に変化しました。
- Explicit ヒープへのオブジェクト生成後の Explicit ヒープの確保済みサイズは 150532K です。最大サイズは 1048576K です。
- Explicit ヒープへのオブジェクト生成後に，サブ状態が Enable である Explicit メモリブロックの数は 200 個です。
- "BEM"という名称のメモリブロックの利用済みサイズが，30K から 31K に変化しました。オブジェクト生成後の BEM の確保済みサイズは 32K です。

(5) Explicit メモリブロックへの移動（詳細情報出力）

Explicit メモリブロックにオブジェクトが移動する場合の詳細な情報が出力されます。「5.11.3(1) GC 発生（Explicit ヒープ利用状況出力）」で示した出力内容に加えて，移動先になった Explicit メモリブロックすべての利用状況が出力されます。

(a) 出力の契機

GC 発生による，Explicit メモリブロックへのオブジェクトの移動です。

(b)　出力形式

```
<GC発生時のExplicitヒープ利用状況>※
[EVS]{["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_USED_BF>-><EM_USED_AF>(<EM_TOTAL>)]}{1,5}
...
```

注

Explicit メモリブロックの情報が 5 個出力されるごとに改行されます。

注※

出力項目については，「5.11.3(1)　GC 発生（Explicit ヒープ利用状況出力）」を参照してください。

(c)　出力項目

「(b)　出力形式」で示した各項目について説明します。

表 5-47　出力項目（Explicit メモリブロックへの移動（詳細情報出力））

出力項目	出力内容	意味
GC 発生時の Explicit ヒープ利用状況		「5.11.3(1)　GC 発生（Explicit ヒープ利用状況出力）」を参照してください。
<EM_NAME>	<letters>	GC 発生時にオブジェクト移動先となった Explicit メモリブロックの名称が出力されます。 Explicit メモリブロックの名称に多バイト文字が含まれている場合，出力内容は不定です（通常は文字化けして出力されます）。
<EID>	<const>	GC 発生時にオブジェクト移動先となった Explicit メモリブロックの ID が出力されます。
<EM_TYPE>	R|A	GC 発生時にオブジェクト移動先となった Explicit メモリブロックの種別が出力されます。 R または A が出力されます。R は，Explicit メモリブロックの JavaVM 内部での種別を示します。A は，自動配置設定ファイルを使って指定した Explicit メモリブロックを示します。
<EM_USED_BF>	<const>K	GC 発生時にオブジェクト移動先となった Explicit メモリブロックの，GC 発生前の利用済みサイズが出力されます。単位はキロバイトです。
<EM_USED_AF>	<const>K	GC 発生時にオブジェクト移動先となった Explicit メモリブロックの，GC 発生後の利用済みサイズが出力されます。単位はキロバイトです。
<EM_TOTAL>	<const>K	GC 発生時にオブジェクト移動先となった Explicit メモリブロックの，GC 発生後の確保済みサイズが出力されます。単位はキロバイトです。

(d)　出力例

出力例を示します。

```
[ENS]<Thu Oct 21 14:55:50 2007>[EH: 150528K->162816K(162816K/1048576K)][E/F/D: 200/0/0][caus
e:GC][CF: 0]
[EVS]["REM2" eid=2/R: 0K->88K(128K)]["REM3" eid=3/R: 30K->230K(256K)]["REM6" eid=6/R: 30K->2
030K(2048K)]¥
["Session1" eid=8/R: 30K->2530K(2548K)]["Session2" eid=10/R: 30K->2530K(2548K)]
[EVS]["Session3" eid=12/R: 30K->5030K(5048K)]
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 10 月 21 日(木)14 時 55 分 50 秒に発生した GC での，Explicit ヒープへのオブジェクトの移動です。
- Explicit ヒープの利用済みサイズが，150528K から 162816K に変化しました。
- GC 実行後の Explicit ヒープの確保済みサイズは 162816K です。最大サイズは 1048576K です。
- GC 実行後に，サブ状態が Enable である Explicit メモリブロックは 200 個あります。
- GC の種類は CopyGC です。
- 移動先になった Explicit メモリブロックは 5 個以上あります。このため，5 個分出力されたところで 1 回改行されています。
- Explicit ヒープへの 2288K(162816K-150528K)移動の内訳は次のとおりです。
  - Explicit メモリブロック"REM2"(eid=2)に 88K 移動。
  - Explicit メモリブロック"REM3"(eid=3)に 200K 移動。
  - Explicit メモリブロック"REM6"(eid=6)に 2000K 移動。
  - Explicit メモリブロック"Session1"(eid=8)に 2500K 移動。
  - Explicit メモリブロック"Session2"(eid=10)に 2500K 移動。
  - Explicit メモリブロック"Session3"(eid=12)に 5000K 移動。

  なお，移動先は，すべてアプリケーションサーバの内部で利用されている Explicit メモリブロックです。

(6) Explicit メモリブロックの明示解放処理（詳細情報出力）

Explicit メモリブロックの明示解放処理が起こった場合の詳細な情報が出力されます。「5.11.3(2) Explicit メモリブロックの明示解放処理」で示した出力内容に加えて，明示解放したすべての Explicit メモリブロックの情報が出力されます。

Explicit メモリブロックの明示解放時に Java ヒープがあふれた場合は，あふれる前に明示解放した Explicit メモリブロックの情報が追加されて出力されます。ただし，「5.11.3(3) Explicit メモリブロックの明示解放処理時の Java ヒープあふれ」で示した出力内容には，ここで説明する情報は追加出力されません。

(a) 出力の契機

Explicit メモリブロックの明示解放処理です。

(b)　出力形式

```
<Explicitメモリブロックの明示解放処理>※
[EVS]{["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_TOTAL>]}{1,5}
...
```

注

Explicit メモリブロックの情報が 5 個出力されるごとに改行されます。

注※

出力項目については，「5.11.3(2)　Explicit メモリブロックの明示解放処理」を参照してください。

(c)　出力項目

「(b)　出力形式」で示した各項目について説明します。

表 5-48　出力項目（Explicit メモリブロックの明示解放処理（詳細情報出力））

出力項目	出力内容	意味
Explicit メモリブロックの明示解放処理で出力される情報		「5.11.3(2)　Explicit メモリブロックの明示解放処理」を参照してください。
<EM_NAME>	<letters>	明示解放した Explicit メモリブロックの名称が出力されます。 Explicit メモリブロックの名称に多バイト文字が含まれている場合，出力内容は不定です（通常は文字化けして出力されます）。
<EID>	<const>	明示解放した Explicit メモリブロックの ID が出力されます。
<EM_TYPE>	R|B|A	明示解放した Explicit メモリブロックの種別が出力されます。 R は，アプリケーションサーバの内部で利用されている Explicit メモリブロックを示します。B は，Explicit メモリブロックの JavaVM 内部での種別を示します。A は，自動配置設定ファイルを使って指定した Explicit メモリブロックを示します。
<EM_TOTAL>	<const>K	明示解放した Explicit メモリブロックによって確保済みだったメモリサイズ（明示解放処理によって明示解放されたメモリサイズ）が出力されます。単位はキロバイトです。

(d)　出力例

出力例を示します。

```
[ENS]<Tue Jul 24 01:23:51 2007>[EH: 150528K->149528K(162816K/1048576K), 0.1129602 secs][E/F/
D: 523/0/0]¥
[DefNew::Eden: 0K->0K(243600K)][DefNew::Survivor: 0K->0K(17400K)][Tenured: 103400K->103400K(
556800K)][cause:Reclaim]
[EVS]["REM2" eid=3/R: 300K]["BEM3" eid=5/B: 300K]["BEM1" eid=7/B: 400K]
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 7 月 24 日(火)1 時 23 分 51 秒に発生した Explicit メモリブロックの明示解放処理です。
- Explicit メモリブロックの明示解放処理によって，Explicit ヒープの利用済みサイズが 150528K から 149528K に減少しました。
- Explicit メモリブロックの明示解放処理後の Explicit ヒープの確保済みサイズは 162816K です。最大サイズは 1048576K です。
- Explicit メモリブロックの明示解放処理には，0.1129602 秒掛かりました。
- Explicit メモリブロックの明示解放処理後，サブ状態が Enable である Explicit メモリブロックは 523 個あります。
- Explicit メモリブロックの明示解放処理によって，Java ヒープの各領域のメモリサイズは変化していません。つまり，Java ヒープへのオブジェクトの移動はありませんでした。
- 明示解放処理の対象になったのは，次の Explicit メモリブロックです。
  - 確保済みのメモリサイズが 300K だった Explicit メモリブロック"REM2"(eid=3)
  - 確保済みのメモリサイズが 300K だった Explicit メモリブロック"BEM3"(eid=5)
  - 確保済みのメモリサイズが 400K だった Explicit メモリブロック"BEM1"(eid=7)

  なお，REM2 は，アプリケーションサーバの内部で利用されていた Explicit メモリブロックです。BEM3 と BEM1 は，アプリケーションが利用していた Explicit メモリブロックです。

(7)　ファイナライザによる Explicit メモリブロックの解放予約

ExplicitMemory インスタンスへの参照切れによって，ExplicitMemory クラスの finalize メソッドで対応する Explicit メモリブロックが解放予約された場合に出力されます。

(a)　出力の契機

ExplicitMemory.finalize()メソッドによって Explicit メモリブロックの解放が予約された場合です。

(b)　出力形式

```
[EVO]<ctime>[Finalized]["<EM_NAME>" eid=<EID>/<EM_TYPE>: <EM_TOTAL>]
```

(c)　出力項目

「(b)　出力形式」で示した各項目について説明します。

表 5-49　出力項目（ファイナライザによる Explicit メモリブロックの解放予約）

出力項目	出力内容	意味
<ctime>	<letters>	解放が予約された日時を示します。拡張 verbosegc 情報と同じ形式で出力されます。

出力項目	出力内容	意味
		-XX:+HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<EM_NAME>	<letters>	解放予約した Explicit メモリブロックの名称が出力されます。 Explicit メモリブロックの名称に多バイト文字が含まれている場合，出力内容は不定です（通常は文字化けして出力されます）。
<EID>	<const>	解放予約した Explicit メモリブロックの ID が出力されます。
<EM_TYPE>	B	解放予約した Explicit メモリブロックの種別が出力されます。 B は，Explicit メモリブロックの JavaVM 内部での種別を示します。アプリケーションサーバのバージョンが 08-50 より前の場合は，アプリケーションが利用している Explicit メモリブロックを示します。
<EM_TOTAL>	<const>K	解放を予約した Explicit メモリブロックによって確保済みサイズ（解放処理によって解放されるメモリサイズ）が出力されます。単位はキロバイトです。

(d) 出力例

出力例を示します。

```
[EVO]<Tue Jul 24 01:23:51 2007>[Finalized][”REM1” eid=3/R: 512K]
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 7 月 24 日(火)1 時 23 分 51 秒に Explicit メモリブロック"REM1"(eid=3)が解放予約されたことです。解放予約は，finalize()によって実行されました。

(8) 明示管理ヒープへの自動配置

指定したクラスが明示管理ヒープ上への自動配置に成功した場合，ログメッセージが出力されます。

(a) 出力の契機

指定したクラスが明示管理ヒープ上への自動配置に成功した場合です。

(b) 出力形式

```
[EVA]<ctime> creation in explicit memory is succeeded. [class=<CLASSNAME>]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-50　出力項目（明示管理ヒープへの自動配置）

出力項目	出力内容	意味
<ctime>	<letters>	指定したクラスが明示管理ヒープ上への自動配置に成功した日時を示します。拡張 VerboseGC 機能で出力しているものと同一の時刻形式で出力されます。HitachiOutputMilliTime オプションが設定されている場合は，ミリ秒単位まで出力されます。
<CLASSNAME>	<letters>	明示管理ヒープへの自動配置に成功したクラスの完全修飾クラス名が出力されます。

(d)　出力例

出力例を示します。

```
[EVA]<Tue Jul 24 01:23:51 2007> creation in explicit memory is succeeded. [class=jp.co.sampl
e.Main]
```

この出力例では次の内容が確認できます。

- 2007 年 7 月 24 日(火)1 時 23 分 51 秒に，明示管理ヒープ機能がクラス jp.co.sample.Main を明示管理ヒープ上への自動配置に成功した場合です。

5.11.5　出力レベルが debug の場合に出力される内容

ここでは，ログ出力レベルに debug を指定した場合に出力される内容をイベントごとに説明します。

補足

debug は，デバッグなどを実施する場合に，verbose で出力する内容よりもさらに詳細な情報が必要なときに使用します。debug で出力するログには，出力するために FullGC 実行時以上のオーバーヘッドが必要なログもあります。

(1)　Explicit メモリブロックの明示解放による Java ヒープへのオブジェクトの移動

Explicit メモリブロックを明示解放する場合に，Explicit メモリブロック内のオブジェクトが明示解放対象の Explicit メモリブロック以外から参照されているとき，そのオブジェクトは Java ヒープに移動します。このログでは，Java ヒープに移動したオブジェクトとその参照元のオブジェクトの情報が出力されます。

(a)　出力の契機

Explicit メモリブロックの明示解放処理で，Java ヒープへのオブジェクトの移動が発生した場合です。

(b) 出力形式

```
[EDO][eid=<EID>: Reference to <REFED_NAME>(<REFED_PTR>), total <R_SIZE>]
[EDO]   <REF_NAME>(<REF_PTR>)<REF_GEN>
```

注

明示解放対象の Explicit メモリブロック以外から参照されているオブジェクトごとに出力されます。

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-51 出力項目（Explicit メモリブロックの明示解放による Java ヒープへのオブジェクト移動）

出力項目	出力内容	意味
<EID>	<const>	Explicit メモリブロックの明示解放処理時に，明示解放対象の Explicit ヒープ以外から参照されているオブジェクトを含む Explicit メモリブロックの ID が出力されます。
<REFED_NAME>	<letters>	Explicit メモリブロックの明示解放処理時に，明示解放対象の Explicit ヒープ以外のオブジェクト (<REF_NAME>(<REF_PTR>)で示すオブジェクト)から参照されているオブジェクトの完全クラス名が出力されます。
<REFED_PTR>	<ptr>	<REFED_NAME>が示すオブジェクトの Java ヒープへの移動前のメモリアドレスが出力されます。
<R_SIZE>	<const>K	<REF_NAME>(<REF_PTR>)から参照されていることによって，Java ヒープに移動することになるオブジェクトの総サイズが出力されます。<REF_NAME>(<REF_PTR>)から間接参照している，明示解放対象 Explicit メモリブロック内のオブジェクトも含めた値になります※。単位はキロバイトです。
<REF_NAME>	<letters>|JVM	<REFED_NAME>(<REFED_PTR>)を参照しているオブジェクトの完全クラス名が出力されます。ただし，参照元がスタックや JavaVM 内部の場合は，"JVM"と出力されます。
<REF_PTR>	<ptr>	<REF_NAME>が示すオブジェクト，スタックまたは JavaVM 内部のメモリアドレスが出力されます。
<REF_GEN>	(eid=<EID>)|(DefNew)|(Tenured)|(JVM)	<REF_NAME>(<REF_PTR>)の属する領域または世代の名称が出力されます。Explicit メモリブロックの場合は Explicit メモリブロックの ID が出力されます。スタックや JavaVM 内部から参照されている場合は"JVM"と出力されます。

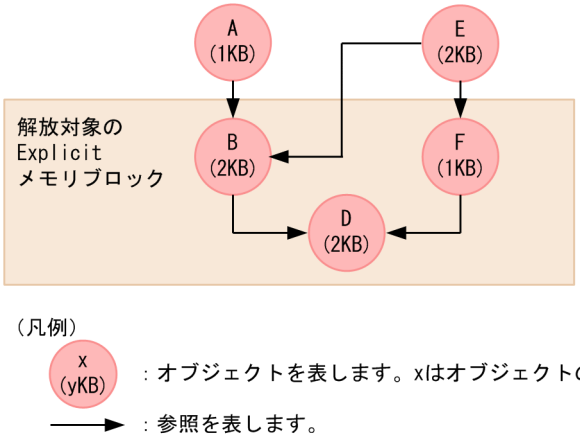
注※ 次に示す「補足」も参照してください。

補足

Java ヒープに移動するオブジェクトへの参照経路が複数ある場合について説明します。

ここでは，次の図を例にして説明します。

図 5-9　Java ヒープに移動するオブジェクトへの参照経路が複数ある例



（凡例）

x (yKB)　：オブジェクトを表します。xはオブジェクトの名称，yはサイズです。

→　：参照を表します。

Java ヒープに移動するオブジェクトが複数のオブジェクトから参照されている場合の出力

Java ヒープに移動するオブジェクトは，複数のオブジェクトから参照されていることがあります。図の場合，オブジェクト B に対して，A→B，E→B の 2 種類の参照経路があります。

この場合，ログには，A→B，E→B の参照についての情報が別々に出力されます。

複数のオブジェクトから間接参照されている場合の<R_SIZE>の算出

<R_SIZE>には，<REF_NAME>(<REF_PTR>)から間接参照している，明示解放対象 Explicit メモリブロック内のオブジェクトも含めた値が出力されます。ただし，Java ヒープに移動するオブジェクトが複数のオブジェクトから間接参照されている場合は，算出方法が異なります。

図の場合，オブジェクト D は，次の 3 種類の経路で間接参照されています。

- A→B→D
- E→B→D
- E→F→D

複数の経路で参照される場合，最初に算出される参照関係の<R_SIZE>にオブジェクトのサイズが計上されます。例えば，A→B，E→B，E→F の順で参照関係が出力される場合，オブジェクト D のサイズは，A→B の<R_SIZE>に計上されます。この場合の各参照関係の<R_SIZE>の出力は次のようになります。

- A→B

  オブジェクト B とオブジェクト D のサイズの合計で 4K になります。
- E→B

  オブジェクト B のサイズである 2K になります。
- E→F

  オブジェクト F のサイズである 1K になります。

(d)　出力例

出力例を示します。

```
[EDO][eid=5: Reference to java.lang.HashMap$Entry(0x1234568), total 125K]
[EDO]   java.util.HashMap$KeyIterator(0x1134428)(eid=1)
[EDO][eid=5: Reference to ClassA(0x1234580), total 19250K]
[EDO]   ClassV(0x1234468)(Tenured)
[EDO][eid=5: Reference to ClassZ(0x1234680), total 12K]
[EDO]   ClassU(0x1233468)(DefNew)
[EDO][eid=9: Reference to JP.co.Hitachi.soft.jvm.BBB(0x1034428), total 1250K]
[EDO]   JVM(0x23456780)(JVM)
```

この出力例では次の内容が確認できます。

- eid=5 の Explicit メモリブロック内にある java.lang.HashMap$Entry のオブジェクトは，明示解放対象ではない eid=1 の Explicit メモリブロック内の java.util.HashMap$KeyIterator のオブジェクトから参照されています。このため，125 キロバイトのオブジェクトが Java ヒープに移動しました。
- eid=5 の Explicit メモリブロック内にある ClassA のオブジェクトは，Tenured 領域内の ClassV のオブジェクトから参照されています。このため，19,250 キロバイトのオブジェクトが Java ヒープに移動しました。
- eid=5 の Explicit メモリブロック内にある ClassZ のオブジェクトは，New 領域内の ClassU のオブジェクトから参照されています。このため，12 キロバイトのオブジェクトが Java ヒープに移動しました。
- eid=9 の Explicit メモリブロック内にある JP.co.Hitachi.soft.jvm.BBB のオブジェクトは，スタックまたは JavaVM 内部から参照されています。このため，1,250 キロバイトのオブジェクトが Java ヒープに移動しました。

(2) Explicit メモリブロックの初期化（詳細情報出力）

新たに Explicit メモリブロックを初期化した場合の詳細な情報が出力されます。「5.11.4(1) Explicit メモリブロックの初期化」で示した出力内容に加えて，初期化処理を実行した Java プログラムのスタックトレースが出力されます。このログは，複数行にわたって，Java プログラムの実行と非同期に出力されます。このため，各行の間に別なログが出力されることがあります。ただし，1 行中にほかのログが出力されることはありません。

(a) 出力の契機

Explicit メモリブロックの初期化です。

(b) 出力形式

```
<Explicitメモリブロックの初期化の情報（verbose）>※[Thread: <TH_PTR>]
[EDO][Thread: <TH_PTR>] at <FRAME><SOURCE>
...
```

注※

出力項目については，「5.11.4(1) Explicit メモリブロックの初期化」を参照してください。

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-52 出力項目（Explicit メモリブロックの初期化（詳細情報出力））

出力項目	出力内容	意味
Explicit メモリブロックの初期化時の情報（vervose）		「5.11.4(1) Explicit メモリブロックの初期化」を参照してください。
<TH_PTR>	<ptr>	Explicit メモリブロックを初期化したスレッドのスレッド ID が出力されます。スレッドダンプに出力される tid と同一です。
<FRAME>	<letters>.<letters>	Explicit メモリブロック初期化時のスタックトレース中の 1 フレームが出力されます。完全クラス名とメソッド名が"."で区切られて出力されます。
<SOURCE>	(<letters>:<const>)| (Native Method)| (Unknown Source)	<FRAME>で出力されたメソッドが記述されているソースファイル名と，スタックトレースと一致する行番号が出力されます。ファイル名と行番号は":"で区切られて出力されます。 ネイティブメソッドの場合は，"(Native Method)"と出力されます。 ソースファイル名が取得できない場合は，"(Unknown Source)"と出力されます。

(d) 出力例

出力例を示します。

```
[EVO]<Tue Jul 24 01:23:51 2007>[Created][”BasicExplicitMemory-2” eid=2(0x1234568)/B][Thread:
 0x00035a60]
[EDO][Thread: 0x00035a60] at ExplicitMemory.registerExplicitMemory(Native Method)
[EDO][Thread: 0x00035a60] at BasicExplicitMemory.<init>(Unknown Source)
[EDO][Thread: 0x00035a60] at AllocTest.test(AllocTest.java:64)
[EDO][Thread: 0x00035a60] at java.lang.Thread.run(Thread.java:2312)
```

この出力例では次の内容が確認できます。

- 出力契機は 2007 年 7 月 24 日(火)1 時 23 分 51 秒に実行したメモリブロックの初期化です。Explicit メモリブロックの名称は"BasicExplicitMemory-2"です。Explicit メモリブロックの ID は eid=2 です。
- AllocTest.java の 64 行目で BasicExplicitMemory クラスのコンストラクタを実行して，Explicit メモリブロックの初期化を試みています。

(3) Explicit メモリブロックの自動解放処理の詳細（詳細情報出力）

Explicit メモリブロックの自動解放処理が起こった場合の詳細情報を出力します。「5.11.3(4) Explicit メモリブロックの自動解放処理」で示した出力内容に加えて，自動解放処理を実施した Explicit メモリブロックの EID 情報を出力します。Explicit メモリブロックの自動解放処理は，オブジェクトの移動が発生しない自動解放処理，多対 1 の自動解放処理，および 1 対 1 の自動解放処理が実行されることがあります。このため，オブジェクトの移動が発生しなかった EID，多対 1 の EID，および 1 対 1 の EID の情報を出力します。

(a) 出力の契機

Explicit メモリブロックの自動解放処理が起こった場合です。

(b) 出力形式

```
[EDO][migrate:(<EID_DEL>{,<EID_DEL>}*|)/(<EID_MBF>{,<EID_MBF>}*-><EID_MAF>|)/(<EID_MIG>{,<EI
D_MIG>}*|)]
```

(c) 出力項目

「(b) 出力形式」で示した各項目について説明します。

表 5-53 出力項目（Explicit メモリブロックの自動解放処理の詳細（詳細情報出力））

出力項目	出力内容	意味
<EID_DEL>	<const>	Explicit メモリブロックの自動解放処理によって解放される Explicit メモリブロックのうち，オブジェクトの移動が発生しなかった Explicit メモリブロックの EID を出力します。
<EID_MBF>	<const>	Explicit メモリブロックの自動解放処理によって解放される Explicit メモリブロックのうち，多対 1 の自動解放が起こった Explicit メモリブロックの自動解放予約される前の EID を出力します。
<EID_MAF>	<const>	Explicit メモリブロックの自動解放処理によって生成される Explicit メモリブロックのうち，多対 1 の自動解放によって生成された Explicit メモリブロックの EID を出力します。
<EID_MIG>	<const>	Explicit メモリブロックの自動解放処理によって解放される Explicit メモリブロックのうち，1 対 1 の自動解放が起こった Explicit メモリブロックの EID を出力します。

(d) 出力例

出力例を示します。

```
[EVS]<Tue Jul 14 02:31:22 2009>[EH: 256512K->256128K(256256K/1048576K), 0.1124626 secs][E/F/
D: 423/0/0][target:584K/384K/200K]¥
[cause:Migrate]
[EDO][migrate:()/(2,4,6,9->10)/(1,8)]
```

この出力例では次の内容が確認できます。

- 2009 年 7 月 14 日（火）2 時 31 分 22 秒に発生した，GC での Explicit メモリブロックの自動解放処理です。
- 自動解放処理が発生したことによって，Explicit ヒープの利用済みサイズが 256,512K から 256,128K に変化しています。

- 自動解放処理後の Explicit ヒープの確保済みサイズは 256,256K，最大サイズは 1,048,576K となっています。
- 自動解放処理に 0.1124626 秒掛かっています。
- 自動解放処理後のサブ状態が Enable である Explicit メモリブロック数は 423 です。
- Explicit ヒープの利用済みサイズ 584K に対して自動解放処理を実行しました。
- ID（2，4，6，9）の Explicit メモリブロックが ID（10）の Explicit メモリブロックに移動しています。
- ID（1，8）の Explicit メモリブロックが，同じ ID（1，8）に移動しています。

6 トラブルシューティングの手順

ここでは，アプリケーションサーバのトラブルシューティングの手順について説明します。

6.1　この章の構成

この章では，アプリケーションサーバで発生する主なトラブルと，そのトラブルシュートについて説明します。

この章の構成を次の表に示します。

表 6-1　この章の構成（トラブルシューティングの手順）

分類	タイトル	参照先
解説	主なトラブルの一覧	6.2
	ログを出力するプロセス	6.3
	構築時のトラブルシュート	6.4.1
	稼働（運用）時のトラブルシュート	6.4.2
	サーバ管理コマンドでのトラブルシュート	6.4.3
	運用時のトラブルシュートの例	6.5

6.2　主なトラブルの一覧

トラブルが発生したタイミングごとに，主なトラブルの原因や対処に必要なログの調査個所について説明します。

ここでは，次に示すタイミングごとにトラブルの主な原因や対処などについて説明します。

- インストール時
- サーバ構築時
- サーバ起動時
- アプリケーションの開始時
- 稼働（運用）時
- サーバ／アプリケーションの保守（メンテナンス）時

それぞれについて説明します。

6.2.1　インストール時に発生する主なトラブル

インストール時に発生する主なトラブルについて次の表に示します。

表 6-2　インストール時に発生する主なトラブル一覧

トラブルを知らせるツール	現象	主な影響	主な原因	調査個所
インストーラ	エラーメッセージ出力（ダイアログまたはコマンドプロンプト）	インストールが中断します。場合によってはアンインストールが必要です。	環境（OS，ディスクなど）に問題があります。	• ダイアログ • install.log

（凡例）　－：該当なし。

参考

インストールおよびアンインストールに関する注意については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「付録Ｉ　インストールおよびアンインストールするときの注意事項」を参照してください。

6.2.2　サーバ構築時に発生する主なトラブル

サーバ構築時に発生する主なトラブルについて説明します。

表 6-3　サーバ構築時に発生する主なトラブル一覧

トラブルを知らせるツール	現象	主な影響	主な原因	調査個所	参照先
セットアップウィザード	エラーメッセージ出力（コマンドプロンプト）	構築作業が中断します。場合によってはアンセットアップが必要です。	次に示す原因が考えられます。 • 環境（OS，ネットワーク，メモリ，ディスクなど）に問題がある • 操作誤り • 設定誤り	• コマンドエラー • セットアップウィザードログ • Manager ログ • J2EE サーバログ • Web サーバログ • OS	6.4.1
SmartComposer	エラーメッセージ出力（コマンドプロンプト）			• コマンドエラー • Manager ログ • J2EE サーバログ • Web サーバログ • OS	
運用管理ポータル	エラーメッセージ出力（GUI）			• ログ表示 • Manager ログ • J2EE サーバログ • Web サーバログ • OS	

6.2.3　サーバ起動時に発生する主なトラブル

サーバ起動時に発生する主なトラブルについて説明します。

表 6-4　サーバ起動時に発生する主なトラブル一覧

トラブルを知らせるツール	現象	主な影響	主な原因	調査個所	参照先
SmartComposer	エラーメッセージ出力（コマンドプロンプト）	デプロイ作業が中断します。	次に示す原因が考えられます。 • 環境（OS，ネットワーク，メモリ，ディスクなど）に問題がある • 操作誤り • 設定誤り	• コマンドエラー • Manager ログ • J2EE サーバログ • Web サーバログ • OS	6.4.1

トラブルを知らせるツール	現象	主な影響	主な原因	調査個所	参照先
運用管理ポータル	エラーメッセージ出力（GUI）			• ログ表示 • Manager ログ • J2EE サーバログ • Web サーバログ • OS	

6.2.4　アプリケーションの開始時に発生する主なトラブル

アプリケーションの開始時に発生する主なトラブルについて説明します。

表 6-5　アプリケーションの開始時に発生する主なトラブル一覧

トラブルを知らせるツール	現象	主な影響	主な原因	調査個所	参照先
SmartComposer（互換用）	エラーメッセージ出力（コマンドプロンプト）	デプロイ作業が中断します。	次に示す原因が考えられます。 • 環境（OS，ネットワーク，メモリ，ディスクなど）に問題がある • 操作誤り • 設定誤り • 作成したアプリケーションに問題がある • アプリケーションとアプリケーションサーバとの間に問題がある	• コマンドエラー • Manager ログ • J2EE サーバログ • Web サーバログ • OS	6.4.1
運用管理ポータル	エラーメッセージ出力（GUI）			• ログ表示 • Manager ログ • admin ログ • J2EE サーバログ • Web サーバログ • OS	
サーバ管理コマンド※	エラーメッセージ出力（コマンドプロンプト）			• コマンドエラー • admin ログ • J2EE サーバログ • アプリケーション	6.4.3

注※

サーバ管理コマンドを使用している場合は，トラブルの対処が異なります。表に示す参照先の対処を確認してください。

6.2.5　稼働（運用）時に発生する主なトラブル

稼働（運用）時に発生する主なトラブルについて説明します。

表 6-6　稼働時（運用時）に発生する主なトラブル一覧

トラブルを知らせるツール	現象	主な影響	主な原因	調査個所	参照先
運用管理ポータル	エラーメッセージ出力（GUI）	業務が継続できません。	次に示す原因が考えられます。 • 環境（OS，ネットワーク，メモリ，ディスクなど）に問題がある • サーバ内部のリソース枯渇（OOM など） • 連携システム（DB など）に問題がある • アプリケーションの上長な処理 • HA 切り替え	• ログ表示 • Manager ログ • J2EE サーバログ • Web サーバログ • OS • JavaVM，コンテナリソース • PRF トレース • スレッドダンプ • アプリケーション	6.4.2
（クライアント）※	• 業務のエラー出力 • ブラウザ固有のエラー（サーバが見つからない，404，500，503 など） • 無応答 • 応答遅延（性能劣化）	業務が継続できないことがあります。		• ログ表示 • Manager ログ • J2EE サーバログ • Web サーバログ • OS • JavaVM，コンテナリソース • PRF トレース • スレッドダンプ • アプリケーション	

注※　サービスを利用しているクライアント側の現象として現れます。

6.2.6　サーバ／アプリケーションの保守（メンテナンス）時に発生する主なトラブル

サーバ／アプリケーションの保守（メンテナンス）時に発生する主なトラブルについて説明します。

表 6-7　サーバ／アプリケーションの保守（メンテナンス）時に発生する主なトラブル一覧

トラブルを知らせるツール	現象	主な影響	主な原因	調査個所	参照先
SmartComposer	エラーメッセージ出力（コマンドプロンプト）	• サーバの設定変更ができません • アプリケーションの更新ができません	次に示す原因が考えられます。 • 環境（OS，ネットワーク，メモリ，ディスクなど）に問題がある • 操作誤り • 設定誤り • 作成したアプリケーションに問題がある • アプリケーションとアプリケーションサーバとの間に問題がある	• コマンドエラー • Manager ログ • J2EE サーバ ログ • Web サーバ ログ • OS	6.4.1
運用管理ポータル	エラーメッセージ出力（GUI）			• ログ表示 • Manager ログ • admin ログ • J2EE サーバ ログ • Web サーバ ログ • OS	
サーバ管理コマンド※	エラーメッセージ出力（コマンドプロンプト）			• コマンドエラー • admin ログ • J2EE サーバ ログ • OS • アプリケーション	6.4.3

注※

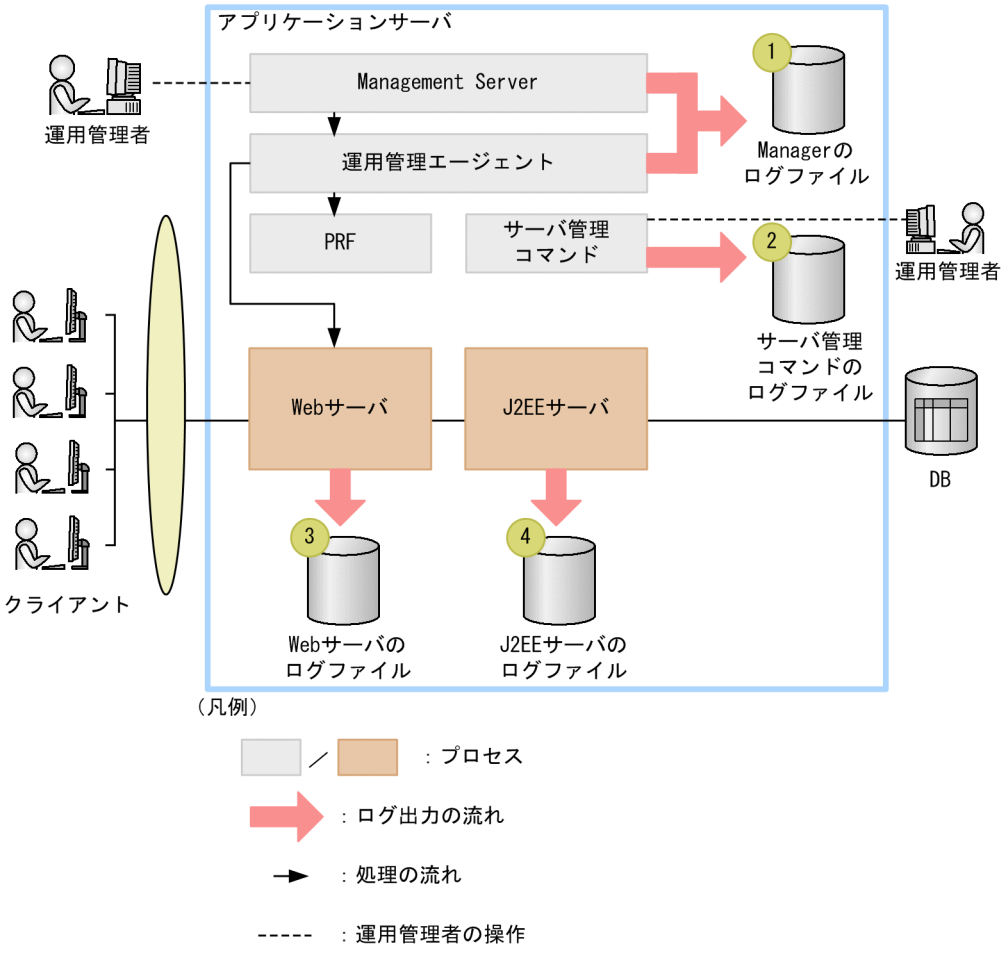
サーバ管理コマンドを使用している場合は，トラブルの対処が異なります。表に示す参照先の対処を確認してください。

6.3　ログを出力するプロセス

アプリケーションサーバでは，複数のログを出力します。これらのログを参照して，トラブルの原因を特定し対処します。

ここでは，トラブルシュートで主に使用するログと，そのログを出力するプロセスについて説明します。トラブルシュートで主に使用するログと出力するプロセスを次の図に示します。

図 6-1　ログを出力するプロセス



注　ここで示すログファイル以外に，セットアップウィザードのログも調査します。

アプリケーションサーバのトラブルシュートで主に使用するログは 4 種類です。それぞれについて次の表に示します。

表 6-8　ログの種類と出力内容

図中の番号	ログの種類	出力内容
1	Manager のログ	Management Server または運用管理エージェントが出力するログです。Manager 機能を使用した構築・運用・保守で出力されるログです。
2	サーバ管理コマンドのログ	サーバ管理コマンドの操作で出力されるログです。

図中の番号	ログの種類	出力内容
3	Web サーバのログ	HTTP Server から出力されるログです。
4	J2EE サーバのログ	J2EE サーバから出力されるログです。

トラブルシュートでは，これらのログを確認して対処します。以降では，使用するツールごとに具体的なトラブルシュートの流れについて説明します。

6.4　トラブルシュートの流れ

6.4.1　構築時のトラブルシュート

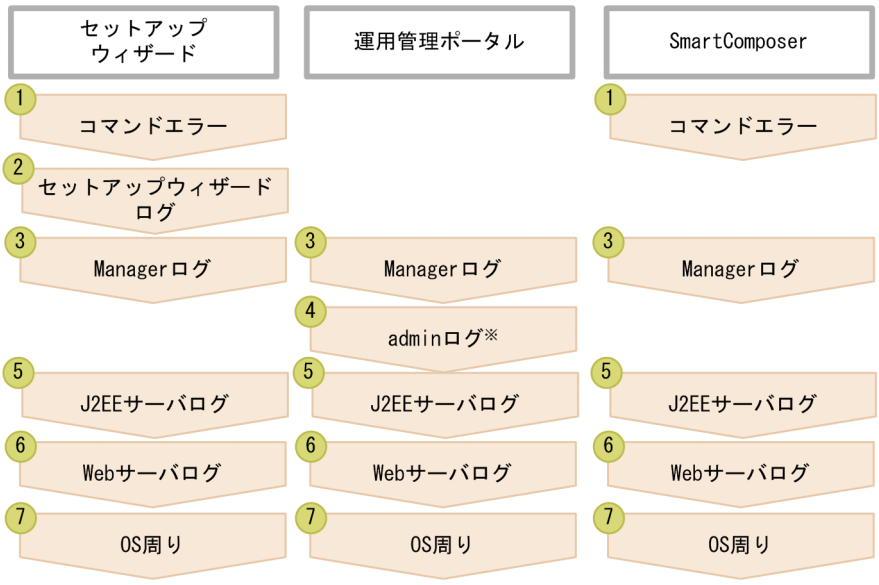
ここでは，構築時に発生するトラブルの対処について説明します。

構築時とは，次に示すタイミングを指します。

- サーバの構築時
- サーバの起動時
- アプリケーションの開始時
- サーバ／アプリケーションの保守（メンテナンス）時

構築時の調査の流れを示します。

図 6-2　使用しているツールごとの調査の流れ（構築時）



注※　アプリケーションの開始時にエラーが発生した場合に必要な調査です。

使用しているツールごとに，図で示した流れに従って，必要な調査を実施します。以降の説明では，それぞれの調査の詳細について説明します。

参考

Manager ログの調査では，運用管理ポータルを利用します。運用管理ポータルについては，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」を参照してください。

(1)　コマンドエラーの調査

ポイント

次のツールを使用している場合に必要な調査です。

- セットアップウィザード
- SmartComposer

セットアップウィザードや SmartComposer の画面にエラーが表示されます。表示されたエラーの内容を確認して対処します。エラーメッセージの対処については，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」を参照してください。

(2)　セットアップウィザードログの調査

ポイント

次のツールを使用している場合に必要な調査です。

- セットアップウィザード

セットアップウィザードのログを確認します。セットアップウィザードのログの出力先は，setup.cfg の setup.log.dir で指定した出力先です。setup.cfg については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.18　setup.cfg（セットアップウィザード用設定ファイル)」を参照してください。

(3)　ログの表示，および Manager ログの調査

ポイント

次のツールを使用している場合に必要な調査です。

- セットアップウィザード
- 運用管理ポータル
- SmartComposer

運用管理ポータルを使って，Manager で出力されたログを確認します。ログの表示方法を次に示します。

1. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックします。
   ツリーペインで次のどちらかの操作をします。

J2EE サーバの場合

［論理 J2EE サーバ］－［J2EE サーバ］－［< J2EE サーバ名>］－［アプリケーション］をクリックします。

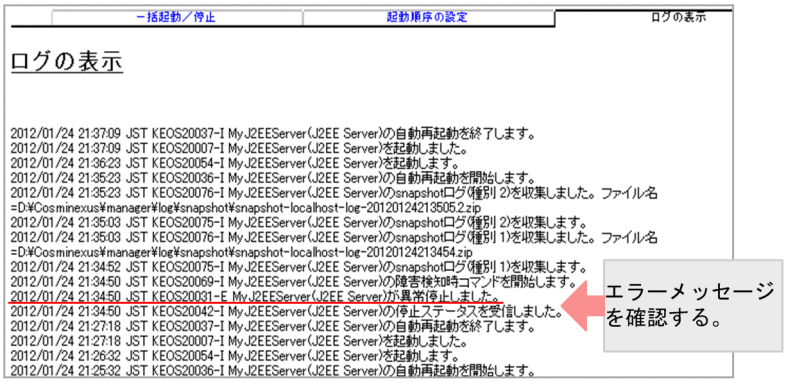
J2EE サーバクラスタの場合

［論理 J2EE サーバ］－［J2EE サーバクラスタ］－［< J2EE サーバクラスタ名>］－［アプリケーション］をクリックします。または，［論理 J2EE サーバ］－［J2EE サーバクラスタ］－［< J2EE サーバクラスタ名>］－［メンバ］－［< J2EE サーバ名>］－［アプリケーション］をクリックします。

2. ［ログの表示］タブをクリックします。

運用管理ポータルの画面でログを表示した場合の画面を次に示します。

図 6-3　運用管理ポータルの画面でログを表示した場合の出力例



［ログの表示］タブでログを表示させたあと，システムの障害が判明したタイミングに近い時刻に出力されたメッセージを探します。そして，エラーメッセージ（メッセージ ID が E で終わっているもの）を特定し，障害が発生した論理サーバの種類と名前を確認してください。

(4)　admin ログの調査

ポイント

次のツールを使用している場合に必要な調査です。

- 運用管理ポータル

なお，この調査はアプリケーションの開始時にエラーが発生した場合に必要な調査です。

サーバ管理コマンドのログを確認します。

サーバ管理コマンドのログには，メッセージログ，例外ログ，および保守用ログの 3 種類があります。これらのログには，サーバ管理コマンドの稼働状態，アプリケーション中で出力される標準出力，および標準エラー出力の情報，保守員が Component Container の障害解析用に使用する情報などが出力されます。

ログファイルのデフォルトの出力先や，ログファイル名については，「4.3.1　Component Container のログの取得」のサーバ管理コマンドのログに関する説明を参照してください。

(5)　J2EE サーバログの調査

ポイント

次のツールを使用している場合に必要な調査です。

- セットアップウィザード
- 運用管理ポータル
- SmartComposer

J2EE サーバで出力されたログファイルの内容を確認します。確認するログファイルの一覧を次に示します。

表 6-9　採取するログファイルの一覧

ログファイル名※	ログの説明	確認方法
cjmessage?.log	J2EE サーバ稼働時のログ	Manager でエラーメッセージが出力された時刻のエラーメッセージ ID（メッセージ種別が E のもの）の内容を確認します。
cjexception?.log	J2EE サーバ稼働時の例外情報ログ	
web_servlet?.log	Web サーブレットログ（JSP／Servlet でのメッセージの出力）	
user_out?.log	ユーザ出力ログ（アプリケーションでのメッセージの標準出力）	
user_err?.log	ユーザエラーログ（アプリケーションでのメッセージの標準エラー出力）	
javalog?.log	JavaVM の保守情報および GC 情報のログ	
hs_err?	JavaVM のエラーリポートファイル	保守員に送付してください。

注※　「?」はログの面数を示します。

なお，ログファイルのデフォルトの出力先については，「4.3.1　Component Container のログの取得」を参照してください。

hs_err の取得方法については，「4.11　JavaVM 出力メッセージログ（標準出力またはエラーリポートファイル）」を参照してください。

(6) Web サーバログの調査

ポイント

次のツールを使用している場合に必要な調査です。

- セットアップウィザード
- 運用管理ポータル
- SmartComposer

Web サーバで出力されたログファイルを確認します。確認するログファイルの一覧を次に示します。

表 6-10 採取するログファイルの一覧

ログファイル名※	ログの説明	確認方法
processConsole?.log	Manager のコンソールログ（Web サーバ起動時のエラー情報）	Manager でエラーメッセージが出力された時刻のエラーメッセージ ID（メッセージ種別が E のもの）の内容を確認します。
error?	Web サーバのエラーログ	Manager でエラーメッセージが出力された時刻のエラーメッセージ（「emerg」，「alert」，「crit」のもの）の内容を確認します。

注※ 「?」はログの面数を示します。

なお，Manager のコンソールログのデフォルトの出力先については，「4.11 JavaVM 出力メッセージログ（標準出力またはエラーリポートファイル）」を参照してください。

Web サーバのエラーログのデフォルトの出力先については，マニュアル「HTTP Server」の「6.2.4 E，F，G，H，I で始まるディレクティブ」を参照してください。

(7) OS 周りの調査

ポイント

次のツールを使用している場合に必要な調査です。

- セットアップウィザード
- 運用管理ポータル
- SmartComposer

アプリケーションサーバをインストールしているマシンのメモリサイズや稼働状態を確認してください。

6.4.2　稼働（運用）時のトラブルシュート

ここでは，稼働（運用）時に発生するトラブルの対処について説明します。

稼働時とは，サーバ起動完了直後／サーバ稼働中のタイミングです。

使用しているツールごとに稼働時の調査の流れを示します。

図 6-4　使用しているツールごとの調査の流れ（稼働時）

注※　サービスを利用しているクライアント側の現象として現れます。

使用しているツールごとに，図で示した流れに従って，必要な調査を実施します。以降の説明では，それぞれの調査の詳細について説明します。

参考

Manager ログの調査では，運用管理ポータルを利用します。運用管理ポータルについては，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」を参照してください。

(1)　ログの表示，および Manager ログの調査

運用管理ポータルを使って，Manager で出力されたログを確認します。ログの表示，および Manager ログの調査については，「6.4.1(3)　ログの表示，および Manager ログの調査」を参照してください。

(2)　J2EE サーバログの調査

J2EE サーバで出力されたログファイルを確認します。J2EE サーバログの調査については，「6.4.1(5)　J2EE サーバログの調査」を参照してください。

(3) Web サーバログの調査

Web サーバで出力されたログファイルを確認します。Web サーバログの調査については，「6.4.1(6) Web サーバログの調査」を参照してください。

(4) OS 周りの調査

アプリケーションサーバをインストールしているマシンのメモリサイズや稼働状態を確認してください。

(5) JavaVM，コンテナリソースの調査

JavaVM に問題が発生していないか（OutOfMemoryError が発生していないか），また，リソースに問題が発生していないか確認します。JavaVM に問題が発生した場合の対処については，「2.5.4 JavaVM が異常終了した場合」を参照してください。

(6) PRF トレースの調査

性能解析トレースを調査して，ボトルネック個所がないか，処理が滞っている個所がないか確認します。性能解析トレースについては，「7. 性能解析トレースを使用した性能解析」を参照してください。

(7) スレッドダンプの調査

スレッドダンプを調査して，デッドロックしている処理がないか，Java プログラムに問題がないか確認します。スレッドダンプに出力される情報については，「5.5 JavaVM のスレッドダンプ」を参照してください。

(8) アプリケーションの調査

問題が発生したと考えられるアプリケーションの内容を調査します。アプリケーションの作成元に確認を依頼してください。

6.4.3 サーバ管理コマンドのトラブルシュート

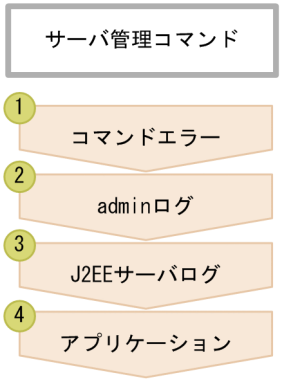
ここでは，サーバ管理コマンドを使用している場合のトラブルの対処について説明します。

ここで説明するトラブルの対処を次に示します。

- アプリケーションの開始時
- サーバ／アプリケーションの保守（メンテナンス）時

サーバ管理コマンドを使用した調査の流れを示します。

図 6-5　サーバ管理コマンドを使用した調査の流れ



それぞれの作業の詳細を示します。

(1)　コマンドエラーの調査

コマンドプロンプトにエラーが表示されます。表示されたエラーの内容を確認して対処します。

(2)　admin ログの調査

サーバ管理コマンドのログを確認します。ログファイルのデフォルトの出力先については，「4.3.1 Component Container のログの取得」のサーバ管理コマンドのログに関する説明を参照してください。

(3)　J2EE サーバログの調査

J2EE サーバで出力されたログファイルを確認します。J2EE サーバログの調査については，「6.4.1(5) J2EE サーバログの調査」を参照してください。

(4)　アプリケーションの調査

問題が発生したと考えられるアプリケーションの内容を調査します。アプリケーションの作成元に確認を依頼してください。

6.5　運用時のトラブルシュートの例

ここでは，運用時のトラブルシュートの例として，次に示すトラブルが発生した場合のトラブルシュートの流れを示します。

- プロセスダウン
- 応答遅延

注意事項

この節では，アプリケーションサーバのデフォルトの設定のパスを記載しています。また，使用する OS は Windows です。

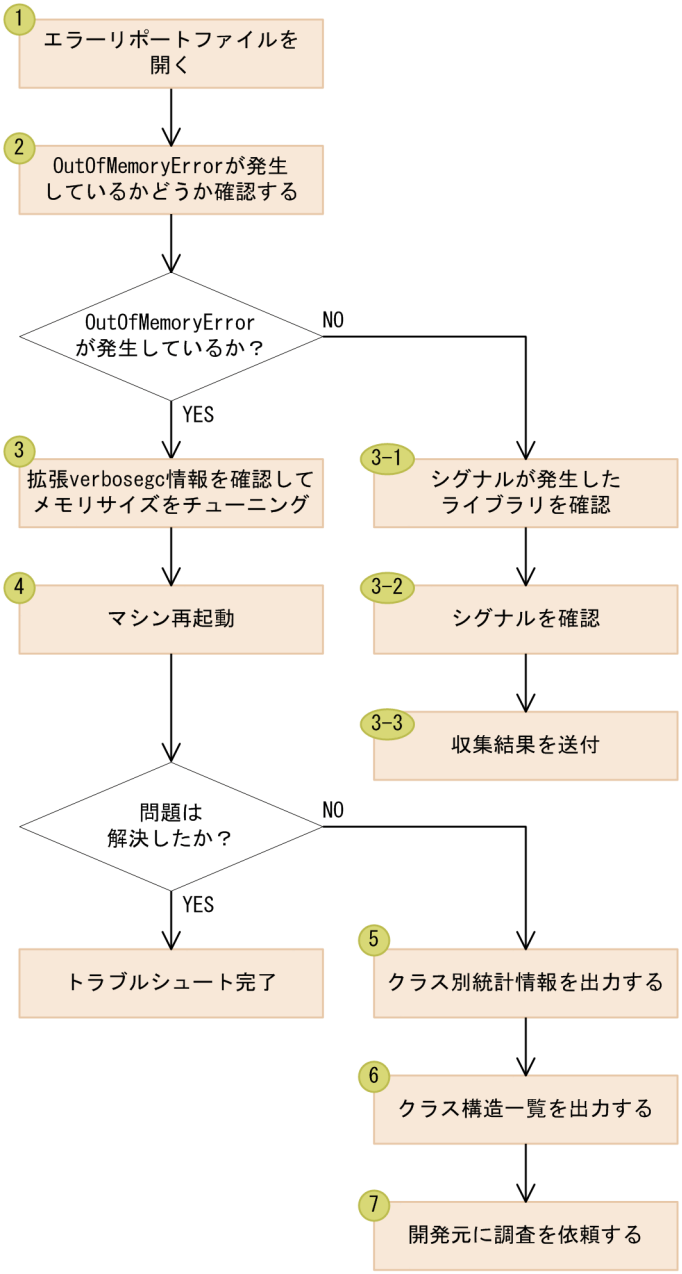
6.5.1　プロセスダウンのトラブルシュート

プロセスダウンのトラブルシュートについて説明します。

(1)　プロセスダウン発生時の対処フロー

プロセスダウンが発生した場合の，トラブルシュートの流れを次に示します。

図 6-6　プロセスダウン発生時の対処フロー



フローの詳細で示した処理の詳細は(2)で説明します。

(2)　プロセスダウン発生時の対処の流れ

プロセスダウンのフローの内容に沿って，それぞれの作業について説明します。

1. エラーリポートファイル（hs_err_pid<プロセス ID>.log）を開く

   エラーが発生した時刻近辺に出力されたエラーリポートファイルを開き，Manager のログと突き合わせます。

   - エラーリポートファイルの出力先および出力ファイル名

C:¥Program Files¥Hitachi¥Cosminexus¥CC¥server¥public¥ejb¥<サーバ名称>¥hs_err_pid<サーバプロセス ID>.log

2. OutOfMemoryError が発生しているかどうか確認する

OutOfMemoryError が発生している場合，「java.lang.OutOfMemoryError occurred.」が表示されています。

OutOfMemoryError が発生している場合，および OutOfMemoryError が発生していない場合について，それぞれエラーリポートファイルの出力例と対処を次に示します。

OutOfMemoryError が発生している場合

出力例

```
   :
#
# java.lang.OutOfMemoryError occurred.
# JavaVM aborted because of specified -XX:+HitachiOutOfMemoryAbort options.
# Please check Javacorefile:D:¥Cosminexus¥CC¥server¥public¥ejb¥MyJ2EEServer¥javacore21
00.120124144835.txt
#
```

背景色付きの太字で示した個所が OutOfMemoryError が発生している場合に出力される文字列です。

対処

エラーリポートファイルを調査する必要があります。手順 3.へ進んでください。

OutOfMemoryError が発生していない場合

出力例

```
   :
#
# A fatal error has been detected by the Java Runtime Environment
#
#  EXCEPTION_ACCESS_VIOLATION (0xc0000005) at pc=0x0000000008303b00, pid=1356, tid=260
4
#
# Java VM: Java HotSpot(TM) 64-Bit Server VM (25.20-b23-CDK0970-20150127 mixed mode wi
ndows-amd64)
# Problematic frame:
# V  [C:¥Program Files¥Hitachi¥Cosminexus¥jdk¥jre¥bin¥server¥jvm.dll+0x303b00]
#
```

対処

次のような調査が必要です。手順に従って調査してください。なお，手順の番号は図 6-6 と対応しています。

3-1　シグナルが発生したライブラリを確認する

出力例の赤字部分を確認し，ダウンしたライブラリ名，およびダウンしたネイティブ関数を控えてください。

出力例の場合，ライブラリ名は「NativeCrash.dll」，ダウンしたネイティブ関数は「(null) +0x77C785BA」となります。

3-2　シグナルを確認する

「An unexpected error has been detected by HotSpot Virtual Machine:」で続く個所がシグナル 6（SIGABRT，SIGIOT）でダウンしている場合，上位の abort 関数でアボートしていることから呼び出し先のライブラリを調査する必要があります。

上記の例はアボートしていません。

この例では，Oracle のサポートに詳細な調査を依頼します。

3-3　収集結果を問い合わせ窓口に送付する

ひかえておいたライブラリ，およびシグナルの内容をモジュール開発元，またはご購入契約に基づくお問い合わせ窓口に送付して調査を依頼してください。

3. 拡張 verbosegc 情報を確認してメモリサイズをチューニングする

拡張 verbosegc 情報を確認して，Java ヒープのメモリサイズを調整します。

設定済みの-Xmx オプションの値を目安として 1.5 倍～2 倍にしてください。ただし，ハード搭載メモリサイズを超えないように注意してください。

4. マシンを再起動する

マシンを再起動して，問題なく動作できるか確認します。

ポイント

マシンを再起動し，問題なく動作した場合は，トラブルシュートは完了です。

マシンを再起動してもうまく動作しない場合は，手順 5.へ進んでください。

5. クラス別統計情報を出力する

JavaVM の GC のログ（javalog[n].log），または稼働情報ログ（HJVMStats_<YYYYMMDDhhmm><TZ>.csv）を確認し，メモリが 10MB～20MB 増加するごとに jheapprof コマンドを実行してクラス別統計情報（txt ファイル）を出力します。

メモリが 100MB 以上増加するまで繰り返してください。

コマンドの実行間隔が短い場合，リークしているメモリサイズも小さいためメモリ増加しているクラスを探すのが難しくなります。

実行間隔を長くしメモリの増分を大きくすることで，リークしているクラスを目立たせます。

実行例

```
% jheapprof -p 2463
```

注意

実行例の「2463」はプロセス ID になります。コマンドを実行する場合は問題となっているプロセス ID を指定してください。

6. クラス構造一覧を出力する

前の手順でピックアップしたクラスをメンバに持つクラスの構造を一覧に出力し，調査対象のクラスリストを作成し，調査対象をリストアップします。

実行例

% jheapprof -class org.apache.catalina.loader.WebappClassLoder -p 2463

注意

実行例の「2463」はプロセス ID になります。コマンドを実行する場合は問題となっているクラスを指定してください。

実行例の「org.apache.catalina.loader.WebappClassLoder」はクラス名になります。
Metaspace 領域不足による OutOfMemoryError はこのクラスを使って調べます。

7. 開発元に調査を依頼する

前回の手順で収集した情報を開発元へ送付して，調査を依頼してください。

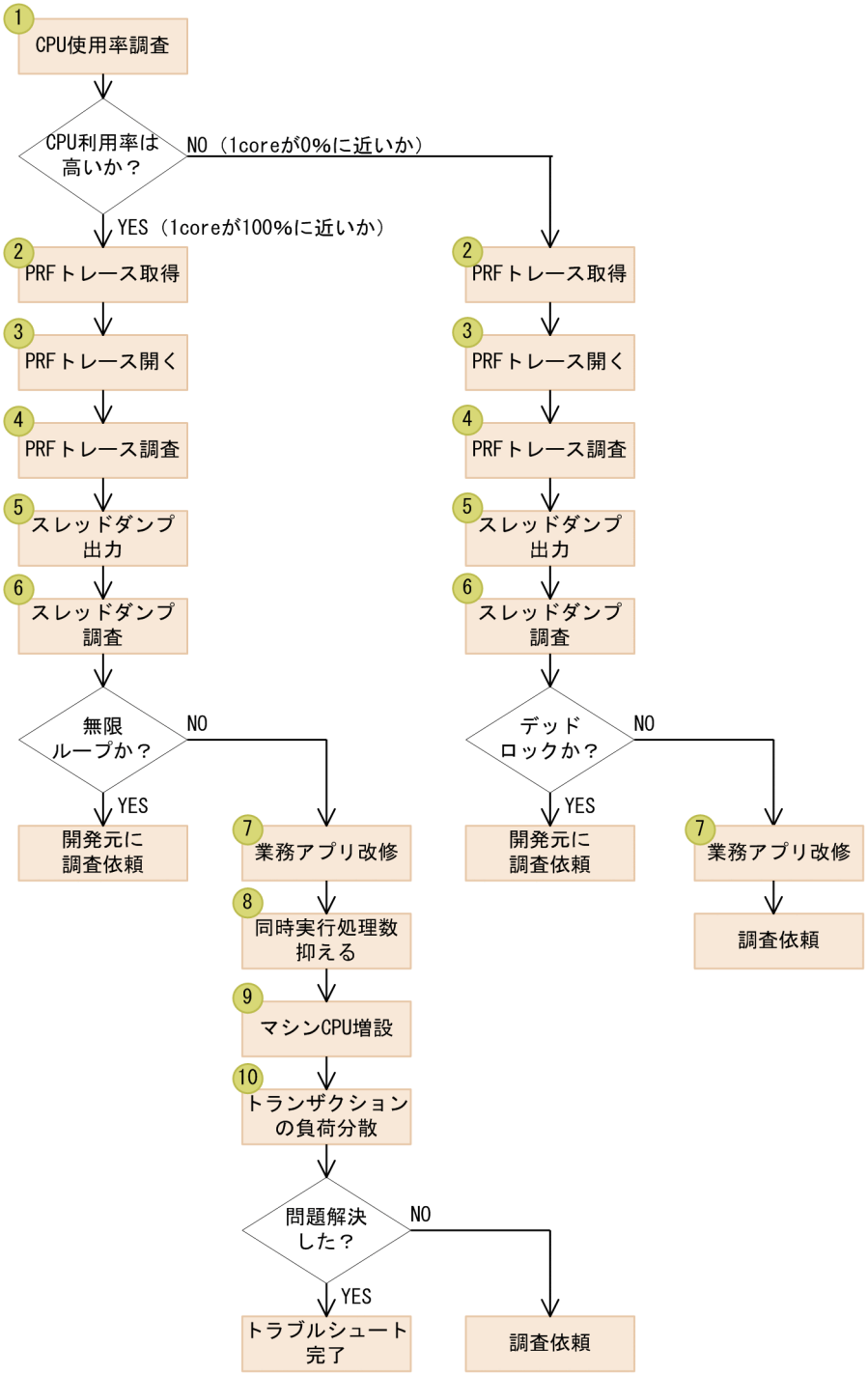
6.5.2 応答遅延のトラブルシュート

応答遅延のトラブルシュートについて説明します。

(1) 応答遅延発生時の対処のフロー

応答遅延が発生した場合の，トラブルシュートの流れを次に示します。

図 6-7　応答遅延発生時の対処フロー



フローの詳細で示した処理の詳細は(2)で説明します。

(2)　応答遅延発生時の対処の流れ

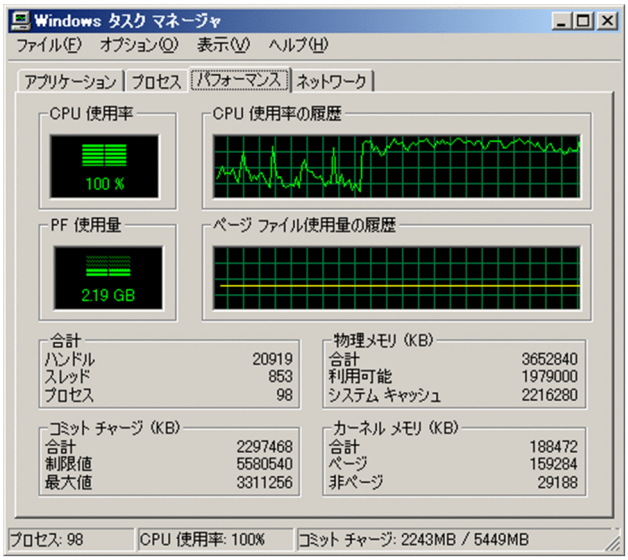
応答遅延のフローの内容に沿って，それぞれの作業について説明します。

1. CPU の利用率を調査する

該当するプロセスの CPU 利用率を調査します。

タスクマネージャで CPU 利用率を表示した例を次に示します。

図 6-8　CPU 利用率



ポイント

1core が 100%に近い場合

無限ループや再帰呼び出しに陥っているケースが考えられます。CPU ネックが原因と考えられます。手順 2.へ進んで調査を進めてください。

1core が 0%に近い場合

バックプロセスから応答が返ってきていないことが原因による，バックプロセス無応答やデッドロックが考えられます。手順 2.へ進んで調査を進めてください。

2. PRF トレースを取得する

mngsvrutil コマンドを実行すると，PRF トレースを出力できます。

実行例

```
mngsvrutil -m 123.45.67.89 -u admin2 collect allPrfTraces
```

3. PRF トレースを開く

PRF トレースを開きます。

出力先

C:¥Program Files¥Hitachi¥Cosminexus¥manager¥log¥prf

ファイル名

トレース情報の収集対象別に次に示すファイル名で出力されます。

なお，<日時>には，PRF トレースを収集した日時が表示されます。

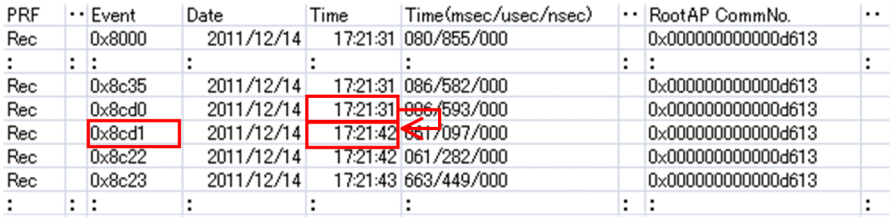
パフォーマンストレーサの種類	ファイル名
運用管理ドメイン内のホスト上で動作しているすべてのパフォーマンストレーサ	<運用管理ドメイン名>－<日時>.zip
特定のホスト上で動作しているすべてのパフォーマンストレーサ	<ホスト名>－<日時>.zip
特定のパフォーマンストレーサ	<論理サーバ名>－<日時>.zip

4. PRF トレースを調査する

PRF トレースの Time 列を確認して，長時間を要している処理を探します。

PRF トレースは，プロセスにわたりイベントを出力するトレース情報で性能解析・障害解析に有効な資料です。

図 6-9　PRF トレースの出力例



PRF	Event	Date	Time	Time(msec/usec/nsec)	RootAP CommNo.
Rec	0x8000	2011/12/14	17:21:31	080/855/000	0x000000000000d613
:	:	:	:	:	:
Rec	0x8c35	2011/12/14	17:21:31	086/582/000	0x000000000000d613
Rec	0x8cd0	2011/12/14	17:21:31	086/593/000	0x000000000000d613
Rec	0x8cd1	2011/12/14	17:21:42	617/097/000	0x000000000000d613
Rec	0x8c22	2011/12/14	17:21:42	061/282/000	0x000000000000d613
Rec	0x8c23	2011/12/14	17:21:43	663/449/000	0x000000000000d613
:	:	:	:	:	:

例の場合，SQL 発行後 11 分の空白があります。さらに SQL の実行は終了していません。したがって，SQL 実行中に DB に何らかの問題が発生したものと考えられます。

なお，PRF トレースは表計算ソフトで表示すると確認しやすくなります。

5. スレッドダンプを出力する

mngsvrutil コマンドを実行すると，スレッドダンプを出力できます。

実行例

```
mngsvrutil -m 123.45.67.89 -u admin2 dump server
```

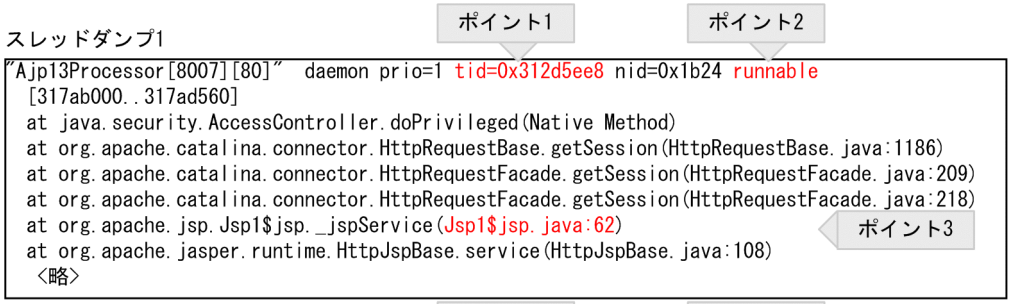
6. スレッドダンプを調査する

無限ループの場合

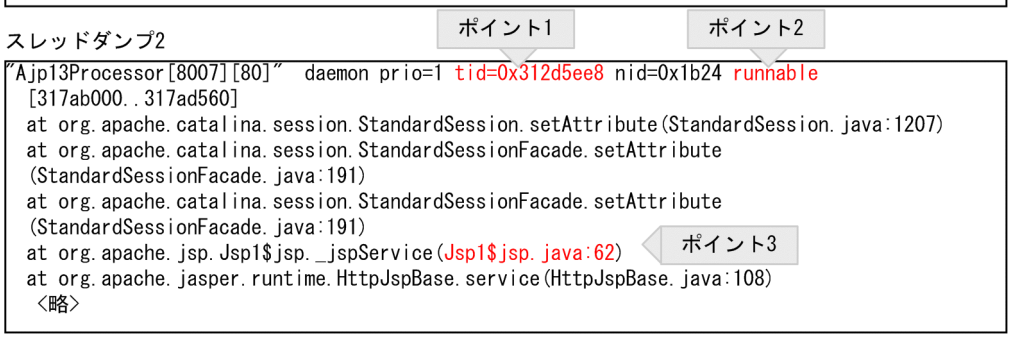
無限ループの場合のスレッドダンプの出力例，および調査のポイントについて説明します。

図 6-10　スレッドダンプ（無限ループ）の出力例

スレッドダンプ1



スレッドダンプ2



スレッドダンプ3

スレッドダンプを複数回出力して，時系列で観察し，それぞれのスレッドダンプで tid の同じスレッドのスタックトレースを比較調査します。

ポイント 1

スレッド属性が runnable の場合，このスレッドは実行可能状態にあります。このスレッドが CPU 利用率の増加に関与しています（waiting for monitor entry の場合は実行可能状態ではないので，CPU 利用率を増加させません）。

ポイント 2

複数のスレッドダンプファイルで，同一の tid のスレッド属性がすべて runnable です。

長時間にわたって実行中の可能性があります。

ポイント 3

同一メソッド内の特定の行が，繰り返し実行されているような場合，無限ループの疑いがあります。

ポイント

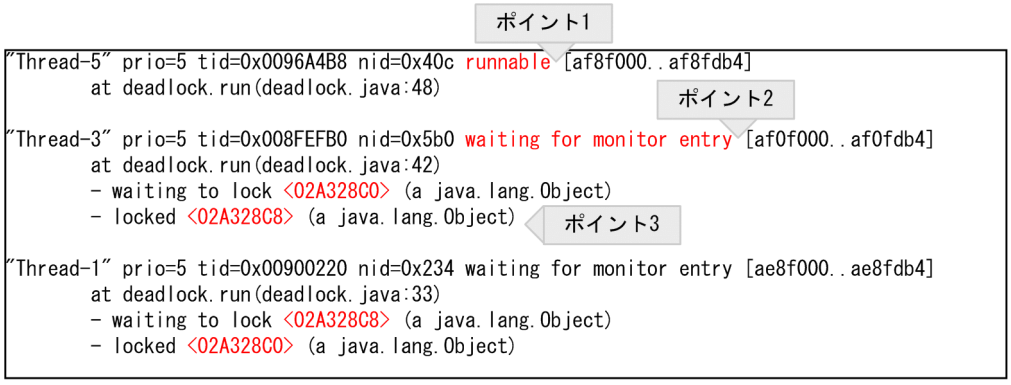
ここまでの調査で，無限ループの疑いがある場合は，開発元へ調査を依頼してください。

無限ループの疑いがない場合は手順 7.へ進んでください。

デッドロックの場合

デッドロックの場合のスレッドダンプの出力例，および調査のポイントについて説明します。

図 6-11　スレッドダンプ（デッドロック）の出力例



デッドロックが発生している場合のスレッドダンプの例を示します。

出力例の「nid:...」に続いてスレッド属性が出力されます。

「waiting for monitor entry」となっているスレッドを探します。

「-waiting to lock...」，および「-locked...」の内容を確認し，お互いがロックしている領域のロック取得待ちが発生していればデッドロック状態です。

ポイント 1

スレッド属性が runnable の場合，このスレッドは実行可能状態にありますので，このスレッドはデッドロックとは無関係です。

ポイント 2

スレッド属性が waiting for monitor entry である場合，このスレッドはロックの取得待ちであることを示しています。

デッドロックを起こしているスレッドである可能性があります。

ポイント 3

スレッドがロックを取得していて，かつ，ポイント 2 でスレッドのロック待ちの場合，デッドロックを起こしているスレッドである可能性がさらに高いです。

ポイント 2，ポイント 3 に当てはまるスレッドに対して，ロックしているオブジェクトのアドレスを突き合わせながらデッドロックを検出します。

例の場合，Thread-3 は，<02A328C8>のロックを取得して，かつ<02A328C0>の取得待ちを示します。

一方，Thread-1 は，<02A328C0>のロックを取得していて，かつ<02A328C8>の取得待ちとなっていて，Thread-3 と Thread-1 がデッドロックしていることがわかります。

ポイント

ここまでの調査で，デッドロックの疑いがある場合は，開発元へ調査を依頼してください。

デッドロックの疑いがない場合は手順 7.へ進んでください。

7. 業務アプリケーションを改修する。冗長な処理を取り除く

PRF トレース，およびスレッドダンプの調査の結果を基に，業務アプリケーションで遅延している疑いがある場合は調査して対策します。

ポイント

問題が解決した場合は，これでトラブルシュートは完了です。

問題が解決しない場合で，CPU 使用率が高い場合は，手順 8.へ進んでください。

問題が解決しない場合で，CPU 使用率が低い場合は，ご購入契約に基づくお問い合わせ窓口に調査を依頼してください。

8. 同時実行スレッド数のパラメタを小さくして，同時実行処理数を抑える

実行待ちリクエストが溜まることがありますが，多少処理を待たせた方がよいです。

9. マシンの CPU を増設する

CPU の増設では，ミドルウェアのライセンス費用の追加に注意してください。

10. マシンを追加し，トランザクションを負荷分散させる

マシンを追加する場合は，ハードウェア／ソフトウェアのライセンス費用の追加に注意してください。

ポイント

問題が解決した場合は，これでトラブルシュートは完了です。

問題が解決しない場合は，ご購入契約に基づくお問い合わせ窓口に調査を依頼してください。

7 性能解析トレースを使用した性能解析

性能解析トレースは，クライアントからのリクエストを処理するときにアプリケーションサーバの各機能が出力する性能解析情報（トレース情報），およびアプリケーションの処理が出力する性能解析情報（トレース情報）を収集する機能です。

この情報を基に，システムおよびアプリケーションの処理性能を解析できます。この章では，性能解析トレースを使用して，システムおよびアプリケーションの性能を解析する方法について説明します。なお，性能解析トレースによって性能解析情報が取得されるポイント（トレース取得ポイント）や，情報の取得範囲（PRF トレース取得レベル）については，「8.　性能解析トレースのトレース取得ポイントと PRF トレース取得レベル」を参照してください。

7.1　この章の構成

性能解析トレースを利用すると，システムを構成する各種ソフトウェアの稼働状況やリクエストの処理過程で各サーバが出力したトレース情報などを監視して，システム全体の処理性能を確認できます。処理性能を監視すると，アプリケーションサーバのボトルネックの調査や，パフォーマンスチューニングを実施する必要があるかどうかの判断ができるようになります。また，アプリケーションの各処理が出力したトレース情報を解析して，アプリケーションの性能を確認したり比較したりできます。性能を解析すると，アプリケーションのボトルネックの調査や，アプリケーションの改善を実施する必要があるかどうかの判断ができるようになります。

この章の構成を次の表に示します。

表 7-1　この章の構成（性能解析トレース）

分類	タイトル	参照先
解説	性能解析トレースの概要	7.2
	Management Server を利用した性能解析トレースファイルの収集	7.3
実装	性能解析トレースのルートアプリケーション情報取得のための実装	7.4
設定	実行環境での設定	7.5
運用	ユーザ拡張性能解析トレース実行時に出力されるログ情報	7.6
	性能解析トレースファイルを使用した処理性能の解析作業	7.7
注意事項	ユーザ拡張性能解析トレース使用時の注意事項	7.8

7.2　性能解析トレースの概要

性能解析トレースは，アプリケーションサーバの各機能やアプリケーションの処理が出力する性能解析情報を使用して，アプリケーションサーバおよびアプリケーションの処理性能を解析する機能です。

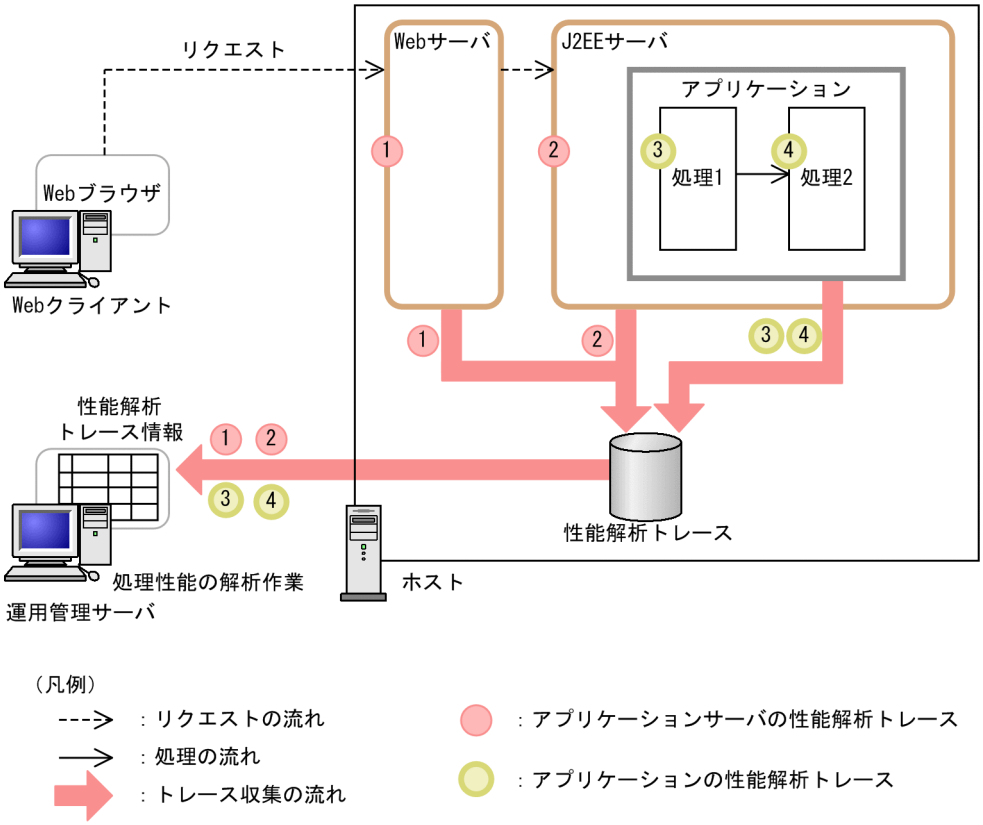
性能解析トレース機能には，使用する性能解析情報によって次の種類があります。

- アプリケーションサーバの性能解析トレース
- アプリケーションの性能解析トレース

なお，どちらの場合も処理性能の解析に，Performance Tracer という構成ソフトウェアを使用します。

性能解析トレースの概要を次の図に示します。

図 7-1　性能解析トレースの概要



図中の 1.および 2.はアプリケーションサーバの性能解析トレース，3.および 4.はアプリケーションの性能解析トレースです。どちらの性能解析トレースも，性能解析トレース情報としてまとめて解析できます。

7.2.1　アプリケーションサーバの性能解析トレースの概要

アプリケーションサーバの性能解析トレースは，クライアントからのリクエストを処理する過程でアプリケーションサーバの各機能が出力する性能解析情報（トレース情報）や，セッションのライフサイクルを判断するための情報を使用して，アプリケーションサーバの処理性能を解析する機能です（以降，アプリ

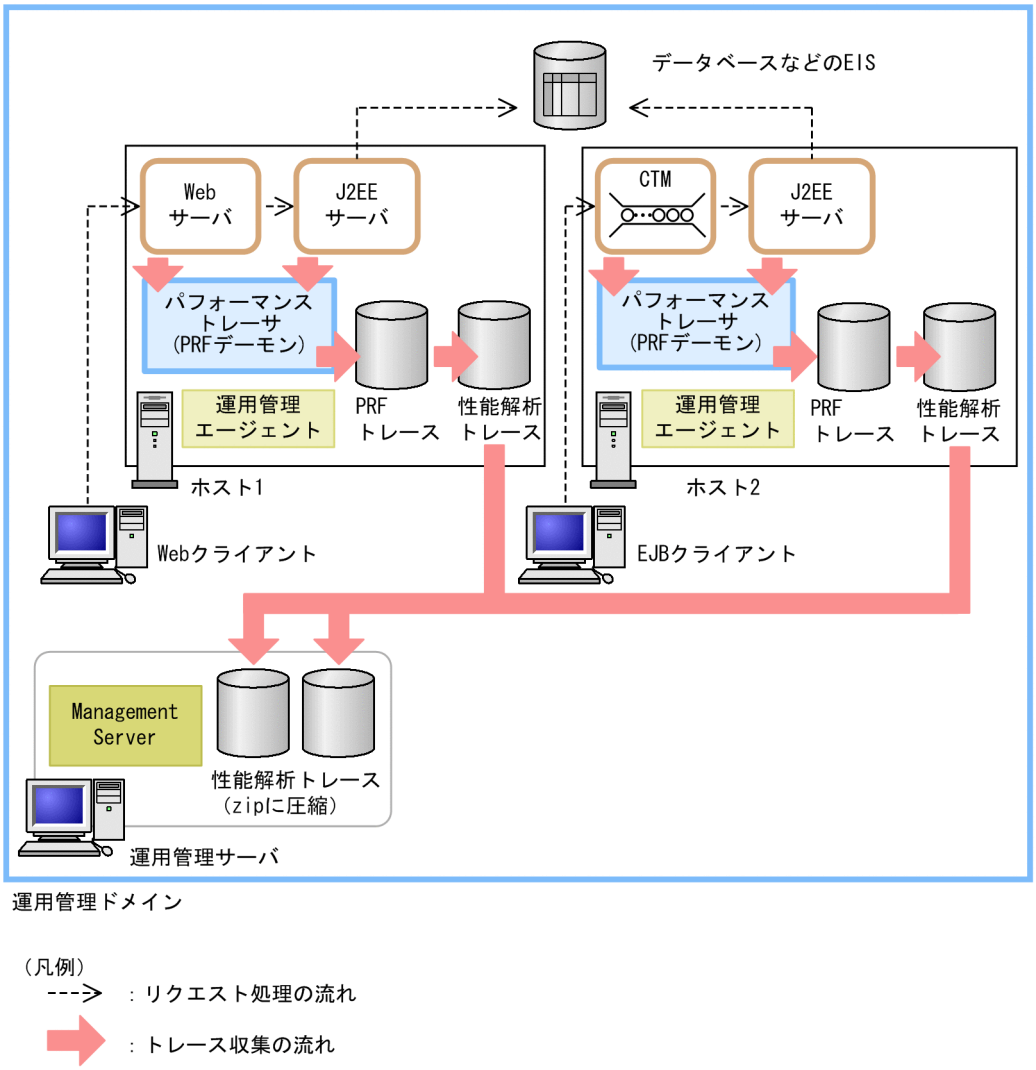
ケーションサーバの性能解析トレースを性能解析トレースと呼びます)。アプリケーションサーバのボトルネックを解析したり，障害が発生した場合にリクエストの処理がどこまで到達したかを調べてトラブルシュートの効率向上を図ったり，セッションやグローバルセッション情報のライフサイクルを把握したりできます。

(1) 性能解析トレースのトレース情報収集

性能解析トレースのトレース情報には，クライアントからデータベースなどの EIS に至るまで，およびその処理結果がクライアントに返却されるまでのリクエストの一連の処理で出力される性能解析情報が収集されます。

性能解析トレースのトレース情報収集の概要を，次の図に示します。

図 7-2 性能解析トレースのトレース情報収集の概要



Web クライアントまたは EJB クライアントからリクエストが送信された場合に，Web サーバ，J2EE サーバおよび CTM では，決まった処理のポイントでトレース情報がバッファに出力されます。出力された情報は，一定量たまると**パフォーマンストレーサ（PRF デーモン）**によって，トレースファイル（**PRF トレースファイル**）に出力されます。トレースが出力されるポイントを，**トレース取得ポイント**といいます。パ

フォーマンストレーサには，トレース取得レベル（標準または詳細）を設定できます。パフォーマンストレーサに設定するトレース取得レベルを**PRF トレース取得レベル**といいます。

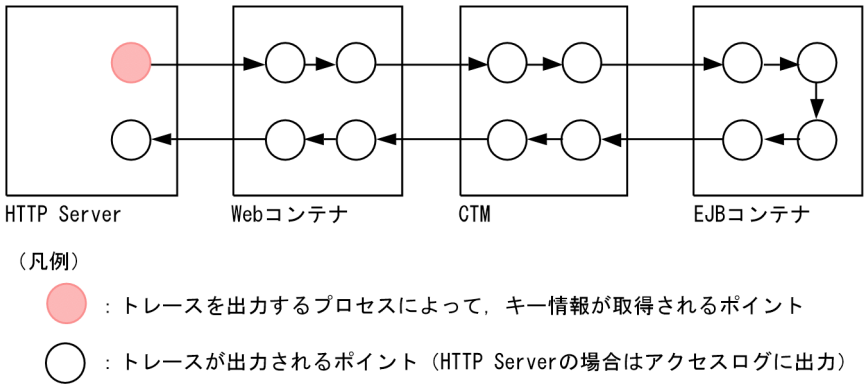
Management Server を利用して運用している場合は，PRF トレースファイルをテキスト形式に編集した性能解析トレースファイルを収集できます。運用管理者は，収集した性能解析トレースファイルを基に，運用管理ドメイン内全体の性能解析およびボトルネックの解析ができるようになります。性能解析トレースファイルの収集方法や出力情報については，「7.3　Management Server を利用した性能解析トレースファイルの収集」を参照してください。

(2)　性能解析トレースの仕組み

性能解析トレースでは，システム内部のイベント単位で，複数のノードおよびプロセス間にわたるトレース情報を取得できます。これによって，一連の処理の中で，どの処理がボトルネックになっているかをトレースできます。

イベント単位でトレースを取得するために，性能解析トレースでは，イベント単位の一連の処理に一貫したキーを設定して管理します。イベント内のトレース取得ポイントで出力するトレースには，キーの情報が付加されます。これによって，一連の処理がトレースできます。

図 7-3　性能解析トレースによるトレース出力の概要



なお，トレースを出力する EJB コンテナ，Web コンテナなどを，**機能レイヤ**といいます。性能解析トレースでは，次の機能レイヤの入り口と出口でトレース情報を出力します。必要に応じて，各機能レイヤ内の処理のうち，性能に影響を与える処理ごとにも，トレース情報を出力します。アプリケーションの実行環境と該当する機能レイヤを次の表に示します。

表 7-2　アプリケーションの実行環境と該当する機能レイヤ

機能レイヤ	アプリケーションの実行環境	
	J2EE アプリケーションの実行環境	バッチアプリケーションの実行環境
CTM	○	－
Web コンテナ	○	－
EJB コンテナ	○	－
Timer Service	○	－

機能レイヤ	アプリケーションの実行環境	
	J2EE アプリケーションの実行環境	バッチアプリケーションの実行環境
JNDI	○	○
JTA	○	○
JCA コンテナ	○	○
DB Connector	○	○
RMI（通信処理）※1	○	○
OTS	○	○
標準出力/標準エラー出力/ユーザログ	○	○
DI	○	－
バッチアプリケーション実行機能	－	○※2
JPA	○	－
TP1 インバウンド連携機能	○	－
CJMS プロバイダ	○	－
JavaMail	○	－
CDI	○	－
JSF 2.3	○	－
JAX-RS	○	－
Java Batch	○	－

（凡例）○：該当する　－：該当しない

注※1

RMI（通信処理）の機能レイヤについては，トレース情報の取得を抑止することができます。その場合，トレース取得レベルに「抑止」を設定します。設定方法については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cprfstart（PRF デーモンの開始）」，またはマニュアル「アプリケーションサーバ リファレンス コマンド編」の「cprflevel（PRF トレース取得レベルの表示と変更）」を参照してください。

注※2

トレース情報は，バッチアプリケーションの実行直前（main メソッドを呼ぶ直前），およびバッチアプリケーション終了直後に出力されます。cjexecjob コマンドおよび cjkilljob コマンドの実行では出力されません。

また，性能解析トレースでは，これらの機能レイヤのほかに，J2EE サーバの開始処理，終了処理，トランザクションタイムアウト発生時，およびセッションの生成／破棄でもトレースを出力します。

なお，トレース情報の内容には，トレース情報を取得したプロセス ID，取得ポイントを示すイベント ID，トレース取得年月日やトレース情報を取得したクライアントアプリケーションの IP アドレスなどの情報が含まれます。

参考

これらの機能レイヤのほか，Application Server の構成ソフトウェアおよび関連プログラムでも，次の機能レイヤで PRF トレースが取得できます。

- Web Services - Base
- TP1 Connector
- TP1/Client/J
- TP1/MQ Access
- Reliable Messaging
- HCSC サーバ
- HCSC サーバ（Object Access アダプタ）
- Service Coordinator Interactive Workflow
- HCSC サーバ（ファイルアダプタ）
- HCSC サーバ（Message Queue アダプタ）
- HCSC サーバ（FTP アダプタ）
- JAX-WS エンジン
- Elastic Application Data store

トレース情報のキー情報は，次の要素で構成されています。

キー情報の構成

- キー情報を取得したプロセス ID
- キー情報を取得したプロセスが起動しているホストの IP アドレス
- PRF トレースの I／O プロセス（PRF デーモン）単位で割り当てられる通信番号
  なお，PRF デーモンが起動していない場合は，通信番号として時刻が返却されます。ただし，この場合，通信番号の一意性を保てないおそれがあるので，PRF デーモンは必ず起動してください。

PRF トレースには，次の 2 種類のキー情報が付与されます。

- **ルートアプリケーション情報**
  各イベントで一連の処理の先頭になるプロセスで取得した情報です。
  - J2EE アプリケーションの場合
    HTTP Server，NIO HTTP サーバ，または EJB クライアントで取得した情報になります。
  - バッチアプリケーションの場合
    バッチアプリケーションの実行直前に取得した情報になります。
- **クライアントアプリケーション情報**

J2EE アプリケーションの場合，次に示す Enterprise Bean を呼び出す処理単位で設定される情報です。

- Web コンテナから EJB コンテナの呼び出し
- EJB クライアントから EJB コンテナの呼び出し
- EJB コンテナから EJB コンテナの呼び出し

なお，バッチアプリケーションの場合は，バッチアプリケーションの実行直前に設定される情報になります。

(3) 性能解析トレースの構成

性能解析トレースは，次のプログラムで構成されています。

- PRF トレース出力ライブラリ

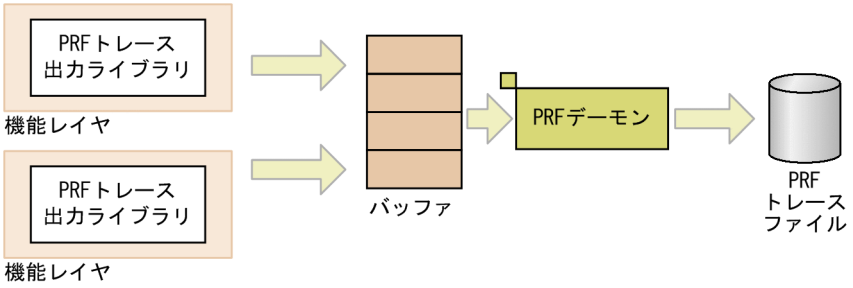
  アプリケーションサーバの各機能レイヤに組み込まれています。各機能レイヤが出力した PRF トレースを，共用メモリに作成されたバッファに出力します。

- PRF デーモン

  バッファに出力された PRF トレースを，一定量たまったらファイルに出力する，I／O プロセスです。PRF トレースを取得するホストごとに，一つ以上起動します。一つのホストに一つ配置することをお勧めします。

PRF トレース出力ライブラリと PRF デーモンの関係を次の図に示します。

図 7-4 PRF トレース出力ライブラリと PRF デーモンの関係



PRF トレース出力ライブラリによってトレースが出力されるバッファ領域は，PRF デーモン起動時に作成されます。バッファ領域は，共用メモリに作成されます。ただし，前回 PRF デーモンを起動した時に作成したバッファ領域が残っている場合は，その領域を再利用します。バッファ領域が削除されないで残っているのは，前回起動した PRF デーモンが異常終了した場合です。

PRF デーモンが正常終了した場合，バッファ領域内のバッファデータは PRF トレースファイルに出力され，バッファ領域は削除されます。

バッファ領域が不足すると，KFCT26999-W のメッセージが出力されて，PRF トレースが完全に出力されない場合があります。そのため，このメッセージが出力される場合は，バッファサイズのチューニングをしてください。

(4) トレース情報の取得によるトラブルシュート

トレース情報を利用したトラブルシュートについて説明します。

性能解析トレースに出力される情報を取得することで，トラブルシュートとして次のように使用できます。

- J2EE アプリケーションのトランザクションがタイムアウトした場合や HTTP Server のリバースプロキシでのレスポンス受信時にタイムアウトが発生した場合，性能解析トレースに出力されるルートアプリケーション情報を使用して，タイムアウトしたトランザクションやリクエストを特定できます。
- データベースとの接続中にトラブルが発生した場合，性能解析トレースに出力されるコネクション ID を利用して，トラブルが発生したコネクションを特定できます。

7.2.2 アプリケーションの性能解析トレースの概要

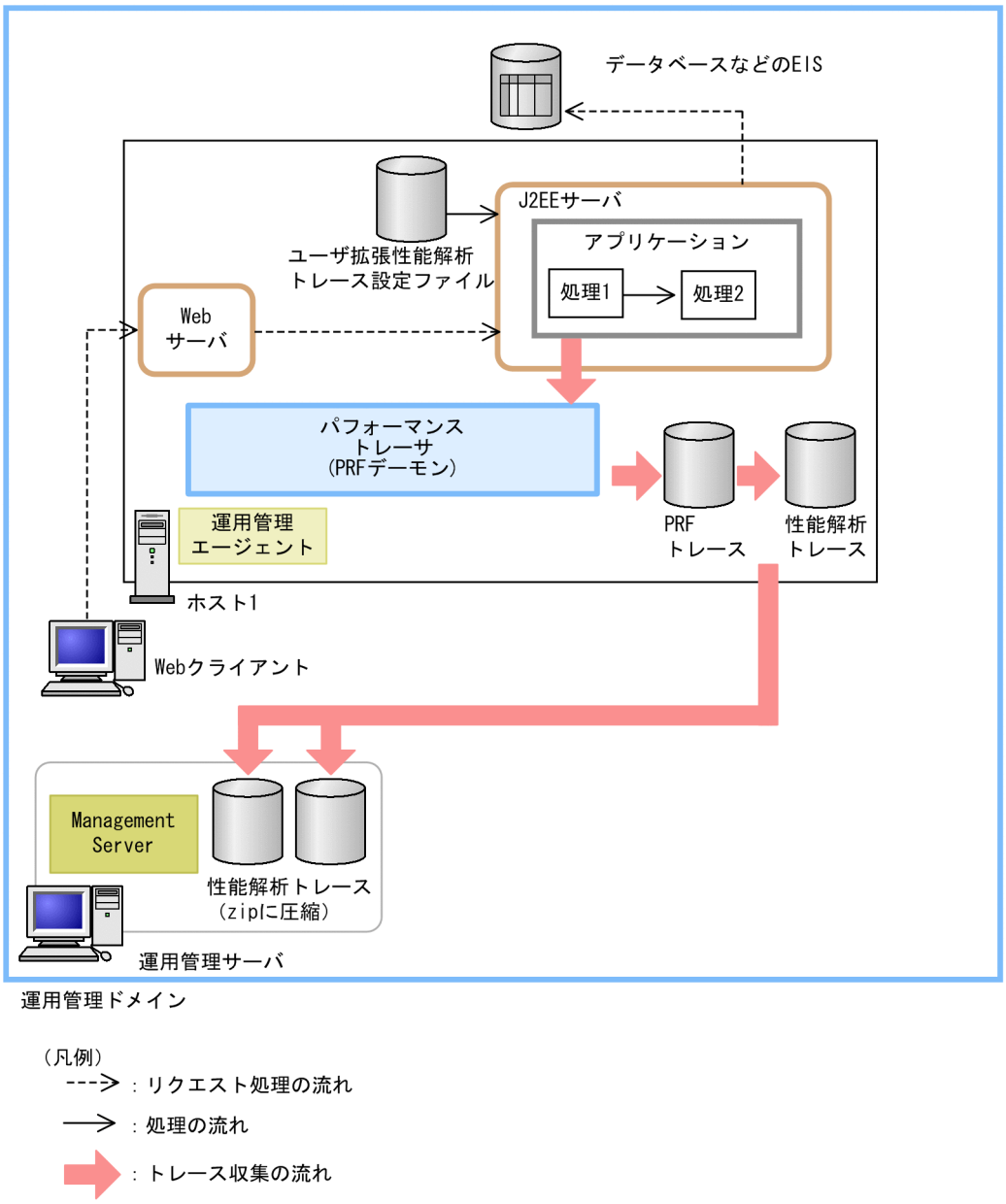
アプリケーションの性能解析トレースは，アプリケーションの開始から終了までの過程で，アプリケーションのトレース対象の処理が実行された時点で出力される性能解析情報（トレース情報）を使用して，アプリケーションの処理性能を解析する機能です（以降，アプリケーションの性能解析トレースをユーザ拡張性能解析トレースと呼びます)。ユーザ拡張性能解析トレースは，アプリケーションサーバの性能解析トレースを使用していることが前提となります。ボトルネックとなる処理を特定したり，障害が発生した場合に処理がどこまで到達したかを調べてトラブルシュートの効率向上を図ったりできます。また，どの処理を性能解析情報の取得対象とするかはユーザ拡張性能解析トレース設定ファイルで指定します。アプリケーションに性能解析情報の取得処理を実装する必要がないため，アプリケーションの性能を確認したり比較したりする作業を効率良く実施できます。

(1) ユーザ拡張性能解析トレースのトレース情報収集

ユーザ拡張性能解析トレースのトレース情報には，ユーザ拡張性能解析トレース設定ファイルで指定した処理が実行された時点で出力される性能解析情報が収集されます。

ユーザ拡張性能解析トレースのトレース情報収集の概要を，次の図に示します。

図 7-5　ユーザ拡張性能解析トレースのトレース情報収集の概要



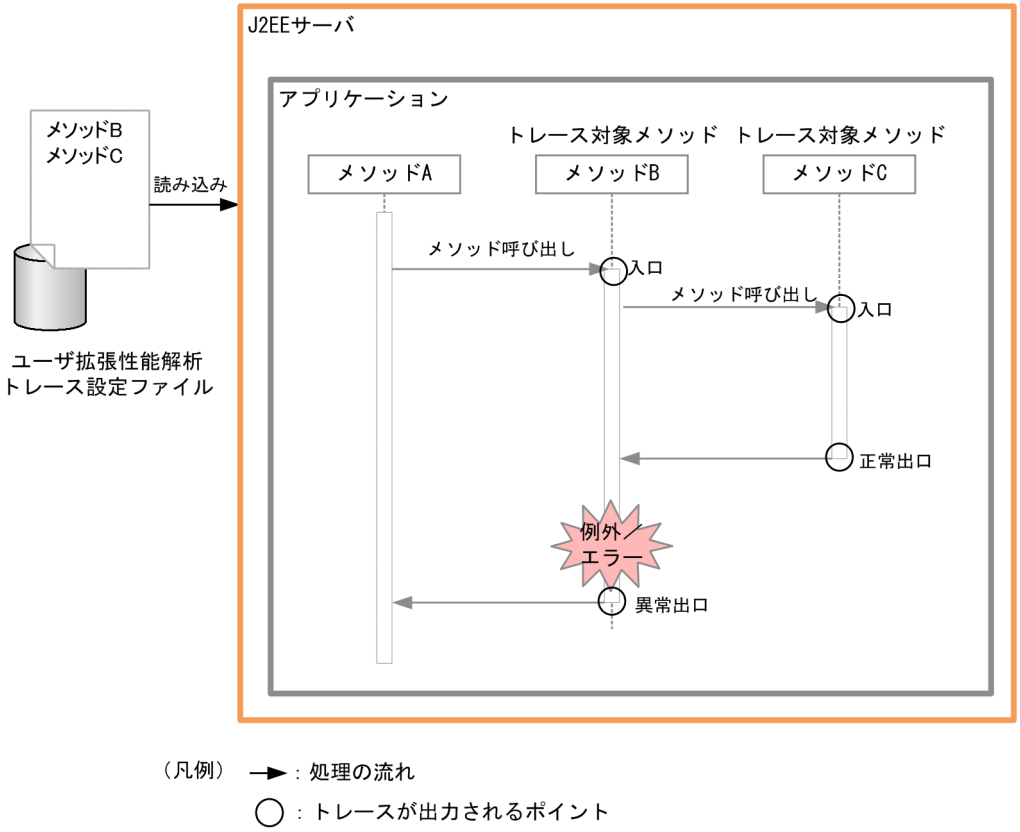
J2EE サーバが起動すると，ユーザ拡張性能解析トレース設定ファイルが読み込まれます。ユーザ拡張性能解析トレース設定ファイルに指定された処理（トレース取得ポイント）が呼ばれると，トレース情報がバッファに出力されます。出力された情報は，一定量溜まるとパフォーマンストレーサ（PRF デーモン）によって，トレースファイル（PRF トレースファイル）に出力されます。

Management Server を利用して運用している場合は，PRF トレースファイルをテキスト形式に編集した性能解析トレースファイルを収集できます。運用管理者は，収集した性能解析トレースファイルを基に，運用管理ドメイン内全体の性能解析およびボトルネックの解析ができるようになります。性能解析トレースファイルの収集方法や出力情報については，「7.3　Management Server を利用した性能解析トレースファイルの収集」を参照してください。

(2)　ユーザ拡張性能解析トレースの仕組み

ユーザ拡張性能解析トレースでは，トレース情報を取得する対象メソッドの名前をユーザ拡張性能解析トレース設定ファイルで指定します。指定したメソッド名のトレース出力の概要を次の図に示します。

図 7-6　ユーザ拡張性能解析トレースによるトレース出力の概要



ユーザ拡張性能解析トレースを有効に設定している場合，ユーザ拡張性能解析トレースはユーザ拡張性能解析トレース設定ファイルを読み込みます。アプリケーションが実行され，ユーザ拡張性能解析トレース設定ファイルに指定したメソッド（トレース対象メソッド）が呼び出されると，次の位置でトレース情報を出力します。

- メソッドの入口

  メソッドが開始された直後のトレース情報。
- メソッドの正常出口

  メソッドが正常終了する直前のトレース情報。
- メソッドの異常出口

  メソッドで例外またはエラーが発生した直後のトレース情報。ただし，メソッドの呼び出し元にスローされない例外やエラーは除きます。

(3)　ユーザ拡張性能解析トレースの構成

ユーザ拡張性能解析トレースは次の要素で構成されています。

なお，ユーザ拡張性能解析トレースは，性能解析トレースを出力するために，インストゥルメンテーション機能のクラスロードフック処理を利用して，トレース対象のアプリケーションを書き換えています。

- **ユーザ拡張性能解析トレース設定ファイル**

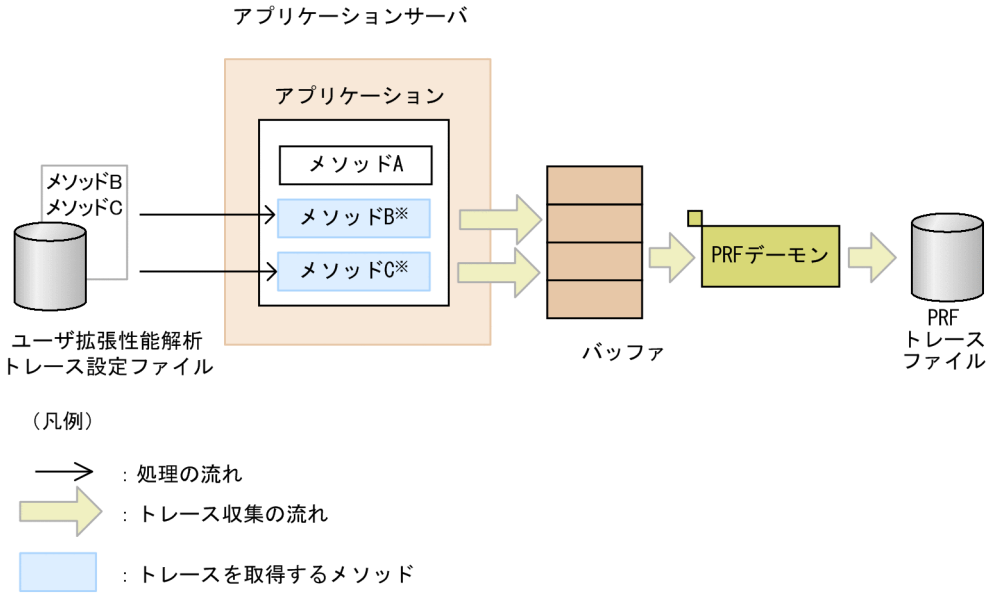
  ユーザ拡張性能解析トレースのトレース対象メソッドの情報は，ユーザ拡張性能解析トレース設定ファイルを使って設定します。ユーザ拡張性能解析トレース設定ファイルの内容については，「7.5.3　ユーザ拡張性能解析トレースのトレース対象メソッドの設定」を参照してください。

- **PRF デーモン**

  バッファに出力された PRF トレースを，一定量たまったらファイルに出力する，I／O プロセスです。PRF トレースを取得するホストごとに，一つ以上起動します。一つのホストに一つ配置することをお勧めします。

ユーザ拡張性能解析トレース設定ファイルおよび PRF デーモンの関係を次の図に示します。

図 7-7　ユーザ拡張性能解析トレース設定ファイルおよび PRF デーモンの関係



注※　ユーザ拡張性能解析トレースは，性能解析トレースを出力するために，インストゥルメンテーション機能のクラスロードフック処理を利用して，トレース対象のアプリケーションを書き換えています。

7.3 Management Server を利用した性能解析トレースファイルの収集

Management Server を利用して運用している場合，各ホストに出力されたトレースファイルの内容は，Management Server の運用管理コマンド（mngsvrutil）で，管理サーバに一括収集できます。なお，PRF デーモンが出力した PRF トレースファイルは，バイナリ形式のファイルです。Management Server では，運用管理エージェントに指示を出し，PRF トレースファイルをテキスト（CSV）形式のファイルに編集して，それを圧縮（ZIP 形式）したものを収集します。Management Server で収集できるのは，運用管理ドメイン内のホストで出力された PRF トレースです。

なお，テキスト形式に編集したトレースファイルを**性能解析トレースファイル**といいます。

運用管理コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「mngsvrutil（Management Server の運用管理コマンド)」を参照してください。

この節では，性能解析トレースファイルの収集方法，出力先や出力情報について説明します。

7.3.1 性能解析トレースファイルの収集方法

性能解析トレースファイルを収集する場合，Management Server の運用管理コマンド（mngsvrutil）を使用します。mngsvrutil コマンドには，サブコマンド「collect」を指定します。性能解析トレースファイルの収集時，トレース情報の収集対象を選択します。運用管理コマンドを使用した場合のトレース情報の収集対象を次に示します。

- 運用管理ドメイン内のホスト上で動作しているすべてのパフォーマンストレーサ
- 特定のホスト上で動作しているすべてのパフォーマンストレーサ
- 特定のパフォーマンストレーサ

それぞれの場合の実行形式と実行例を次に示します。

運用管理ドメイン内のホスト上で動作しているすべてのパフォーマンストレーサを対象にする場合

実行形式

```
mngsvrutil -m <Management Serverのホスト名>[:<ポート番号>] -u <管理ユーザID> -p <管理パスワード> collect allPrfTraces
```

実行例

```
mngsvrutil -m mnghost -u user01 -p pw1 collect allPrfTraces
```

特定のホスト上で動作しているすべてのパフォーマンストレーサを対象にする場合

実行形式

```
mngsvrutil -m <Management Serverのホスト名>[:<ポート番号>] -u <管理ユーザID> -p <管理パスワード> -t <ホスト名> -k host collect prfTrace
```

実行例

```
mngsvrutil -m mnghost -u user01 -p pw1 -t host01 -k host collect prfTrace
```

特定のパフォーマンストレーサを対象にする場合

実行形式

```
mngsvrutil -m <Management Serverのホスト名>[:<ポート番号>] -u <管理ユーザID> -p <管理パスワード> -t <論理パフォーマンストレーサ名> -k logicalServer collect prfTrace
```

実行例

```
mngsvrutil -m mnghost -u user01 -p pw1 -t ID01 -k logicalServer collect prfTrace
```

7.3.2 性能解析トレースファイルの出力先

- Windows の場合
  <Manager のログ出力ディレクトリ>¥prf
- UNIX の場合
  <Manager のログ出力ディレクトリ>/prf

なお，性能解析トレースファイルは，トレース情報の収集対象別に次の表に示すファイル名で出力されます。

表 7-3 性能解析トレースファイルのファイル名

トレース情報の収集対象	ファイル名
運用管理ドメイン内のホスト上で動作しているすべてのパフォーマンストレーサ	<運用管理ドメイン名>-<日時※1>.zip
特定のホスト上で動作しているすべてのパフォーマンストレーサ	<ホスト名>-<日時※1>.zip
特定のパフォーマンストレーサ	<論理サーバ名※2>-<日時※1>.zip

注※1
性能解析トレースファイルを収集した日時が表示されます。

注※2
指定したパフォーマンストレーサの名称が表示されます。

7.3.3 性能解析トレースファイルの出力情報（性能解析トレースの場合）

性能解析トレースでは，各機能レイヤのトレース情報を収集します。

各機能レイヤが出力する情報を次の表に示します。なお，CTM とそれ以外の機能レイヤでは，取得項目が異なります。また，付加情報の有無など，取得ポイントごとに，出力される項目は異なります。各取得

ポイントで出力される項目の詳細については，「8. 性能解析トレースのトレース取得ポイントと PRF トレース取得レベル」を参照してください。

表 7-4 性能解析トレースファイルに出力する情報（性能解析トレースの場合）

トレース情報のヘッダ	説明	値の範囲
PRF	そのプロセスのレコードの状態（正常または異常）。	次のどちらかが出力されます。 正常：Rec 異常：ErrRec
Process	トレース情報を取得したプロセスのプロセス ID。	10 けたの 10 進数が出力されます。
Thread	トレース情報を取得したプロセス内スレッドのスレッド ID，およびスレッドのハッシュ値※1。	スレッド ID：20 けた以内の 10 進数が出力されます。 ハッシュ値：10 けた以内の 10 進数が出力されます。
Trace	トレース情報を取得したプロセス内スレッドでのトレース通番。	10 けたの 10 進数が出力されます。
ProcessName	プロセス名。	32 文字以内のプロセスを表す文字列※2 が出力されます。
Event	トレース取得ポイントを示すイベント ID。	6 けたの 16 進数（6 けたには，"0x"も含みます）が出力されます※3。
Date	トレース情報を取得した年月日。	年月日が「yyyy/mm/dd」のフォーマットで出力されます。 yyyy：年 mm：月 dd：日
Time	トレース情報を取得した時刻（時：分：秒）。	時刻が「hh:mm:ss」のフォーマットで出力されます。 hh：時 mm：分 ss：秒
Time(msec/usec/nsec)	トレース情報を取得した時刻（ミリ秒/マイクロ秒/ナノ秒）。	時刻が「ms/us/ns」のフォーマットで出力されます。 ms：ミリ秒 us：マイクロ秒 ns：ナノ秒
Rc	リターンコード。	16 けたの 16 進数（16 けたには，"0x"も含む）が出力されます。 正常：0 異常：1（または 0 以外）
ClientAP IP※4	トレース情報を取得したクライアントアプリケーションの IP アドレス。	IP アドレスが，「aaa.bbb.ccc.ddd」のフォーマットで出力されます。

トレース情報のヘッダ	説明	値の範囲
ClientAP PID※4	トレース情報を取得したクライアントアプリケーションのプロセス ID。	10 けたの 10 進数が出力されます。
ClientAP CommNo.※4	トレース情報を取得したクライアントアプリケーションの通信番号。	18 けたの 16 進数（18 けたには，"0x"も含みます）が出力されます。
RootAP IP※5	トレース情報を取得したルートアプリケーションの IP アドレス。	IP アドレスが「aaa.bbb.ccc.ddd」のフォーマットで出力されます。
RootAP PID※5	トレース情報を取得したルートアプリケーションのプロセス ID。	10 けたの 10 進数が出力されます。
RootAP CommNo.※5	トレース情報を取得したルートアプリケーションの通信番号。	18 けたの 16 進数（18 けたには，"0x"も含む）が出力されます
SendSCD IP※5	リクエスト要求元 CTM の IP アドレス。	IP アドレスが「aaa.bbb.ccc.ddd」のフォーマットで出力されます。
SendSCD PID※6	リクエスト要求元 CTM のプロセス ID。	10 けたの 10 進数が出力されます。
ReceiveSCD IP※6	リクエスト要求先 CTM の IP アドレス。	IP アドレスが「aaa.bbb.ccc.ddd」のフォーマットで出力されます。
ReceiveSCD PID※6	リクエスト要求先 CTM のプロセス ID。	10 けたの 10 進数
INT	取得ポイントごとのインタフェース名。	33 文字以内の文字列※7 が出力されます。
OPR	取得ポイントに関連するオペレーション情報。	33 文字以内の文字列※7 が出力されます。
LookupName※6	ルックアップ名。	33 文字以内の文字列※7 が出力されます。
OPT※8	取得ポイントごとの付加情報。	514 文字以内の 16 進数値文字列が出力されます。
ASCII	取得ポイントごとの付加情報を ASCII 文字出力。	OPT の内容が 514 文字以内の ASCII 文字列として出力されます。

注※1

CTM で取得したトレース情報には，スレッドのハッシュ値は出力されない場合があります。

注※2

プロセス名称は次のように決定されます。

- **EJB クライアントアプリケーションの場合**

  EJB クライアントアプリケーションのシステムプロパティ ejbserver.server.prf.processName に指定された名称。

  このシステムプロパティを指定していない場合，またはこのプロパティに null 文字を指定した場合は，"EJBClient"になります。

- **J2EE サーバ，バッチサーバ，および Web コンテナサーバの場合**

  サーバ名がプロセス名称となります。

- **CTM の場合**

CTM の各プロセス名称となります。

注※3

イベント ID は，機能レイヤのトレース取得ポイントごとに割り当てられています。詳細は，「8.　性能解析トレースのトレース取得ポイントと PRF トレース取得レベル」を参照してください。

注※4

クライアントアプリケーション情報の構成要素です。

注※5

ルートアプリケーション情報の構成要素です。Web コンテナで出力されるトレースポイントでは，トレース情報を取得したルートアプリケーションの IP アドレス，プロセス ID，および通信番号が「0.0.0.0/0/0x0000000000000000」と出力される場合があります。詳細は「7.7.7　ルートアプリケーション情報を利用したログ調査」を参照してください。

なお，HTTP Server を IPv4 と IPv6 のデュアルスタック環境で動作させて運用する場合，ルートアプリケーションのプロセス ID には，IPv4 アドレスが出力されます。また，クライアントのアドレスが IPv6 の場合で 33 文字以上のときは，トレース情報のイベント ID「0x8000」および「0x8100」を編集して出力します。

- イベント ID「0x8000」の場合："先頭から 32 文字"+"*"
- イベント ID「0x8100」の場合："先頭から 16 文字"+"*"+"後ろから 16 文字"

この場合，イベント ID「0x8000」および「0x8100」の両方を参照すると，クライアントのアドレス全体を取得できます。

注※6

CTM の場合だけ出力される情報です。CTM 以外のレイヤでは，「****」が表示されます。なお，バッチアプリケーションを実行するシステムではスケジュールグループ名が表示されます。

注※7

インタフェース名，オペレーション名，およびルックアップ名が 32 文字を超える場合，次のどれかの方法で，33 文字になって出力されます。詳細については，「8.　性能解析トレースのトレース取得ポイントと PRF トレース取得レベル」を参照してください。

先頭 32 文字+*

先頭 16 文字+*+末尾 16 文字

*+末尾 32 文字

注※8

機能レイヤによっては，「OPT」に入り口時刻が出力されるトレース取得ポイントがあります。入り口時刻とは，そのトレース取得ポイントのトレースに対応する入り口トレースが出力された時間を示します。

なお，入り口時刻は，「1970 年 1 月 1 日 00:00:00」からの通算時間が 16 バイトの値で出力されます。前半の 8 バイトの値は秒単位を，後半の 8 バイトの値はマイクロ秒単位を示します。

7.3.4　性能解析トレースファイルの出力情報（ユーザ拡張性能解析トレースの場合）

ユーザ拡張性能解析トレースでは，アプリケーションの処理が出力するトレース情報を収集します。

性能解析トレースの出力項目については，「7.3.3　性能解析トレースファイルの出力情報（性能解析トレースの場合）」を参照してください。

なお，オペレーション情報の有無など，取得ポイントごとに，出力される項目は異なります。各取得ポイントで出力される項目の詳細については，「8.25　アプリケーションのトレース取得ポイント」を参照してください。

7.4　性能解析トレースのルートアプリケーション情報取得のための実装

この節では，性能解析トレースのルートアプリケーション情報を取得する機能の概要および実装方法を説明します。

ルートアプリケーション情報とは，性能解析トレース機能で取得した情報のうち，HTTP Server，NIO HTTP サーバ，または EJB クライアントで取得した情報をいいます。API を使用して，ルートアプリケーション情報の文字列表現を取得する機能を J2EE アプリケーションおよびバッチアプリケーションに実装できます。実装すると，取得した文字列表現をログファイルなどに記録することで，任意のタイミングで性能解析トレースファイルとの突き合わせができるため，トラブル発生時のトラブルシュートに役立ちます。

性能解析トレースのルートアプリケーション情報の取得機能を実装するには，CprfTrace クラスを使用します。CprfTrace クラスを使用するには，次のパスをクラスパスに指定してコンパイルします。

- Windows の場合
  <Application Server のインストールディレクトリ>¥CC¥lib¥ejbserver.jar
- UNIX の場合
  /opt/Cosminexus/CC/lib/ejbserver.jar

なお，ルートアプリケーション情報を利用した調査については，「7.7.7　ルートアプリケーション情報を利用したログ調査」を参照してください。

7.5　実行環境での設定

この節では，性能解析トレースおよびユーザ拡張性能解析トレースを使用するための設定について説明します。

設定する項目は，性能解析トレースとユーザ拡張性能解析トレースで異なります。取得するトレース情報に必要な設定項目を次の表に示します。

表 7-5　取得するトレース情報に必要な設定項目

設定項目	取得するトレース情報		参照先
	性能解析トレース	ユーザ拡張性能解析トレース	
性能解析トレースを使用するための設定	○	○	7.5.1
ユーザ拡張性能解析トレースの使用するための設定	－	○	7.5.2
ユーザ拡張性能解析トレースのトレース対象メソッドの設定	－	○	7.5.3

（凡例）　○：設定が必要。　－：設定は不要。

7.5.1　性能解析トレースを使用するための設定

性能解析トレースを使用する場合，次の設定が必要です。

- Management Server
- パフォーマンストレーサ

注意事項

AIX の場合は，環境変数の設定には次の点に注意してください。

- Performance Tracer の実行環境では，環境変数 PSALLOC に「early」を設定してください。設定しない場合にメモリ不足が発生すると，正しい動作が保証できません。
- AIX の早期ページングスペース割り当てを指定する，環境変数 PSALLOC に「early」を指定しています。早期ページングスペース割り当てでは，ページングスペース見積もり上の考慮事項があります。詳細は，AIX のマニュアルの「システム・マネージメント・コンセプト：オペレーティング・システムおよびデバイス」を参照してください。
- Performance Tracer の実行環境では，環境変数 NODISCLAIM に「true」を設定してください。環境変数 PSALLOC に「early」を設定した場合，環境変数 NODISCLAIM に「true」を設定しないと，レスポンス，スループットおよび CPU 利用率が極端に低下することがあります。

- Performance Tracer で使用するユーザデータ領域と共用メモリ領域を拡張するため，環境変数 LDR_CNTRL に「MAXDATA=0x40000000」を設定してください。割り当てるメモリの値を 1 ギガバイトにしてください。
- Performance Tracer の実行環境では，環境変数 EXTSHM に「ON」を設定してください。設定しない場合，共有メモリが参照できなくなることがあります。

(1) Management Server の設定

Management Server の設定は，mserver.properties（Management Server 環境設定ファイル）で実施します。指定するパラメタを次に示します。

- com.cosminexus.mngsvr.trace

  Management Server が収集する性能解析トレースファイルのファイル面数を指定します。

mserver.properties およびキーの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.6　mserver.properties（Management Server 環境設定ファイル)」を参照してください。なお，性能解析トレースファイルについては，「7.3　Management Server を利用した性能解析トレースファイルの収集」を参照してください。

(2) パフォーマンストレーサの設定

パフォーマンストレーサ（PRF デーモン）の設定は，簡易構築定義ファイルで実施します。性能解析トレースの定義は，簡易構築定義ファイルの論理パフォーマンストレーサ（performance-tracer）の<configuration>タグ内に指定します。

簡易構築定義ファイルでの性能解析トレースの定義について次の表に示します。

表 7-6　簡易構築定義ファイルでの性能解析トレースの定義

指定するパラメタ	設定内容
PrfTraceLevel	アプリケーションサーバの各機能（Web サーバ，J2EE サーバ，CTM などの機能レイヤ）がバッファに出力する，パフォーマンストレーサのトレース取得レベルを指定します。
PrfTraceCount	パフォーマンストレーサの PRF トレースファイルの面数を指定します。
PrfTraceFileSize	パフォーマンストレーサのファイルサイズを指定します。PrfTraceFileSize≧PrfTraceBufferSize の関係が成り立つ値を設定してください。
PrfTraceBufferSize	パフォーマンストレーサのバッファサイズを指定します。PrfTraceFileSize≧PrfTraceBufferSize の関係が成り立つ値を設定してください。

注　簡易構築定義ファイルおよび指定する各パラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

7.5.2　ユーザ拡張性能解析トレースを使用するための設定

ユーザ拡張性能解析トレースを使用する場合，サーバの起動前に次の設定が必要です。

- J2EE サーバ
- バッチサーバ
- EJB クライアントアプリケーション
- ユーザ拡張性能解析トレースのトレース対象メソッドの設定

(1)　J2EE サーバの設定

J2EE サーバの設定は，簡易構築定義ファイルで実施します。

ユーザ拡張性能解析トレースの定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に JavaVM 起動パラメタ（add.jvm.arg）で指定します。

簡易構築定義ファイルでのユーザ拡張性能解析トレースの定義を次の表に示します。

表 7-7　簡易構築定義ファイルでのユーザ拡張性能解析トレースの定義

項目	指定するパラメタ	設定内容
ユーザ拡張性能解析トレースを有効にするための設定	jvm.userprf.Enable	ユーザ拡張性能解析トレースを使用する場合，有効に設定してください。
ユーザ拡張性能解析トレース設定ファイルを指定するための設定	jvm.userprf.File※1	ユーザ拡張性能解析トレースで使用するユーザ拡張性能解析トレース設定ファイルのファイルパスを指定します。
アプリケーションの書き換え処理に関する設定	jvm.userprf.Limit	ユーザ拡張性能解析トレース設定ファイルで指定したメソッドのうち，書き換え対象とするメソッド数の上限値を指定します。ユーザ拡張性能解析トレース設定ファイルで指定したメソッドで，この値を超えたメソッドはトレース対象となりません。
	jvm.userprf.Trace	ユーザ拡張性能解析トレース設定ファイルで指定したクラスファイルの書き換えに成功した場合に，ログを出力するかどうかを指定します。なお，クラスファイルの書き換えに失敗した場合は，ログは必ず出力されます。
ユーザ拡張性能解析トレースの出力情報に関する設定	jvm.userprf.ExtendedSetting※2	トレース対象をメソッド単位に加えて，パッケージ単位またはクラス単位で指定できるようにするかどうかを指定します。

項目	指定するパラメタ	設定内容
	jvm.userprf.LineNumber	メソッドの正常出口でメソッドが最後に実行した行の行番号をトレース情報に出力するかどうかを指定します。
	jvm.userprf.ThrowableName	例外またはエラーのクラス名をトレース情報に出力するかどうかを指定します。
	jvm.userprf.ThrowableNameEditMethod	例外またはエラーのクラス名が文字数制限 32 文字を超えた場合の名前の編集方法を指定します。
	jvm.userprf.LogLevel	ユーザ拡張性能解析トレースのトレース出力レベルを指定します。

注※1

デフォルト値を次に示します。

Windows の場合

<JDK インストールディレクトリ>¥usrconf¥userprf.cfg

UNIX の場合

/opt/Cosminexus/jdk/usrconf/userprf.cfg

注※2

jvm.userprf.ExtendedSetting プロパティを設定して許可されるユーザ拡張性能解析トレース設定ファイルの指定形式はトレース対象となるメソッドが大量になります。そのため，出力量が増えたり性能に影響を与えたりする場合があるので注意が必要です。ユーザ拡張性能解析トレース使用時の注意事項についての詳細は「7.8　ユーザ拡張性能解析トレース使用時の注意事項」を参照してください。

簡易構築定義ファイルおよび指定する各パラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

定義例を次に示します。

```
   :
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
<param-value>-Djvm.userprf.File=<Application Serverのインストールディレクトリ>/CC/server/usrconf/ejb/<実サーバ名>/userprf.cfg</param-value>
<param-value>-Djvm.userprf.LogLevel=class</param-value>
</param>
   :
```

参考

J2EE サーバの設定は，運用管理ポータルの［起動パラメタの設定］画面（論理 J2EE サーバの定義）でも設定できます。運用管理ポータルでの設定方法については，マニュアル「アプリケーショ

ンサーバ 運用管理ポータル操作ガイド」の「10.8.23　起動パラメタの設定（J2EE サーバ）」を参照してください。

(2)　バッチサーバの設定

バッチサーバの設定は，簡易構築定義ファイルで実施します。

ユーザ拡張性能解析トレースの定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に JavaVM 起動パラメタ（add.jvm.arg）で指定します。指定するパラメタ値は，「(1)　J2EE サーバの設定」を参照してください。

(3)　EJB クライアントアプリケーションの設定

cjclstartap コマンドで動作させる EJB クライアントアプリケーションの設定は，EJB クライアントアプリケーション用オプション定義ファイル（usrconf.cfg）で実施します。

ユーザ拡張性能解析トレースの定義は，EJB クライアントアプリケーション用オプション定義ファイル（usrconf.cfg）の JavaVM 起動パラメタ（add.jvm.arg）で指定します。指定するパラメタ値は，「(1) J2EE サーバの設定」を参照してください。

EJB クライアントアプリケーション用オプション定義ファイル（usrconf.cfg）での定義例を次に示します。

```
   :
add.jvm.arg=-Djvm.userprf.Enable=true
add.jvm.arg=-Djvm.userprf.File=<ユーザ定義ファイル格納ディレクトリ>※/userprf.cfg
add.jvm.arg=-Djvm.userprf.LogLevel=class
   :
```

注※

ユーザ定義ファイル格納先環境変数（CJCLUSRCONFDIR）で指定したディレクトリです。ユーザ定義ファイル格納先環境変数が設定されていない場合，カレントディレクトリが参照されます。

(4)　ユーザ拡張性能解析トレース設定ファイルの設定

ユーザ拡張性能解析トレースを使用するためには，ユーザ拡張性能解析トレース設定ファイルを作成して，トレース対象メソッドの設定が必要です。

トレース対象メソッドの設定は，簡易構築定義ファイルの論理 J2EE サーバ（J2EE-Server）の<configuration>タグ内に UserPrfText パラメタで指定します。

定義例を次に示します。

```
   :
<param>
<param-name>UserPrfText</param-name>
<param-value>
```

```
<![CDATA[
org.apache.struts.action.Action.execute(*), struts, true
Info.getInfo(*), struts, true
]]>
</param-value>
</param>
   :
```

ユーザ拡張性能解析トレース設定ファイルの作成方法については，「7.5.3　ユーザ拡張性能解析トレースのトレース対象メソッドの設定」を参照してください。

参考

ユーザ拡張性能解析トレース設定ファイルは，運用管理ポータルの［起動パラメタの設定］画面（論理 J2EE サーバの定義）またはユーザ任意のファイル（jvm.userprf.File プロパティで指定したファイル）でも設定できます。

7.5.3　ユーザ拡張性能解析トレースのトレース対象メソッドの設定

ユーザ拡張性能解析トレースのトレース対象メソッドの設定は，ユーザ拡張性能解析トレース設定ファイルを使って設定します。ユーザ拡張性能解析トレース設定ファイルの内容は次のどれかに記述できます。

- 簡易構築定義ファイル
- 運用管理ポータルの［起動パラメタの設定］画面（論理 J2EE サーバの定義）
- デフォルトのファイル，またはユーザが指定した任意のファイル（jvm.userprf.File プロパティで指定したファイル）

(1)　記述形式

ユーザ拡張性能解析トレース設定ファイルの記述形式を次に示します。

```
<指定形式>※,<識別ID>,<サブクラスフラグ> [, [<イベントID>] [, [<トレース取得レベル>] ] ] [ #コメント]
```

注 1

［ ］で囲んだ項目は省略できます。

注 2

クラス名およびパッケージ名が一致するすべてのメソッドを対象とする場合，jvm.userprf.ExtendedSetting プロパティの値を true に設定してください。

注※

<指定形式>は次のどれかの形式で指定します。

- メソッド名と引数の型が一致するメソッドをトレースの対象とする場合

<パッケージ名>.<クラス名>.<メソッド名>(メソッドの引数の型名)

- メソッド名が一致するメソッドをトレースの対象とする場合
  <パッケージ名>.<クラス名>.<メソッド名>(*)
- クラス名が一致するすべてのメソッドをトレースの対象とする場合
  <パッケージ名>.<クラス名>
- パッケージ名が一致するすべてのメソッドをトレースの対象とする場合
  <パッケージ名>.*

記述形式の例

```
com.sample.Test.method(),TEST1,false,0xae02,A
com.sample.Test.method(),TEST1,false,0xae02
com.sample.Test.method(),TEST1,false,0xae02,
com.sample.Test.method(),TEST1,false,,A
com.sample.Test.method(),TEST1,false
com.sample.Test.method(),TEST1,false,
com.sample.Test.method(),TEST1,false,,
```

(2) 記述項目

ユーザ拡張性能解析トレース設定ファイルの記述項目を次に示します。

指定形式

トレース対象メソッドを指定します。

パッケージ名

トレース対象メソッドのクラスまたはインタフェースのパッケージ名を指定します。

クラス名

トレース対象メソッドのクラス名またはインタフェース名を指定します。クラス名の代わりにインタフェース名を指定し，かつ，サブクラスフラグに true を指定した場合，そのインタフェースを実装しているメソッドがトレース対象となります。

メソッド名

トレース対象メソッドの名前を指定します。

メソッドの引数の型名

トレース対象メソッドの引数の型を完全修飾名で指定します。

識別 ID

トレース対象メソッドを識別するための文字列を指定します。

使用できる文字は ASCII 文字で 0x21（!）から 0x7e（~）までです。ただし，0x22（"），0x23（#），および 0x2c（,）は指定できません。

識別 ID は PRF トレースファイルに出力されます。なお，PRF トレースファイルに出力される文字数は 32 文字までです。33 文字目以降は省略されて，33 文字目に*が出力されます。

注意事項

トレース対象メソッドが定義されているクラスにサブクラスが存在し，そのサブクラスでトレース対象メソッドがオーバーライドされていない場合，スーパークラスのトレース対象メソッドに対して指定された識別 ID が出力されます。

サブクラスフラグ

指定したメソッドのクラスまたはインタフェースと継承関係にあるクラスのメソッドをトレース対象に含めるかどうかを true または false で指定します。

- true を指定した場合，指定したメソッドと，それをオーバーライドしているメソッドがトレース対象となります。
- false を指定した場合，指定したメソッドだけがトレース対象となり，指定したメソッドをオーバーライドしているメソッドはトレース対象となりません。

イベント ID

トレース情報を取得するポイント（トレース取得ポイント）を 16 進数（0xae02～0xae7e および 0xc000～0xcffe）の値で指定します。デフォルト値は 0xae00 です。

メソッドの入口で出力されるトレース情報にはこの値が出力され，メソッドの出口で出力されるトレース情報にはこの値＋ 1 が出力されます。

トレース取得レベル

トレース対象メソッドのトレース取得レベルに，A，B，C，またはそれぞれの小文字のどれかを指定します。デフォルト値は A です。

- A：標準レベル
- B：詳細レベル
- C：保守レベル

参考

トレース取得レベルは cprflevel コマンドの PRF トレース取得レベルと同じです。トレース対象メソッドはすべて Java VM レイヤのレベルを参照します。PRF 取得レベルとレイヤについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cprflevel（PRF トレース取得レベルの表示と変更）」を参照してください。

コメント

コメントは「#」で始まり，「#」から行末までの間の文字すべてをコメントとします。

(3) 記述規則

ユーザ拡張性能解析トレース設定ファイルの記述規則を次に示します。

- 記述にはシングルバイト文字だけが使用できます。

- 空白文字は，半角スペース文字（「0x20」）またはタブ文字（「¥t」もしくは「0x09」）となります。なお，ユーザ拡張性能解析トレース設定ファイルの読み込みでは空白文字は無視されます。
- トレース対象メソッドは 1 行に一つずつ記述します。
- 1 行に指定できる文字数は 2,048 文字までです。この文字数には空白やコメントを含みます。
- 行末は改行文字（「¥n」もしくは「0x0A」）または復帰文字（「¥r」もしくは「0x0D」）が 1 文字以上続いたものとなります。
- ユーザ拡張性能解析トレース設定ファイルの記述に誤りがある場合，誤りの内容を示すメッセージが出力されます。また，各項目に無効な値を記述した場合，誤りの内容を示すメッセージが出力され，その行の設定は無効になります。

  メッセージについては，「7.6　ユーザ拡張性能解析トレース実行時に出力されるログ情報」を参照してください。
- ユーザ拡張性能解析トレースでは，JavaVM 内のクラスや Cosminexus 内のクラスはトレース対象のメソッドに指定できません。次のパッケージが該当します。
  - java 以下
  - javax 以下
  - com.hitachi 以下
  - JP.co.Hitachi 以下

  そのため，jvm.userprf.ExtendedSetting プロパティを指定して，パッケージ名指定をする場合は，アプリケーションのクラスだけが含まれるようにパッケージ名を記述してください。
- 次のメソッドはトレース対象のメソッドに指定できません。
  - 存在しないパッケージ名，クラス名，およびメソッド名
  - native メソッド
  - abstract メソッド
  - JavaVM 内のクラス，およびそのクラスのメソッド
    （例）java および javax で始まるパッケージのクラス
  - -Xbootclasspath で指定したクラス，およびそのクラスのメソッド
  - 製品内のクラス
- 次のメソッドは次の記述でトレース対象のメソッドに指定できます。
  - コンストラクタは，クラス名と同じメソッド名，または<init>で指定できます。
    （例）MyMain クラスのコンストラクタの場合，次のように指定します。
    MyMain.MyMain()または MyMain.<init>()
  - コンストラクタではないクラス名と同じ名前のメソッドを指定した場合は，コンストラクタを指定しているのか，メソッドを指定しているのか判別できないためコンストラクタとメソッドがトレースの対象となります。
  - 引数が可変長であるメソッドを指定する場合は，可変長の引数を配列として記述してください。

（例）引数が可変長であるメソッドの指定

正しい指定例：com.sample.Test.method(java.lang.String[])

誤った指定例：com.sample.Test.method(java.lang.String...)

- ネストクラスは，「.」ではなく「$」で区切った名前を指定してください。

  （例）ネストクラスの指定

  正しい指定例：com.sample.Test$NestClass

  誤った指定例：com.sample.Test.NestClass

- ジェネリクスではなく，パラメタ化されていないクラス名（raw 型）であれば指定できます。

  （例）ジェネリクスではないクラスの指定

  正しい指定例：com.sample.Test.method()

  誤った指定例：com.sample.Test<java.lang.String,java.lang.Object>.method()

- <指定形式>の<クラス名>にクラス名を指定した場合とインタフェース名を指定した場合ではトレース対象メソッドが異なります。

  クラスまたはインタフェースの指定によるトレース対象メソッドを次の表に示します。

表 7-8　クラスまたはインタフェースの指定によるトレース対象メソッド

クラスまたはインタフェース	トレース対象メソッド	
	サブクラスフラグに false を指定した場合	サブクラスフラグに true を指定した場合
クラス	指定したクラスのメソッド	指定したクラスのメソッドと，そのメソッドをオーバーライドしているメソッド
インタフェース	なし	指定したインタフェースを直接実装しているクラス※1 と間接実装しているクラス※2 のメソッド，およびそのメソッドをオーバーライドしているメソッド

注※1　クラスの宣言時に implements を使用して指定したインタフェースを実装しているクラスです。

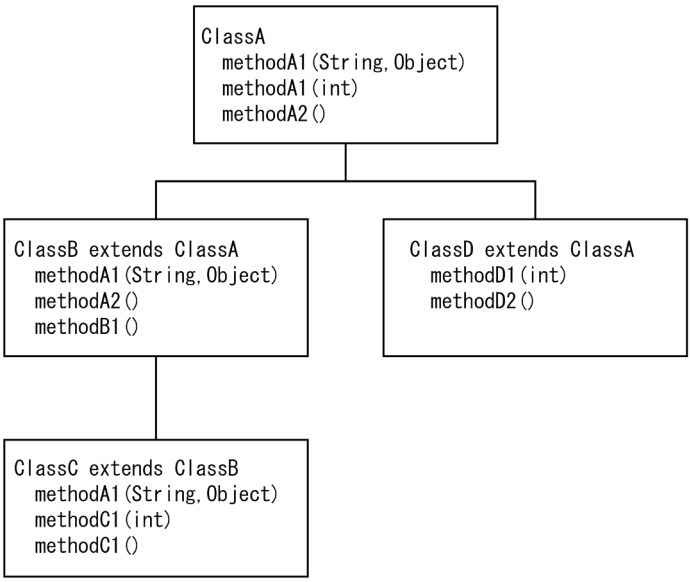
注※2　指定したインタフェースを直接実装しているクラスのサブクラス，または指定したインタフェースを継承しているインタフェースを直接実装しているクラスです。

(4)　ユーザ拡張性能解析トレース設定ファイルの記述例

ここでは，トレース対象メソッドごとに，ユーザ拡張性能解析トレース設定ファイルの記述例について説明します。

なお，ここで説明する記述例は，パッケージ名が「com.sample」で，次の図に示すクラス構造を持っているアプリケーションとします。

図 7-8　アプリケーションのクラス構造の例



(a)　メソッド名と引数の型が一致するメソッドをトレースの対象とする場合

メソッド名と引数の型が一致するメソッドをトレースの対象とする場合のユーザ拡張性能解析トレース設定ファイルの記述例を次に示します。

- サブクラスフラグが false の場合

```
com.sample.ClassA.methodA1(java.lang.String,java.lang.Object),1000,false
```

サブクラスフラグが false の場合，記述したメソッド名と引数の型が一致するメソッドがトレース対象となります。

トレース対象のメソッド

・ClassA クラスの methodA1(String,Object)

トレース対象メソッドに対してイベント ID の設定を省略しているため，トレース対象となったメソッドが呼び出されると，イベント ID としてメソッドの入口で 0xae00，メソッドの出口で 0xae01 のデフォルト値が出力されます。また，識別 ID に「1000」を設定しているため，識別 ID として「1000」が出力されます。

- サブクラスフラグが true の場合

```
com.sample.ClassA.methodA2(),2000,true
```

サブクラスフラグが ture の場合，メソッド名と引数の型が記述と一致するメソッドに加えて，記述したメソッドをオーバーライドしているメソッドもトレース対象となります。

トレース対象のメソッド

・ClassA クラスの methodA2()

・ClassA クラスの methodA2()をオーバーライドしている ClassB クラスの methodA2()

トレース対象メソッドに対してイベント ID の設定を省略しているため，トレース対象となったメソッドが呼び出されると，イベント ID としてメソッドの入口で 0xae00，メソッドの出口で 0xae01 のデ

フォルト値が出力されます。また，識別 ID に「2000」を設定しているため，トレース対象のすべてのメソッドに識別 ID として「2000」が出力されます。

(b) メソッド名が一致するメソッドをトレースの対象とする場合

メソッド名が一致するメソッドをトレースの対象とする場合のユーザ拡張性能解析トレース設定ファイルの記述例を次に示します。

- サブクラスフラグが false の場合

```
com.sample.ClassA.methodA1(*),methodA1,false,0xae30
```

サブクラスフラグが false の場合，記述したメソッド名と一致するメソッドのすべてがトレース対象となります。

トレース対象のメソッド

・ClassA クラスの methodA1(String,Object)

・ClassA クラスの methodA1(int)

トレース対象となったメソッドが呼び出されると，イベント ID としてメソッドの入口で 0xae30，メソッドの出口で 0xae31 が出力されます。また，トレース対象のすべてのメソッドに識別 ID として「methodA1」が出力されます。

- サブクラスフラグが true の場合

```
com.sample.ClassA.methodA1(*),methodA1,true,0xae30
```

サブクラスフラグが true の場合，記述したメソッド名と一致するメソッドのすべてがトレース対象となります。また，記述したメソッドに加えて，記述したメソッドをオーバーライドしているメソッドもトレース対象となります。

トレース対象のメソッド

・ClassA クラスの methodA1(String,Object)と methodA1(int)

・ClassA クラスの methodA1(String,Object)をオーバーライドしている ClassB クラスの methodA1(String,Object)

・ClassB クラスの methodA1(String,Object)をオーバーライドしている ClassC クラスの methodA1(String,Object)

トレース対象となったメソッドが呼び出されると，イベント ID としてメソッドの入口で 0xae30，メソッドの出口で 0xae31 が出力されます。また，トレース対象のすべてのメソッドに識別 ID として「methodA1」が出力されます。

(c) クラス名が一致するすべてのメソッドをトレースの対象とする場合

メソッドと引数を省略して，クラス名が一致するすべてのメソッドをトレースの対象とする場合のユーザ拡張性能解析トレース設定ファイルの記述例を次に示します。

- サブクラスフラグが false の場合

```
com.sample.ClassA,TEST01,false
```

サブクラスフラグが false の場合，記述したクラス名（ClassA クラス）のすべてのメソッドがトレース対象となります。

トレース対象のメソッド

ClassA クラスの methodA1(String,Object)，methodA1(int)，および methodA2()

トレース対象となったメソッドが呼び出されると，イベント ID が省略されているため，メソッド入口で 0xae00，メソッド出口で 0xae01 が出力されます。また，これらすべてのメソッドに識別 ID として「TEST01」が出力されます。

- サブクラスフラグが true の場合

```
com.sample.ClassB,TEST02,true
```

サブクラスフラグが true の場合，記述したクラス名（ClassB クラス）のすべてのメソッドと，そのメソッドをオーバーライドしているメソッドのすべてがトレース対象となります。

トレース対象のメソッド

・ClassB クラスの methodA1(String,Object)

・ClassB クラスの methodA2()

・ClassB クラスの methodB1()

・ClassB クラスの methodA1(String,Object)をオーバーライドしている ClassC クラスの methodA1(String,Object)

トレース対象となったメソッドが呼び出されると，イベント ID が省略されているため，メソッド入口で 0xae00，メソッド出口で 0xae01 が出力されます。また，これらすべてのメソッドに識別 ID として「TEST02」が出力されます。

(d) パッケージ名が一致するすべてのメソッドをトレースの対象とする場合

クラス名とメソッド名と引数を省略して，パッケージ名が一致するすべてのクラスのすべてのメソッドをトレースの対象とする場合のユーザ拡張性能解析トレース設定ファイルの記述例を次に示します。

注意事項

この指定ではサブパッケージも対象に含まれます。トレース対象となった場合，対象となったメソッドが呼び出されるとトレースが出力されます。

com.sample パッケージにサブパッケージがある場合，そのサブパッケージのすべてのクラスのすべてのメソッドもトレース対象となります。

- サブクラスフラグが false の場合

```
com.sample.*,6000,false
```

サブクラスフラグが false の場合，記述したパッケージ（com.sample）にあるすべてのクラスのすべてのメソッドがトレース対象となります。

トレース対象のメソッド

ClassA，ClassB，ClassC，ClassD のすべてのメソッドがトレース対象となります。

トレース対象となったメソッドが呼び出されると，イベント ID が省略されているため，メソッド入口で 0xae00，メソッド出口で 0xae01 が出力されます。また，これらすべてのメソッドに識別 ID として「6000」が出力されます。

- サブクラスフラグが true の場合

```
com.sample.*,6000,false
```

サブクラスフラグが true の場合，記述したパッケージ（com.sample）にあるすべてのクラスのすべてのメソッドと，そのメソッドをオーバーライドしているメソッドのすべてがトレース対象となります。

トレース対象のメソッド

ClassA，ClassB，ClassC，ClassD のすべてのメソッドと，そのメソッドをオーバーライドしているすべてのメソッドもトレース対象となります。

トレース対象となったメソッドが呼び出されると，イベント ID が省略されているため，メソッド入口で 0xae00，メソッド出口で 0xae01 が出力されます。また，これらすべてのメソッドに識別 ID として「6001」が出力されます。

(5) ユーザ拡張性能解析トレース設定ファイル作成の注意事項

ユーザ拡張性能解析トレース設定ファイルを作成するときの注意事項を次に示します。

- ユーザ拡張性能解析トレース設定ファイルで，同じメソッドに対して複数行で異なるイベント ID または識別 ID を指定した場合，ユーザ拡張性能解析トレース設定ファイルの前方に記述された設定が優先されます。
- ユーザ拡張性能解析トレース設定ファイルで，複数のメソッドを対象とする記述をした場合，対象となるすべてのメソッドが同一イベント ID または識別 ID で出力されます。その場合，出力されるメソッド名が出力できる文字数を超えるとメソッドを特定できないことがあります。メソッドを一つに特定できるように記述してください。
- トレース対象にインタフェースを指定した場合，サブクラスフラグに true を指定してください。インタフェースにはトレース対象となるメソッドが存在しないため，サブクラスフラグに false を指定するとトレース情報は出力されません。
- 複数のメソッドをユーザ拡張解析トレースの対象とする指定形式は，アプリケーションの動作を把握したい場合などに限り指定してください。

  ユーザ拡張性能解析トレース設定ファイルの記述で，サブクラスフラグに true を指定した場合や，メソッド名が一致するメソッドをトレースの対象とする指定形式で指定した場合，ユーザが意図しない多くのメソッドがユーザ拡張性能解析トレースの対象となり，性能劣化要因の特定が困難になることがあります。性能劣化要因の特定のためには，ユーザ拡張性能解析トレース設定ファイルのサブクラスフラ

グに false を指定する，またはメソッド名と引数の型が一致するメソッドをトレースの対象とする指定形式で指定するなどで，ユーザ拡張性能解析トレースの対象を限定することを推奨します。

7.6　ユーザ拡張性能解析トレース実行時に出力されるログ情報

ユーザ拡張性能解析トレースの実行時に，JavaVM ログファイルに次のログが出力されます。

- ユーザ拡張性能解析トレース設定ファイルの読み込み時のログ
- アプリケーションの書き換え時のログ

それぞれのログについて説明します。

7.6.1　ユーザ拡張性能解析トレース設定ファイルの読み込み時のログ

ユーザ拡張性能解析トレースでユーザ拡張性能解析トレース設定ファイルの読み込み時に出力されるログ情報を次の表に示します。

表 7-9　ユーザ拡張性能解析トレース設定ファイルの読み込み時に出力されるログ

出力形式	出力内容	説明
[UPR※1]<DATE>Setting file not found.<file=FILEPATH>	• DATE※2 ユーザ拡張性能解析トレース設定ファイルの読み込みに失敗した日時。 • FILEPATH 読み込みに失敗したユーザ拡張性能解析トレース設定ファイルの絶対ファイルパス。	デフォルト，または jvm.userprf.File プロパティで指定したユーザ拡張性能解析トレース設定ファイルが存在しません。
[UPR※1]<DATE>Failed to open setting file.<file=FILEPATH>	• DATE※2 ユーザ拡張性能解析トレース設定ファイルの読み込みに失敗した日時。 • FILEPATH 読み込みに失敗したユーザ拡張性能解析トレース設定ファイルの絶対ファイルパス。	ユーザ拡張性能解析トレース設定ファイルのオープンまたは読み込みができません。
[UPR※1]<DATE>Failed to parse setting file.<file=FILEPATH><line=LINE>	• DATE※2 ユーザ拡張性能解析トレース設定ファイルの誤りを検出した日時。 • FILEPATH ユーザ拡張性能解析トレース設定ファイルの絶対ファイルパス。 • LINE ユーザ拡張性能解析トレース設定ファイルの行番号。	ユーザ拡張性能解析トレース設定ファイルの内容に記述フォーマット上の誤りがあります。
[UPR※1]<DATE>Event ID is invalid value.<file=FILEPATH><eventID=EventID>	• DATE※2 ユーザ拡張性能解析トレース設定ファイルの誤りを検出した日時。	ユーザ拡張性能解析トレース設定ファイルのイベント

出力形式	出力内容	説明
	• FILEPATH ユーザ拡張性能解析トレース設定ファイルの絶対ファイルパス。 • EventID：ユーザ拡張性能解析トレース設定ファイルに指定したイベントID。	ID が有効な値の範囲外です。
[UPR※1]<DATE>No valid settings in setting file.<file=FILEPATH>	• DATE※2 ユーザ拡張性能解析トレース設定ファイルに有効な設定がないことを検出した日時。 • FILEPATH ユーザ拡張性能解析トレース設定ファイルの絶対ファイルパス。	ユーザ拡張性能解析トレース設定ファイルに有効な設定がありません。
[UPR※1]<DATE>User Extended PRF started successfully.<file=FILEPATH>	• DATE※2 ユーザ拡張性能解析トレース設定ファイルの読み込みに成功した日時。 • FILEPATH 有効に読み込みが行われたユーザ拡張性能解析トレース設定ファイルの絶対ファイルパス。	ユーザ拡張性能解析トレース設定ファイルが正常に読み込まれ，ユーザ拡張性能解析トレースが有効になりました。

注※1　ユーザ拡張性能解析トレースが出力したログであることを示す識別子です。

注※2　拡張 verbosegc 情報と同じ形式で出力します。

7.6.2　アプリケーションの書き換え時のログ

ユーザ拡張性能解析トレースでは性能解析トレースを出力するために，インストゥルメンテーション機能を利用して，クラスロード時にアプリケーションのクラスを書き換えます。このときに出力するログを次の表に示します。

表 7-10　アプリケーションの書き換え時のログ

出力形式	出力内容	説明
[UPR※1]<DATE>BCI process failed.<class=CLASS>	• DATE※2 書き換えに失敗した日時。 • CLASS 書き換えに失敗した完全修飾クラス名。	ユーザ拡張性能解析トレース設定ファイルに指定されたクラスの書き換えに失敗しました。
[UPR※1]<DATE>BCI process finished successfully.<class=CLASS>	• DATE※2 書き換えに成功した日時。 • CLASS 書き換えに成功した完全修飾クラス名。	ユーザ拡張性能解析トレース設定ファイルに指定されたクラスの書き換えに成功しました。なお，このログは jvm.userprf.Trace プ

出力形式	出力内容	説明
		ロパティに true を指定している場合に出力します。
[UPR※1]<DATE>BCI count exceeded a limit.	• DATE※2 書き換え数が書き換え数の上限値を超えた日時。	jvm.userprf.Limit プロパティで指定した数を超えてトレース対象メソッドを書き換えようとしました。次のどちらかで対処してください。 • ユーザ拡張性能解析トレースの対象メソッド数を減らす。 • jvm.userprf.Limit プロパティで指定した値を大きくする。 なお，このログは jvm.userprf.Limit プロパティで指定した値を初めて超えたときに出力します。

注※1　ユーザ拡張性能解析トレースが出力したログであることを示す識別子です。

注※2　拡張 verbosegc 情報と同じ形式で出力します。

7.7　性能解析トレースファイルを使用した処理性能の解析作業

この節では，性能解析トレースファイルを使用した処理性能の解析作業について説明します。

7.7.1　処理性能の解析作業の概要

アプリケーションサーバの処理性能は，クライアントからデータベースなどの EIS までの一連の処理，およびその処理結果がクライアントに返却されるまでのリクエストの一連の処理で，EJB コンテナや Web コンテナなどの機能レイヤから出力されるトレース情報を基に解析できます。

各機能レイヤのトレース情報は，バイナリ形式で PRF トレースファイルに出力されます。アプリケーションサーバの処理性能を解析する際は，PRF トレースファイルをテキスト形式（CSV 形式）に変換した性能解析トレースファイルを使用します。

次に，アプリケーションサーバの処理性能を解析する作業について説明します。

参考

保守レベルのトレース情報について

PRF トレース取得レベルには，標準レベルと詳細レベルのほかに，**保守レベル**があります。保守レベルとは，障害発生時などの保守情報を取得するためのレベルを指します。通常は指定しないでください。

PRF トレース取得レベルに保守レベルを設定する方法については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cprflevel（PRF トレース取得レベルの表示と変更）」を参照してください。

(1)　性能解析トレースファイルの収集

アプリケーションサーバの処理性能の解析で使用する性能解析トレースファイルを，運用管理コマンド（mngsvrutil）を使用して出力，収集します。性能解析トレースファイルの収集方法，性能解析トレースファイルの出力先および出力情報については，「7.3　Management Server を利用した性能解析トレースファイルの収集」を参照してください。

(2)　性能解析トレースファイルを利用したアプリケーションサーバの処理性能の解析

性能解析トレースファイルを使用して性能を解析する場合，CSV 形式のファイルを編集できるアプリケーションプログラムで表示して，目的に合わせてフィルタリングや並べ替えの機能を利用します。

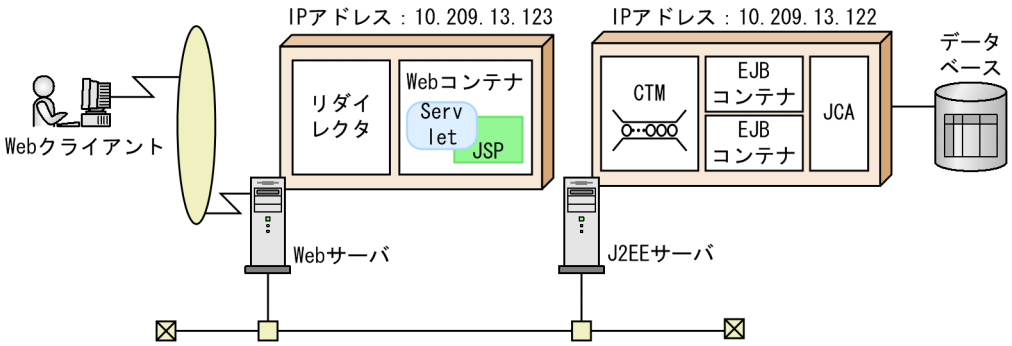
例えば，CSV 形式で出力すると，CSV 形式のファイルを編集できるアプリケーションプログラムで「イベントごと」「プロセスごと」などによって出力項目をフィルタリングして，注目する個所を絞り込んで，目的に合わせた解析ができます。

次に示す性能解析トレースファイルの利用方法について，例を使用して説明します。

- Web サーバのレスポンスタイムの解析（7.7.2 参照）
- アプリケーションサーバ内でのリクエストの処理状況の調査（7.7.3 参照）
- セッションのライフサイクルの調査（7.7.4 参照）
- タイムアウトが発生したトランザクションの特定（7.7.5 参照）
- タイムアウトが発生したリクエストの特定（7.7.6 参照）
- ルートアプリケーション情報を利用したログ調査（7.7.7 参照）
- トラブルが発生したコネクションの特定（7.7.8 参照）
- 性能解析トレースとスレッドダンプの対応づけた問題個所の調査（7.7.9 参照）

なお，ここでは，次のような Web クライアント構成の環境で性能解析トレースファイルを収集したことを前提とします。

図 7-9　Web クライアント構成の例



7.7.2　Web サーバのレスポンスタイムの解析

ここでは，Web サーバがクライアントからリクエストを受けてから返却するまでに掛かった時間を解析する方法を，例を使用して説明します。

Web サーバのアクセスログでリクエスト処理に掛かった時間を出力することができます。詳細は，マニュアル「HTTP Server」の「6.2.3(6)　CustomLog」ディレクティブを参照してください。

デフォルトのコンフィグファイルで指定しているアクセスログの出力例を次に示します。

（出力例）

```
192.168.10.10 - - [12/Feb/2020:20:52:16.352 +0900] "GET /index.html HTTP/1.1" 200 53 0.013 1
1340 "192.168.10.10/5636/0x0000000000000022"
```

```
192.168.10.10 - - [12/Feb/2020:20:52:19.439 +0900] "GET /wait.pl HTTP/1.1" 504 319 3.004 125
48 "192.168.10.10/5636/0x0000000000000023"
```

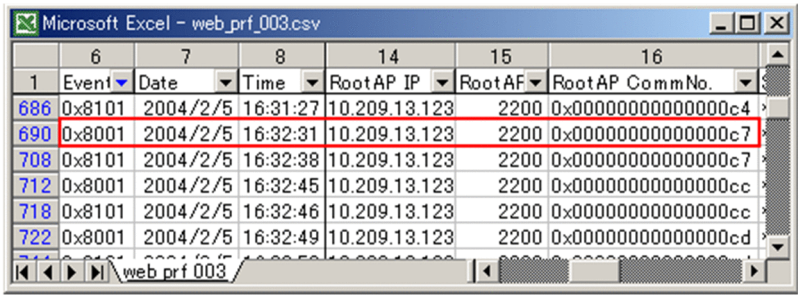
出力例では index.html へのリクエスト処理に 0.013 秒が掛かり，wait.pl へのリクエスト処理に 3.004 秒が掛かっています。また，それぞれのメッセージに出力されているルートアプリケーション情報を，性能解析トレースファイルに出力されたルートアプリケーション情報と突き合わせることで，各機能レイヤごとの処理時間を解析できます。

7.7.3　アプリケーションサーバ内でのリクエストの処理状況の調査

ここでは，Web サーバがクライアントから受けたリクエストがアプリケーションサーバで処理されているかどうか調査する方法を，例を使用して説明します。

次に示す Web サーバで収集した性能解析トレースファイルから，2004/2/5 16:32:31 にリクエストを受け付けたことがわかります。

図 7-10　Web サーバで収集した性能解析トレースファイルの例

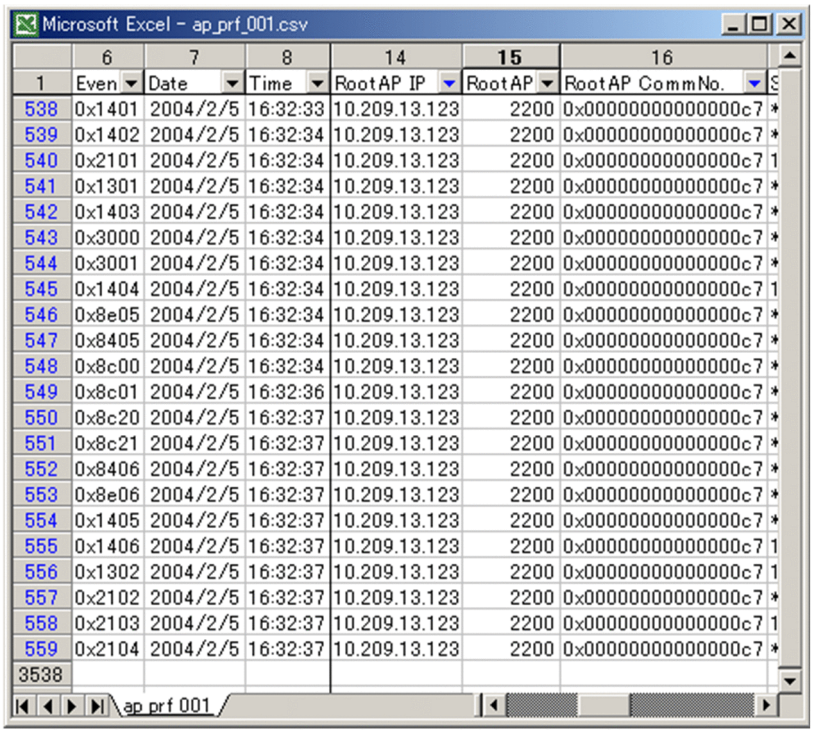


	6	7	8	14	15	16
1	Event	Date	Time	RootAP IP	RootAP	RootAP CommNo.
686	0x8101	2004/2/5	16:31:27	10.209.13.123	2200	0x00000000000000c4
690	0x8001	2004/2/5	16:32:31	10.209.13.123	2200	0x00000000000000c7
708	0x8101	2004/2/5	16:32:38	10.209.13.123	2200	0x00000000000000c7
712	0x8001	2004/2/5	16:32:45	10.209.13.123	2200	0x00000000000000cc
718	0x8101	2004/2/5	16:32:46	10.209.13.123	2200	0x00000000000000cc
722	0x8001	2004/2/5	16:32:49	10.209.13.123	2200	0x00000000000000cd

このリクエストのルートアプリケーション情報をキーにして，アプリケーションサーバで収集した性能解析トレースファイルを調査します。

「RootAP IP：10.209.13.123」，「RootAP ID：2200」，および「RootAP CommNo.：0x00000000000000c7」をキーにして，アプリケーションサーバで収集した性能解析トレースファイルをフィルタリングした例を次に示します。

図 7-11　アプリケーションサーバで収集した性能解析トレースファイルをフィルタリングした例



Microsoft Excel - ap_prf_001.csv

	6	7	8	14	15	16	
1	Even	Date	Time	RootAP IP	RootAP	RootAP CommNo.	S
538	0x1401	2004/2/5	16:32:33	10.209.13.123	2200	0x00000000000000c7	*
539	0x1402	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
540	0x2101	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	1
541	0x1301	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
542	0x1403	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
543	0x3000	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
544	0x3001	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
545	0x1404	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	1
546	0x8e05	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
547	0x8405	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
548	0x8c00	2004/2/5	16:32:34	10.209.13.123	2200	0x00000000000000c7	*
549	0x8c01	2004/2/5	16:32:36	10.209.13.123	2200	0x00000000000000c7	*
550	0x8c20	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
551	0x8c21	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
552	0x8406	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
553	0x8e06	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
554	0x1405	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
555	0x1406	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	1
556	0x1302	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	1
557	0x2102	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
558	0x2103	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	1
559	0x2104	2004/2/5	16:32:37	10.209.13.123	2200	0x00000000000000c7	*
3538							

ap_prf_001

このようにアプリケーションサーバで収集した性能解析トレースファイルを，ルートアプリケーション情報をキーにフィルタリングすると，アプリケーションサーバ内でのリクエストの一連の処理を調査できます。

7.7.4　セッションのライフサイクルの調査

ここでは，性能解析トレースを利用したセッションのライフサイクルの調査方法について説明します。

セッションのライフサイクルは，PRF トレース取得レベルを「詳細」に設定している場合に出力されるトレース（**セッショントレース**）を利用して調査できます。

セッショントレース情報は，出力された性能解析トレースファイルのセッション ID をキーにして調査できます。

セッション ID は，セッショントレース情報として出力されたイベントのオプション領域に，次の形式で出力されます。

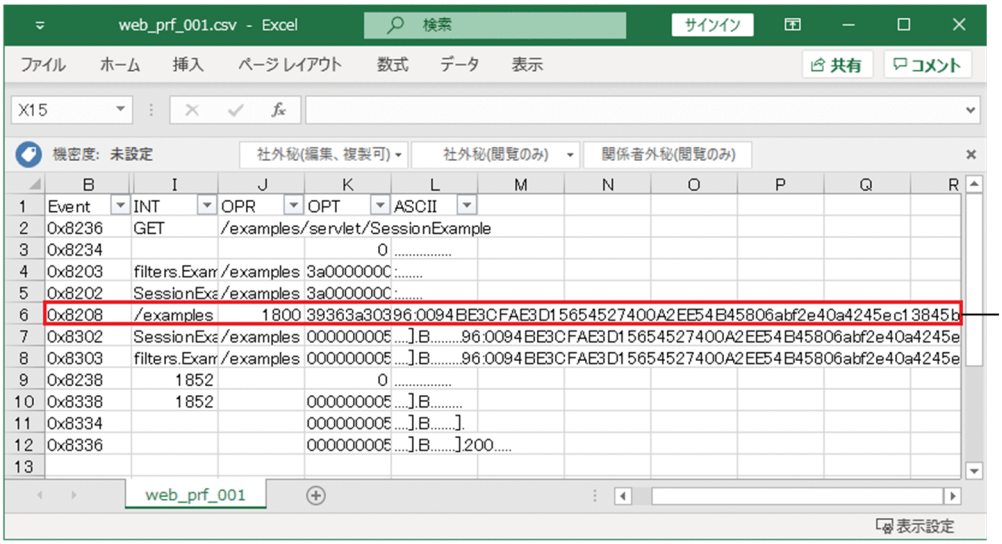
```
セッション文字数:セッションID
```

例えば，「abc123」というセッション ID の場合，オプション領域には「6:abc123」と出力されます。セッション ID が空文字の場合は，セッション文字数として「0:」と出力されます。セッション ID を取得できなかった場合は，何も出力されません。

セッショントレース情報が出力されている性能解析トレースファイルの例を次の図に示します。この例は，一つのセッションの有効期間内での，セッションを生成するリクエスト，セッションを利用するリクエス

ト，およびセッションを破棄するリクエストのトレース情報を出力したものです。また，ここでは，セッションを生成するリクエスト，セッションを利用するリクエスト，およびセッションを破棄するリクエストを分けて示します。実際はこれら三つの出力例は，続けて出力されます。また，これらの画面では，セッショントレースに関係ない出力項目は表示していません。

図 7-12　セッショントレース情報が出力されている性能解析トレースファイルの例（セッションを生成するリクエスト部分）



1. イベント「0x8208」の ASCII の列に，生成されたセッション ID が出力されます。オペレーション列の 1800 は，セッションの有効期間を示します。

このトレースファイルで確認できることについて説明します。

セッションの有効期間の確認

セッション生成時には，イベント「0x8208」が出力されます。セッション破棄時には，イベント「0x8209」が出力されます。これらの取得時刻から，セッションの有効範囲を確認できます。また，イベント「0x8208」では，オペレーション名から生成されたセッションの有効期間（秒）も確認できます。

セッション ID は，セッションを破棄するリクエスト内のイベント「0x8209」でセッションが破棄されるまで，同じ ID が出力されます。なお，イベント「0x8209」のオペレーション名から破棄されたセッションの生成時刻も確認できます。

7.7.5　タイムアウトが発生したトランザクションの特定

ここでは，J2EE アプリケーションまたはバッチアプリケーションのトランザクションがタイムアウトした場合に，メッセージまたは性能解析トレースファイルを使用してタイムアウトしたトランザクションを特定する方法について説明します。

ここでは，次の 2 種類の方法について説明します。

- メッセージを利用した特定方法

- 性能解析トレースファイルの出力内容による特定方法

(1) メッセージを利用した特定方法

トランザクションでタイムアウトが発生した場合，メッセージ KDJE31002-W，および KDJE50080-W が出力されます。これらのメッセージには，次の情報が出力されます。

KDJE31002-W

- トランザクションを開始した J2EE アプリケーション名，またはバッチアプリケーションのクラス名
- トランザクションを開始した J2EE コンポーネント（Enterprise Bean，サーブレットまたは JSP）のクラス名とインスタンスのハッシュコード
- 保守情報
- 性能解析トレースのルートアプリケーション情報

メッセージに出力された性能解析トレースのルートアプリケーション情報を，性能解析トレースファイルに出力されたルートアプリケーション情報と突き合わせて確認することで，性能解析トレース中のどこでトランザクションタイムアウトが発生したかを確認できます。

また，その前後のトレース情報を確認して，タイムアウトが発生したトランザクションでの処理内容を確認できます。

KDJE50080-W

- IP アドレス，プロセス ID，ルートアプリケーション情報
- SQL を実行したコネクションのコネクション ID
- SQL を実行したメソッド名
- SQL を実行したメソッドが実行中かどうか
- 最後に実行された SQL

メッセージに出力された SQL の情報を参照することで，タイムアウトが発生した原因となった個所を調査できます。

なお，メッセージの詳細については，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」の「KDJE31002-W」，およびマニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」の「KDJE50080-W」を参照してください。

(2) 性能解析トレースファイルの出力内容による特定方法

性能解析トレースファイルには，トランザクションタイムアウトが発生する直前と直後のトレース取得ポイントで，トレース情報が出力されています。

次のイベント ID の処理内容を確認してください。

表 7-11　トランザクションタイムアウト時に出力される性能解析トレース

イベント ID	説明
0x8819	トランザクションタイムアウトの直前の処理で出力される情報です。インタフェース名に，タイムアウトするトランザクションのルートアプリケーション情報が出力されます。
0x8820	トランザクションタイムアウトの直後の処理で出力される情報です。
0x8C41	障害調査用 SQL が出力されたときに出力される情報です。

7.7.6　タイムアウトが発生したリクエストの特定

ここでは，Web サーバのリバースプロキシでレスポンス受信時にタイムアウトが発生した場合に，性能解析トレースを利用してタイムアウトが発生したリクエストを特定する方法について説明します。

リバースプロキシでのレスポンス受信時にタイムアウトが発生した場合，Web サーバのエラーログにメッセージ AH01102 が出力され，その詳細情報としてタイムアウトが発生したことが示されます。このメッセージに含まれるスレッド ID と時刻を基に，Web サーバのリクエストログからプロキシトレースを検索します。プロキシトレースには次の情報が出力されています。

- バックエンドサーバとの接続状況
- 接続先 IP アドレス
- 接続先ポート番号
- 性能解析トレースのルートアプリケーション情報

プロキシトレースのメッセージに出力された性能解析トレースのルートアプリケーション情報を，性能解析トレースファイルに出力されたルートアプリケーション情報と突き合わせて確認することで，性能解析トレース中のどこでリクエストのタイムアウトが発生したかを確認できます。

また，該当するルートアプリケーション情報を，Web サーバのアクセスログに出力されたルートアプリケーション情報と突き合わせることで，タイムアウトが発生したリクエストの URI を確認できます。これらの情報を基に，タイムアウトが発生したリクエストを特定してください。

7.7.7　ルートアプリケーション情報を利用したログ調査

J2EE アプリケーションまたはバッチアプリケーション内で，アプリケーションサーバの API を使用すると，性能解析情報のルートアプリケーション情報の文字列表現を任意のタイミングでログファイルに出力できます。

ルートアプリケーション情報の文字列表現は，次の形式で出力されます（最大 48 文字）。

```
IPアドレス/プロセスID/通信番号
（例：10.209.15.130/1234/0x0000000000000001）
```

API を使用してログファイルにルートアプリケーション情報を出力すると，ログファイルと性能解析トレースファイルを突き合わせた調査ができるようになります。

Web コンテナでは，新しいリクエストを受け付けた時点で，リクエスト単位で新しいルートアプリケーション情報が割り当てられます。新しいルートアプリケーション情報が割り当てられるトレースポイントを次に示します。

表 7-12　新しいルートアプリケーション情報が割り当てられるトレースポイント

イベント ID（処理内容）	処理内容
0x8236	リクエスト取得時，リクエストヘッダ解析完了時

注※　Web サーバと連携している場合，HTTP Server で取得したルートアプリケーション情報が割り当てられます。HTTP Server または HTTP Server が使用する PRF トレース出力ライブラリのロードに失敗してルートアプリケーション情報が発行されなかった場合，Web コンテナで新しいルートアプリケーション情報が割り当てられます。

また，次のトレース取得ポイントでは，「IP アドレス/プロセス ID/通信番号」が「0.0.0. 0/0/0x0000000000000000」と出力されることがあります。

- 0x8237
  Web クライアントからのデータ読み込み開始時
- 0x8238
  Web クライアントへのデータ書き込み開始時

「IP アドレス/プロセス ID/通信番号」が「0.0.0. 0/0/0x0000000000000000」と出力されるのは，次の場合です。

- HTTP リクエストヘッダを受信した場合
- HTTP リクエストではない，不正なデータを受信した場合
- リクエスト処理中に例外が発生した場合

なお，ルートアプリケーション情報の文字列表現での出力に使用する API は，com.hitachi.software.ejb.application.prf パッケージで提供されている，CprfTrace クラスの getRootApInfo メソッドです。

J2EE アプリケーションまたはバッチアプリケーション開発時のルートアプリケーション情報の取得の実装については，「7.4　性能解析トレースのルートアプリケーション情報取得のための実装」を参照してください。API の詳細については，マニュアル「アプリケーションサーバ リファレンス API 編」を参照してください。

7.7.8　トラブルが発生したコネクションの特定

データベースとして HiRDB または Oracle を使用している場合，性能解析トレースファイルに出力されたコネクション ID と，HiRDB クライアントおよび Oracle クライアントのログファイルやトレースファイルに出力されたコネクション ID を突き合わせて確認することで，トラブルが発生したコネクションを確認できます。確認方法については，「付録 B　データベースと接続中にトラブルが発生したコネクションの特定」を参照してください。

7.7.9　性能解析トレースファイルとスレッドダンプを対応づけた問題個所の調査

J2EE アプリケーションまたはバッチアプリケーションでスローダウンやハングアップが発生したときに，性能解析トレースファイルとスレッドダンプを対応づけることで，問題が発生した個所を調査できます。

性能解析トレースファイルとスレッドダンプの対応づけには，性能解析トレースファイルに出力されたスレッド ID と，スレッドダンプに出力されたスレッド情報の nativeID（OS レベルのスレッド ID）を使用します。ここでは，性能解析トレースファイルから対応するスレッドダンプを特定する方法について説明します。

性能解析トレースファイルから対応するスレッドダンプを特定する手順を次に示します。

1. 性能解析トレースファイル，およびスレッドダンプを収集します。

   性能解析トレースファイルの収集方法については，「7.3.1　性能解析トレースファイルの収集方法」を参照してください。スレッドダンプの収集方法については，「4.7　JavaVM のスレッドダンプ」を参照してください。

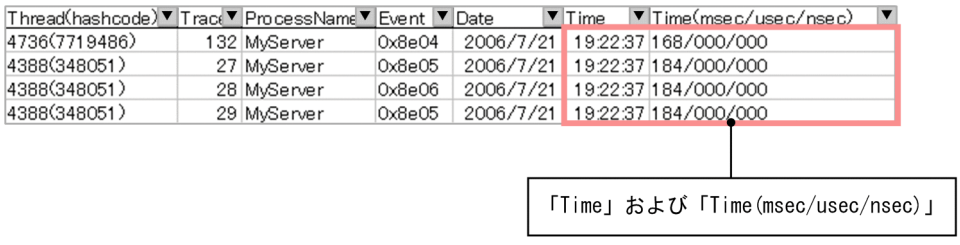
2. 使用する性能解析トレースファイル，およびスレッドダンプを選びます。

   性能解析トレースファイルとスレッドダンプが出力された時刻を基に，調査に使用する性能解析トレースファイル，およびスレッドダンプを選びます。出力された時刻は，次の情報を参考にしてください。

   性能解析トレースファイル

   「Time」および「Time(msec/usec/nsec)」

   性能解析トレースファイルの「Time」および「Time(msec/usec/nsec)」の例を次の図に示します。



   スレッドダンプ

   ファイル名，およびファイルの末尾に出力される時刻

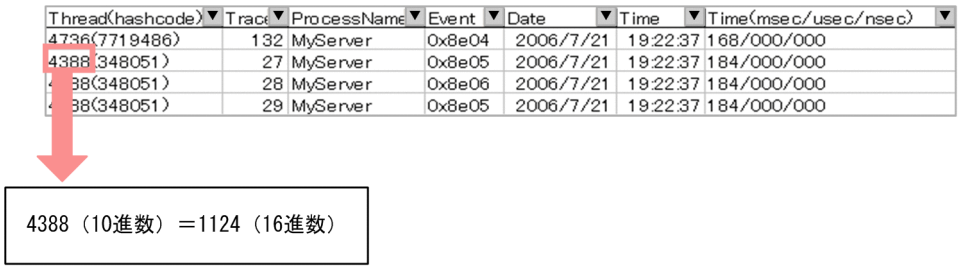
   ファイルの末尾に出力される時刻の例を次に示します。

```
   :
   :
Full thread dump completed.    Fri Jul 21 19:22:47 2006
```

3. 性能解析トレースファイルの調査したいイベント ID に対応する「Thread（hashcode)」の値を 16 進数に変換します。

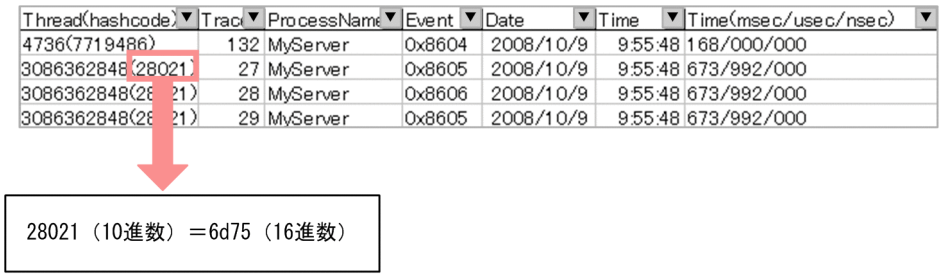
- Windows および AIX の場合

  Thread（スレッド ID）の値を 10 進数から 16 進数に変換します。



- Linux の場合

  hashcode（ハッシュコード）の値を 10 進数から 16 進数に変換します。



4. 手順 3.で 16 進数に変換した「Thread（hashcode)」の値と一致するスレッド情報を探して，対応するスレッド情報を特定します。

- Windows および AIX の場合

  スレッドダンプの「nid」が，16 進数に変換した Thread（スレッド ID）の値（1124）と一致するスレッド情報を探します。

```
   :
   :
"VBJ ThreadPool Worker" daemon prio=5 jid=0x00054f93 tid=0x04cef380 nid=0x1124 in Obje
ct.wait() [0x0632f000..0x0632fd18]
  stack=[0x06330000..0x062f5000..0x062f1000..0x062f0000]
  [user cpu time=0ms, kernel cpu time=15ms] [blocked count=1, waited count=29]
at java.lang.Object.wait(Native Method)
   :
   :
```

- Linux の場合

  スレッドダンプの「jid」が，16 進数に変換した hashcode（ハッシュコード）の値（6d75）と一致するスレッド情報を探します。

```
   :
   :
"main" prio=1 jid=0x00006d75 tid=0x00201d70 nid=0x1e51 waiting on condition [0x0000000
0..0xbfe80488]
  stack=[0xbfe87000..0xbfc8c000..0xbfc88000..0xbfc87000]
  [user cpu time=1320ms, kernel cpu time=4280ms] [blocked count=5, waited count=4]
   :
   :
```

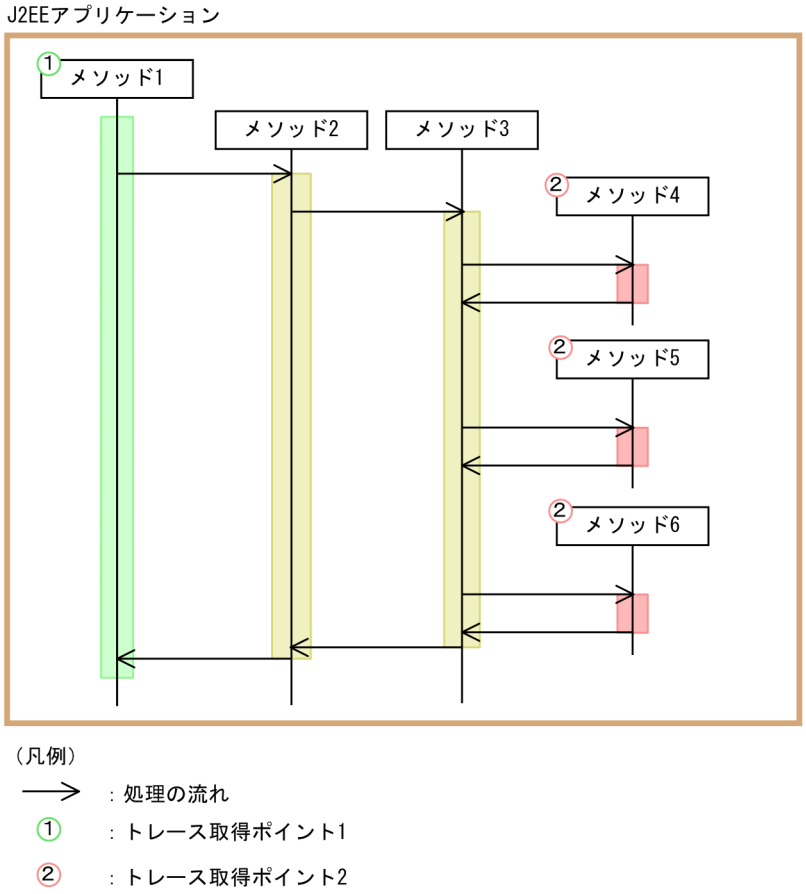
7.8　ユーザ拡張性能解析トレース使用時の注意事項

ユーザ拡張性能解析トレースを使用する場合の注意事項を次に示します。

- トレース対象メソッドが頻繁に呼び出される場合，ユーザ拡張性能解析トレースの処理のオーバーヘッドによる性能劣化によって，劣化の要因となるメソッドの特定が困難になることがあります。そのため，トレース対象メソッドには劣化要因が特定できる処理の入口となるメソッドを指定してください。

  例えば，次の図に示すようなメソッドの呼び出し関係のあるアプリケーションの場合，処理の入口となるメソッド 1（トレース取得ポイント 1）を対象としてください。トレース取得ポイント 2 のメソッド 4，メソッド 5，メソッド 6 を指定すると，これらのメソッドに対するユーザ拡張性能解析トレースの処理に時間が掛かると，このアプリケーションの性能測定に影響を与えることがあります。

図 7-13　アプリケーションのメソッドの呼び出し関係



- ルートアプリケーション情報を取得する前に呼び出されるメソッドにユーザ拡張性能解析トレースを使用した場合，ルートアプリケーション情報が 0 で出力されることがあります。
- トレース対象メソッドの呼び出し回数を増やしたり，トレースの出力量を増やすプロパティを指定したりした場合，PRF トレースファイルの出力量が増えます。その場合は，PRF トレースファイル面数または PRF トレースファイルサイズの量を増やしてください。PRF トレースファイル数と PRF トレースファイルサイズの変更の方法については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.3.1　パフォーマンストレーサの設定」を参照してください。

  なお，PRF トレースファイルの出力量の増加は，次の現象に対する原因となります。

- トレースファイルへの書き込みによるオーバーヘッドが発生して性能劣化につながります。
- 大量の書き込みでトレースファイルの面の切り替えが発生して，J2EE サーバが取得しているトレースが取得できなくなることがあります。

通常運用時では，ユーザ拡張性能解析トレース設定ファイルの指定内容を適切に設定してトレース取得量を調整してください。

- ユーザ拡張性能解析トレースでは，アプリケーションのクラスを書き換えます。このとき，クラスの書き換え後のサイズが 64 キロバイトを超える場合は，書き換えに失敗します。その場合は，クラスを分割するなどしてクラスのサイズを小さくしてください。

8 性能解析トレースのトレース取得ポイントと PRF トレース取得レベル

性能解析トレースには，アプリケーションサーバの性能を解析する性能解析トレースと，性能解析トレースの対象を拡張してアプリケーションサーバ上に構築したアプリケーションの性能を解析するユーザ拡張性能解析トレースの 2 種類があります。この章では，性能解析トレースおよびユーザ拡張性能解析トレースによって出力されるトレース取得ポイントと PRF トレース取得レベルについて説明します。

性能解析トレースの概要と，解析方法については，「7.　性能解析トレースを使用した性能解析」を参照してください。

8.1　この章の構成

この章では，次の性能解析トレースのトレース取得ポイントと PRF トレース取得レベルについて説明します。

- 性能解析トレースによって，機能レイヤごとに出力されるトレース取得ポイントと PRF トレース取得レベル

  Web クライアントまたは EJB クライアントからリクエストが送信された場合に，J2EE サーバ，EJB クライアント，および CTM では，決まった処理のポイント（トレース取得ポイント）でトレース情報を出力します。また，性能解析トレースでは，標準，詳細などの PRF トレース取得レベルを設定してトレースを出力できます。

- ユーザ拡張性能解析トレースによって，アプリケーションの処理の起点となるメソッドごとに出力されるトレース取得ポイントと PRF トレース取得レベル

  ユーザ拡張性能解析トレースでは，アプリケーションの性能解析情報を取得したい処理ポイントとして，アプリケーションの処理の起点となるメソッドを指定することで，そのメソッドが呼ばれたポイントでトレースを出力できます。

この章の構成を次の表に示します。

表 8-1　この章の構成（性能解析トレース）

分類	タイトル	参照先
解説	性能解析トレースのトレース取得ポイントと PRF トレース取得レベルの概要	8.2
	CTM のトレース取得ポイント	8.3
	Web コンテナのトレース取得ポイント（リクエスト処理のトレース）	8.4
	Web コンテナのトレース取得ポイント（セッショントレース）	8.5
	Web コンテナのトレース取得ポイント（フィルタのトレース）	8.6
	Web コンテナのトレース取得ポイント（データベースセッションフェイルオーバ機能のトレース）	8.7
	Web コンテナのトレース取得ポイント（セッションマネージャの指定機能のトレース）	8.8
	EJB コンテナのトレース取得ポイント	8.9
	JNDI のトレース取得ポイント	8.10
	JTA のトレース取得ポイント	8.11
	DB Connector，JCA コンテナのトレース取得ポイント	8.12
	RMI のトレース取得ポイント	8.13
	OTS のトレース取得ポイント	8.14
	標準出力／標準エラー出力／ユーザログのトレース取得ポイント	8.15
	DI のトレース取得ポイント	8.16

分類	タイトル	参照先
	バッチアプリケーション実行機能のトレース取得ポイント	8.17
	JPA でのトレース取得ポイント	8.18
	TP1 インバウンド連携機能のトレース取得ポイント	8.19
	CJMS プロバイダのトレース取得ポイント	8.20
	JavaMail のトレース取得ポイント	8.21
	JSF 2.3 のトレース取得ポイント	8.22
	CDI のトレース取得ポイント	8.23
	J2EE サーバの開始・終了時のトレース取得ポイント	8.24
	アプリケーションのトレース取得ポイント	8.25
	JAX-RS のトレース取得ポイント	8.26
	Java Batch のトレース取得ポイント	8.27
	WebSocket のトレース取得ポイント	8.28
	Concurrency Utilities のトレース取得ポイント	8.29

注 「実装」,「設定」,「運用」および「注意事項」について，この機能固有の説明はありません。

なお，性能解析トレースの概要と，解析方法については，「7. 性能解析トレースを使用した性能解析」を参照してください。

8.2　性能解析トレースのトレース取得ポイントと PRF トレース取得レベルの概要

ここでは，トレース取得ポイントと PRF トレース取得レベルについて説明します。

8.2.1　トレース取得ポイント

トレース取得ポイントは大きく分けて，J2EE サーバの開始・終了時でのトレース取得と，各機能レイヤでの処理中のトレース取得およびアプリケーションのメソッドの開始・終了時でのトレース取得があります。

(1)　J2EE サーバの開始・終了時のトレース取得

J2EE サーバの開始処理の完了時，および J2EE サーバの終了処理の開始時に，トレース情報を取得できます。取得できるイベント ID と参照先を次に示します。

- イベント ID

  0x8FFE～0x8FFF
- 参照先

  J2EE サーバの開始・終了時のトレース取得の詳細については，「8.24　J2EE サーバの開始・終了時のトレース取得ポイント」を参照してください。

(2)　各機能レイヤでのトレース取得

取得できるイベント ID と機能レイヤの対応を次の表に示します。

表 8-2　取得できるイベント ID と機能レイヤ

イベント ID	機能レイヤ	図中の番号※	参照先
0x1101～0x1102 0x1301～0x1302 0x1401～0x1406 0x2002～0x2003 0x2101～0x2104 0x3000～0x3008	CTM	5	8.3
0x8202～0x8203 0x8206～0x8210 0x8214～0x8216 0x8219～0x8225 0x8234～0x8252 0x8302～0x8303 0x8306～0x8310	Web コンテナ	2	8.4，8.5，8.6，8.7，8.8

イベント ID	機能レイヤ	図中の番号※	参照先
0x8314～0x8316 0x8319～0x8325 0x8334～0x8352			
0x8401～0x840A 0x8425～0x8428 0x842D～0x8434 0x8453～0x8454 0x8460～0x846D 0x8470～0x8477 0x8490～0x8491 0x84A0～0x84D9 0x8C41	EJB コンテナ	6	8.9
0x8603～0x861C	JNDI	4	8.10
0x8435～0x843F 0x8811～0x8820 0x8C41	JTA	7	8.11
0x8B00～0x8B01 0x8B80～0x8B89 0x8B8A～0x8C03 0x8C10～0x8C13 0x8C20～0x8C41 0x8C60～0x8C65 0x8C80～0x8C93 0x8CC0～0x8CD9 0x8CDA～0x8CE3 0x8D00～0x8D19 0x8D1A～0x8D25 0x8D42～0x8D4D 0x8D60～0x8D63 0x8D80～0x8D89 0x8D8A～0x8D8F 0x8D90～0x8D99	DB Connector，JCA コンテナ	8	8.12
0x8E01～0x8E06	RMI	3	8.13
0x9400～0x9413	OTS	9	8.14
0x9C00～0x9C03	標準出力／標準エラー出力／ユーザログ	－	8.15
0x9D00，0x9D01	DI	10	8.16
0xA100，0xA101	バッチアプリケーション実行機能	11	8.17
0xD100～0xD10F 0xD110～0xD11F	JPA	12	8.18

イベント ID	機能レイヤ	図中の番号※	参照先
0xD120～0xD12F 0xD130～0xD13F 0xD140～0xD14F 0xD150～0xD15F 0xD160～0xD16F 0xD170～0xD17F 0xD180～0xD18F 0xD190～0xD19F 0xD1A0～0xD1AF 0xD1B0～0xD1BF 0xD1C0～0xD1CF 0xD1D0～0xD1DF 0xD1E0～0xD1EF 0xD1F0～0xD1FF			
0x842F 0x8430～0x8432 0x8825 0x8826 0x8B86，0x8B87 0x8B8A～0x8B93 0xAA00～0xAA06 0xAA08～0xAA0D 0xAA10～0xAA19	TP1 インバウンド連携機能	14	8.19
0xA600～0xA60F 0xA610～0xA619 0xA61E，0xA61F 0xA620～0xA62F 0xA630～0xA63F 0xA640～0xA64F 0xA650～0xA65F 0xA660～0xA667 0xA61A，0xA61B 0xA668～0xA66F 0xA670～0xA67F 0xA686～0xA68D 0xA692～0xA69B 0xA69E，0xA69F 0xA6A0，0xA6A1 0xA6A6～0xA6AB	CJMS プロバイダ	15	8.20
0xAD00～0xAD15 0xAD80～0xAD93	JavaMail	16	8.21

イベント ID	機能レイヤ	図中の番号※	参照先
0xAF20～0xAF2D	JSF 2.3	17	8.22
0xb002～0xb009	CDI	18	8.23
0xD000～0xD005	JAX-RS	19	8.26
0xD020～0xD04D	Java Batch	20	8.27
0xE100～0xE147	WebSocket	21	8.28
0xD050～0xD057	Concurrency Utilities	22	8.29

（凡例）－：該当しない

注※　図 8-1～図 8-4 中の番号と対応しています。

PRF トレースを出力する機能レイヤとトレース取得ポイントについて，システムの構成ごとに次の図に示します。

図 8-1　機能レイヤとトレース取得ポイント（Web クライアント構成の場合）

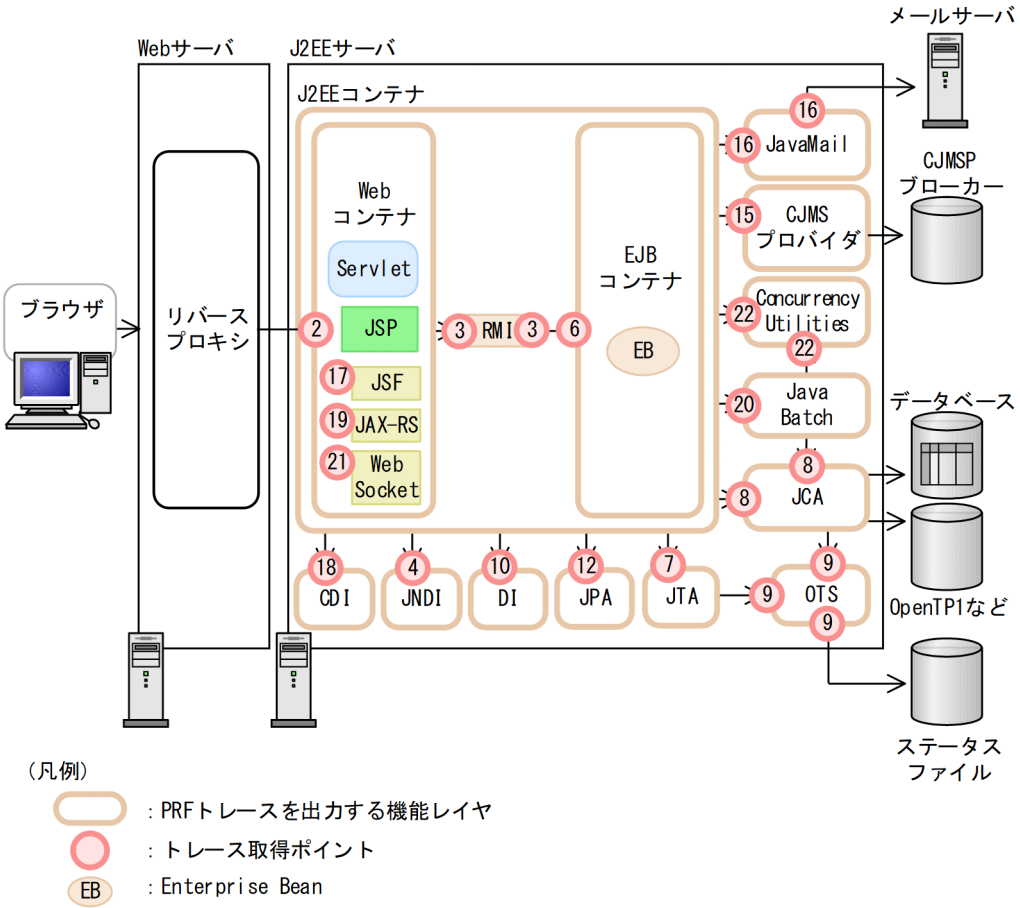
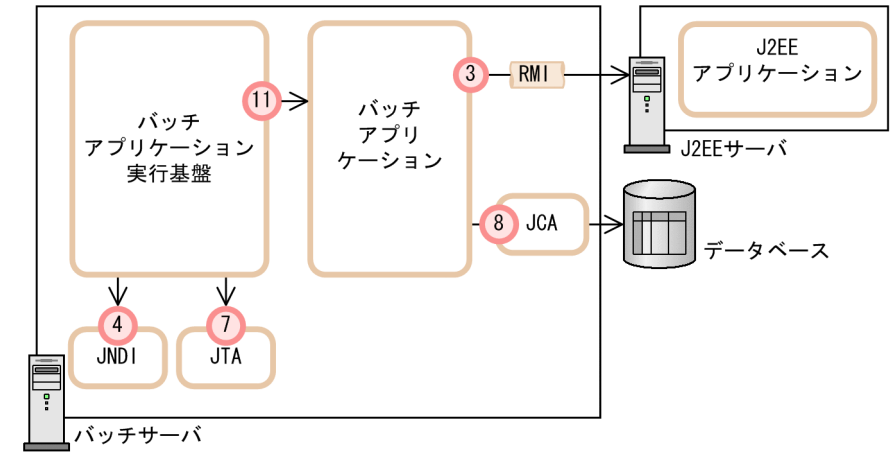


図 8-2　機能レイヤとトレース取得ポイント（バッチアプリケーションを実行するシステムの場合）

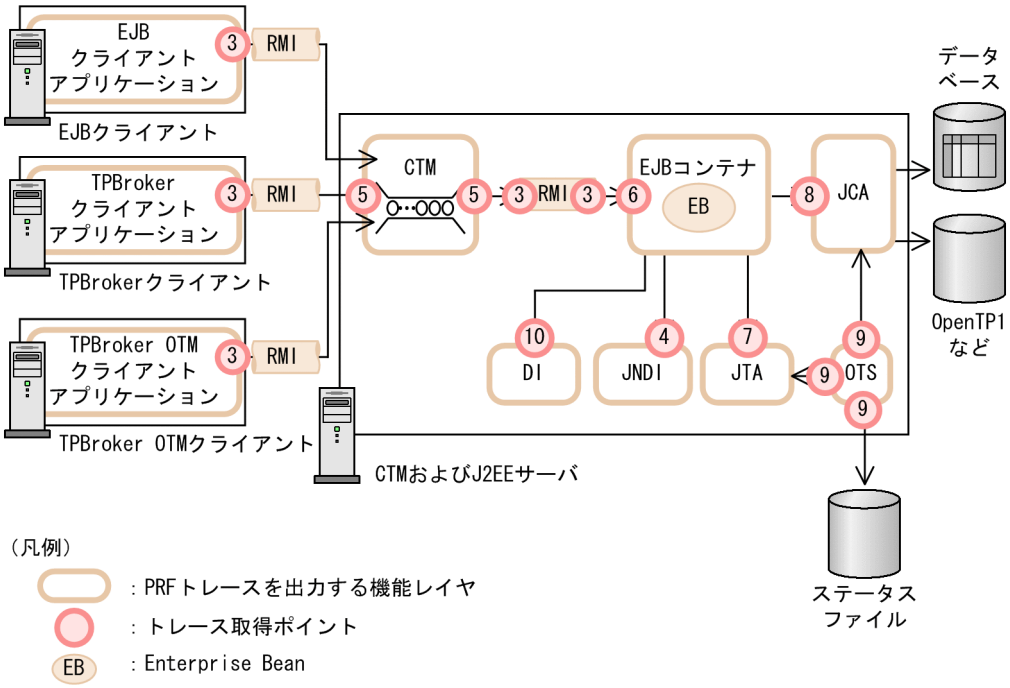


（凡例）

：PRFトレースを出力する機能レイヤ

：トレース取得ポイント

図 8-3　機能レイヤとトレース取得ポイント（EJB クライアント/TPBroker クライアント/TPBroker OTM クライアント構成（CTM 使用）の場合）



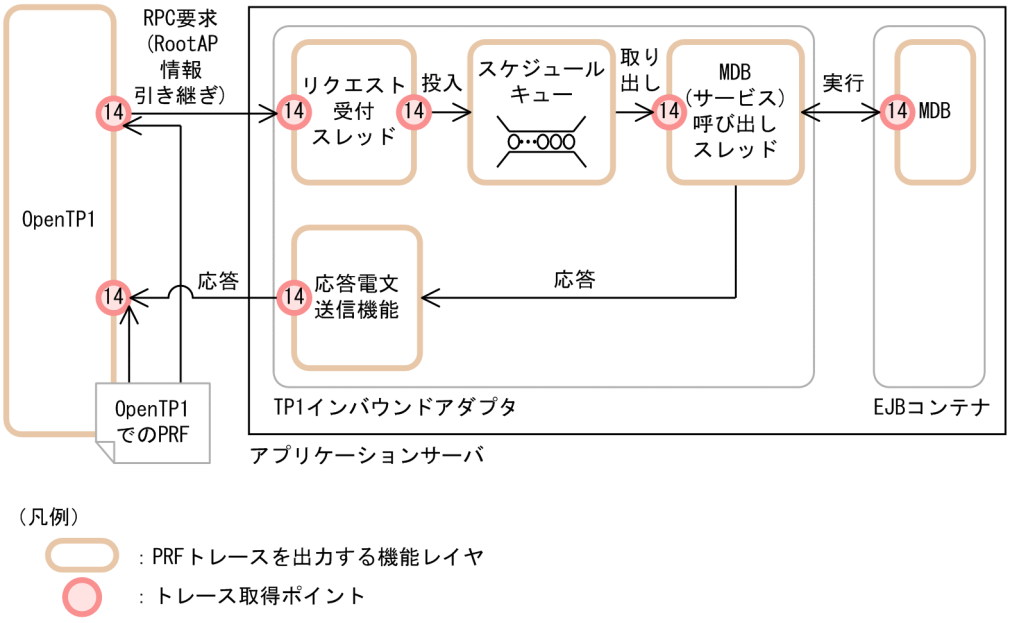
（凡例）

：PRFトレースを出力する機能レイヤ

：トレース取得ポイント

EB　：Enterprise Bean

図 8-4　機能レイヤとトレース取得ポイント（TP1 インバウンド連携機能の場合）



トレース取得ポイントは，各機能レイヤ内でさらに詳細に分かれており，トレース取得ポイントによって，PRF トレース取得レベルが異なります。各機能レイヤの詳細なトレース取得ポイント，および PRF トレース取得レベルについては，表 8-2 の参照先を参照してください。

参考

表 8-2 に示した機能レイヤのほか，Application Server のプロセス，構成ソフトウェアおよび関連プログラムでも，PRF トレースが取得できるものがあります。

表 8-2 に示した以外で PRF トレースを取得できる機能レイヤとイベント ID の対応を，次の表に示します。

表 8-3　表 8-2 に示した以外で PRF トレースを取得できるイベント ID と機能レイヤの対応

イベント ID	機能レイヤ
0x9000～0x90FF 0xA400～0xA4FF	SOAP 通信基盤
0x9100～0x91FF	• TP1 Connector • TP1/Client/J
0x9200～0x92FF	TP1/MQ Access
0x9300～0x93FF	Reliable Messaging
0x9800～0x9B6E	HCSC サーバ
0x9E00～0x9EFF	Service Coordinator Interactive Workflow
0x9F00～0x9FFF	HCSC サーバ（Object Access アダプタ）

イベント ID	機能レイヤ
0xA000～0xA0FF	HCSC サーバ（ファイルアダプタ）
0xA200～0xA2FF	HCSC サーバ（Message Queue アダプタ）
0xAB00～0xABFF 0xAC00～0xACFF	HCSC サーバ（FTP アダプタ）
0xA400～0xA4FF	JAX-WS エンジン
0xE000～0xE0FF	Elastic Application Data store

(3)　アプリケーションのメソッドの開始・終了時トレース取得

アプリケーションのメソッドの開始時，および終了時にトレース情報を取得できます。取得できるトレース情報を次に示します。

- メソッドの開始時刻，終了時刻
- 識別 ID
- パッケージ名，クラス名，メソッド名
- メソッドで実行された最後の行の行番号
- 発生した例外またはエラーのクラス名

トレース情報の詳細については，「8.25　アプリケーションのトレース取得ポイント」を参照してください。

(4)　各トレースのリターンコード

各トレースのリターンコードは入口トレースの場合，常に「0」が出力されます。

出口トレースおよび呼び出し後トレースの場合，次のように出力されます。

正常終了：0

異常終了：0 以外

8.2.2　PRF トレース取得レベル

性能解析トレースでは，次の 4 種類の PRF トレース取得レベルを設定してトレースを出力できます。レベルによって，トレース取得ポイントの数が異なります。トレース取得ポイント，および PRF トレース取得レベルについては，表 8-2 の参照先を参照してください。

- **標準レベル**
  各機能レイヤの境界（入り口と出口）を識別できるトレース情報を出力します。

- **詳細レベル**
  標準レベルの出力内容に加えて，各機能レイヤ内処理のトレース情報も出力します。
- **保守レベル**
  障害発生時などの保守情報を取得するためのレベルです。
- **抑止レベル**
  トレース情報の出力を抑止するためのレベルです。RMI，JSF 2.3，JAX-RS，Java Batch，WebSocket，および Concurrency Utilities の機能レイヤに設定できます。

Management Server を利用して運用している場合は，簡易構築定義ファイルで，すべての機能レイヤに対して共通のレベルを設定して，トレース情報を出力します。

なお，次節以降は，トレース取得レベルが標準レベルと詳細レベルのトレース取得ポイントについて説明します。保守レベルは，障害発生時などの保守情報を取得するためのレベルのため，通常は収集する必要はありません。

8.3　CTM のトレース取得ポイント

ここでは，CTM でのトレース取得ポイントと，取得できるトレース情報について説明します。

8.3.1　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-4　CTM でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x1101	21	OTM ゲートウェイからのリクエスト送信直前	A
0x1102	22	OTM ゲートウェイからのレスポンス受信直前	A
0x1301	4	EJB レギュレータからのリクエスト送信直前	A
0x1302	13	EJB レギュレータへのレスポンス受信直後	A
0x1401	1	CTM 内の Create メソッドの入り口	A
0x1402	2	CTM 内の Create メソッドの出口	A
0x1403	5	EJB レギュレータからのリクエスト受信直後	A
0x1404	8	CTM から J2EE サーバまたはバッチサーバへのリクエスト送信直前	A
0x1405	9	CTM での，J2EE サーバまたはバッチサーバからのリクエスト受信直後	A
0x1406	12	CTM から EJB レギュレータへのレスポンス送信直前	A
0x2002	23	OTM クライアントからのリクエスト受信直前	A
0x2003	24	OTM クライアントからのレスポンス受信直前	A
0x2101	3	EJB クライアントまたは cjexecjob コマンドからのリクエスト受信直後	A
0x2102	14	EJB クライアントまたは cjexecjob コマンドへのレスポンス送信直前	A
0x2103	15	CTM 内の Remove メソッドの入り口	A
0x2104	16	CTM 内の Remove メソッドの出口	A
0x3000	6	リクエストをキューイングする直前	A
0x3001	7	キューからリクエストを取り出した直後	A
0x3002	17	他 CTM へのリクエスト送信直前	A
0x3003	18	他 CTM からのリクエスト受信直後	A
0x3004	10	リクエスト応答をキューイングする直前	B
0x3005	11	キューからリクエスト応答を取り出した直後	B

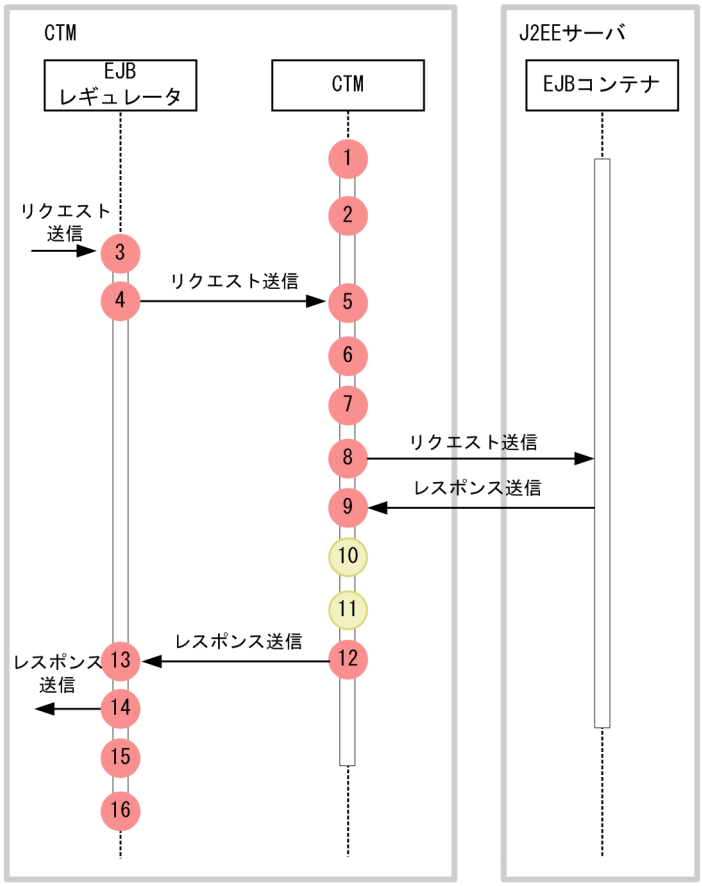
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x3006	19	他 CTM へのリクエスト応答送信直前	A
0x3007	20	他 CTM からのリクエスト応答受信直後	A
0x3008	－	サーバ状態変更時に取得されます。（リクエスト処理中には取得されません）	B

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-5～図 8-8 中の番号と対応しています。

CTM でのトレース取得ポイントを次の図に示します。

図 8-5　CTM のトレース取得ポイント

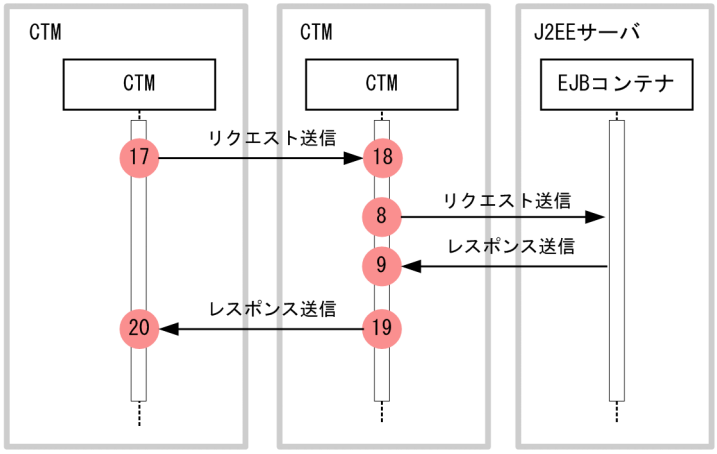


（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

　　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

ほかの CTM と連携している場合は，次の図に示す場所でもトレース情報を取得します。上記の図とあわせて参照してください。なお，この図では，ほかの CTM と連携している個所だけを取り上げています。

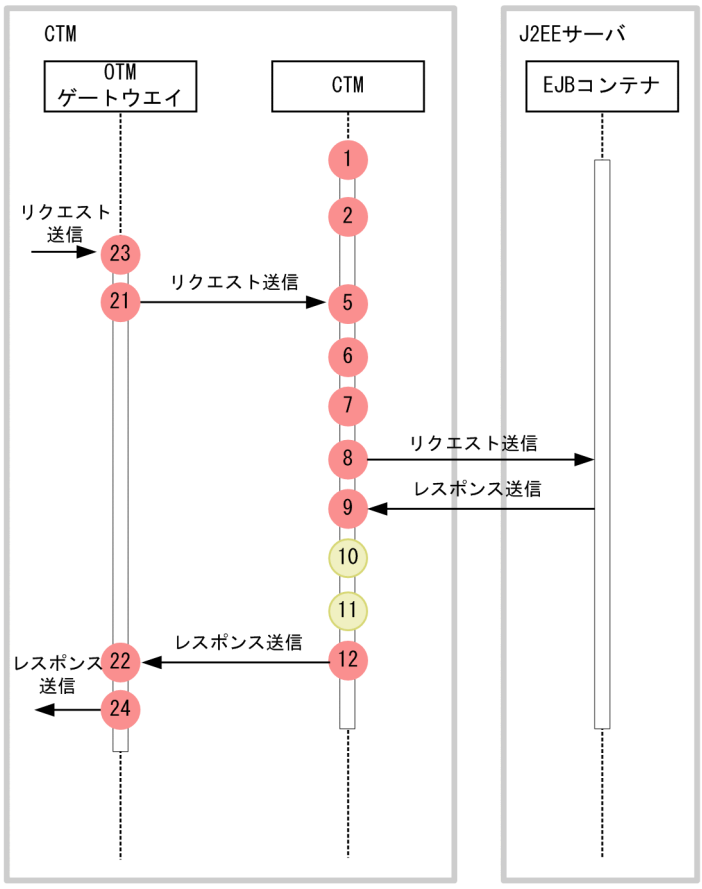
図 8-6　CTM のトレース取得ポイント（他 CTM と連携している場合）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

また，OTM ゲートウェイまたは ORB ゲートウェイを使用する場合は，次の図に示す場所でもトレース情報を取得します。上記の図とあわせて参照してください。なお，この図では，ほかの CTM と連携している個所だけを取り上げています。

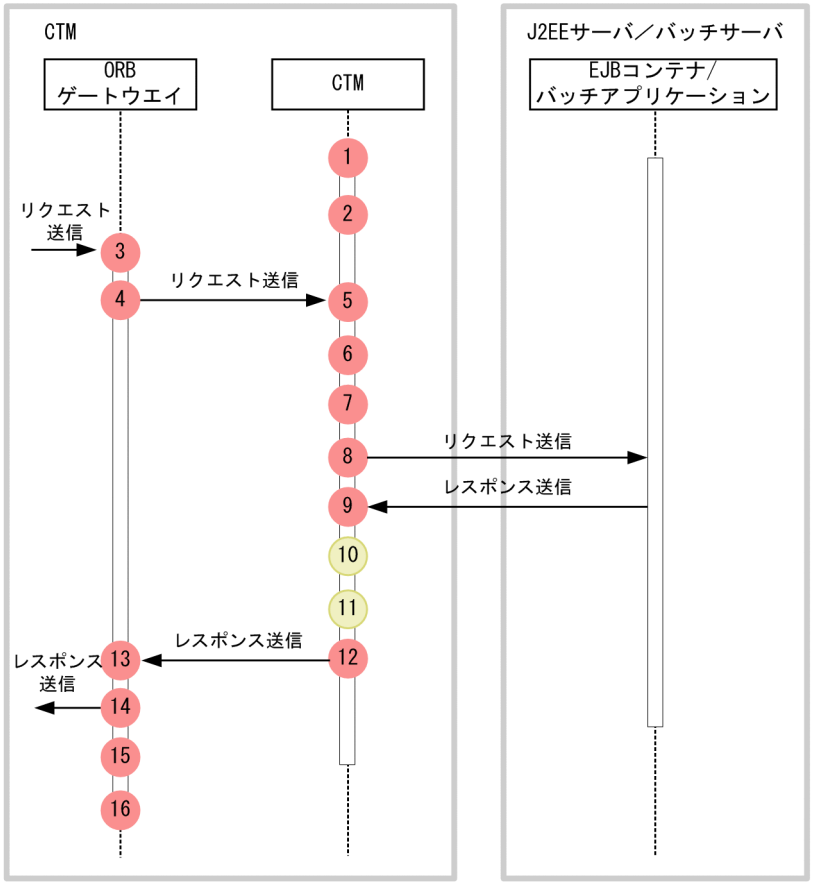
図 8-7　CTM のトレース取得ポイント（OTM ゲートウェイを使用する場合）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

図 8-8　CTM のトレース取得ポイント（ORB ゲートウェイを使用する場合）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

8.3.2　取得できるトレース情報

CTM で取得できるトレース情報を次の表に示します。

表 8-5　CTM で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x1401	A	リモートインタフェース名	メソッド名	−
2	0x1402	A	リモートインタフェース名	メソッド名	内部情報
3	0x2101	A	リモートインタフェース名	メソッド名	−
4	0x1301	A	リモートインタフェース名	メソッド名	−

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
5	0x1403	A	リモートインタフェース名	メソッド名	－
6	0x3000	A	－	－	－
7	0x3001	A	－	－	内部情報
8	0x1404	A	リモートインタフェース名	メソッド名	内部情報
9	0x1405	A	リモートインタフェース名	メソッド名	内部情報
10	0x3004	B	－	－	－
11	0x3005	B	－	－	内部情報
12	0x1406	A	リモートインタフェース名	メソッド名	－
13	0x1302	A	リモートインタフェース名	メソッド名	－
14	0x2102	A	リモートインタフェース名	メソッド名	－
15	0x2103	A	リモートインタフェース名	メソッド名	－
16	0x2104	A	リモートインタフェース名	メソッド名	－
17	0x3002	A	－	－	－
18	0x3003	A	－	－	－
19	0x3006	A	－	－	－
20	0x3007	A	－	－	－
21	0x1101	A	リモートインタフェース名	メソッド名	－
22	0x1102	A	リモートインタフェース名	メソッド名	－
23	0x2002	A	リモートインタフェース名	メソッド名	－
24	0x2003	A	リモートインタフェース名	メソッド名	－
－	0x3008	B	－	－	内部情報

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-5～図 8-8 中の番号と対応しています。

8.4 Web コンテナのトレース取得ポイント（リクエスト処理のトレース）

ここでは，Web コンテナのトレース取得ポイントと，取得できるトレース情報について説明します。なお，Web コンテナでは，リクエスト処理のトレースとセッショントレースが出力されます。ここでは，リクエスト処理のトレースについて説明します。

8.4.1 トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-6 Web コンテナでのトレース取得ポイントの詳細（リクエスト処理のトレース）

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0x8202※2	6	サーブレット／JSP の呼び出し直前	A／B
0x8203	5	フィルタの呼び出し直前	B
0x8206	7	RequestDispatcher の forward()／include()呼び出し直前	B
0x8207	6	静的コンテンツの呼び出し	A／B
0x8234	4	同期処理の開始直後	A
0x8235	14	非同期処理の開始直後	A
0x8236	3	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8237	1	リバースプロキシからのデータ読み込み開始直前	B
0x8238	8	リバースプロキシへのデータ書き込み開始直前	B
0x8239	16	非同期サーブレット処理の開始直後	A
0x8260	19	javax.servlet.http.PushBuilder.push メソッドによる push 処理開始直前	A
0x8302	11	サーブレット／JSP の処理完了直後	A／B
0x8303	12	フィルタの処理完了直後	B
0x8306	10	RequestDispatcher の forward()／include()処理完了直後	B
0x8307	11	静的コンテンツの処理完了直後	A／B
0x8334	13	同期処理の完了直前	A
0x8335	15	非同期処理の完了直前	A
0x8336	18	リクエスト処理完了直後	A
0x8337	2	リバースプロキシからのデータ読み込み完了直後	B
0x8338	9	リバースプロキシへのデータ書き込み完了直後	B
0x8339	17	非同期サーブレット処理の完了直前	A

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0x8360	20	javax.servlet.http.PushBuilder.push メソッドによる push 処理開始直後	A

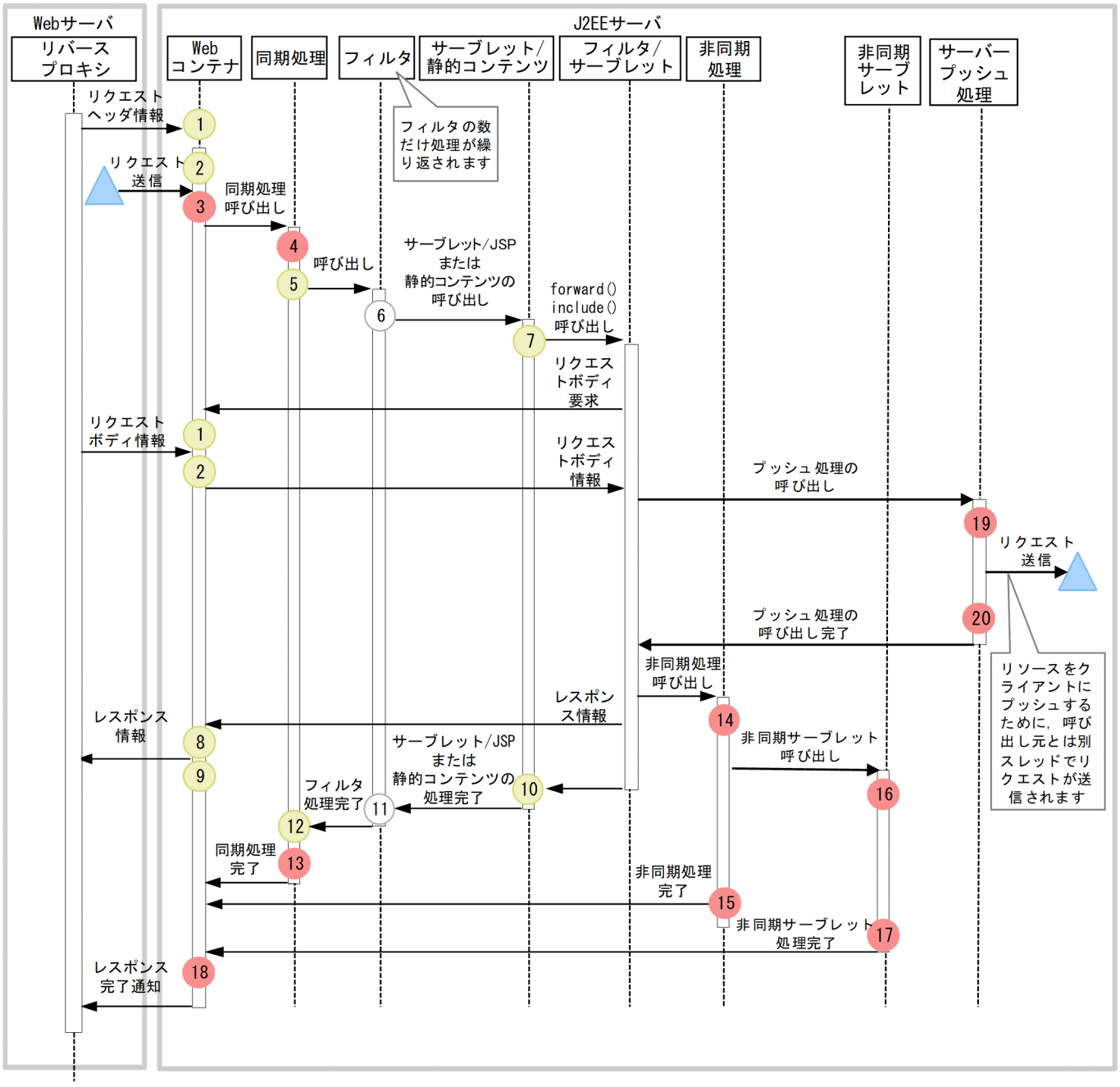
（凡例）A：標準　B：詳細　A／B：標準と詳細で異なる情報を取得

注※1　図 8-9 中の番号と対応しています。

注※2　JSP のコンパイルが必要な場合は，JSP のコンパイルを実行したあと，トレースを取得します。

Web コンテナでのトレース取得ポイントを次の図に示します。

図 8-9　Web コンテナのトレース取得ポイント（リクエスト処理のトレース）



（凡例）

：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRFトレース取得レベルは「詳細」です。

：トレース取得ポイントを示します。
サーブレットの呼び出しの場合，PRFトレース取得レベルは「標準」です。
静的コンテンツの呼び出しの場合，PRFトレース取得レベルは「詳細」です。

：2つの は繋がっており，サーバープッシュ処理で，Webコンテナへリクエストが送信されます。

8.4.2　取得できるトレース情報

Web コンテナで取得できるトレース情報を次の表に示します。

表 8–7　Web コンテナで取得できるトレース情報（リクエスト処理のトレース）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8237	B	要求サイズ	−	−
2	0x8337	B	読み込めたサイズ	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値（ヘッダ名未指定の場合はなし）
4	0x8234	A	−	−	−
5	0x8203	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
6	0x8202	A	クラス名（JSP 呼び出し時は JSP ファイル名）	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時 <入口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8207	B	−	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
7	0x8206	B	クラス名	ディスパッチのタイプ コンテキストルート	<セッション ID 文字数:セッション ID:グローバル

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					セッション ID 文字数:グローバルセッション ID>
8	0x8238	B	書き込みサイズ	−	−
9	0x8338	B	書き込めたサイズ	−	• 正常時 <入口時刻> • 例外発生時 <入口時刻><例外名>
10	0x8306	B	クラス名	ディスパッチのタイプ コンテキストルート名	• 正常時 <入口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入口時刻><例外名:セッション ID 文字数:セッション ID>
11	0x8302	A	クラス名（JSP 呼び出し時は JSP ファイル名）	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時 <入口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8307	B	−	コンテキストルート名	• 正常時 <入口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入口時刻><例外名:セッション ID 文字数:セッション ID>
12	0x8303	B	クラス名	コンテキストルート名	• 正常時 <入口時刻><セッション ID 文字数:セッション ID> • 例外発生時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入口時刻><例外名:セッション ID 文字数:セッション ID>
13	0x8334	A	−	−	<入り口時刻>
14	0x8235	A	非同期処理の実装クラス名	−	−
15	0x8335	A	非同期処理の実装クラス名	−	• 正常時 <入口時刻> • 例外発生時 <入口時刻><例外名>
16	0x8239	A	−	−	−
17	0x8339	A	−	−	<入り口時刻>
18	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>
19	0x8260	A	HTTP メソッド	push するリクエストの URI	−
20	0x8360	A	HTTP メソッド	push するリクエストの URI	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　B：詳細　−：該当なし

注※　図 8-9 中の番号と対応しています。

参考

SOAP クライアント以外のリクエストを受信した場合，トレース情報のキー情報である「クライアントアプリケーション情報」には，常に 0 が表示されます。SOAP クライアントのリクエストを受信した場合だけ，「クライアントアプリケーション情報」が出力されます。

8.5 Web コンテナのトレース取得ポイント（セッショントレース）

ここでは，Web コンテナのトレースのトレース取得ポイントと，取得できるトレース情報について説明します。なお，Web コンテナでは，リクエスト処理のトレースとセッショントレースが出力されます。ここでは，セッショントレース，およびグローバルセッションについてのトレース取得ポイントと取得できるトレース情報について説明します。

8.5.1 トレース取得ポイントと PRF トレース取得レベル（セッショントレース）

セッショントレースに関連するトレースの，イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。なお，「0x8203」，「0x8202」，「0x8207」，および「0x8206」では，グローバルセッションについての情報も出力されます。

表 8-8 Web コンテナでのトレース取得ポイントの詳細（セッショントレース）

イベント ID	図中の番号※1	トレース取得ポイント	レベル※2
0x8202	4，9	サーブレット／JSP 呼び出し直前	A／B
0x8203	2，3	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの呼び出し直前 （web.xml の<filter-mapping>タグの<dispatcher>タグを省略した場合，または<dispatcher>タグで"REQUEST"を指定したフィルタを呼び出した場合）	B
0x8206	7	RequestDispatcher 経由のサーブレット／JSP 呼び出し直前	B
0x8207	4，9	静的コンテンツ呼び出し直前（DefaultServlet）	B
0x8208	5	セッション生成後	B
0x8209	6	セッション破棄後	B
0x8210	17	セッションタイムアウト後	B
0x8214	8	フォワード時に実行されるフィルタの呼び出し直前 （web.xml の<filter-mapping>タグの<dispatcher>タグで"FORWARD"を指定したフィルタを呼び出した場合）	B
0x8215	8	インクルード時に実行されるフィルタの呼び出し直前 （web.xml の<filter-mapping>タグの<dispatcher>タグで"INCLUDE"を指定したフィルタを呼び出した場合）	B
0x8216	2	エラーページに転送される際に実行されるフィルタの呼び出し直前 （web.xml の<filter-mapping>タグの<dispatcher>タグで"ERROR"を指定したフィルタを呼び出した場合）	B
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A

イベント ID	図中の番号※1	トレース取得ポイント	レベル※2
0x8302	10，13	サーブレット／JSP 処理完了直後	A／B
0x8303	14，15	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの処理完了直後 （web.xml の<filter-mapping>タグの<dispatcher>タグで""REQUEST""を指定したフィルタの処理が完了した場合）	B
0x8306	12	RequestDispatcher 経由のサーブレット／JSP 処理完了直後	B
0x8307	10，13	静的コンテンツ処理完了直後（DefaultServlet）	B
0x8314	11	フォワード時に実行されるフィルタの処理完了直後 （web.xml の<filter-mapping>タグの<dispatcher>タグで"FORWARD"を指定したフィルタの処理が完了した場合）	B
0x8315	11	インクルード時に実行されるフィルタの処理完了直後 （web.xml の<filter-mapping>タグの<dispatcher>タグで"INCLUDE"を指定したフィルタの処理が完了した場合）	B
0x8316	15	エラーページに転送される際に実行されるフィルタの処理完了直後 （web.xml の<filter-mapping>タグの<dispatcher>タグで"ERROR"を指定したフィルタの処理が完了した場合）	B
0x8336	16	リクエスト処理完了直後	A

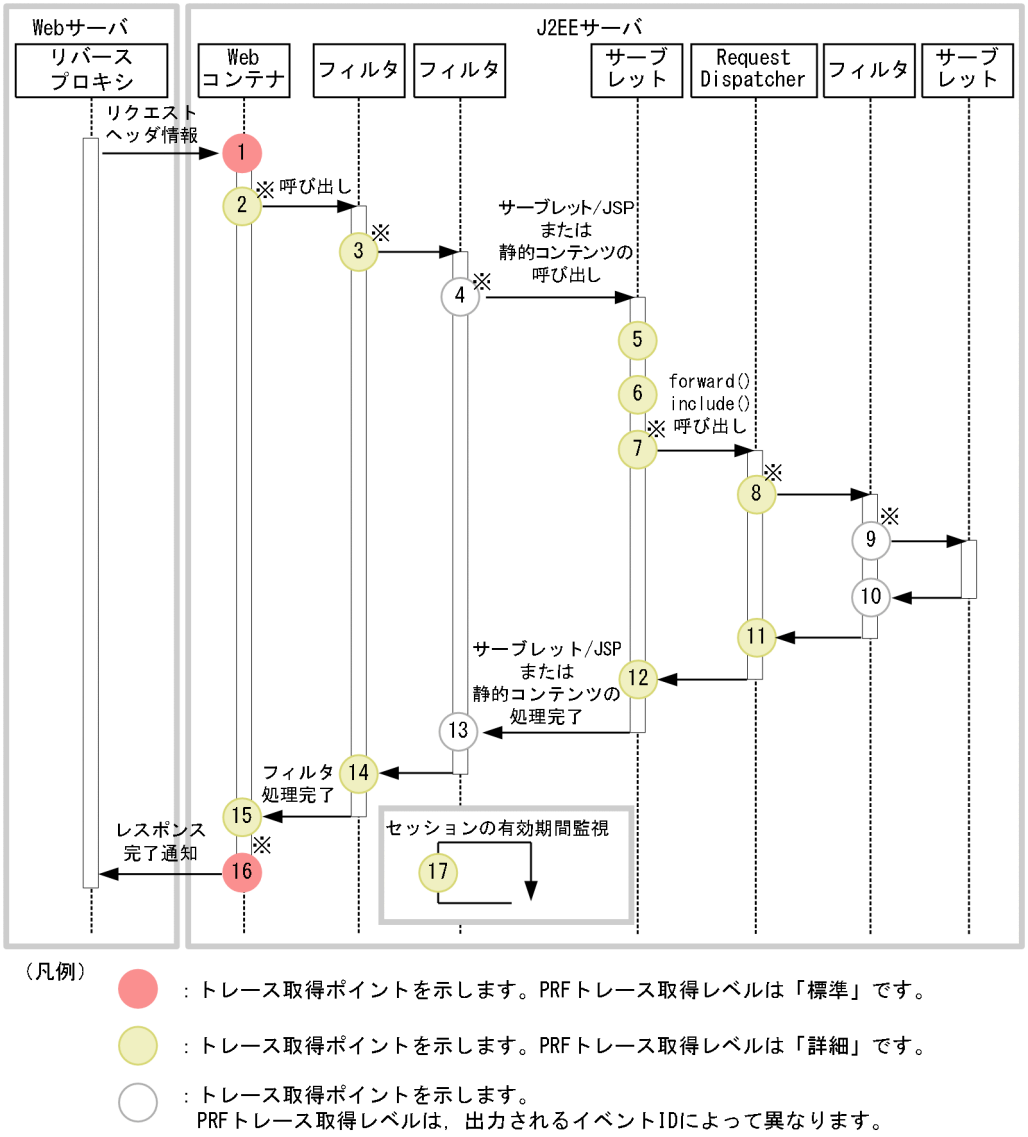
（凡例）A：標準　B：詳細　A／B：標準と詳細で異なる情報を取得

注※1　図 8-10 中の番号と対応しています。

注※2　セッショントレースについての情報はレベルが「詳細」の場合だけ出力されます。

Web コンテナでのセッショントレースのトレース取得ポイントを次の図に示します。

図 8-10　Web コンテナのトレース取得ポイント（セッショントレース）



（凡例）

：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRFトレース取得レベルは「詳細」です。

：トレース取得ポイントを示します。
PRFトレース取得レベルは，出力されるイベントIDによって異なります。

注※
グローバルセッションIDも取得できるトレース取得ポイントを示します。

それぞれのポイントで取得できるセッション ID について説明します。

2，3，4，7，8，9 のポイント

トレース取得ポイントで有効なセッション ID を取得できます。ただし，J2EE アプリケーションでセッションが破棄される場合があります。

また，これらのポイントでは，グローバルセッション ID も取得できます。取得できるグローバルセッション ID の内容は，トレース取得ポイントごとに異なります。

- 2 のポイントは，一つのリクエストで最初にイベント ID「0x8203」が出力されるトレース取得ポイントです。このトレース取得ポイントでは，Web クライアントからリクエストとして送信されたグローバルセッション ID が取得できます。ただし，このポイントでは，すでに無効になっているグローバルセッション ID が出力される場合もあります。

- 3，4，7，8，9 のポイントで出力される，イベント ID が「0x8216」「0x8202」「0x8203」「0x8206」「0x8207」「0x8214」「0x8215」のトレースには，その時点で有効なグローバルセッション ID が取得できます。

5 のポイント

J2EE アプリケーションでセッションが生成された場合だけ，トレース取得ポイントで有効なセッション ID を取得できます。ただし，J2EE アプリケーションでセッションが破棄される場合があります。

6 のポイント

J2EE アプリケーションでセッションが破棄された場合だけ，トレース取得ポイントで無効になったセッション ID を取得できます。ただし，J2EE アプリケーションでセッションが破棄される場合があります。

10，11，12，13 のポイント

トレース取得ポイントで有効なセッション ID を取得できます。ただし，J2EE アプリケーションでセッションが破棄される場合があります。

14，15 のポイント

トレース取得ポイントで有効なセッション ID を取得できます。なお，このトレース取得ポイントでリクエスト処理が完了すると，以降の J2EE アプリケーションでセッションが破棄されることはありません。

17 のポイント

有効期限を超えたセッションが破棄された場合だけ，無効になったセッション ID を取得できます。

8.5.2 取得できるトレース情報

セッショントレースについて，Web コンテナで取得できるトレース情報を次の表に示します。なお，イベント ID が「0x8202」，「0x8203」，「0x8206」，「0x8207」，「0x8214」，および「0x8215」のトレース取得ポイントでは，グローバルセッションについての情報も出力されます。

表 8–9 Web コンテナで取得できるトレース情報（セッショントレース）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値（ヘッダ名未指定の場合はなし）
4，9	0x8202	A	クラス名（JSP 呼び出し時は JSP ファイル名）	－	－
		B		コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
2，3	0x8203	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					文字数:グローバルセッション ID>
7	0x8206	B	クラス名	ディスパッチのタイプ コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
4，9	0x8207	B	−	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
5	0x8208	B	コンテキストルート名	セッション有効期間	<セッション ID 文字数:セッション ID>
6	0x8209	B	コンテキストルート名	セッション生成時刻	<セッション ID 文字数:セッション ID>
17	0x8210	B	コンテキストルート名	セッション有効期間:セッション生成時刻	<セッション ID 文字数:セッション ID>
8	0x8214	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
8	0x8215	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
2	0x8216	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
16	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>
10，13	0x8302	A	クラス名（JSP 呼び出し時は JSP ファイル名）	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻><例外名:セッション ID 文字数:セッション ID>
14，15	0x8303	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
12	0x8306	B	クラス名	ディスパッチのタイプ コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
10，13	0x8307	B	−	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
11	0x8314	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
11	0x8315	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
15	0x8316	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					• 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-10 中の番号と対応しています。

8.6 Web コンテナのトレース取得ポイント（フィルタのトレース）

ここでは，フォワード時，またはインクルード時に呼び出されるフィルタを設定した場合の Web コンテナのトレースのトレース取得ポイントと，取得できるトレース情報について説明します。

なお，フォワード時，またはインクルード時に呼び出されるフィルタを設定した場合の Web コンテナでは，正常に処理が終了した場合と障害が発生した場合で取得できるトレース情報が異なります。ここでは，それぞれの場合について説明します。

なお，JSP の page ディレクティブで errorPage 属性を使用しエラーページを設定して，JSP で例外が発生した場合，リクエストの forward でエラーページが表示されます。したがって，JSP でエラーページを表示する場合も，forward 時のトレースが出力されます。

8.6.1 正常に処理が終了した場合の Web コンテナのトレース取得ポイント（フィルタのトレース）

(1) トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-10 正常終了時の Web コンテナでのトレース取得ポイントの詳細（フィルタのトレース）

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8202	3	サーブレット／JSP の呼び出し直前	A／B
0x8202	6	サーブレット／JSP の呼び出し直前	A／B
0x8203	2	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの呼び出し直前（web.xml の<filter-mapping>タグの<dispatcher>タグを省略，または<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8206	4	RequestDispatcher 経由のサーブレット／JSP 呼び出し直前	B
0x8207	6	静的コンテンツ呼び出し直前（DefaultServlet）	B
0x8214	5	フォワード時に実行されるフィルタの呼び出し直前（web.xml の<filter-mapping>タグの<dispatcher>タグで"FORWARD"を指定したフィルタ）	B
0x8215	5	インクルード時に実行されるフィルタの呼び出し直前（web.xml の<filter-mapping>タグの<dispatcher>タグで"INCLUDE"を指定したフィルタ）	B
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8302	7	サーブレット／JSP の処理完了直後	A／B

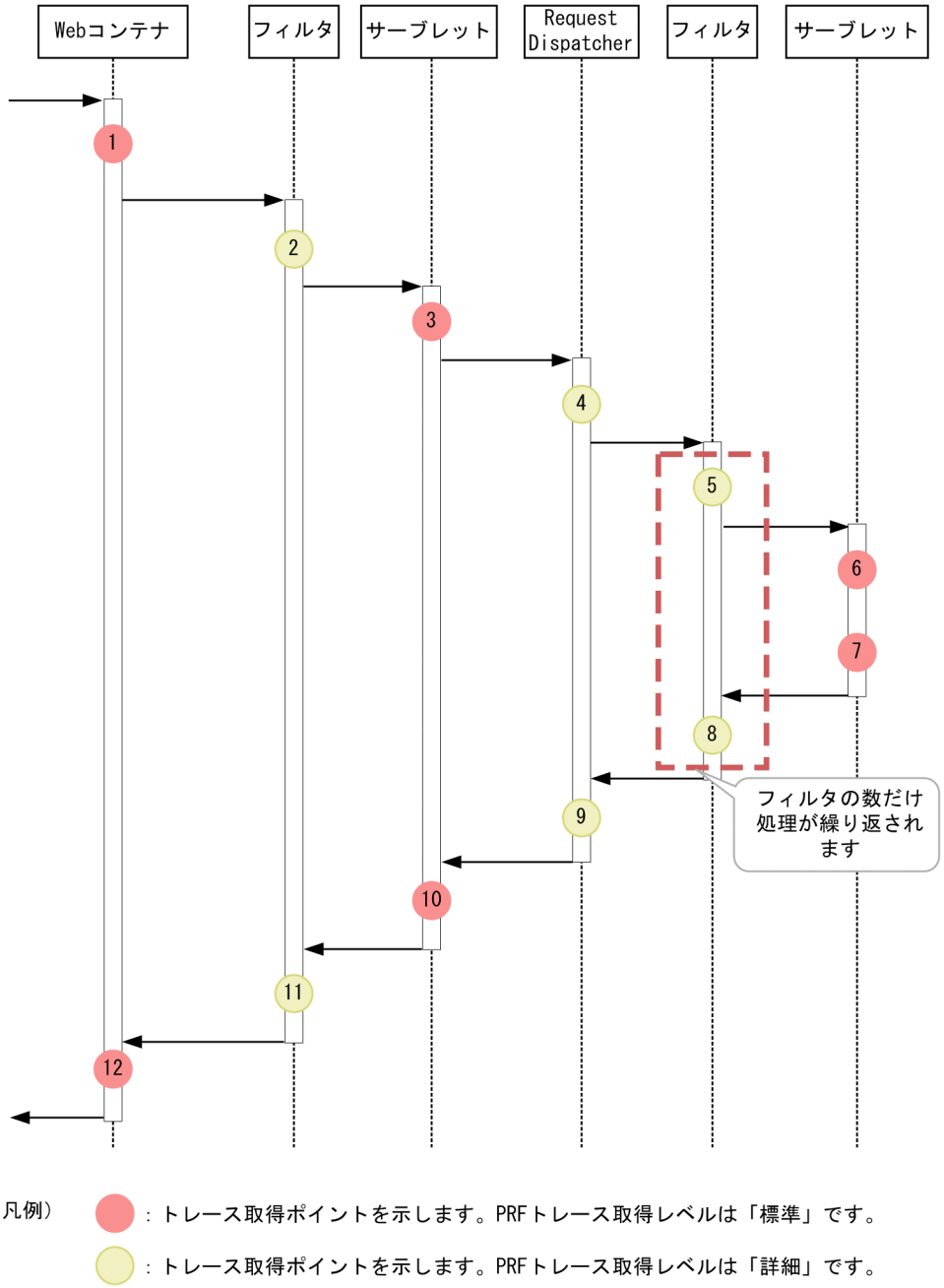
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8302	10	サーブレット／JSP の処理完了直後	A／B
0x8303	11	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8306	9	RequestDispatcher 経由のサーブレット／JSP 処理完了直後	B
0x8307	7	静的コンテンツ処理完了直後(DefaultServlet)	B
0x8314	8	フォワード時に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"FORWARD"を指定したフィルタ）	B
0x8315	8	インクルード時に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"INCLUDE"を指定したフィルタ）	B
0x8336	12	リクエスト処理完了直後	A

（凡例）A：標準　B：詳細　A／B：標準と詳細で異なる情報を取得

注※　図 8-11 中の番号と対応しています。

フォワード時，またはインクルード時に呼び出されるフィルタを設定した場合の Web コンテナでのトレース取得ポイントを次の図に示します。

図 8-11　正常終了時の Web コンテナでのトレース取得ポイント（フィルタのトレース）



(2)　取得できるトレース情報

フォワード時，またはインクルード時に呼び出されるフィルタを設定した場合の Web コンテナで取得できるトレース情報を次の表に示します。

表 8-11　正常終了時の Web コンテナで取得できるトレース情報（フィルタのトレース）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					(ヘッダ名未指定の場合はなし)
2	0x8203	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
3	0x8202	A	クラス名(JSP 呼び出しの場合は JSP ファイル名)	−	−
		B		コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
4	0x8206	B	クラス名	ディスパッチのタイプ コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
5	0x8214	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
	0x8215	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
6	0x8202	A	クラス名(JSP 呼び出しの場合は JSP ファイル名)	−	−
		B		コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
	0x8207	B	−	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
7	0x8302	A	クラス名(JSP 呼び出しの場合は JSP ファイル名)	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					• 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8307	B	−	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
8	0x8314	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8315	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
9	0x8306	B	クラス名	ディスパッチのタイプ コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
10	0x8302	A	クラス名（JSP 呼び出しの場合は JSP ファイル名）	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
11	0x8303	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
12	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-11 中の番号と対応しています。

8.6.2　例外が発生した場合の Web コンテナのトレース取得ポイント（フィルタのトレース）

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-12　例外が発生した場合の Web コンテナでのトレース取得ポイントの詳細（フィルタのトレース）

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8202	3	サーブレット／JSP の呼び出し直前	A／B
0x8202	8	サーブレット／JSP の呼び出し直前	A／B
0x8203	2	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの呼び出し直前（web.xml の<filter-mapping>タグの<dispatcher>タグを省略，または<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8206	6	RequestDispatcher 経由のサーブレット／JSP 呼び出し直前	B

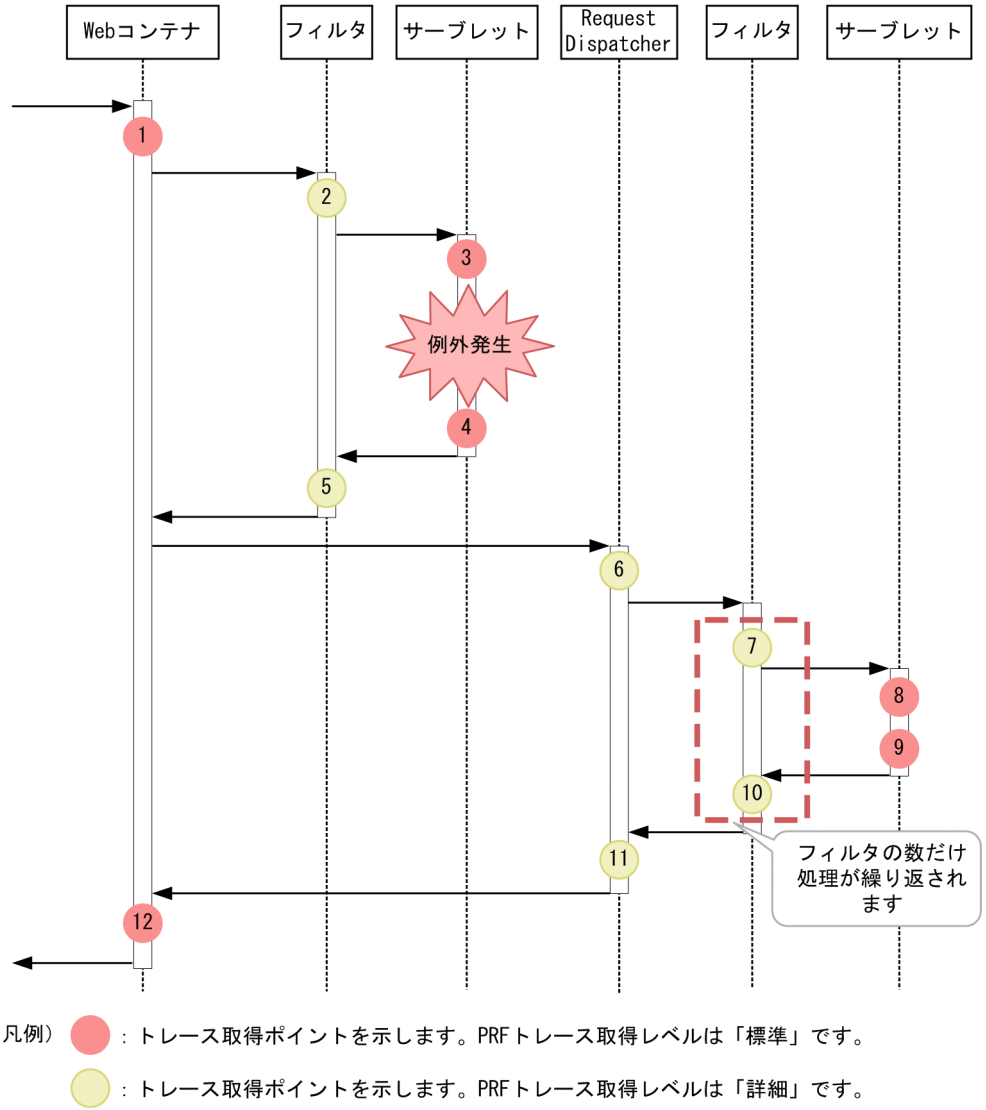
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8207	8	静的コンテンツ呼び出し直前（DefaultServlet）	B
0x8216	7	エラーページに転送される際に実行されるフィルタの呼び出し直前（web.xml の<filter-mapping>タグの<dispatcher>タグで"ERROR"を指定したフィルタ）	B
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8302	4，9	サーブレット／JSP の処理完了直後	A／B
0x8303	5	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8306	11	RequestDispatcher 経由のサーブレット／JSP 処理完了直後	B
0x8307	9	静的コンテンツ処理完了直後(DefaultServlet)	B
0x8316	10	エラーページに転送される際に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"ERROR"を指定したフィルタ）	B
0x8336	12	リクエスト処理完了直後	A

（凡例）A：標準　B：詳細　A／B：標準と詳細で異なる情報を取得

注※　図 8-12 中の番号と対応しています。

例外が発生した場合の Web コンテナでのトレース取得ポイントを次の図に示します。

図 8-12　例外が発生した場合の Web コンテナでのトレース取得ポイント



(2)　取得できるトレース情報

フォワード時，またはインクルード時に呼び出されるフィルタを設定した場合の Web コンテナで取得できるトレース情報を次の表に示します。

表 8-13　例外が発生した場合の Web コンテナで取得できるトレース情報（フィルタのトレース）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値（ヘッダ名未指定の場合はなし）
2	0x8203	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッ

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					ション ID 文字数:グローバルセッション ID>
3	0x8202	A	クラス名（JSP 呼び出しの場合は JSP ファイル名）	－	－
		B		コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
4	0x8302	A	クラス名（JSP 呼び出しの場合は JSP ファイル名）	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
5	0x8303	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
6	0x8206	B	クラス名	ディスパッチのタイプ コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
7	0x8216	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
8	0x8202	A	クラス名（JSP 呼び出しの場合は JSP ファイル名）	－	－
		B		コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0x8207	B	−	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
9	0x8302	A	クラス名（JSP 呼び出しの場合は JSP ファイル名）	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8307	B	−	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
10	0x8316	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
11	0x8306	B	クラス名	ディスパッチのタイプ コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
12	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>

（凡例）A：標準　B：詳細　－：該当なし
注※　図 8-12 中の番号と対応しています。

8.7 Web コンテナのトレース取得ポイント（データベースセッションフェイルオーバ機能のトレース）

ここでは，データベースセッションフェイルオーバ機能を使用した場合のトレース取得ポイントと，取得できるトレース情報について説明します。

8.7.1 HTTP セッションを作成するリクエスト処理のトレース取得ポイントと取得できるトレース情報（データベースセッションフェイルオーバ機能のトレース）

HTTP セッションを作成するリクエスト処理のトレース取得ポイントと取得できるトレース情報について説明します。

(1) トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-14 HTTP セッションを作成するリクエスト処理のトレース取得ポイントの詳細（データベースセッションフェイルオーバ機能）

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8202	2	サーブレット／JSP 呼び出し直前	A／B
0x8203	2	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの呼び出し直前（web.xml の<filter-mapping>タグの<dispatcher>タグを省略，または<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8207	2	静的コンテンツ呼び出し直前（DefaultServlet）	B
0x8219	6	データベースセッションフェイルオーバ機能による HTTP セッションの属性情報のシリアライズ開始直前	A
0x8222	8	データベースセッションフェイルオーバ機能による Web アプリケーション処理後のデータベースアクセス開始直前	A
0x8223	3	データベースセッションフェイルオーバ機能による HTTP セッション作成時のデータベースアクセス開始直前	A
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8302	5	サーブレット／JSP 処理完了直後	A／B
0x8303	5	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8307	5	静的コンテンツ処理完了直後（DefaultServlet）	B

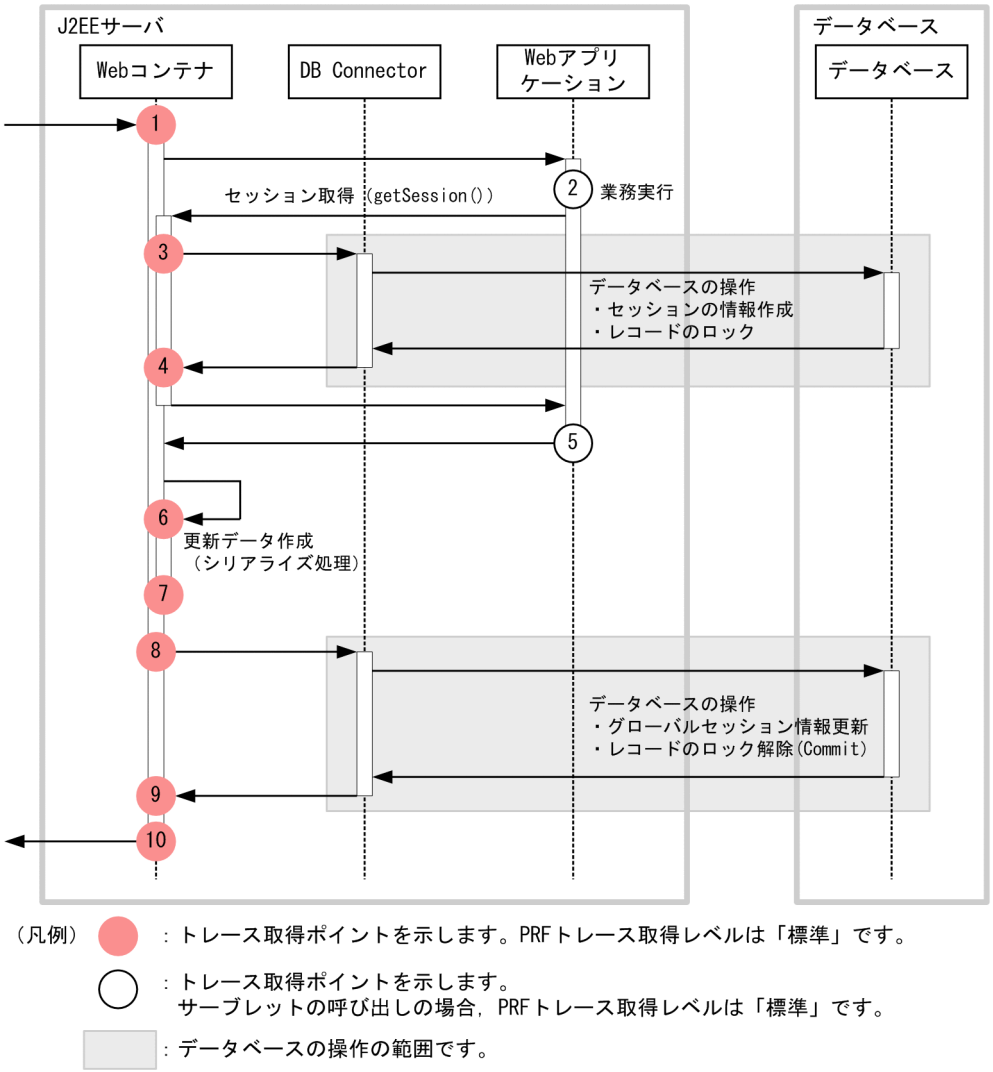
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8316	5	エラーページに転送される際に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"ERROR"を指定したフィルタ）	B
0x8319	7	データベースセッションフェイルオーバ機能による HTTP セッションの属性情報のシリアライズ終了直後	A
0x8322	9	データベースセッションフェイルオーバ機能による Web アプリケーション処理後のデータベースアクセス終了直後	A
0x8323	4	データベースセッションフェイルオーバ機能による HTTP セッション作成時のデータベースアクセス終了直後	A
0x8336	10	リクエスト処理完了直後	A

（凡例）A：標準　B：詳細　A／B：標準と詳細で異なる情報を取得

注※　図 8-13 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-13　HTTP セッションを作成するリクエスト処理のトレース取得ポイント（データベースセッションフェイルオーバ機能）



(2)　取得できるトレース情報

HTTP セッションを作成するリクエスト処理で取得できるトレース情報を次の表に示します。

表 8-15　HTTP セッションを作成するリクエスト処理で取得できるトレース情報（データベースセッションフェイルオーバ機能）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値（ヘッダ名未指定の場合はなし）
2	0x8202	A	クラス名	−	−

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
		B	(JSP 呼び出し時は JSP ファイル名)	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
	0x8207	B	−	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
	0x8203	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
3	0x8223	A	−	−	−
4	0x8323	A	完全性保障モードが有効な場合 排他を取得したレコード番号 完全性保障モードが無効な場合 -1	−	• 正常時 <入り口時刻><セッション ID 文字数:作成した HTTP セッションのセッション ID> • 例外発生時 <入り口時刻><例外名>
5	0x8302	A	クラス名 (JSP 呼び出し時は JSP ファイル名)	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8307	B	−	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8303	B	クラス名	コンテキストルート名	• 正常時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8316	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
6	0x8219	A	リクエスト URL	−	<セッション ID 文字数:セッション ID>
7	0x8319	A	リクエスト URL	シリアライズ後の HTTP セッションの属性情報のサイズ(バイト)	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
8	0x8222	A	−	−	<セッション ID の文字数:HTTP セッションのセッション ID>
9	0x8322	A	完全性保障モードが有効な場合 排他を解放したレコード番号 完全性保障モードが無効な場合 -1	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
10	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>>

(凡例) A：標準　B：詳細　−：該当なし

注※　図 8-13 中の番号と対応しています。

8.7.2 HTTP セッションを更新するリクエスト処理のトレース取得ポイントと取得できるトレース情報（データベースセッションフェイルオーバ機能のトレース）

HTTP セッションを更新するリクエスト処理のトレース取得ポイントと取得できるトレース情報について説明します。

(1) トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-16 HTTP セッションを更新するリクエスト処理のトレース取得ポイントの詳細（データベースセッションフェイルオーバ機能）

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0x8202	6	サーブレット／JSP 呼び出し直前	A／B
0x8203	6	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの呼び出し直前（web.xml の<filter-mapping>タグの<dispatcher>タグを省略，または<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8207	6	静的コンテンツ呼び出し直前（DefaultServlet）	B
0x8219※2	8	データベースセッションフェイルオーバ機能による HTTP セッションの属性情報のシリアライズ開始直前	A
0x8220	4	データベースセッションフェイルオーバ機能による HTTP セッションの属性情報のデシリアライズ開始直前	A
0x8221	2	データベースセッションフェイルオーバ機能による Web アプリケーション処理前のデータベースアクセス開始直前	A
0x8222※2	10	データベースセッションフェイルオーバ機能による Web アプリケーション処理後のデータベースアクセス開始直前	A
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8302	7	サーブレット／JSP 処理完了直後	A／B
0x8303	7	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8307	7	静的コンテンツ処理完了直後（DefaultServlet）	B
0x8316	7	エラーページに転送される時に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"ERROR"を指定したフィルタ）	B
0x8319※2	9	データベースセッションフェイルオーバ機能による HTTP セッションの属性情報のシリアライズ終了直後	A

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0x8320	5	データベースセッションフェイルオーバ機能による HTTP セッションの属性情報のデシリアライズ終了直後	A
0x8321	3	データベースセッションフェイルオーバ機能による Web アプリケーション処理前のデータベースアクセス終了直後	A
0x8322※2	11	データベースセッションフェイルオーバ機能による Web アプリケーション処理後のデータベースアクセス終了直後	A
0x8336	12	リクエスト処理完了直後	A

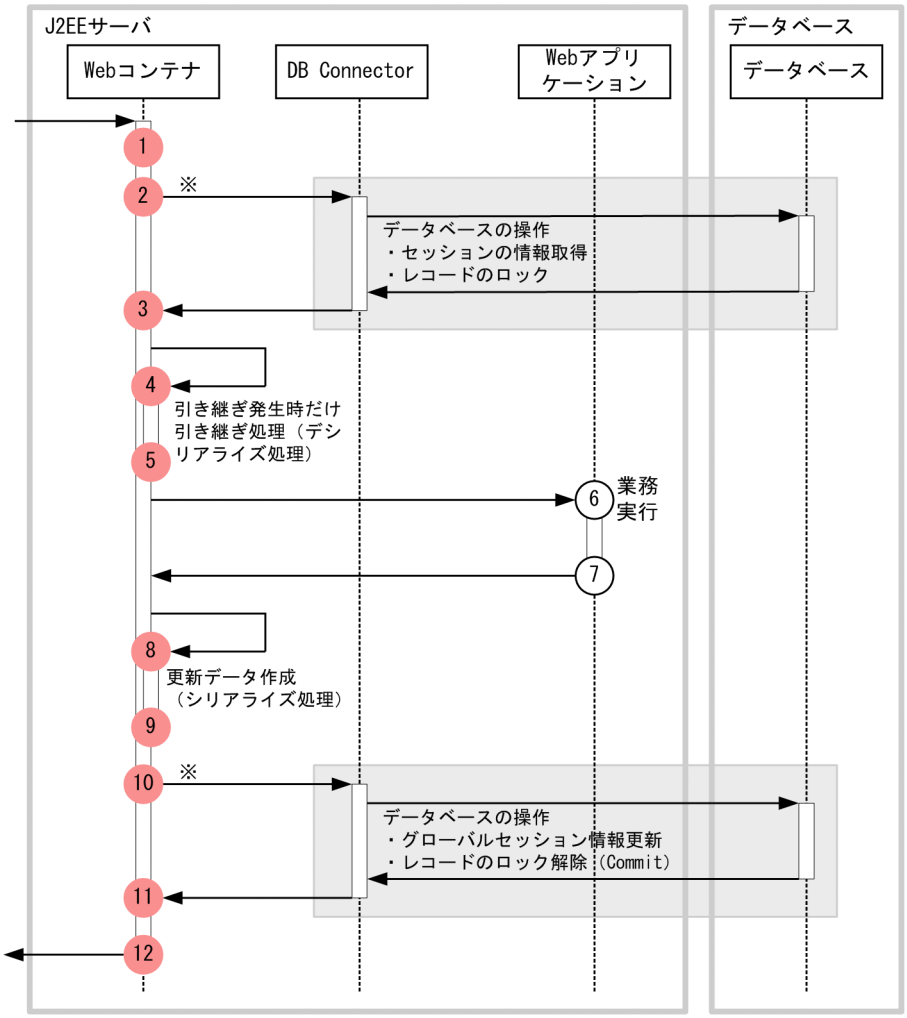
（凡例）A：標準　B：詳細　A／B：標準と詳細で異なる情報を取得

注※1　図 8-14 中の番号と対応しています。

注※2　HTTP セッションの参照専用リクエストの場合，出力されません。

トレース取得ポイントを次の図に示します。

図 8-14　HTTP セッションを更新するリクエスト処理のトレース取得ポイント（データベースセッションフェイルオーバ機能）



（凡例）　：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

：トレース取得ポイントを示します。
サーブレットの呼び出しの場合，PRFトレース取得レベルは「標準」です。

：データベースの操作の範囲です。

注※　実際のDB Connector呼び出しは，複数回発生します。

(2)　取得できるトレース情報

HTTP セッションを更新するリクエスト処理で取得できるトレース情報を次の表に示します。

表 8-17　HTTP セッションを更新するリクエスト処理で取得できるトレース情報（データベースセッションフェイルオーバ機能）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値 (ヘッダ名未指定の場合はなし
2	0x8221	A	−	−	<セッション ID の文字数:リクエストで受信したセッション ID>
3	0x8321	A	完全性保障モードが有効な場合 排他を取得したレコード番号 完全性保障モードが無効な場合 -1	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
4	0x8220	A	リクエスト URL	デシリアライズ前の HTTP セッションの属性情報のサイズ（バイト）	<セッション ID 文字数:セッション ID>
5	0x8320	A	リクエスト URL	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
6	0x8202	A	クラス名 (JSP 呼び出し時は JSP ファイル名)	−	−
		B		コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
	0x8207	B	−	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
	0x8203	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
7	0x8302	A	クラス名 (JSP 呼び出し時は JSP ファイル名)	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
		B		コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8307	B	−	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8303	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8316	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
8	0x8219	A	リクエスト URL	−	<セッション ID 文字数:セッション ID>
9	0x8319	A	リクエスト URL	シリアライズ後の HTTP セッションの属性情報のサイズ(バイト)	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
10	0x8222	A	−	−	<セッション ID の文字数:HTTP セッションのセッション ID>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
11	0x8322	A	完全性保障モードが有効な場合 　排他を解放したレコード番号 完全性保障モードが無効な場合 　-1	－	• 正常時 　<入り口時刻> • 例外発生時 　<入り口時刻><例外名>
12	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-14 中の番号と対応しています。

8.7.3　HTTP セッションを無効化するリクエスト処理のトレース取得ポイントと取得できるトレース情報（データベースセッションフェイルオーバ機能のトレース）

HTTP セッションを無効化するリクエスト処理のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-18　HTTP セッションを無効化するリクエスト処理のトレース取得ポイントの詳細（データベースセッションフェイルオーバ機能）

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8202	6	サーブレット／JSP 呼び出し直前	A／B
0x8203	6	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの呼び出し直前（web.xml の<filter-mapping>タグの<dispatcher>タグを省略，または<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8207	6	静的コンテンツ呼び出し直前（DefaultServlet）	B
0x8220	4	データベースセッションフェイルオーバ機能による HTTP セッションの属性情報のデシリアライズ開始直前	A
0x8221	2	データベースセッションフェイルオーバ機能による Web アプリケーション処理前のデータベースアクセス開始直前	A

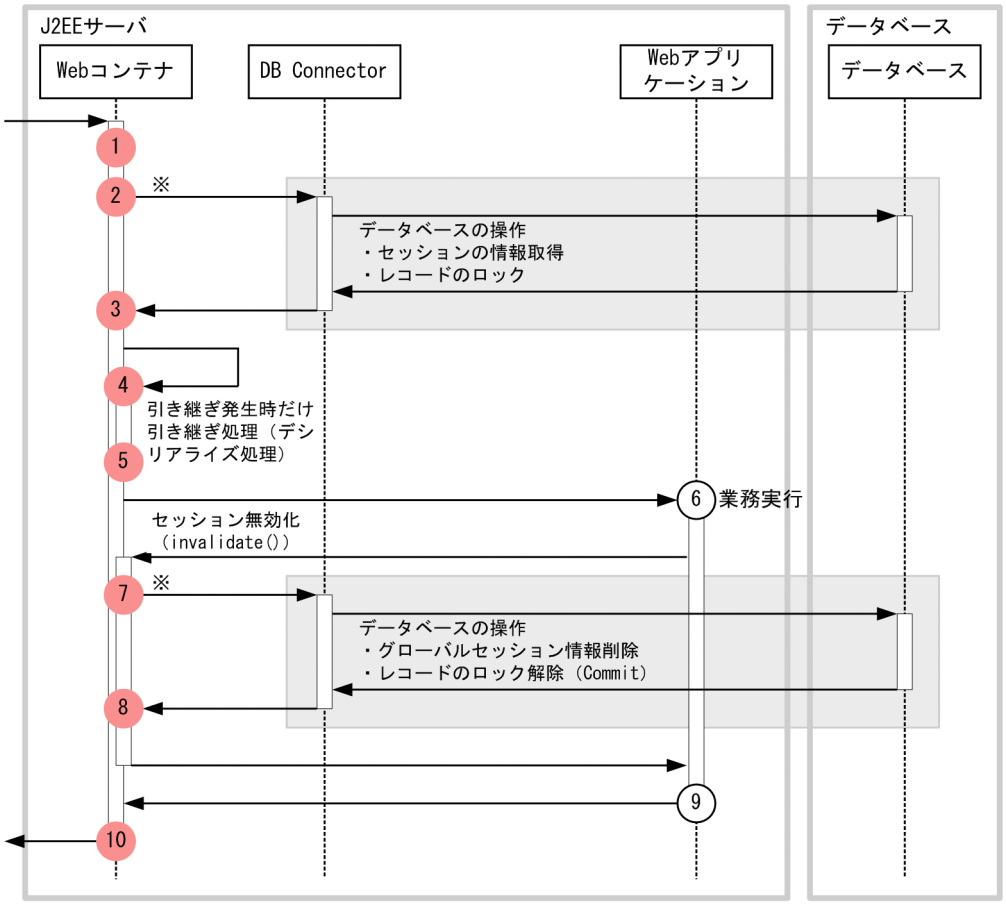
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8224	7	データベースセッションフェイルオーバ機能による HTTP セッション無効化時のデータベースアクセス開始直前	A
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8302	9	サーブレット／JSP 処理完了直後	A／B
0x8303	9	リクエストを受信したサーブレット／JSP の実行前に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"REQUEST"を指定したフィルタ）	B
0x8307	9	静的コンテンツ処理完了直後（DefaultServlet）	B
0x8316	9	エラーページに転送される際に実行されるフィルタの処理完了直後（web.xml の<filter-mapping>タグの<dispatcher>タグで"ERROR"を指定したフィルタ）	B
0x8320	5	データベースセッションフェイルオーバ機能による HTTP セッションの属性情報のデシリアライズ終了直後	A
0x8321	3	データベースセッションフェイルオーバ機能による Web アプリケーション処理前のデータベースアクセス終了直後	A
0x8324	8	データベースセッションフェイルオーバ機能による HTTP セッション無効化時のデータベースアクセス終了直後	A
0x8336	10	リクエスト処理完了直後	A

（凡例）A：標準　B：詳細　A／B：標準と詳細で異なる情報を取得

注※　図 8-15 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-15　HTTP セッションを無効化するリクエスト処理のトレース取得ポイント（データベースセッションフェイルオーバ機能）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。
サーブレットの呼び出しの場合，PRF トレース取得レベルは「標準」です。

：データベースの操作の範囲です。

注※　実際のDB Connector呼び出しは，複数回発生します。

(2)　取得できるトレース情報

HTTP セッションを無効化するリクエスト処理で取得できるトレース情報を次の表に示します。

表 8-19　HTTP セッションを無効化するリクエスト処理で取得できるトレース情報（データベースセッションフェイルオーバ機能）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値（ヘッダ名未指定の場合はなし）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
2	0x8221	A	－	－	<セッション ID の文字数:リクエストで受信したセッション ID>
3	0x8321	A	完全性保障モードが有効な場合 排他を取得したレコード番号 完全性保障モードが無効な場合 -1	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
4	0x8220	A	リクエスト URL	デシリアライズ前の HTTP セッションの属性情報のサイズ（バイト）	<セッション ID 文字数:セッション ID>
5	0x8320	A	リクエスト URL	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
6	0x8202	A	クラス名 (JSP 呼び出し時は JSP ファイル名)	－	－
		B		コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
	0x8207	B	－	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
	0x8203	B	クラス名	コンテキストルート名	<セッション ID 文字数:セッション ID:グローバルセッション ID 文字数:グローバルセッション ID>
7	0x8224	A	－	－	<セッション ID の文字数:無効化した HTTP セッションのセッション ID>
8	0x8324	A	完全性保障モードが有効な場合 排他を解放したレコード番号 完全性保障モードが無効な場合 -1	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
9	0x8302	A	クラス名 (JSP 呼び出し時は JSP ファイル名)	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
		B		コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8307	B	－	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8303	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
	0x8316	B	クラス名	コンテキストルート名	• 正常時 <入り口時刻><セッション ID 文字数:セッション ID> • 例外発生時 <入り口時刻><例外名:セッション ID 文字数:セッション ID>
10	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>

(凡例) A：標準　B：詳細　－：該当なし

注※　図 8-15 中の番号と対応しています。

8.7.4　有効期限監視で HTTP セッションを無効化するリクエスト処理のトレース取得ポイントと取得できるトレース情報（データベースセッションフェイルオーバ機能のトレース）

有効期限監視で HTTP セッションを無効化するリクエスト処理のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8–20　有効期限監視で HTTP セッションを無効化するリクエスト処理のトレース取得ポイントの詳細（データベースセッションフェイルオーバ機能）

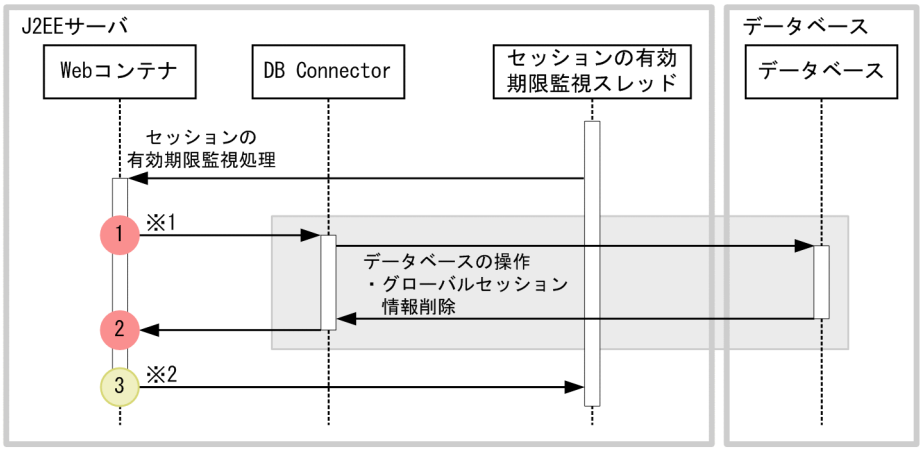
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8210	3	セッションタイムアウト後	B
0x8225	1	データベース上のグローバルセッション情報の有効期限監視処理開始直前	A
0x8325	2	データベース上のグローバルセッション情報の有効期限監視処理終了直後	A

（凡例）A：標準　B：詳細

注※　図 8-16 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-16　有効期限監視で HTTP セッションを無効化するリクエスト処理のトレース取得ポイント（データベースセッションフェイルオーバ機能）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

：データベースの操作の範囲です。

注※1　実際のDB Connector呼び出しは，複数回発生します。

注※2　削除したHTTPセッションの個数分出力します。

(2)　取得できるトレース情報

有効期限監視で HTTP セッションを無効化するリクエスト処理で取得できるトレース情報を次の表に示します。

表 8-21　有効期限監視で HTTP セッションを無効化するリクエスト処理で取得できるトレース情報（データベースセッションフェイルオーバ機能）

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8225※2	A	－	－	－
2	0x8325※2	A	グローバルセッション情報の有効期限チェックを実施した場合 　無効化したグローバルセッションの個数 グローバルセッション情報の有効期限チェックを実施しなかった場合 　現在，有効期限チェックを担当している J2EE サーバの J2EE サーバ名（IP アドレス）	－	• 正常時 　<入り口時刻> • 例外発生時 　<入り口時刻><例外名>

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
3	0x8210	B	コンテキストルート名	セッション有効期間：セッション生成時刻	<セッション ID 文字数:セッション ID>

（凡例）A：標準　B：詳細　－：該当なし

注※1　図 8-16 中の番号と対応しています。

注※2　完全性保障モードが無効な場合は，出力されません。

8.8 Web コンテナのトレース取得ポイント（セッションマネージャの指定機能のトレース）

ここでは，セッションマネージャの指定機能を使用した場合のトレース取得ポイントと，取得できるトレース情報について説明します。

8.8.1 HTTP セッションを作成するリクエスト処理のトレース取得ポイントと取得できるトレース情報（セッションマネージャの指定機能のトレース）

HTTP セッションを作成するリクエスト処理のトレース取得ポイントと取得できるトレース情報について説明します。

(1) トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-22 HTTP セッションを作成するリクエスト処理のトレース取得ポイントの詳細（セッションマネージャの指定機能）

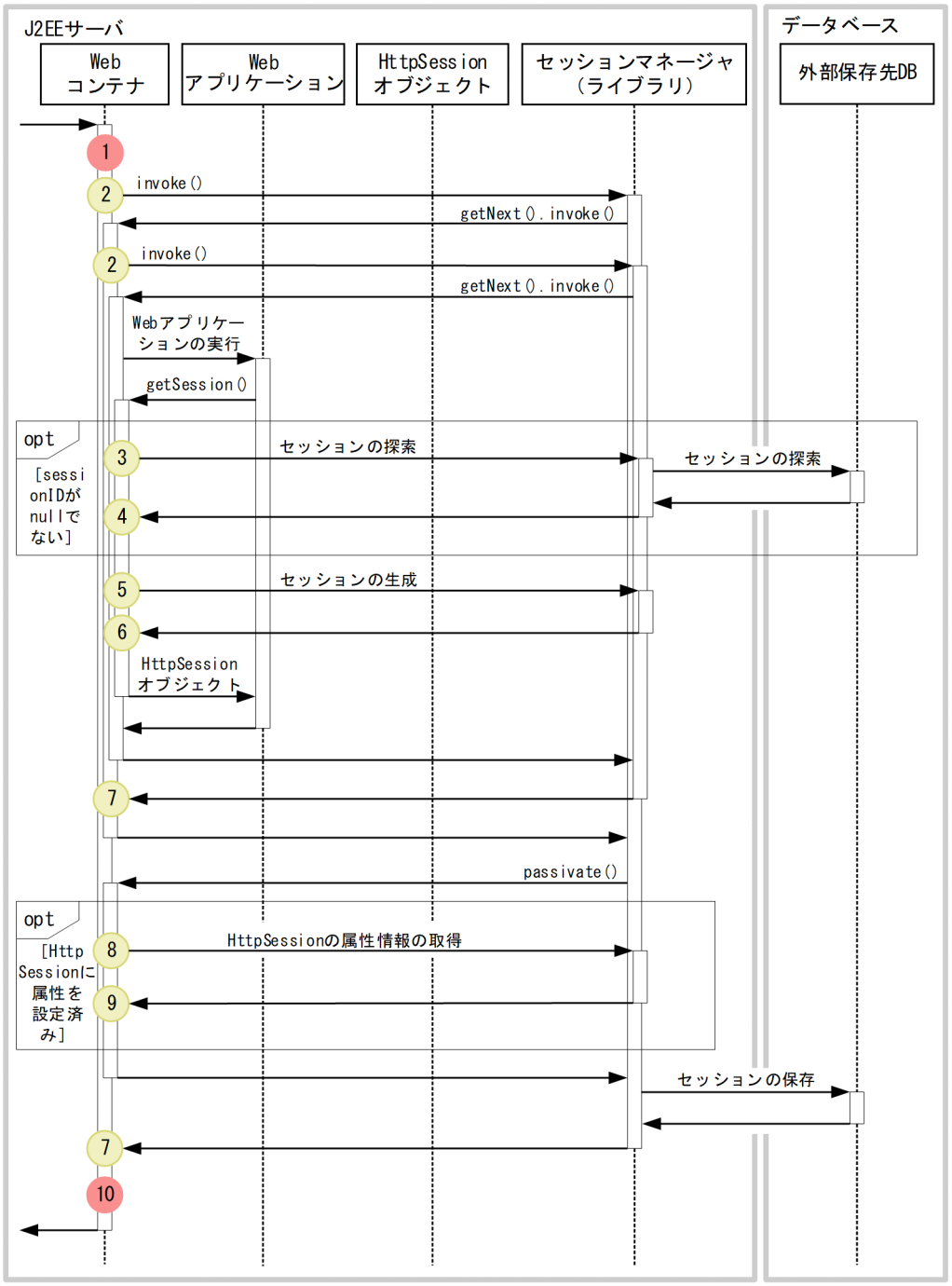
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8240	5	セッションの生成直前（セッションマネージャの指定機能）	B
0x8241	3	セッションの探索直前（セッションマネージャの指定機能）	B
0x8248	8	セッションマネージャの指定機能による HTTP セッションの属性情報の取得直前	B
0x8252	2	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.Valve インタフェースの実装クラスの invoke メソッド呼び出し直前	B
0x8336	10	リクエスト処理完了直後	A
0x8340	6	セッションの生成直後（セッションマネージャの指定機能）	B
0x8341	4	セッションの探索直後（セッションマネージャの指定機能）	B
0x8348	9	セッションマネージャの指定機能による HTTP セッションの属性情報の取得直後	B
0x8352	7	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.Valve インタフェースの実装クラスの invoke メソッド呼び出し直後	B

（凡例）A：標準　B：詳細

注※　図 8-17 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-17　HTTP セッションを作成するリクエスト処理のトレース取得ポイント（セッションマネージャの指定機能）



（凡例）

：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(2) 取得できるトレース情報

HTTP セッションを作成するリクエスト処理で取得できるトレース情報を次の表に示します。

表 8-23 HTTP セッションを作成するリクエスト処理で取得できるトレース情報（セッションマネージャの指定機能）

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名※3	オペレーション名※3※4	オプション※2※4
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値（ヘッダ名未指定の場合はなし）
2	0x8252	B	ライブラリにおける Valve インターフェースの実装クラス名	ライブラリの呼び出し元メソッド名	なし
3	0x8241	B	コンテキストルート名	なし	<セッション ID 文字数:セッション ID>
4	0x8341	B	コンテキストルート名	セッション有効期間	• 正常時 <入り口時刻:セッション ID 文字数:セッション ID> • 異常時 <入り口時刻:例外名>
5	0x8240	B	コンテキストルート名	なし	なし
6	0x8340	B	コンテキストルート名	セッション有効期間	• 正常時 <入り口時刻:セッション ID 文字数:セッション ID> • 異常時 <入り口時刻:例外名>
7	0x8352	B	ライブラリにおける Valve インターフェースの実装クラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>
8	0x8248	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	<セッション ID 文字数:セッション ID:属性名>
9	0x8348	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名※3	オペレーション名※3※4	オプション※2※4
10	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>

（凡例）A：標準　B：詳細

注※1　図 8-17 中の番号と対応しています。

注※2　「例外名」の情報が取得できるトレース取得ポイントでは，PRF トレースに出力した例外のスタックトレースを J2EE サーバの例外ログに出力します。

注※3　33 文字以上の場合，「先頭 16 文字+*+末尾 16 文字」で出力します。

注※4　「セッション有効期間」はセッションが存在する場合だけ出力します。「セッション ID 文字数」「セッション ID」はセッション ID が存在する場合だけ出力します。

8.8.2　HTTP セッションを更新するリクエスト処理のトレース取得ポイントと取得できるトレース情報（セッションマネージャの指定機能のトレース）

HTTP セッションを更新するリクエスト処理のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-24　HTTP セッションを更新するリクエスト処理のトレース取得ポイントの詳細（セッションマネージャの指定機能）

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8241	3	セッションの探索直前（セッションマネージャの指定機能）	B
0x8246	9	セッションマネージャの指定機能による HTTP セッションの属性情報の設定直前	B
0x8247	10	セッションマネージャの指定機能による HTTP セッションの属性情報の消去直前	B
0x8248	4	セッションマネージャの指定機能による HTTP セッションの属性情報の取得直前	B
0x8252	2	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.Valve インタフェースの実装クラスの invoke メソッド呼び出し直前	B
0x8336	8	リクエスト処理完了直後	A

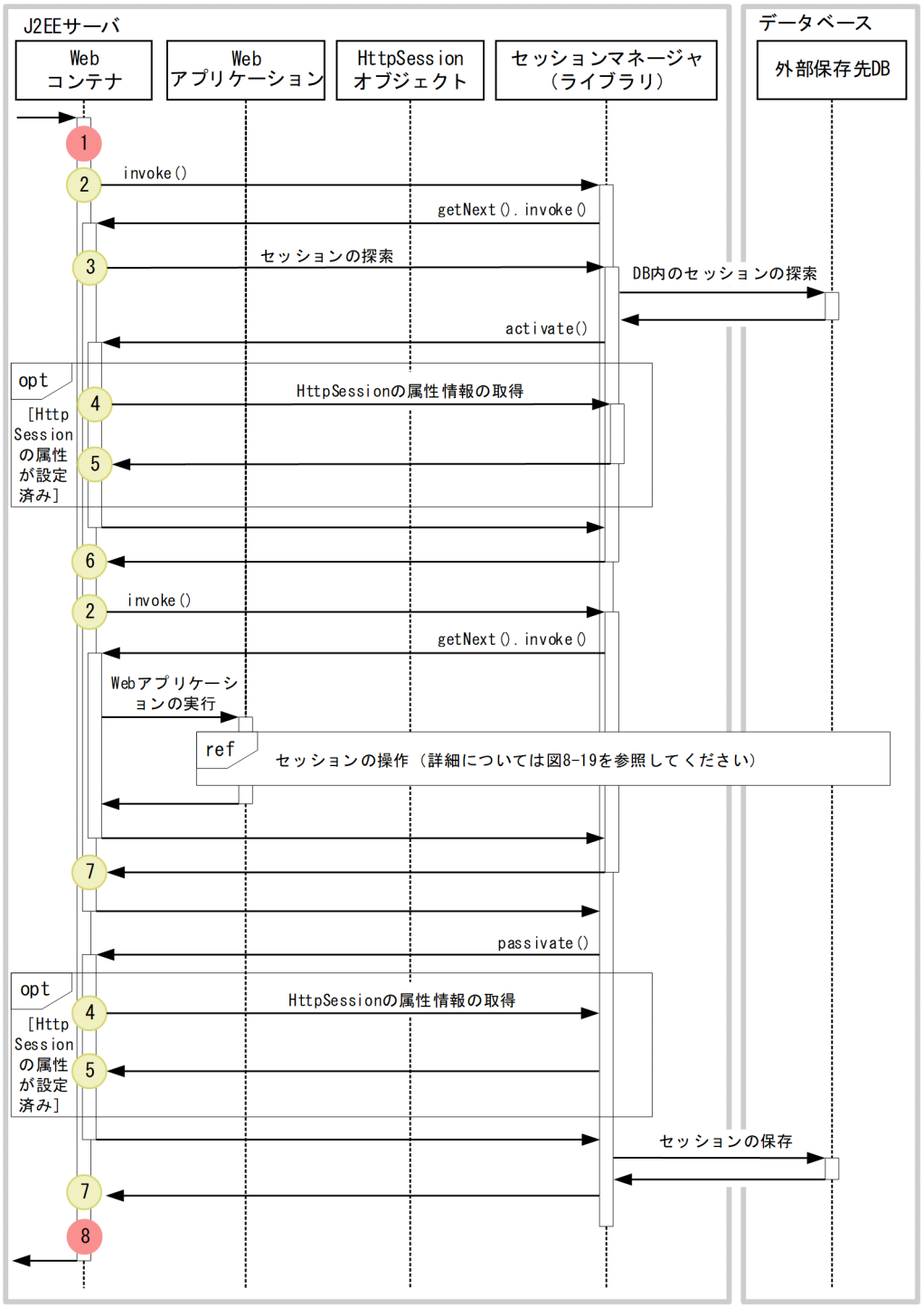
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8341	6	セッションの探索直後（セッションマネージャの指定機能）	B
0x8346	12	セッションマネージャの指定機能による HTTP セッションの属性情報の設定直後	B
0x8347	11	セッションマネージャの指定機能による HTTP セッションの属性情報の消去直後	B
0x8348	5	セッションマネージャの指定機能による HTTP セッションの属性情報の取得直後	B
0x8352	7	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.Valve インタフェースの実装クラスの invoke メソッド呼び出し直後	B

（凡例）A：標準　B：詳細

注※　図 8-18 および図 8-19 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-18　HTTP セッションを更新するリクエスト処理のトレース取得ポイント（セッションマネージャの指定機能）

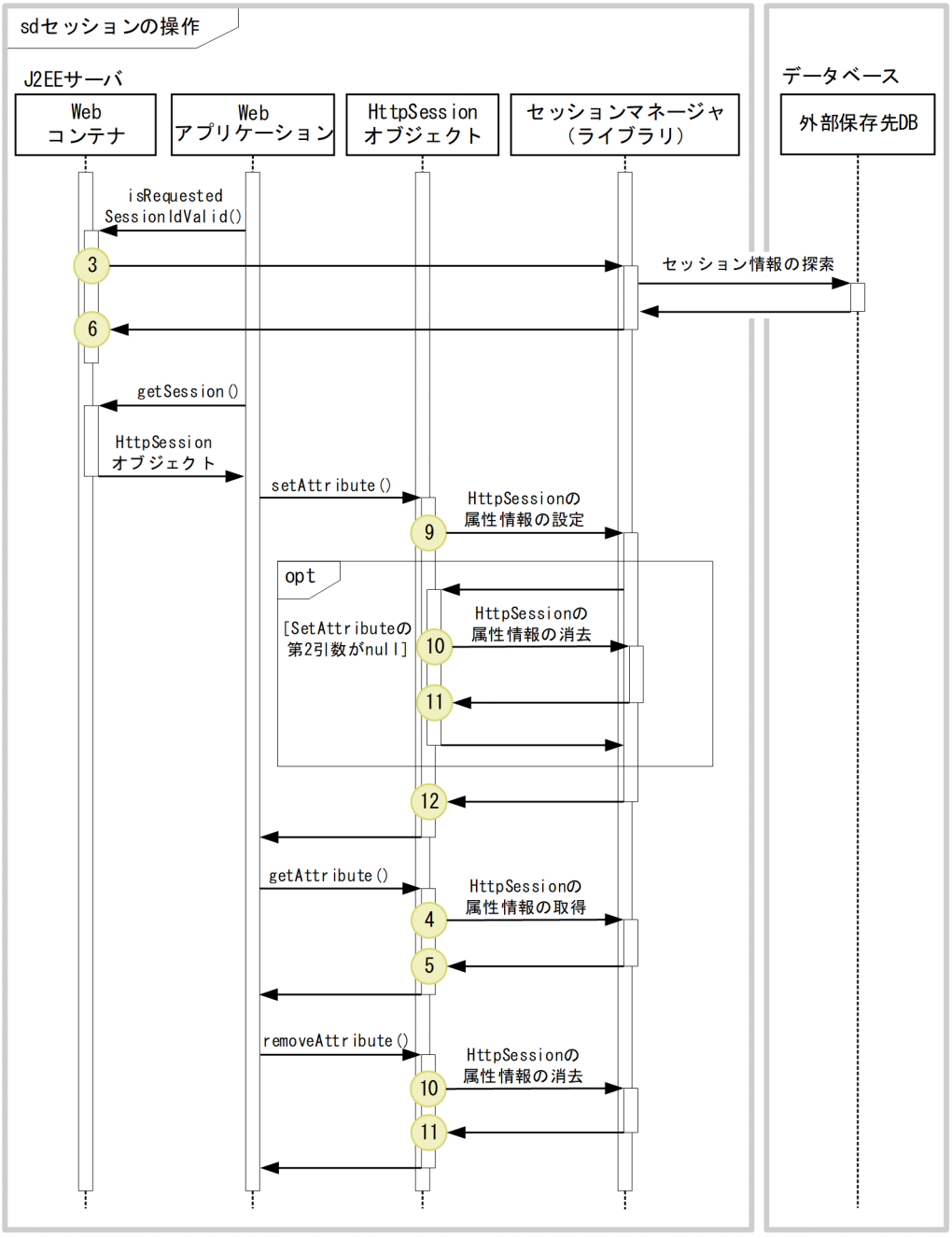


（凡例）

：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRFトレース取得レベルは「詳細」です。

図 8-19　HTTP セッションの操作のトレース取得ポイント（セッションマネージャの指定機能）



(2)　取得できるトレース情報

HTTP セッションを更新するリクエスト処理で取得できるトレース情報を次の表に示します。

表 8-25　HTTP セッションを更新するリクエスト処理で取得できるトレース情報（セッションマネージャの指定機能）

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名※3	オペレーション名※3※4	オプション※2※4
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値 (ヘッダ名未指定の場合はなし)
2	0x8252	B	ライブラリにおける Valve インターフェースの実装クラス名	ライブラリの呼び出し元メソッド名	なし
3	0x8241	B	コンテキストルート名	なし	<セッション ID 文字数:セッション ID>
4	0x8248	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	<セッション ID 文字数:セッション ID:属性名>
5	0x8348	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>
6	0x8341	B	コンテキストルート名	セッション有効期間	• 正常時 <入り口時刻:セッション ID 文字数:セッション ID> • 異常時 <入り口時刻:例外名>
7	0x8352	B	ライブラリにおける Valve インターフェースの実装クラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>
8	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>
9	0x8246	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	<セッション ID 文字数:セッション ID:属性名>
10	0x8247	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	<セッション ID 文字数:セッション ID:属性名>
11	0x8347	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名※3	オペレーション名※3※4	オプション※2※4
					<入り口時刻:例外名>
12	0x8346	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 • <入り口時刻:例外名>

（凡例）A：標準　B：詳細

注※1　図 8-18 および図 8-19 中の番号と対応しています。

注※2　「例外名」の情報が取得できるトレース取得ポイントでは，PRF トレースに出力した例外のスタックトレースを J2EE サーバの例外ログに出力します。

注※3　33 文字以上の場合，「先頭 16 文字+*+末尾 16 文字」で出力します。

注※4　「セッション有効期間」はセッションが存在する場合だけ出力します。「セッション ID 文字数」「セッション ID」はセッション ID が存在する場合だけ出力します。

8.8.3　HTTP セッションを無効化するリクエスト処理のトレース取得ポイントと取得できるトレース情報（セッションマネージャの指定機能のトレース）

HTTP セッションを無効化するリクエスト処理のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-26　HTTP セッションを無効化するリクエスト処理のトレース取得ポイントの詳細（セッションマネージャの指定機能）

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8209	17	セッションの破棄後	B
0x8236	1	リクエスト取得・リクエストヘッダ解析完了直後	A
0x8241	3	セッションの探索直前（セッションマネージャの指定機能）	B
0x8242	7	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.ManagerBase のサブクラスの remove メソッドの呼び出し直前	B
0x8243	11	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.StandardSession のサブクラスの recycle メソッドの呼び出し直前	B

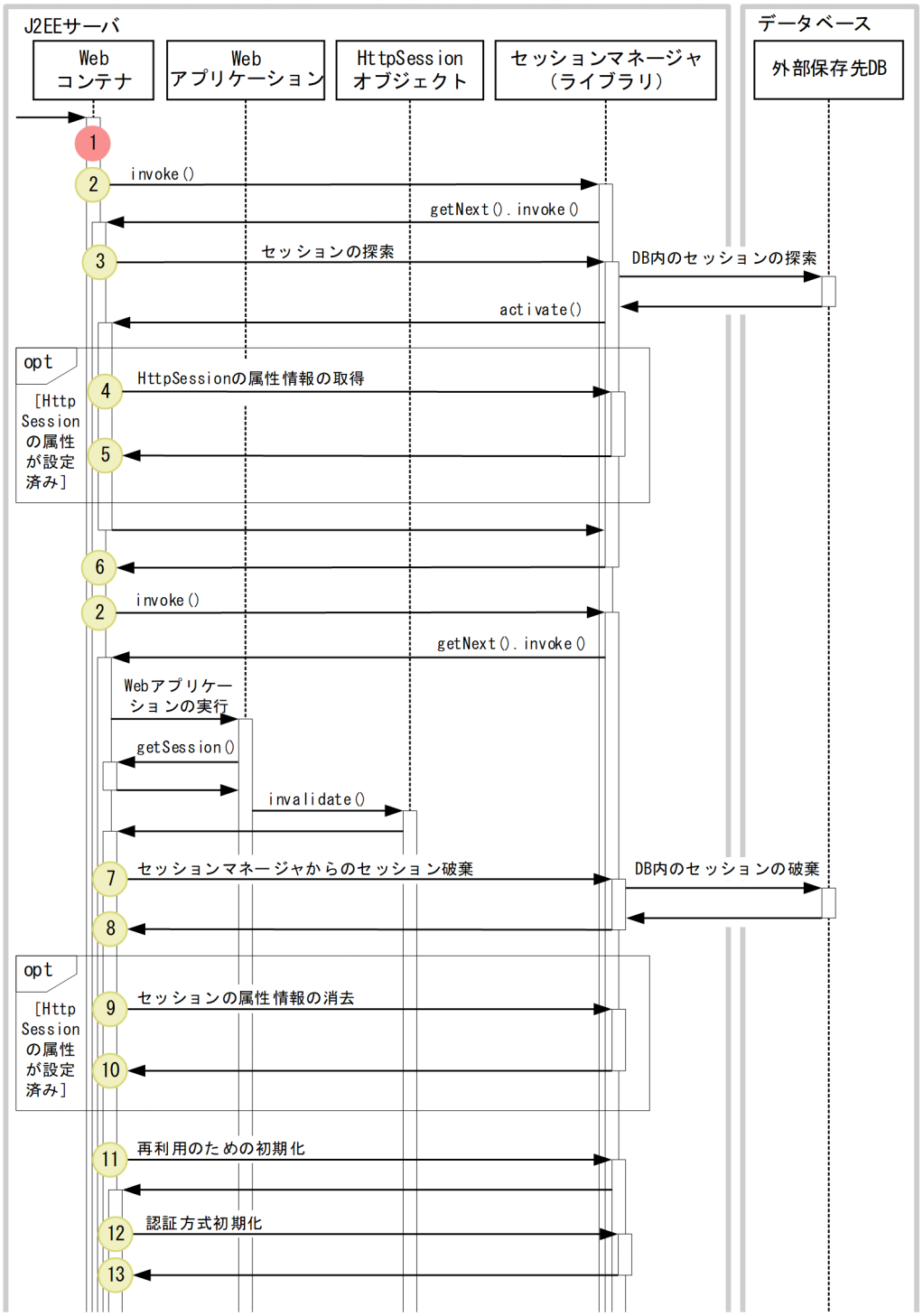
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8244	12	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.StandardSession のサブクラスの setAuthType メソッドの呼び出し直前	B
0x8245	14	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.StandardSession のサブクラスの setPrincipal メソッドの呼び出し直前	B
0x8247	9	セッションマネージャの指定機能による HTTP セッションの属性情報の消去直前	B
0x8248	4	セッションマネージャの指定機能による HTTP セッションの属性情報の取得直前	B
0x8252	2	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.Valve インタフェースの実装クラスの invoke メソッド呼び出し直前	B
0x8336	19	リクエスト処理完了直後	A
0x8341	6	セッションの探索直後（セッションマネージャの指定機能）	B
0x8342	8	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.ManagerBase のサブクラスの remove メソッドの呼び出し直後	B
0x8343	16	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.StandardSession のサブクラスの recycle メソッドの呼び出し直後	B
0x8344	13	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.StandardSession のサブクラスの setAuthType メソッドの呼び出し直後	B
0x8345	15	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.StandardSession のサブクラスの setPrincipal メソッドの呼び出し直後	B
0x8347	10	セッションマネージャの指定機能による HTTP セッションの属性情報の消去直後	B
0x8348	5	セッションマネージャの指定機能による HTTP セッションの属性情報の取得直後	B
0x8352	18	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.Valve インタフェースの実装クラスの invoke メソッド呼び出し直後	B

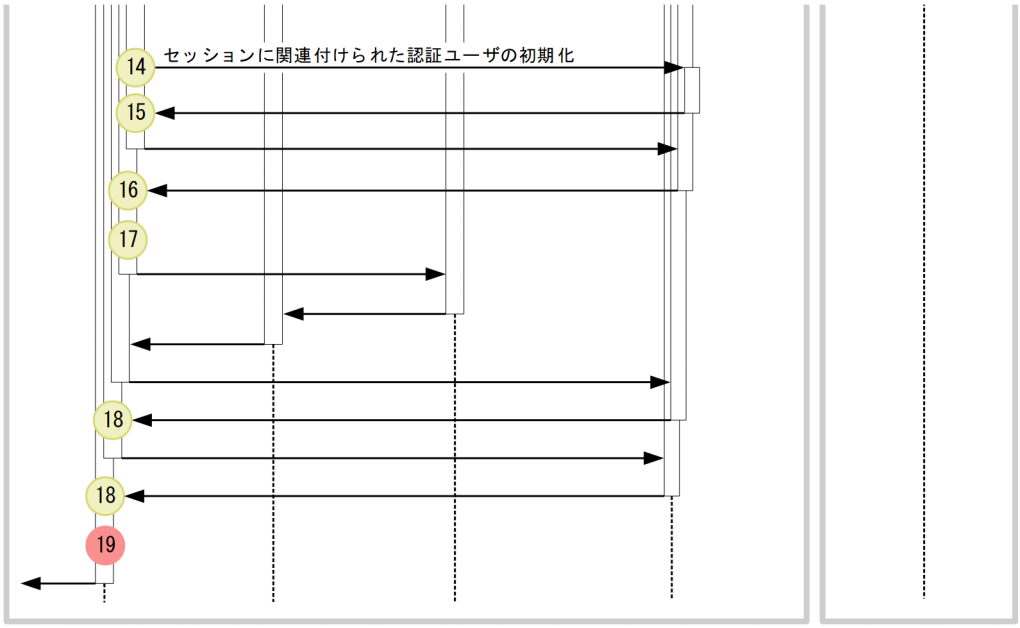
（凡例）A：標準　B：詳細

注※　図 8-20 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-20　HTTP セッションを無効化するリクエスト処理のトレース取得ポイント（セッションマネージャの指定機能）





（凡例）

：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRFトレース取得レベルは「詳細」です。

(2) 取得できるトレース情報

HTTP セッションを無効化するリクエスト処理で取得できるトレース情報を次の表に示します。

表 8-27 HTTP セッションを無効化するリクエスト処理で取得できるトレース情報（セッションマネージャの指定機能）

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名※3	オペレーション名※3※4	オプション※2※4
1	0x8236	A	HTTP メソッド	URI	プロパティで指定したリクエストヘッダ名に対する値 (ヘッダ名未指定の場合はなし)
2	0x8252	B	ライブラリにおける Valve インターフェースの実装クラス名	ライブラリの呼び出し元メソッド名	なし
3	0x8241	B	コンテキストルート名	なし	<セッション ID 文字数:セッション ID>
4	0x8248	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	<セッション ID 文字数:セッション ID:属性名>
5	0x8348	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名※3	オペレーション名※3※4	オプション※2※4
					<入り口時刻:例外名>
6	0x8341	B	コンテキストルート名	セッション有効期間	• 正常時 <入り口時刻:セッション ID 文字数:セッション ID> • 異常時 <入り口時刻:例外名>
7	0x8242	B	コンテキストルート名	なし	<セッション ID 文字数:セッション ID>
8	0x8342	B	コンテキストルート名	なし	• 正常時 <入り口時刻:セッション ID 文字数:セッション ID> • 異常時 <入り口時刻:例外名>
9	0x8247	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	<セッション ID 文字数:セッション ID:属性名>
10	0x8347	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>
11	0x8243	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	なし
12	0x8244	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	なし
13	0x8344	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>
14	0x8245	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	なし
15	0x8345	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名※3	オペレーション名※3※4	オプション※2※4
16	0x8343	B	ライブラリにおける StandardSession のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>
17	0x8209	B	コンテキストルート名	セッション生成時刻	<セッション ID 文字数:セッション ID>
18	0x8352	B	ライブラリにおける Valve インターフェースの実装クラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻> • 異常時 <入り口時刻:例外名>
19	0x8336	A	HTTP メソッド	URI	<入り口時刻><ステータスコード>

（凡例）A：標準　B：詳細

注※1　図 8-20 中の番号と対応しています。

注※2　「例外名」の情報が取得できるトレース取得ポイントでは，PRF トレースに出力した例外のスタックトレースを J2EE サーバの例外ログに出力します。

注※3　33 文字以上の場合，「先頭 16 文字+*+末尾 16 文字」で出力します。

注※4　「セッション有効期間」はセッションが存在する場合だけ出力します。「セッション ID 文字数」「セッション ID」はセッション ID が存在する場合だけ出力します。

8.8.4　セッションマネージャの起動・停止処理のトレース取得ポイントと取得できるトレース情報（セッションマネージャの指定機能のトレース）

セッションマネージャの起動・停止処理のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-28　セッションマネージャの起動・停止処理のトレース取得ポイントの詳細（セッションマネージャの指定機能）

イベント ID	トレース取得ポイント	レベル
0x8250	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.ManagerBase のサブクラスの startInternal メソッド呼び出し直前	B

イベント ID	トレース取得ポイント	レベル
0x8251	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.ManagerBase のサブクラスの stopInternal メソッド呼び出し直前	B
0x8350	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.ManagerBase のサブクラスの startInternal メソッド呼び出し直後	B
0x8351	セッションマネージャの指定機能で使用するライブラリが実装する org.apache.catalina.session.ManagerBase のサブクラスの stopInternal メソッド呼び出し直後	B

（凡例）A：標準　B：詳細

(2)　取得できるトレース情報

セッションマネージャの起動・停止処理で取得できるトレース情報を次の表に示します。

表 8-29　セッションマネージャの起動・停止処理で取得できるトレース情報（セッションマネージャの指定機能）

イベント ID	レベル	取得できる情報		
		インタフェース名※2	オペレーション名※2	オプション※1
0x8250	B	ライブラリにおける ManagerBase のサブクラス名	ライブラリの呼び出し元メソッド名	<ライフサイクルの状態>
0x8251	B	ライブラリにおける ManagerBase のサブクラス名	ライブラリの呼び出し元メソッド名	<ライフサイクルの状態>
0x8350	B	ライブラリにおける ManagerBase のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻:ライフサイクルの状態> • 異常時 <入り口時刻:例外名>
0x8351	B	ライブラリにおける ManagerBase のサブクラス名	ライブラリの呼び出し元メソッド名	• 正常時 <入り口時刻:ライフサイクルの状態> • 異常時 <入り口時刻:例外名>

（凡例）A：標準　B：詳細

注※1 「例外名」の情報が取得できるトレース取得ポイントでは，PRF トレースに出力した例外のスタックトレースを J2EE サーバの例外ログに出力します。

注※2　33 文字以上の場合，「先頭 16 文字+*+末尾 16 文字」で出力します。

8.9 EJB コンテナのトレース取得ポイント

ここでは，EJB コンテナのトレース取得ポイントと，取得できるトレース情報について，Session Bean，Entity Bean の場合，Message-driven Bean の場合，Timer Service を使用した場合，およびメソッドキャンセルが発生した場合に分けて説明します。

8.9.1 Session Bean，Entity Bean の場合

Session Bean，Entity Bean のトレース取得ポイントと，取得できるトレース情報について説明します。

(1) トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8–30 Session Bean，Entity Bean でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8401	1	リモートホームインタフェースの場合	EJB コンテナがリクエストを受信した直後	A
0x8402	4		EJB コンテナがレスポンスを送信する直前	A
0x8403	1	ローカルホームインタフェースの場合	EJB コンテナがリクエストを受信した直後	A
0x8404	4		EJB コンテナがレスポンスを送信する直前	A
0x8405	1	リモートコンポーネントインタフェースの場合	EJB コンテナがリクエストを受信した直後	A
0x8406	4		EJB コンテナがレスポンスを送信する直前	A
0x8407	1	ローカルコンポーネントインタフェースの場合	EJB コンテナがリクエストを受信した直後	A
0x8408	4		EJB コンテナがレスポンスを送信する直前	A
0x8409	2	EJB コンテナが EJB のビジネスメソッドをコールバックする直前		B
0x840A	3	EJB のビジネスメソッドのコールバックからリターンした直後		B
0x8453	2	EJB コンテナが EJB のホームメソッドをコールバックする直前		B
0x8454	3	EJB のホームメソッドのコールバックからリターンした直後		B
0x8470	1	リモートビジネスインタフェースの場合	EJB コンテナがリクエストを受信した直後	A

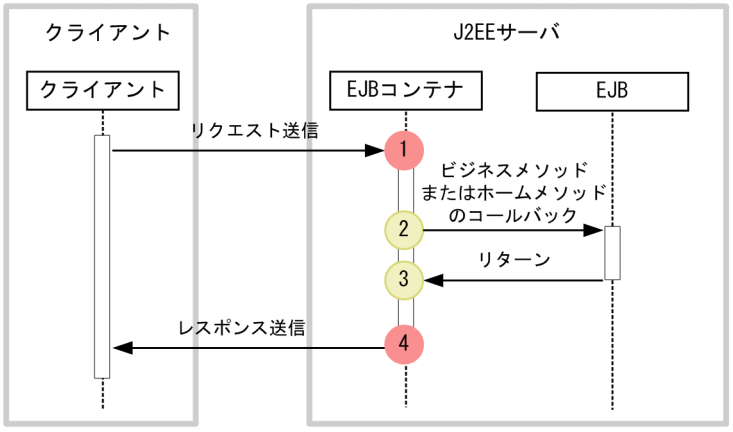
イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8471	4		EJB コンテナがレスポンスを送信する直前	A
0x8472	1	ローカルビジネスインタフェースの場合	EJB コンテナがリクエストを受信した直後	A
0x8473	4		EJB コンテナがレスポンスを送信する直前	A
0x8474	1	リモートビジネスインタフェースの場合	EJB コンテナがリモートビジネスインタフェース経由で Stateful Session Bean のインスタンス作成リクエストを受信した直後	A
0x8475	1		EJB コンテナがリモートビジネスインタフェース経由で Stateful Session Bean のインスタンス作成リクエストのレスポンスを送信する直前	A
0x8476	1	ローカルビジネスインタフェースの場合	EJB コンテナがローカルビジネスインタフェース経由で Stateful Session Bean のインスタンス作成リクエストを受信した直後	A
0x8477	1		EJB コンテナがローカルビジネスインタフェース経由で Stateful Session Bean のインスタンス作成リクエストのレスポンスを送信する直前	A

（凡例）A：標準　B：詳細

注※　図 8-21 中の番号と対応しています。

Session Bean，Entity Bean でのトレース取得ポイントを次の図に示します。

図 8-21　Session Bean，Entity Bean のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(2)　取得できるトレース情報

Session Bean，Entity Bean で取得できるトレース情報を次の表に示します。

表 8-31　Session Bean，Entity Bean で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8401	A	Bean 名	メソッド名，引数の数	−
	0x8403	A	Bean 名	メソッド名，引数の数	−
	0x8405	A	Bean 名	メソッド名，引数の数	−
	0x8407	A	Bean 名	メソッド名，引数の数	−
	0x8470	A	Bean 名	メソッド名，引数の数	−
	0x8472	A	Bean 名	メソッド名，引数の数	−
	0x8474	A	Bean 名	−	−
	0x8475	A	Bean 名	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
	0x8476	A	Bean 名	−	−
	0x8477	A	Bean 名	−	• 正常時 <入り口時刻>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					• 例外発生時 <入り口時刻><例外名>
2	0x8409	B	Bean 名	メソッド名，引数の数	−
	0x8453	B	Bean 名	メソッド名，引数の数	−
3	0x840A	B	Bean 名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
	0x8454	B	Bean 名	メソッド名，引数の数	
4	0x8402	A	Bean 名	メソッド名，引数の数	
	0x8404	A	Bean 名	メソッド名，引数の数	
	0x8406	A	Bean 名	メソッド名，引数の数	
	0x8408	A	Bean 名	メソッド名，引数の数	
	0x8471	A	Bean 名	メソッド名，引数の数	
	0x8473	A	Bean 名	メソッド名，引数の数	

（凡例）A：標準　B：詳細　−：該当なし

注※　図 8-21 中の番号と対応しています。

8.9.2　Message-driven Bean（EJB2.0）の場合

Message-driven Bean（EJB2.0）のトレース取得ポイントと，取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-32　Message-driven Bean（EJB2.0）でのトレース取得ポイントの詳細

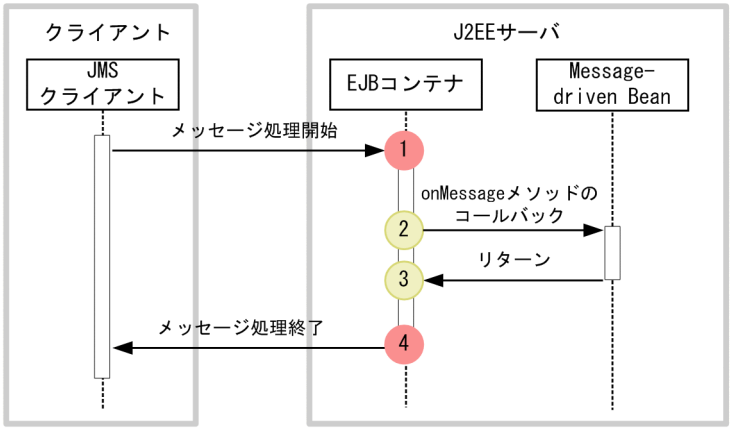
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8425	1	EJB コンテナでのメッセージ処理の始まり	A
0x8426	4	EJB コンテナでのメッセージ処理の終わり	A
0x8427	2	EJB コンテナが Message-driven Bean の onMessage メソッドをコールバックする直前	B
0x8428	3	Message-driven Bean の onMessage メソッドのコールバックからリターンした直後	B

（凡例）A：標準　B：詳細

注※　図 8-22 中の番号と対応しています。

Message-driven Bean（EJB2.0）でのトレース取得ポイントを次の図に示します。

図 8-22　Message-driven Bean（EJB2.0）のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

　　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(2)　取得できるトレース情報

Message-driven Bean（EJB2.0）で取得できるトレース情報を次の表に示します。

表 8-33　Message-driven Bean（EJB2.0）で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8425	A	Bean 名	−	−
2	0x8427	B	−	−	−
3	0x8428	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><<例外名>
4	0x8426	A	Bean 名	−	

（凡例）A：標準　B：詳細　−：該当なし

注※　図 8-22 中の番号と対応しています。

8.9.3　Message-driven Bean（EJB2.1 以降）の場合

Message-driven Bean（EJB2.1 以降）のトレース取得ポイントと，取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-34　Message-driven Bean（EJB2.1 以降）でのトレース取得ポイントの詳細

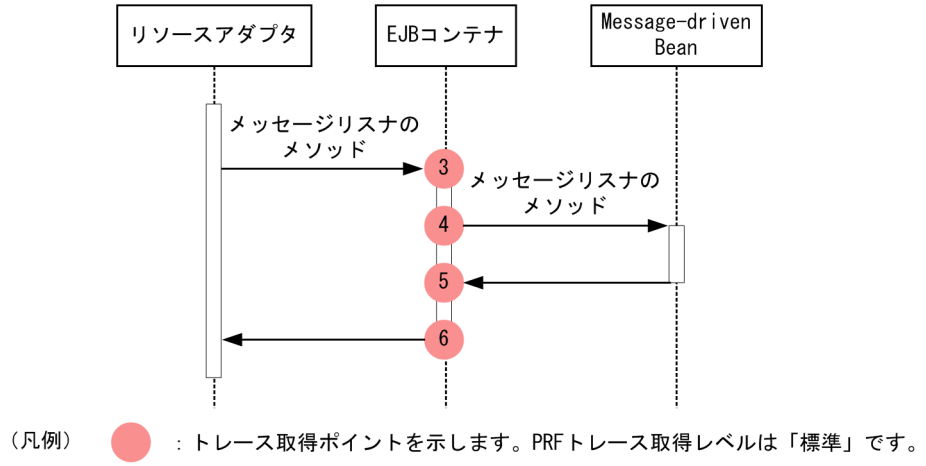
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x842D	1	Message-driven Bean の beforeDelivery()メソッドの呼び出し直後	A
0x842E	2	Message-driven Bean の beforeDelivery()メソッドのリターン直前	A
0x842F	3	リソースアダプタから Message-driven Bean のメッセージリスナのメソッドを呼び出した直後	A
0x8431	4	EJB コンテナが Message-driven Bean のメッセージリスナのメソッドをコールバックする直前	A
0x8432	5	Message-driven Bean のメッセージリスナのメソッドのコールバックからリターンした直後	A
0x8430	6	リソースアダプタから呼び出した Message-driven Bean のメッセージリスナのメソッドのリターン直前	A
0x8433	7	Message-driven Bean の afterDelivery()メソッドの呼び出し直後	A
0x8434	8	Message-driven Bean の afterDelivery()メソッドのリターン直前	A

（凡例）A：標準

注※　図 8-23 および図 8-24 中の番号と対応しています。

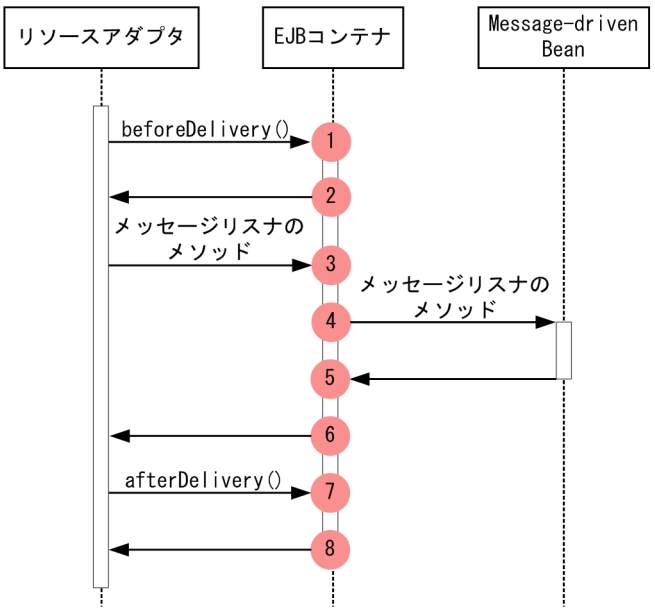
Message-driven Bean（EJB2.1 以降）でのトレース取得ポイントを次の図に示します。

図 8-23　Message-driven Bean（EJB2.1 以降）のトレース取得ポイント（optionA※の場合）



注※　Connector 1.5 仕様に記述されるメッセージ配送オプションを示します。

図 8-24　Message-driven Bean（EJB2.1 以降）のトレース取得ポイント（optionB※の場合）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

注※　Connector 1.5 仕様に記述されるメッセージ配送オプションを示します。

(2)　取得できるトレース情報

Message-driven Bean（EJB2.1 以降）で取得できるトレース情報を次の表に示します。

表 8-35　Message-driven Bean（EJB2.1 以降）で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x842D	A	Bean 名	−	−
2	0x842E	A	Bean 名	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x842F	A	Bean 名	メソッド名	−
4	0x8431	A	Bean 名	メソッド名	−
5	0x8432	A	Bean 名	メソッド名	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
6	0x8430	A	Bean 名	メソッド名	

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
7	0x8433	A	Bean 名	－	－
8	0x8434	A	Bean 名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　－：該当なし

注※　図 8-23 および図 8-24 中の番号と対応しています。

8.9.4　Timer Service の場合

Timer Service のトレース取得ポイントと，取得できるトレース情報について説明します。

(1)　createTimer の場合

(a)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-36　Timer Service でのトレース取得ポイントの詳細（createTimer の場合）

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8460	1	TimerService.createTimer(Date initialExpiration, long intervalDuration, Serializable info)処理開始	A
0x8462	1	TimerService.createTimer(Date expiration, Serializable info) 処理開始	A
0x8464	1	TimerService.createTimer(long initialDuration, long intervalDuration, Serializable info) 処理開始	A
0x8466	1	'TimerService.createTimer(long duration, Serializable info) 処理開始	A
0x8461	2	TimerService.createTimer(Date initialExpiration, long intervalDuration, Serializable info)処理終了	A
0x8463	2	TimerService.createTimer(Date expiration, Serializable info) 処理終了	A
0x8465	2	'TimerService.createTimer(long initialDuration, long intervalDuration, Serializable info) 処理終了	A

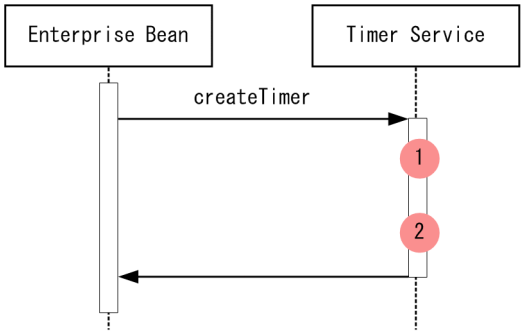
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8467	2	TimerService.createTimer(long duration, Serializable info) 処理終了	A

（凡例）A：標準

注※　図 8-25 中の番号と対応しています。

createTimer の場合の Timer Service でのトレース取得ポイントを次の図に示します。

図 8-25　Timer Service のトレース取得ポイント（createTimer の場合）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b)　取得できるトレース情報

createTimer の場合の Timer Service で取得できるトレース情報を次の表に示します。

表 8-37　Timer Service で取得できるトレース情報（createTimer の場合）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8460	A	コールバック対象 Bean のクラス名	－	引数情報
1	0x8462	A	コールバック対象 Bean のクラス名	－	引数情報
1	0x8464	A	コールバック対象 Bean のクラス名	－	引数情報
1	0x8466	A	コールバック対象 Bean のクラス名	－	引数情報
2	0x8461	A	コールバック対象 Bean のクラス名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻>< 例外名>
2	0x8463	A	コールバック対象 Bean のクラス名	－	
2	0x8465	A	コールバック対象 Bean のクラス名	－	
2	0x8467	A	コールバック対象 Bean のクラス名	－	

（凡例）A：標準　－：該当なし

注※　図 8-25 中の番号と対応しています。

(2) cancel の場合

(a) トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-38 Timer Service でのトレース取得ポイントの詳細（cancel の場合）

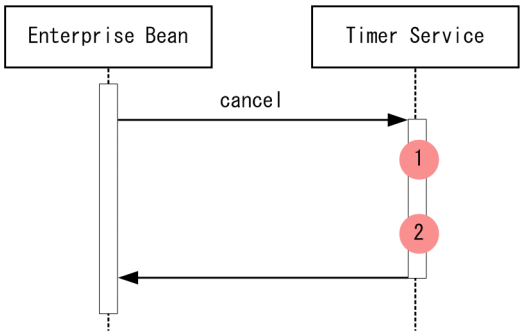
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8468	1	javax.ejb.Timer.cancel()処理開始	A
0x8469	2	javax.ejb.Timer.cancel()処理終了	A

（凡例）A：標準

注※ 図 8-26 中の番号と対応しています。

cancel の場合の Timer Service でのトレース取得ポイントを次の図に示します。

図 8-26 Timer Service のトレース取得ポイント（cancel の場合）



（凡例） ：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b) 取得できるトレース情報

cancel の場合の Timer Service で取得できるトレース情報を次の表に示します。

表 8-39 Timer Service で取得できるトレース情報（cancel の場合）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8468	A	コールバック対象 Bean のクラス名	－	－
2	0x8469	A	コールバック対象 Bean のクラス名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻>< 例外名>

（凡例）A：標準 －：該当なし

注※ 図 8-26 中の番号と対応しています。

(3)　コールバックの場合

(a)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-40　Timer Service でのトレース取得ポイントの詳細（コールバックの場合）

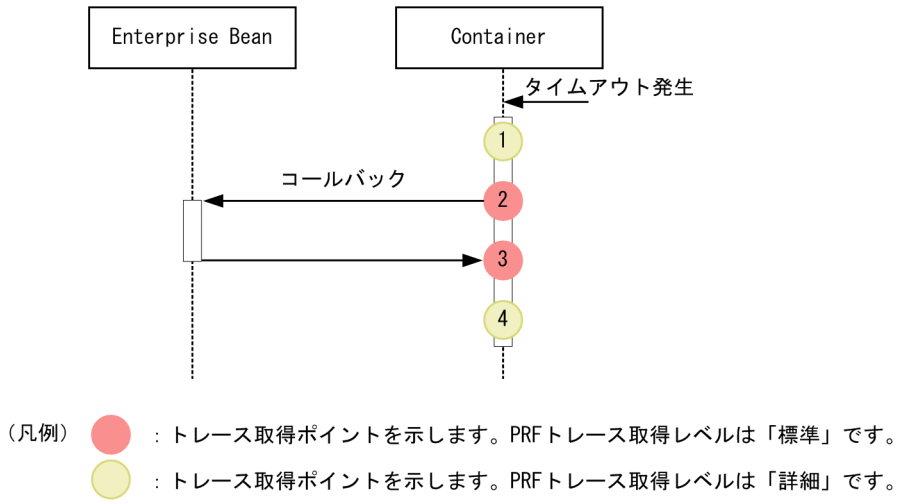
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x846C	1	スレッドがコールバック処理開始	B
0x846A	2	Enterprise Bean のタイムアウトコールバックメソッドをコールバック開始	A
0x846B	3	Enterprise Bean のタイムアウトコールバックメソッドをコールバック終了	A
0x846D	4	スレッドがコールバック処理終了	B

（凡例）A：標準　B：詳細

注※　図 8-27 中の番号と対応しています。

コールバックの場合の Timer Service でのトレース取得ポイントを次の図に示します。

図 8-27　Timer Service のトレース取得ポイント（コールバックの場合）



(b)　取得できるトレース情報

コールバックの場合の Timer Service で取得できるトレース情報を次の表に示します。

なお，ルートアプリケーション情報には，タイムアウトメソッドのコールバック時に取得する情報が出力されます。また，クライアントアプリケーション情報には「0」が出力されます。

表 8-41　Timer Service で取得できるトレース情報（コールバックの場合）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x846C	B	コールバック対象 Bean のクラス名	－	－
2	0x846A	A	コールバック対象 Bean のクラス名	－	－
3	0x846B	A	コールバック対象 Bean のクラス名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻> <例外名>
4	0x846D	B	コールバック対象 Bean のクラス名	－	

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-27 中の番号と対応しています。

(4)　createSingleActionTimer の場合

(a)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-42　Timer Service でのトレース取得ポイントの詳細（createSingleActionTimer の場合）

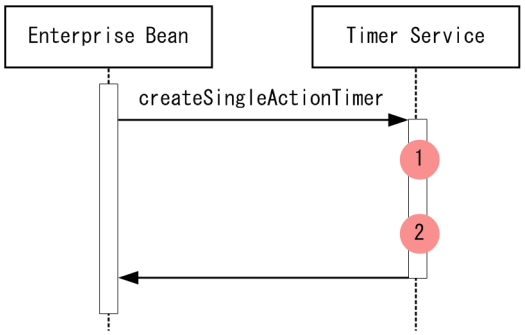
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84A0	1	TimerService.createSingleActionTimer(long duration, TimerConfig timerConfig)処理開始	A
0x84A1	2	TimerService.createSingleActionTimer(long duration, TimerConfig timerConfig)処理終了	A
0x84A2	1	TimerService.createSingleActionTimer(Date expiration, TimerConfig timerConfig)処理開始	A
0x84A3	2	TimerService.createSingleActionTimer(Date expiration, TimerConfig timerConfig)処理終了	A

（凡例）A：標準

注※　図 8-28 中の番号と対応しています。

createSingleActionTimer でのトレース取得ポイントを次の図に示します。

図 8-28　Timer Service のトレース取得ポイント（createSingleActionTimer の場合）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b)　取得できるトレース情報

createSingleActionTimer で取得できるトレース情報を次の表に示します。

表 8-43　Timer Service で取得できるトレース情報（createSingleActionTimer の場合）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84A0	A	コールバック対象 Bean のクラス名	－	引数情報
2	0x84A1	A	コールバック対象 Bean のクラス名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
1	0x84A2	A	コールバック対象 Bean のクラス名	－	引数情報
2	0x84A3	A	コールバック対象 Bean のクラス名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　－：該当なし

注※　図 8-28 中の番号と対応しています。

(5)　createIntervalTimer の場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-44　Timer Service のトレース取得ポイントの詳細（createIntervalTimer の場合）

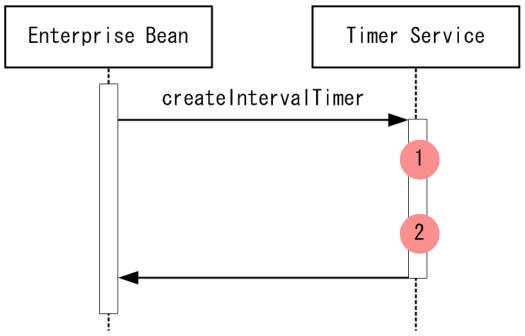
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84A4	1	TimerService.createIntervalTimer(long initialDuration, long intervalDuration, TimerConfig timerConfig)処理開始	A
0x84A5	2	TimerService.createIntervalTimer(long initialDuration, long intervalDuration, TimerConfig timerConfig)処理終了	A
0x84A6	1	TimerService.createIntervalTimer(Date initialExpiration, long intervalDuration, TimerConfig timerConfig)処理開始	A
0x84A7	2	TimerService.createIntervalTimer(Date initialExpiration, long intervalDuration, TimerConfig timerConfig)処理終了	A

（凡例）A：標準

注※　図 8-29 中の番号と対応しています。

createIntervalTimer のトレース取得ポイントを次の図に示します。

図 8-29　Timer Service のトレース取得ポイント（createIntervalTimer の場合）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b)　取得できるトレース情報

createIntervalTimer で取得できるトレース情報を次の表に示します。

表 8-45　Timer Service で取得できるトレース情報（createIntervalTimer の場合）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84A4	A	コールバック対象 Bean のクラス名	−	引数情報
2	0x84A5	A	コールバック対象 Bean のクラス名	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84A6	A	コールバック対象 Bean のクラス名	－	引数情報
2	0x84A7	A	コールバック対象 Bean のクラス名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　－：該当なし

注※　図 8-29 中の番号と対応しています。

(6)　createCalendarTimer の場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-46　Timer Service のトレース取得ポイントの詳細（createCalendarTimer の場合）

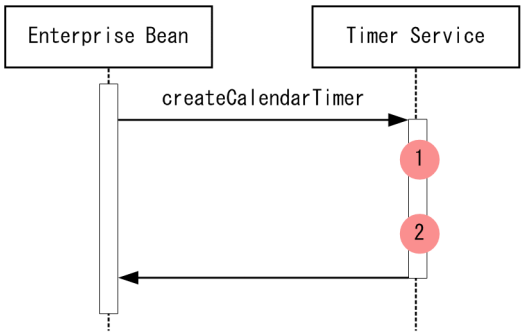
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84A8	1	TimerService.createCalendarTimer(ScheduleExpression schedule)処理開始	A
0x84A9	2	TimerService.createCalendarTimer(ScheduleExpression schedule)処理終了	A
0x84AA	1	TimerService.createCalendarTimer(ScheduleExpression schedule, TimerConfig config)処理開始	A
0x84AB	2	TimerService.createCalendarTimer(ScheduleExpression schedule, TimerConfig config)処理終了	A

（凡例）A：標準

注※　図 8-30 中の番号と対応しています。

createCalendarTimer のトレース取得ポイントを次の図に示します。

図 8-30　Timer Service のトレース取得ポイント（createCalendarTimer の場合）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b)　取得できるトレース情報

createCalendarTimer で取得できるトレース情報を次の表に示します。

表 8-47　Timer Service で取得できるトレース情報（createCalendarTimer の場合）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84A8	A	コールバック対象 Bean のクラス名	－	引数情報
2	0x84A9	A	コールバック対象 Bean のクラス名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
1	0x84AA	A	コールバック対象 Bean のクラス名	－	引数情報
2	0x84AB	A	コールバック対象 Bean のクラス名	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　－：該当なし

注※　図 8-30 中の番号と対応しています。

8.9.5　Session Bean の非同期呼び出しの場合

Session Bean の非同期呼び出し時のトレース取得ポイントと，取得できるトレース情報について説明します。

非同期呼び出しでは，同じスレッドで出力されるイベント ID の出力順序は保証されますが，異なるスレッドで出力されるイベント ID の出力順序は保証されません。このため，次のことが発生する場合があります。

- イベント ID の出力順序がここで説明する順序とは異なることがあります。
- ほかの機能のイベント ID の間に，非同期呼び出しのイベント ID が出力されることがあります。

(1) ローカルクライアントから Session Bean を非同期呼び出しする場合

(a) トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-48 ローカルクライアントから Session Bean を非同期呼び出しした場合のトレース取得ポイントの詳細

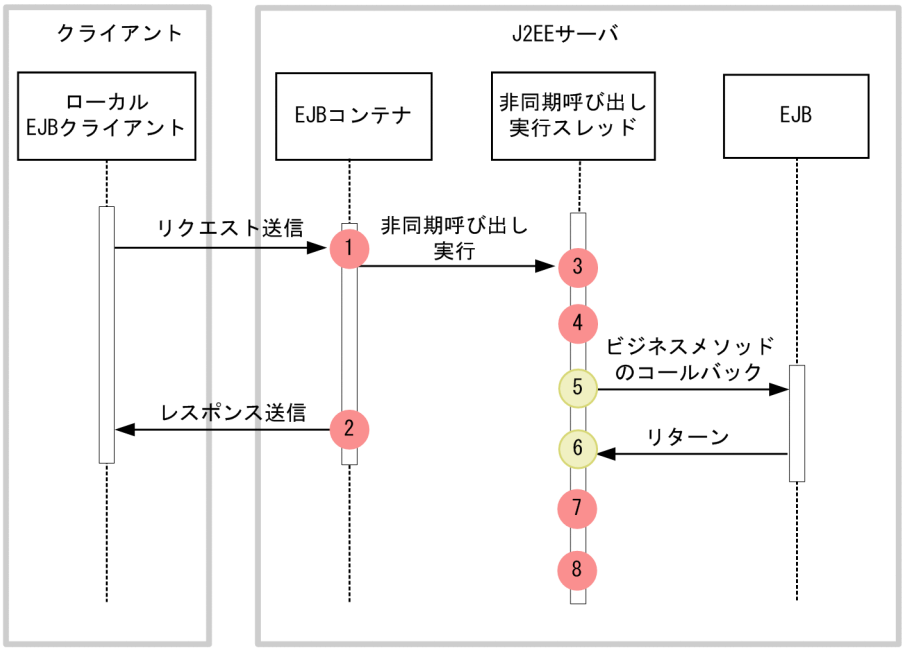
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8472	1	EJB コンテナがリクエストを受信した直後（ローカル呼び出しの場合）	A
0x8473	2	EJB コンテナがレスポンスを送信する直前（ローカル呼び出しの場合）	A
0x84C0	3	EJB コンテナが EJB の非同期のビジネスメソッドをコールバックする直前	A
0x84D6	4	EJB コンテナが EJB の非同期のビジネスメソッドを開始する直前（ローカル呼び出しの場合）	A
0x8409	5	EJB コンテナが EJB のビジネスメソッドをコールバックする直前	B
0x840A	6	EJB のビジネスメソッドのコールバックからリターンした直後	B
0x84D7	7	EJB の非同期のビジネスメソッドが終了した直後（ローカル呼び出しの場合）	A
0x84C1	8	EJB の非同期のビジネスメソッドのコールバックからリターンした直後	A

（凡例）A：標準　B：詳細

注※　図 8-31 中の番号と対応しています。

ローカルクライアントから Session Bean を非同期呼び出しした場合のトレース取得ポイントを次の図に示します。

図 8-31　ローカルクライアントから Session Bean を非同期呼び出しした場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(b)　取得できるトレース情報

ローカルクライアントから Session Bean を非同期呼び出しした場合に取得できるトレース情報を次の表に示します。

表 8-49　ローカルクライアントから Session Bean を非同期呼び出しした場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8472	A	Bean 名	メソッド名，引数の数	－
2	0x8473	A	Bean 名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x84C0	A	コールバック対象 Bean のクラス名	メソッド名，引数の数	非同期メソッド呼び出し元のルートアプリケーション情報
4	0x84D6	A	コールバック対象 Bean のクラス名	メソッド名，引数の数	－

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
5	0x8409	B	Bean 名	メソッド名，引数の数	－
6	0x840A	B	Bean 名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
7	0x84D7	A	コールバック対象 Bean のクラス名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
8	0x84C1	A	コールバック対象 Bean のクラス名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-31 中の番号と対応しています。

(2)　リモートクライアントから Session Bean を非同期呼び出しする場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-50　リモートクライアントから Session Bean を非同期呼び出しした場合のトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8470	1	EJB コンテナがリクエストを受信した直後（リモート呼び出しの場合）	A
0x8471	2	EJB コンテナがレスポンスを送信する直前（リモート呼び出しの場合）	A
0x84C0	3	EJB コンテナが EJB の非同期のビジネスメソッドをコールバックする直前	A
0x84D8	4	EJB コンテナが EJB の非同期のビジネスメソッドを開始する直前（リモート呼び出しの場合）	A
0x8409	5	EJB コンテナが EJB のビジネスメソッドをコールバックする直前	B
0x840A	6	EJB のビジネスメソッドのコールバックからリターンした直後	B

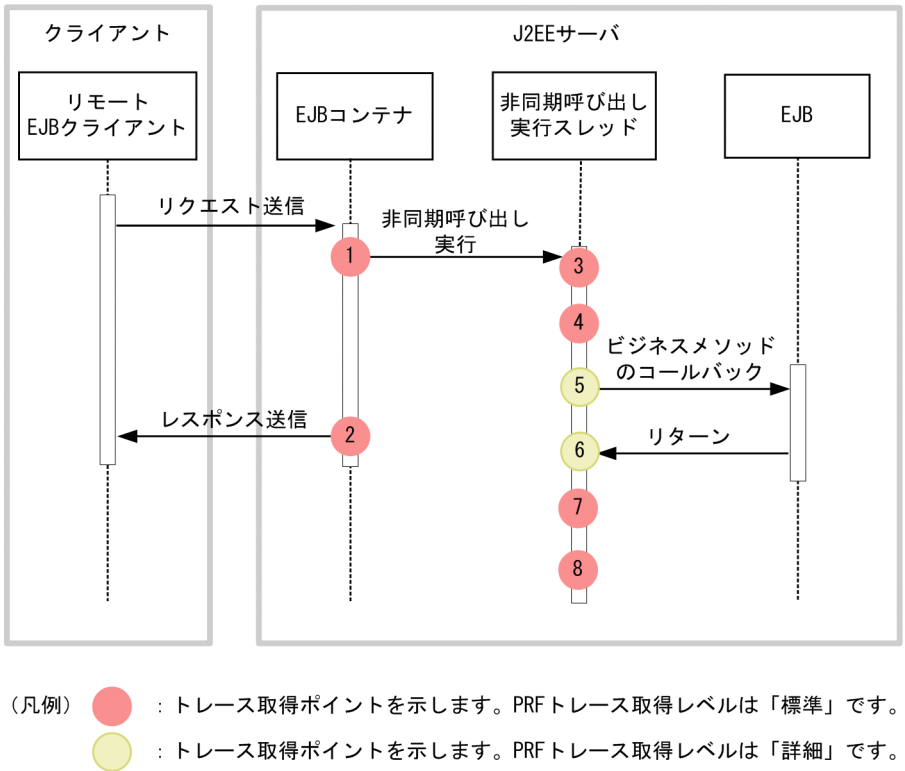
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84D9	7	EJB の非同期のビジネスメソッドが終了した直後（リモート呼び出しの場合）	A
0x84C1	8	EJB の非同期のビジネスメソッドのコールバックからリターンした直後	A

（凡例）A：標準　B：詳細

注※　図 8-32 中の番号と対応しています。

リモートクライアントから Session Bean を非同期呼び出しした場合のトレース取得ポイントを次の図に示します。

図 8-32　リモートクライアントから Session Bean を非同期呼び出しした場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

　　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(b)　取得できるトレース情報

リモートクライアントから Session Bean を非同期呼び出しした場合に取得できるトレース情報を次の表に示します。

表 8-51　リモートクライアントから Session Bean を非同期呼び出しした場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8470	A	Bean 名	メソッド名，引数の数	－
2	0x8471	A	Bean 名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x84C0	A	コールバック対象 Bean のクラス名	メソッド名，引数の数	非同期メソッド呼び出し元のルートアプリケーション情報
4	0x84D8	A	コールバック対象 Bean のクラス名	メソッド名，引数の数	－
5	0x8409	B	Bean 名	メソッド名，引数の数	－
6	0x840A	B	Bean 名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
7	0x84D9	A	コールバック対象 Bean のクラス名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
8	0x84C1	A	コールバック対象 Bean のクラス名	メソッド名，引数の数	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-32 中の番号と対応しています。

(3)　ローカルクライアントから get を呼び出す場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-52　ローカルクライアントから get を呼び出した場合のトレース取得ポイントの詳細

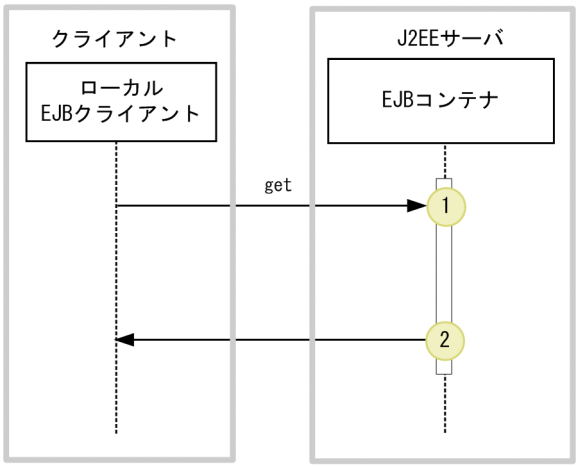
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84C2	1	Future.get()処理開始（ローカル呼び出しの場合）	B
0x84C3	2	Future.get()処理終了（ローカル呼び出しの場合）	B
0x84C4	1	Future.get(long timeout, TimeUnit unit)処理開始（ローカル呼び出しの場合）	B
0x84C5	2	Future.get(long timeout, TimeUnit unit)処理終了（ローカル呼び出しの場合）	B

（凡例）B：詳細

注※　図 8-33 中の番号と対応しています。

ローカルクライアントから get を呼び出した場合のトレース取得ポイントを次の図に示します。

図 8-33　ローカルクライアントから get を呼び出した場合のトレース取得ポイント



（凡例）：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(b)　取得できるトレース情報

ローカルクライアントから get を呼び出した場合に取得できるトレース情報を次の表に示します。

表 8-53　ローカルクライアントから get を呼び出した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84C2	B	−	−	−
2	0x84C3	B	−	−	• 正常時 <入り口時刻> • 例外発生時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻><例外名>
1	0x84C4	B	−	−	メソッド名，引数情報
2	0x84C5	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）B：詳細　−：該当なし

注※　図 8-33 中の番号と対応しています。

(4)　リモートクライアントから get を呼び出す場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-54　リモートクライアントから get を呼び出した場合のトレース取得ポイントの詳細

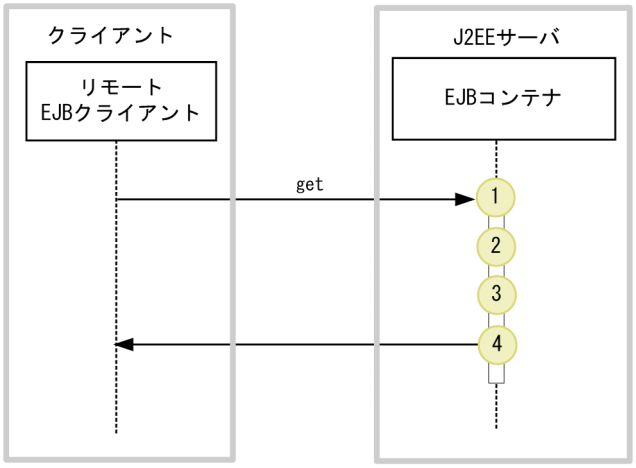
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84C6	1	Future.get()処理開始（リモート呼び出しの場合）	B
0x84C2	2	Future.get()処理開始（ローカル呼び出しの場合）	B
0x84C3	3	Future.get()処理終了（ローカル呼び出しの場合）	B
0x84C7	4	Future.get()処理終了（リモート呼び出しの場合）	B
0x84C8	1	Future.get(long timeout, TimeUnit unit)処理開始（リモート呼び出しの場合）	B
0x84C4	2	Future.get(long timeout, TimeUnit unit)処理開始（ローカル呼び出しの場合）	B
0x84C5	3	Future.get(long timeout, TimeUnit unit)処理終了（ローカル呼び出しの場合）	B
0x84C9	4	Future.get(long timeout, TimeUnit unit)処理終了（リモート呼び出しの場合）	B

（凡例）B：詳細

注※　図 8-34 中の番号と対応しています。

リモートクライアントから get を呼び出した場合のトレース取得ポイントを次の図に示します。

図 8-34　リモートクライアントから get を呼び出した場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(b)　取得できるトレース情報

リモートクライアントから get を呼び出した場合に取得できるトレース情報を次の表に示します。

表 8-55　リモートクライアントから get を呼び出した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84C6	B	−	−	−
2	0x84C2	B	−	−	−
3	0x84C3	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
4	0x84C7	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
1	0x84C8	B	−	−	メソッド名，引数情報
2	0x84C4	B	−	−	メソッド名，引数情報
3	0x84C5	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
4	0x84C9	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）B：詳細　−：該当なし

注※　図 8-34 中の番号と対応しています。

(5)　ローカルクライアントから isDone を呼び出す場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-56　ローカルクライアントから isDone を呼び出した場合のトレース取得ポイントの詳細

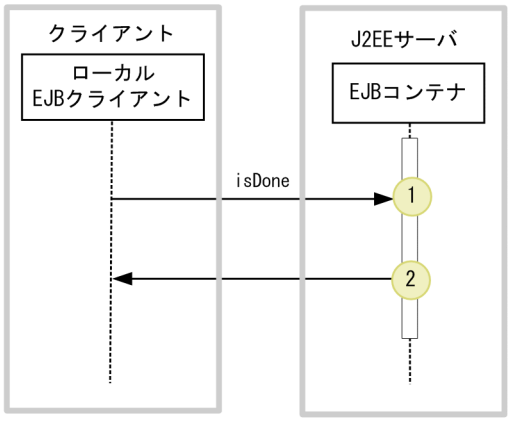
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84CA	1	Future.isDone()処理開始（ローカル呼び出しの場合）	B
0x84CB	2	Future.isDone()処理終了（ローカル呼び出しの場合）	B

（凡例）B：詳細

注※　図 8-35 中の番号と対応しています。

ローカルクライアントから isDone を呼び出した場合のトレース取得ポイントを次の図に示します。

図 8-35　ローカルクライアントから isDone を呼び出した場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(b)　取得できるトレース情報

ローカルクライアントから isDone を呼び出した場合に取得できるトレース情報を次の表に示します。

表 8-57　ローカルクライアントから isDone を呼び出した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84CA	B	−	−	−
2	0x84CB	B	−	−	isDone()の戻り値

（凡例）B：詳細　−：該当なし

注※　図 8-35 中の番号と対応しています。

注

このメソッドの実行では，例外は発生しません。したがって，異常な状態を示すログは生成されません。

(6)　リモートクライアントから isDone を呼び出す場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-58　リモートクライアントから isDone を呼び出した場合のトレース取得ポイントの詳細

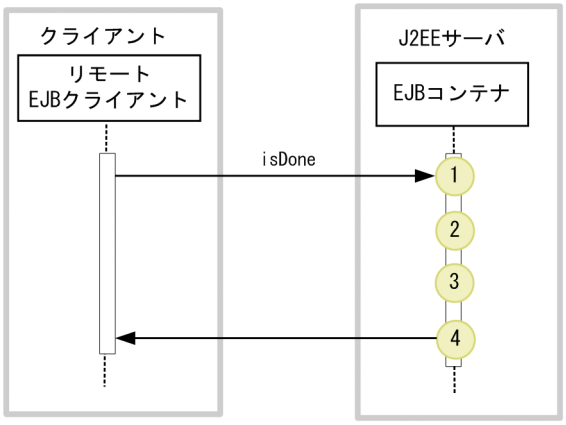
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84CE	1	Future.isDone()処理開始（リモート呼び出しの場合）	B
0x84CA	2	Future.isDone()処理開始（ローカル呼び出しの場合）	B
0x84CB	3	Future.isDone()処理終了（ローカル呼び出しの場合）	B
0x84CF	4	Future.isDone()処理終了（リモート呼び出しの場合）	B

（凡例）B：詳細

注※　図 8-36 中の番号と対応しています。

リモートクライアントから isDone を呼び出した場合のトレース取得ポイントを次の図に示します。

図 8-36　リモートクライアントから isDone を呼び出した場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(b)　取得できるトレース情報

リモートクライアントから isDone を呼び出した場合に取得できるトレース情報を次の表に示します。

表 8-59　リモートクライアントから isDone を呼び出した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84CE	B	−	−	−
2	0x84CA	B	−	−	−
3	0x84CB	B	−	−	isDone()の戻り値
4	0x84CF	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <例外名>

（凡例）B：詳細　−：該当なし

注※　図 8-36 中の番号と対応しています。

(7)　ローカルクライアントから isCancelled を呼び出す場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-60　ローカルクライアントから isCancelled を呼び出した場合のトレース取得ポイントの詳細

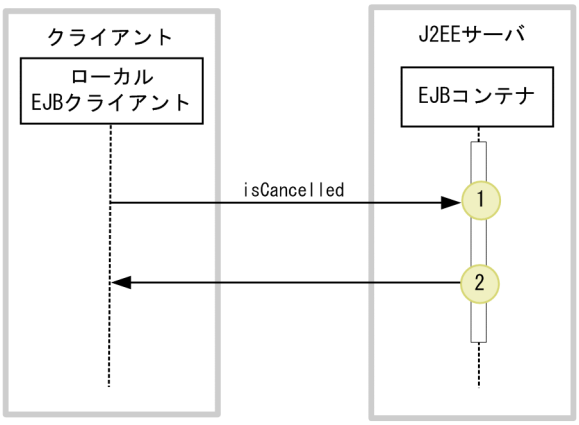
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84CC	1	Future.isCancelled()処理開始（ローカル呼び出しの場合）	B
0x84CD	2	Future.isCancelled()処理終了（ローカル呼び出しの場合）	B

（凡例）B：詳細

注※　図 8-37 中の番号と対応しています。

ローカルクライアントから isCancelled を呼び出した場合のトレース取得ポイントを次の図に示します。

図 8-37　ローカルクライアントから isCancelled を呼び出した場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(b)　取得できるトレース情報

ローカルクライアントから isCancelled を呼び出した場合に取得できるトレース情報を次の表に示します。

表 8-61　ローカルクライアントから isCancelled を呼び出した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84CC	B	−	−	−
2	0x84CD	B	−	−	isCancelled()の戻り値

（凡例）B：詳細　−：該当なし

注※　図 8-37 中の番号と対応しています。

注

　このメソッドの実行では，例外は発生しません。したがって，異常な状態を示すログは生成されません。

(8)　リモートクライアントから isCancelled を呼び出す場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-62　リモートクライアントから isCancelled を呼び出した場合のトレース取得ポイントの詳細

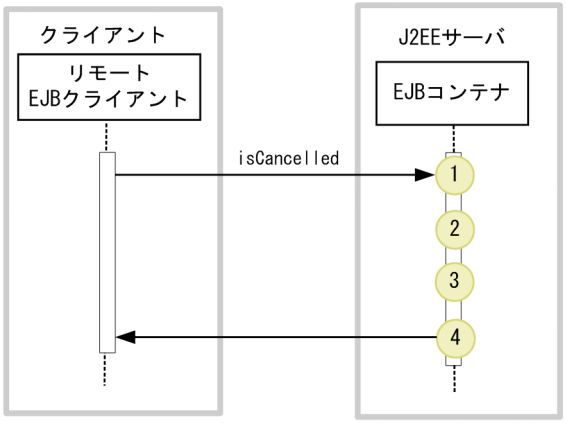
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84D0	1	Future.isCancelled()処理開始（リモート呼び出しの場合）	B
0x84CC	2	Future.isCancelled()処理開始（ローカル呼び出しの場合）	B
0x84CD	3	Future.isCancelled()処理終了（ローカル呼び出しの場合）	B
0x84D1	4	Future.isCancelled()処理終了（リモート呼び出しの場合）	B

（凡例）B：詳細

注※　図 8-38 中の番号と対応しています。

リモートクライアントから isCancelled を呼び出した場合のトレース取得ポイントを次の図に示します。

図 8-38　リモートクライアントから isCancelled を呼び出した場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRFトレース取得レベルは「詳細」です。

(b)　取得できるトレース情報

リモートクライアントから isCancelled を呼び出した場合に取得できるトレース情報を次の表に示します。

表 8-63　リモートクライアントから isCancelled を呼び出した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84D0	B	－	－	－

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
2	0x84CC	B	−	−	−
3	0x84CD	B	−	−	isCancelled()の戻り値
4	0x84D1	B	−	−	• 正常時 isCancelled()の戻り値 • 例外発生時 <例外名>

（凡例）B：詳細　−：該当なし

注※　図 8-38 中の番号と対応しています。

(9)　ローカルクライアントから cancel を呼び出す場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-64　ローカルクライアントから cancel を呼び出した場合のトレース取得ポイントの詳細

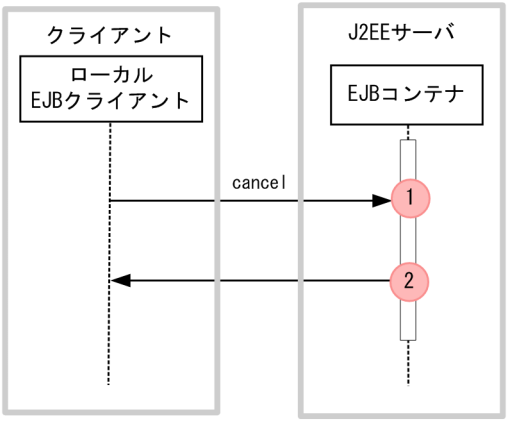
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84D2	1	Future.cancel()処理開始（ローカル呼び出しの場合）	A
0x84D3	2	Future.cancel()処理終了（ローカル呼び出しの場合）	A

（凡例）A：標準

注※　図 8-39 中の番号と対応しています。

ローカルクライアントから cancel を呼び出した場合のトレース取得ポイントを次の図に示します。

図 8-39　ローカルクライアントから cancel を呼び出した場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b)　取得できるトレース情報

ローカルクライアントから cancel を呼び出した場合に取得できるトレース情報を次の表に示します。

表 8-65　ローカルクライアントから cancel を呼び出した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84D2	A	－	－	メソッド名，引数情報
2	0x84D3	A	－	－	cancel()の戻り値

（凡例）A：標準　－：該当なし

注※　図 8-39 中の番号と対応しています。

注

　このメソッドの実行では，例外は発生しません。したがって，異常な状態を示すログは生成されません。

(10)　リモートクライアントから cancel を呼び出す場合

(a)　トレース取得ポイントと PRF トレース

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-66　リモートクライアントから cancel を呼び出した場合のトレース取得ポイントの詳細

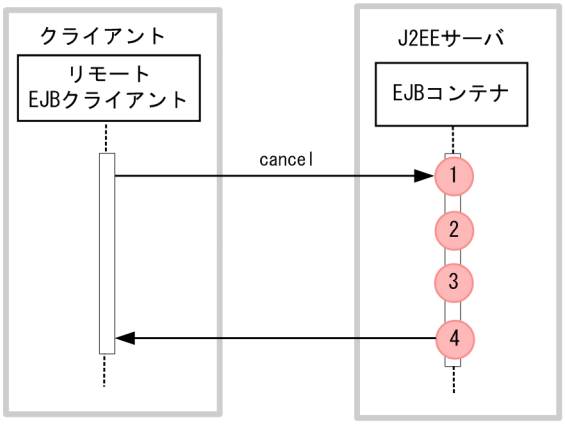
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x84D4	1	Future.cancel()処理開始（リモート呼び出しの場合）	A
0x84D2	2	Future.cancel()処理開始（ローカル呼び出しの場合）	A
0x84D3	3	Future.cancel()処理終了（ローカル呼び出しの場合）	A
0x84D5	4	Future.cancel()処理終了（リモート呼び出しの場合）	A

（凡例）A：標準

注※　図 8-40 中の番号と対応しています。

リモートクライアントから cancel を呼び出した場合のトレース取得ポイントを次の図に示します。

図 8-40　リモートクライアントから cancel を呼び出した場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b)　取得できるトレース情報

リモートクライアントから cancel を呼び出した場合に取得できるトレース情報を次の表に示します。

表 8-67　リモートクライアントから cancel を呼び出した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x84D4	A	－	－	メソッド名，引数情報
2	0x84D2	A	－	－	メソッド名，引数情報
3	0x84D3	A	－	－	cancel()の戻り値
4	0x84D5	A	－	－	• 正常時 cancel()の戻り値 • 例外発生時 <例外名>

（凡例）A：標準　－：該当なし

注※　図 8-40 中の番号と対応しています。

8.9.6　メソッドキャンセルが発生した場合

メソッドキャンセルが発生した場合のトレース取得ポイントと，取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-68　メソッドキャンセルが発生した場合のトレース取得ポイントの詳細

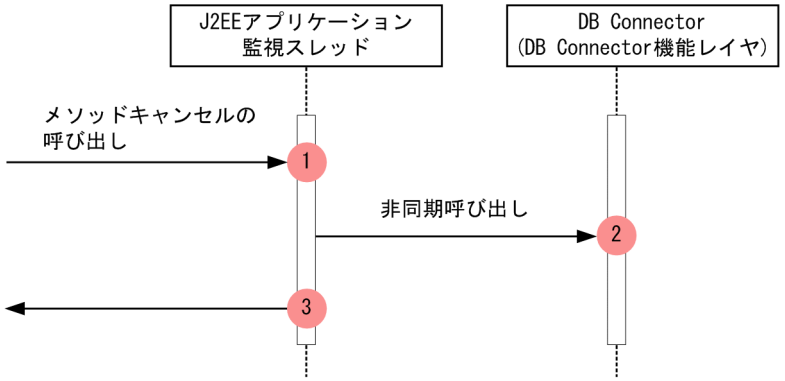
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8490	1	メソッドキャンセル処理開始	A
0x8C41	2	障害調査用 SQL 出力	A
0x8491	3	メソッドキャンセル処理完了	A

（凡例）A：標準

注※　図 8-41 中の番号と対応しています。

メソッドキャンセルが発生した場合のトレース取得ポイントを次の図に示します。

図 8-41　メソッドキャンセルが発生した場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

メソッドキャンセルが発生した場合に取得できるトレース情報を次の表に示します。

表 8-69　メソッドキャンセルが発生した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8490	A	メソッドキャンセル対象のアプリケーションのルートアプリケーション情報	−	−
2	0x8C41	A	トランザクションタイムアウト，J2EE アプリケーション強制停止，またはメソッドキャンセルされたコネクションのルートアプリケーション情報	−	SQL 文
3	0x8491	A	−	−	<入り口時刻>

（凡例）A：標準　−：該当なし

注※　図 8-41 中の番号と対応しています。

8.10　JNDI のトレース取得ポイント

ここでは，JNDI のトレース取得ポイントと，取得できるトレース情報について説明します。

8.10.1　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-70　JNDI でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント			レベル
0x8603	1	JNDI 内部の名前空間，または CORBA ネーミングサービスの検索時	javax.naming.Context.lookup	呼び出し直後	A
0x8604	2			リターン直前	A
0x8605	1		javax.naming.Context.list	呼び出し直後	B
0x8606	2			リターン直前	B
0x8607	1		javax.naming.Context.listBindings	呼び出し直後	B
0x8608	2			リターン直前	B
0x8609	1	javax.ejb.EJBLocalHome が保存されている名前空間の検索時	javax.naming.Context.lookup	呼び出し直後	A
0x860A	2			リターン直前	A
0x860B	1		javax.naming.Context.list	呼び出し直後	B
0x860C	2			リターン直前	B
0x860D	1		javax.naming.Context.listBindings	呼び出し直後	B
0x860E	2			リターン直前	B
0x860F	1	java: 名前空間検索時	javax.naming.Context.lookup	呼び出し直後	A
0x8610	2			リターン直前	A
0x8611	1		javax.naming.Context.list	呼び出し直後	B
0x8612	2			リターン直前	B
0x8613	1		javax.naming.Context.listBindings	呼び出し直後	B
0x8614	2			リターン直前	B
0x8615	1	ラウンドロビン検索機能利用時	javax.naming.Context.lookup	呼び出し直後	A
0x8616	2			リターン直前	A
0x8617	1	CORBA ネーミングサービス切り替え機能利用時	javax.naming.Context.lookup	呼び出し直後	A
0x8618	2			リターン直前	A

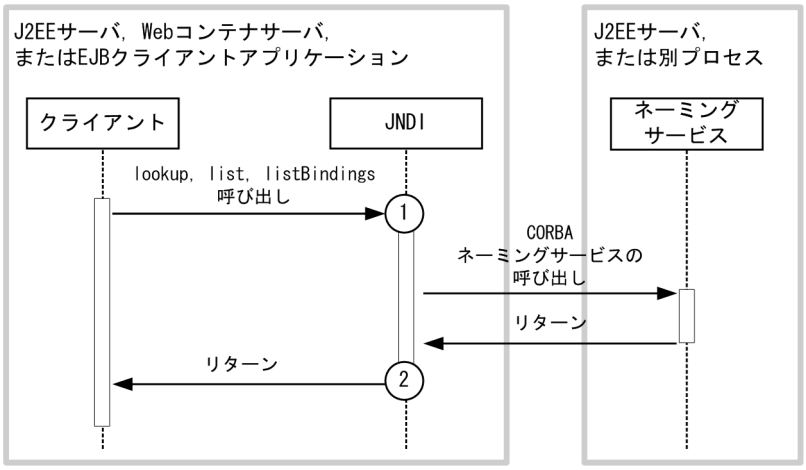
イベント ID	図中の番号※	トレース取得ポイント			レベル
0x8619	1		javax.naming.Context.list	呼び出し直後	B
0x861A	2			リターン直前	B
0x861B	1		javax.naming.Context.listBindings	呼び出し直後	B
0x861C	2			リターン直前	B

（凡例）A：標準　B：詳細

注※　図 8-42 中の番号と対応しています。

JNDI でのトレース取得ポイントを次の図に示します。

図 8-42　JNDI のトレース取得ポイント



（凡例）　○　：性能解析トレース取得ポイントを示します。
javax.naming.Context.lookupメソッドの場合，取得レベルは「標準」です。
javax.naming.Context.listメソッドまたは
javax.naming.Context.list.Bindingsメソッドの場合，取得レベルは「詳細」です。

8.10.2　取得できるトレース情報

JNDI で取得できるトレース情報を次の表に示します。

表 8-71　JNDI で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8603	A	－	指定された名前	－
	0x8609	A			
	0x860F	A			
	0x8615	A			
	0x8617	A			

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0x8605	B			
	0x860B	B			
	0x8611	B			
	0x8619	B			
	0x8607	B			
	0x860D	B			
	0x8613	B			
	0x861B	B			
2	0x8604	A	−		• 正常時 <入り口時刻> • 例外発生時 <入り口時刻:例外名>
	0x860A	A			
	0x8610	A			
	0x8616	A			
	0x8618	A			
	0x8606	B			
	0x860C	B			
	0x8612	B			
	0x861A	B			
	0x8608	B			
	0x860E	B			
	0x8614	B			
	0x861C	B			

（凡例）A：標準　B：詳細　−：該当なし

注※　図 8-42 中の番号と対応しています。

参考

- JNDI でのトレース情報では，次に示す場合，キー情報である「ルートアプリケーション情報」および「クライアントアプリケーション情報」には，0 が表示されます。
  - ・クライアントからの lookup メソッドを呼び出す場合
  - ・サーバが開始処理中または停止処理中の場合
- ビジネスインタフェース利用時には，0x8609 と 0x860A が 2 回ずつ出力されます。

8.11　JTA のトレース取得ポイント

ここでは，JTA のトレース取得ポイントと，取得できるトレース情報について説明します。

8.11.1　CMT および TransactionManager を使用する場合

CMT および javax.transaction.TransactionManager を使用する場合のトレース取得ポイントと，取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-72　CMT および TransactionManager を使用する場合のトレース取得ポイントの詳細

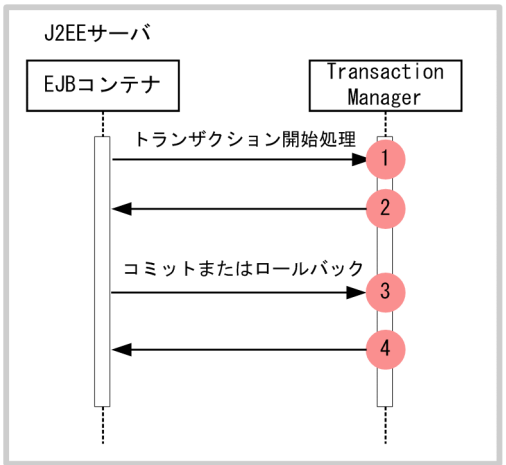
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8811	1	トランザクション開始処理直前	A
0x8812	2	トランザクション開始処理直後	A
0x8815	3	トランザクションコミット処理直前	A
0x8816	4	トランザクションコミット処理直後	A
0x8817	3	トランザクションロールバック処理直前	A
0x8818	4	トランザクションロールバック処理直後	A

（凡例）A：標準

注※　図 8-43 および図 8-44 中の番号と対応しています。

CMT を使用する場合のトレース取得ポイントを，次の図に示します。

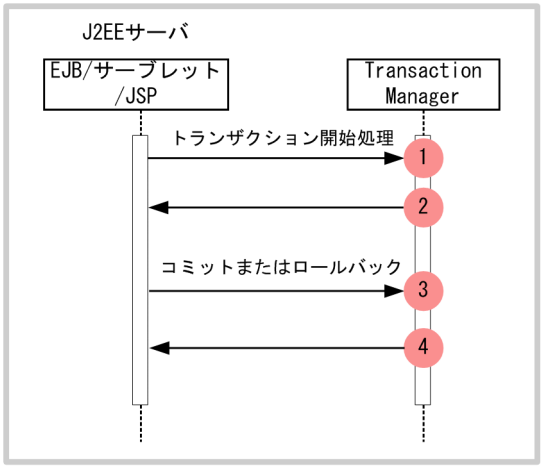
図 8-43　CMT を使用する場合のトレース取得ポイント



（凡例）●：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

TransactionManager を使用する場合のトレース取得ポイントを，次の図に示します。

図 8-44　TransactionManager を使用する場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

CMT および TransactionManager を使用する場合に取得できるトレース情報を次の表に示します。

表 8-73　CMT および TransactionManager を使用する場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8811	A	−	−	−
2	0x8812	A	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8815	A	−	−	−
	0x8817	A	−	−	−
4	0x8816	A	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
	0x8818	A	−	−	

（凡例）A：標準　−：該当なし

注※　図 8-43 および図 8-44 中の番号と対応しています。

8.11.2　UserTransaction を使用する場合

UserTransaction を使用する場合のトレース取得ポイントと，取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-74　UserTransaction を使用する場合のトレース取得ポイントの詳細

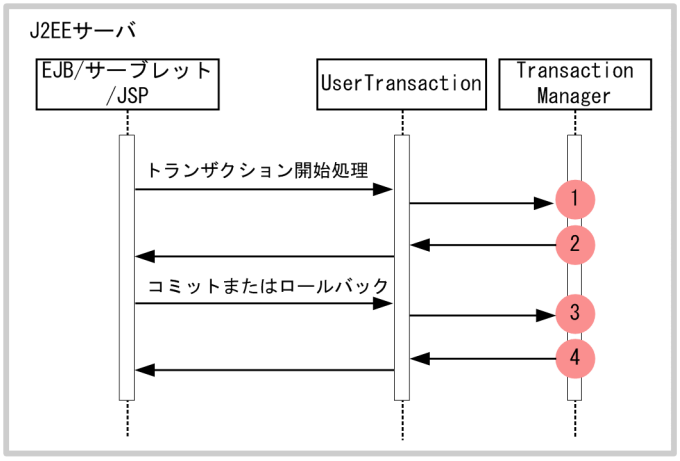
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8813	1	トランザクション開始処理直前	A
0x8814	2	トランザクション開始処理直後	A
0x8815	3	トランザクションコミット処理直前	A
0x8816	4	トランザクションコミット処理直後	A
0x8817	3	トランザクションロールバック処理直前	A
0x8818	4	トランザクションロールバック処理直後	A

（凡例）A：標準

注※　図 8-45 中の番号と対応しています。

UserTransaction を使用する場合のトレース取得ポイントを，次の図に示します。

図 8-45　UserTransaction を使用する場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

UserTransaction を使用する場合に取得できるトレース情報を次の表に示します。

表 8-75　UserTransaction を使用する場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8813	A	−	−	−
2	0x8814	A	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8815	A	−	−	−
	0x8817	A	−	−	−
4	0x8816	A	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
	0x8818	A	−	−	

（凡例）A：標準　−：該当なし

注※　図 8-45 中の番号と対応しています。

8.11.3　トランザクションタイムアウトの場合

トランザクションタイムアウトの場合のトレース取得ポイントと，取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-76　トランザクションタイムアウトでのトレース取得ポイントの詳細

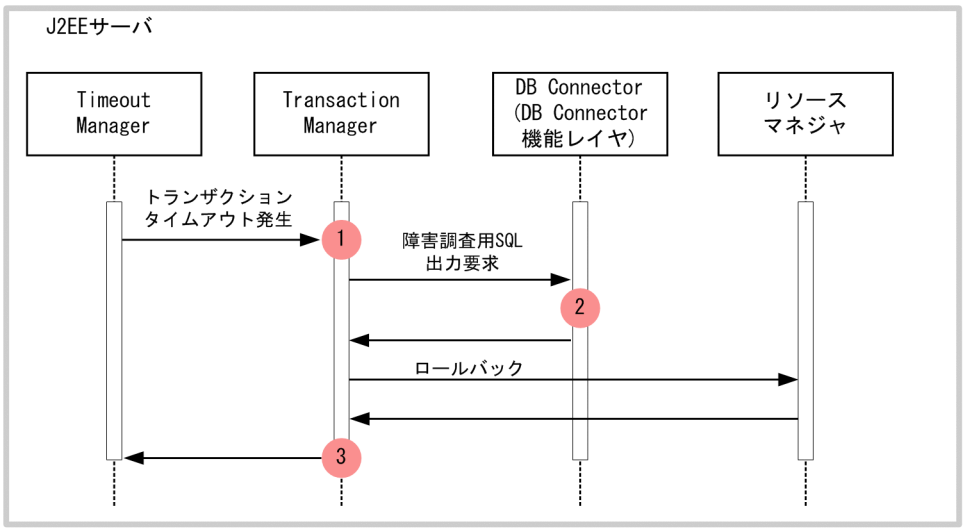
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8819	1	トランザクションタイムアウト処理直前	A
0x8C41	2	障害調査用 SQL 出力	A
0x8820	3	トランザクションタイムアウト処理直後	A

（凡例）A：標準

注※　図 8-46 中の番号と対応しています。

トランザクションタイムアウトのトレース取得ポイントを，次の図に示します。

図 8-46　トランザクションタイムアウトのトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

トランザクションタイムアウトで取得できるトレース情報を次の表に示します。

表 8-77　トランザクションタイムアウトで取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8819	A	タイムアウトしたトランザクションのルートアプリケーション情報	－	－
2	0x8C41	A	トランザクションタイムアウト，J2EE アプリケーション強制停止，またはメソッドキャンセルされたコネクションのルートアプリケーション情報	－	SQL 文
3	0x8820	A	－	－	<入り口時刻>

（凡例）A：標準　－：該当なし

注※　図 8-46 中の番号と対応しています。

8.11.4　スレッドの非同期並行処理を使用する場合

スレッドの非同期並行処理を使用する場合のトレース取得ポイントと，取得できるトレース情報について説明します。

(1) TimerManager のトレース取得ポイント

- トレース取得ポイントと PRF トレース取得レベル

  イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-78 TimerManager のトレース取得ポイントの詳細

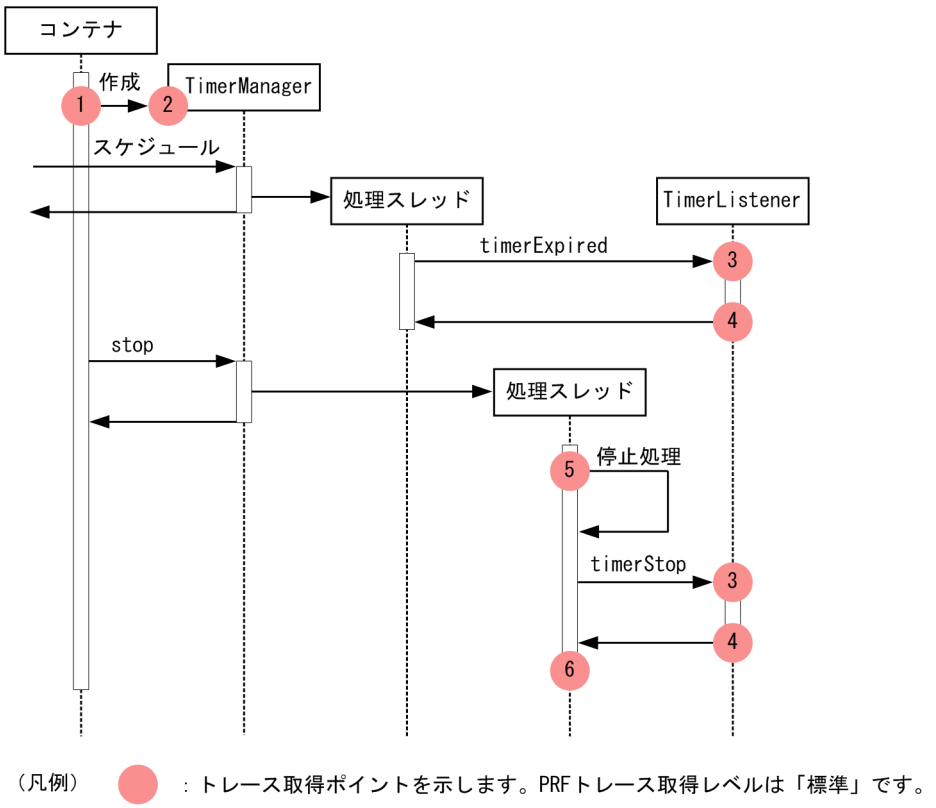
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8435	1	TimerManager の開始直前	A
0x8436	2	TimerManager の開始直後	A
0x8439	5	TimerManager の停止直前	A
0x843A	6	TimerManager の停止直後	A
0x843D	3	TimerManager のリスナ実行の直前	A
0x843E	4	TimerManager のリスナ実行の直後	A

（凡例）A：標準

注※ 図 8-47 中の番号と対応しています。

TimerManager のトレース取得ポイントを次の図に示します。

図 8-47 TimerManager のトレース取得ポイント



（凡例） ：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

- 取得できるトレース情報

  TimerManager で取得できるトレース情報を次の表に示します。

表 8-79　TimerManager で取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8435	A	−	−	−
2	0x8436	A	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x843D	A	メソッド名※2	スケジュールごとに一意の番号	−
4	0x843E	A	メソッド名※2	スケジュールごとに一意の番号	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
5	0x8439	A	−	−	−
6	0x843A	A	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　−：該当なし

注※1　図 8-47 中の番号と対応しています。

注※2　TimerManager.timerExpired，StopTimerListener.timerStop，または CancelTimerListener.timerCancel が出力されます。

(2)　WorkManager のトレース取得ポイント

- トレース取得ポイントと PRF トレース取得レベル

  イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-80　WorkManager のトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8437	1	WorkManager の開始直前	A
0x8438	2	WorkManager の開始直後	A
0x8440	4	WorkManager のリスナ，またはタスク処理の実行直後	A

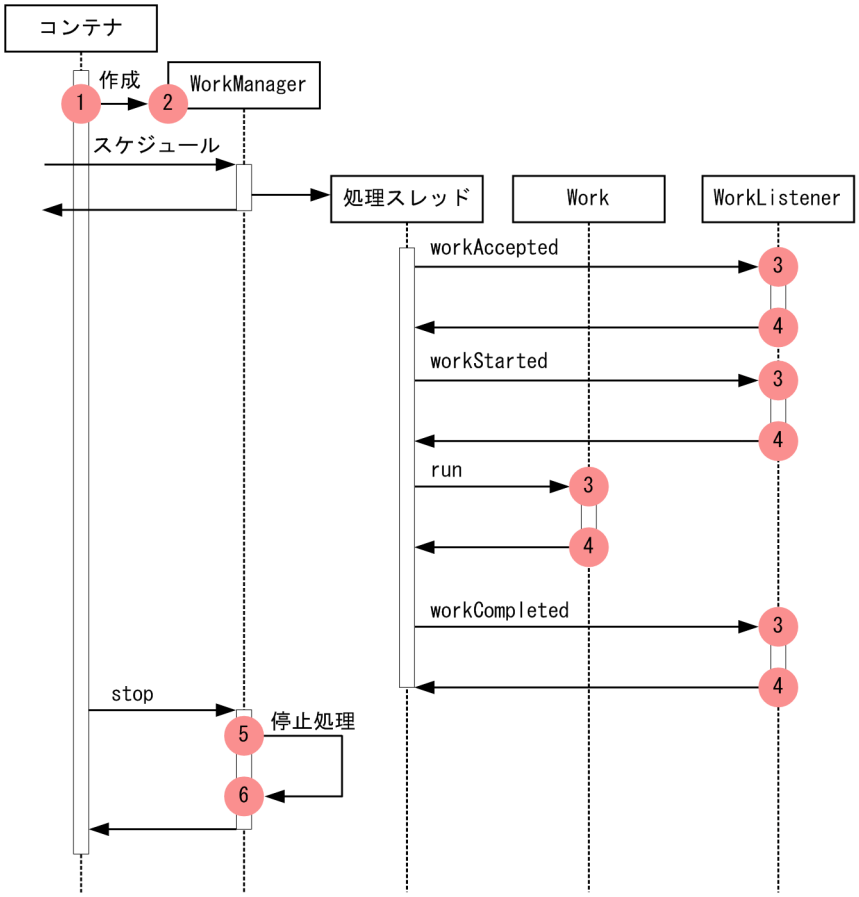
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x843B	5	WorkManager の停止直前	A
0x843C	6	WorkManager の停止直後	A
0x843F	3	WorkManager のリスナ，またはタスク処理の実行直前	A

（凡例）A：標準

注※　図 8-48 中の番号と対応しています。

WorkManager のトレース取得ポイントを次の図に示します。

図 8-48　WorkManager のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

- **取得できるトレース情報**

  WorkManager で取得できるトレース情報を次の表に示します。

表 8-81　WorkManager で取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8437	A	−	−	−
2	0x8438	A	−	−	• 正常時

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x843F	A	メソッド名※2	スケジュールごとに一意の番号	−
4	0x8440	A	メソッド名※2	スケジュールごとに一意の番号	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
5	0x843B	A	−	−	−
6	0x843C	A	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　−：該当なし

注※1　図 8-48 中の番号と対応しています。

注※2　Work.run，WorkListener.workAccepted，WorkListener.workRejected，WorkListener.workStarted，または WorkListener.workCompleted が出力されます。

8.12　DB Connector，JCA コンテナのトレース取得ポイント

ここでは，DB Connector および JCA コンテナのトレース取得ポイントと，取得できるトレース情報について説明します。

コネクション関連のトレース取得ポイントには，ローカルトランザクションを使用する場合だけ取得できるものがあります。ここでは，トランザクションサポートレベルに関係なく取得できるトレース取得ポイントと，ローカルトランザクションを使用する場合だけ取得できるトレース取得ポイントの二つに分けて説明します。なお，ここでは，トランザクションサポートレベルに関係なく取得できるトレース取得ポイントを，「**コネクション関連のトレース取得ポイント**」と呼びます。

8.12.1　コネクション関連のトレース取得ポイントと取得できるトレース情報

コネクション関連のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，イベント ID ごとに，次の四つの表に示します。

- 0x8B00，0x8B01，0x8B80～0x8B83，0x8C00～0x8C03，0x8C10～0xC13，0x8C20～0x8C29，0x8C2A～0x8C3F の場合（コネクション関連の処理実行時）

  参照先：表 8-82
- 0x8C80～0x8C93，0x8D42～0x8D4B の場合（java.sql.Statement インタフェースのメソッド実行時）

  参照先：表 8-83
- 0x8CC0～0x8CD9，0x8CDA～0x8CE3，0x8D4C，0x8D4D の場合（java.sql.PreparedStatement インタフェースのメソッド実行時）

  参照先：表 8-84
- 0x8D00～0x8D19，0x8D1A～0x8D25 の場合（java.sql.CallableStatement インタフェースのメソッド実行時）

  参照先：表 8-85

ポイント

イベント ID「0x8C41」については，「8.9　EJB コンテナのトレース取得ポイント」，および「8.11　JTA のトレース取得ポイント」を参照してください。

表 8-82　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（コネクション関連の処理実行時）1

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8B00	2	リソースアダプタからのコネクション取得要求呼び出し直後		B
0x8B01	5	リソースアダプタからのコネクション取得要求リターン直前		B
0x8B80	3	物理コネクション作成の呼び出し直前		B
0x8B81	4	物理コネクション作成のリターン直後		B
0x8B82	14	物理コネクション破棄の呼び出し直前		B
0x8B83	15	物理コネクション破棄のリターン直後		B
0x8C00	1	javax.sql.DataSource.getConnection()での，データベース接続の確立	処理開始	A
0x8C01	6		処理終了	A
0x8C02	1	javax.sql.DataSource.getConnection(String username, String password)での，データベース接続の確立	処理開始	A
0x8C03	6		処理終了	A
0x8C10	1	クラスタコネクションプール機能（互換機能）使用時の javax.sql.DataSource.getConnection()での，データベース接続の確立	処理開始	A
0x8C11	6		処理終了	A
0x8C12	1	クラスタコネクションプール機能（互換機能）使用時の javax.sql.DataSource.getConnection(String username, String password)での，データベース接続の確立	処理開始	A
0x8C13	6		処理終了	A
0x8C20	13	java.sql.Connection.close()での，Connection オブジェクトのデータベースと JDBC リソースの解除	処理開始	A
0x8C21	16		処理終了	A
0x8C22	11	java.sql.Connection.commit()	処理開始	B
0x8C23	12		処理終了	B
0x8C24	11	java.sql.Connection.rollback()	処理開始	B
0x8C25	12		処理終了	B
0x8C26	11	java.sql.Connection.rollback(Savepoint savepoint)	処理開始	B
0x8C27	12		処理終了	B
0x8C28	7	java.sql.Connection.createStatement()	処理開始	B
0x8C29	8		処理終了	B
0x8C2A	7	java.sql.Connection.createStatement(int resultSetType, int resultSetConcurrency)	処理開始	B
0x8C2B	8		処理終了	B

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8C2C	7	java.sql.Connection.createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)	処理開始	B
0x8C2D	8		処理終了	B
0x8C2E	7	java.sql.Connection.prepareCall(String sql)	処理開始	B
0x8C2F	8		処理終了	B
0x8C30	7	java.sql.Connection.prepareCall(String sql, int resultSetType, int resultSetConcurrency)	処理開始	B
0x8C31	8		処理終了	B
0x8C32	7	java.sql.Connection.prepareCall(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	処理開始	B
0x8C33	8		処理終了	B
0x8C34	7	java.sql.Connection.prepareStatement(String sql)	処理開始	B
0x8C35	8		処理終了	B
0x8C36	7	java.sql.Connection.prepareStatement(String sql, int autoGeneratedKeys)	処理開始	B
0x8C37	8		処理終了	B
0x8C38	7	java.sql.Connection.prepareStatement(String sql, int[] columnIndexes)	処理開始	B
0x8C39	8		処理終了	B
0x8C3A	7	java.sql.Connection.prepareStatement(String sql, int resultSetType, int resultSetConcurrency)	処理開始	B
0x8C3B	8		処理終了	B
0x8C3C	7	java.sql.Connection.prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	処理開始	B
0x8C3D	8		処理終了	B
0x8C3E	7	java.sql.Connection.prepareStatement(String sql, String[] columnNames)	処理開始	B
0x8C3F	8		処理終了	B

（凡例）A：標準　B：詳細

注　SQL Server 2005 を使用している場合，DB Connector のイベント ID は出力されません。

注※　図 8-49 中の番号と対応しています。

表 8-83　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（java.sql.Statement インタフェースのメソッド実行時）2

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8C80	9	execute(String sql)	処理開始	B
0x8C81	10		処理終了	B
0x8C82	9	execute(String sql, int autoGeneratedKeys)	処理開始	B

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8C83	10		処理終了	B
0x8C84	9	execute(String sql, int[] columnIndexes)	処理開始	B
0x8C85	10		処理終了	B
0x8C86	9	execute(String sql, String[] columnNames)	処理開始	B
0x8C87	10		処理終了	B
0x8C88	9	executeBatch()	処理開始	B
0x8C89	10		処理終了	B
0x8C8A	9	executeQuery(String sql)	処理開始	B
0x8C8B	10		処理終了	B
0x8C8C	9	executeUpdate(String sql)	処理開始	B
0x8C8D	10		処理終了	B
0x8C8E	9	executeUpdate(String sql, int autoGeneratedKeys)	処理開始	B
0x8C8F	10		処理終了	B
0x8C90	9	executeUpdate(String sql, int[] columnIndexes)	処理開始	B
0x8C91	10		処理終了	B
0x8C92	9	executeUpdate(String sql, String[] columnNames)	処理開始	B
0x8C93	10		処理終了	B
0x8D42	9	executeLargeBatch()	処理開始	B
0x8D43	10		処理終了	B
0x8D44	9	executeLargeUpdate(String sql)	処理開始	B
0x8D45	10		処理終了	B
0x8D46	9	executeLargeUpdate(String sql,int autoGeneratedKeys)	処理開始	B
0x8D47	10		処理終了	B
0x8D48	9	executeLargeUpdate(String sql, int[] columnIndexes)	処理開始	B
0x8D49	10		処理終了	B
0x8D4A	9	executeLargeUpdate(String sql, String[] columnNames)	処理開始	B
0x8D4B	10		処理終了	B

（凡例）B：詳細

注※　図 8-49 中の番号と対応しています。

表 8-84　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（java.sql.PreparedStatement インタフェースのメソッド実行時）3

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8CC0	9	execute()	処理開始	B
0x8CC1	10		処理終了	B
0x8CC2	9	execute(String sql)	処理開始	B
0x8CC3	10		処理終了	B
0x8CC4	9	execute(String sql, int autoGeneratedKeys)	処理開始	B
0x8CC5	10		処理終了	B
0x8CC6	9	execute(String sql, int[] columnIndexes)	処理開始	B
0x8CC7	10		処理終了	B
0x8CC8	9	execute(String sql, String[] columnNames)	処理開始	B
0x8CC9	10		処理終了	B
0x8CCA	9	executeBatch()	処理開始	B
0x8CCB	10		処理終了	B
0x8CCC	9	executeQuery()	処理開始	B
0x8CCD	10		処理終了	B
0x8CCE	9	executeQuery(String sql)	処理開始	B
0x8CCF	10		処理終了	B
0x8CD0	9	executeUpdate()	処理開始	B
0x8CD1	10		処理終了	B
0x8CD2	9	executeUpdate(String sql)	処理開始	B
0x8CD3	10		処理終了	B
0x8CD4	9	executeUpdate(String sql, int autoGeneratedKeys)	処理開始	B
0x8CD5	10		処理終了	B
0x8CD6	9	executeUpdate(String sql, int[] columnIndexes)	処理開始	B
0x8CD7	10		処理終了	B
0x8CD8	9	executeUpdate(String sql, String[] columnNames)	処理開始	B
0x8CD9	10		処理終了	B
0x8CDA	9	executeLargeBatch()	処理開始	B
0x8CDB	10		処理終了	B
0x8CDC	9	executeLargeUpdate(String sql)	処理開始	B

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8CDD	10		処理終了	B
0x8CDE	9	executeLargeUpdate(String sql,int autoGeneratedKeys)	処理開始	B
0x8CDF	10		処理終了	B
0x8CE0	9	executeLargeUpdate(String sql, int[] columnIndexes)	処理開始	B
0x8CE1	10		処理終了	B
0x8CE2	9	executeLargeUpdate(String sql, String[] columnNames)	処理開始	B
0x8CE3	10		処理終了	B
0x8D4C	9	executeLargeUpdate()	処理開始	B
0x8D4D	10		処理終了	B

（凡例）B：詳細

注※　図 8-49 中の番号と対応しています。

表 8-85　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（java.sql.CallableStatement インタフェースのメソッド実行時）4

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8D00	9	execute()	処理開始	B
0x8D01	10		処理終了	B
0x8D02	9	execute(String sql)	処理開始	B
0x8D03	10		処理終了	B
0x8D04	9	execute(String sql, int autoGeneratedKeys)	処理開始	B
0x8D05	10		処理終了	B
0x8D06	9	execute(String sql, int[] columnIndexes)	処理開始	B
0x8D07	10		処理終了	B
0x8D08	9	execute(String sql, String[] columnNames)	処理開始	B
0x8D09	10		処理終了	B
0x8D0A	9	executeBatch()	処理開始	B
0x8D0B	10		処理終了	B
0x8D0C	9	executeQuery()	処理開始	B
0x8D0D	10		処理終了	B
0x8D0E	9	executeQuery(String sql)	処理開始	B
0x8D0F	10		処理終了	B
0x8D10	9	executeUpdate()	処理開始	B

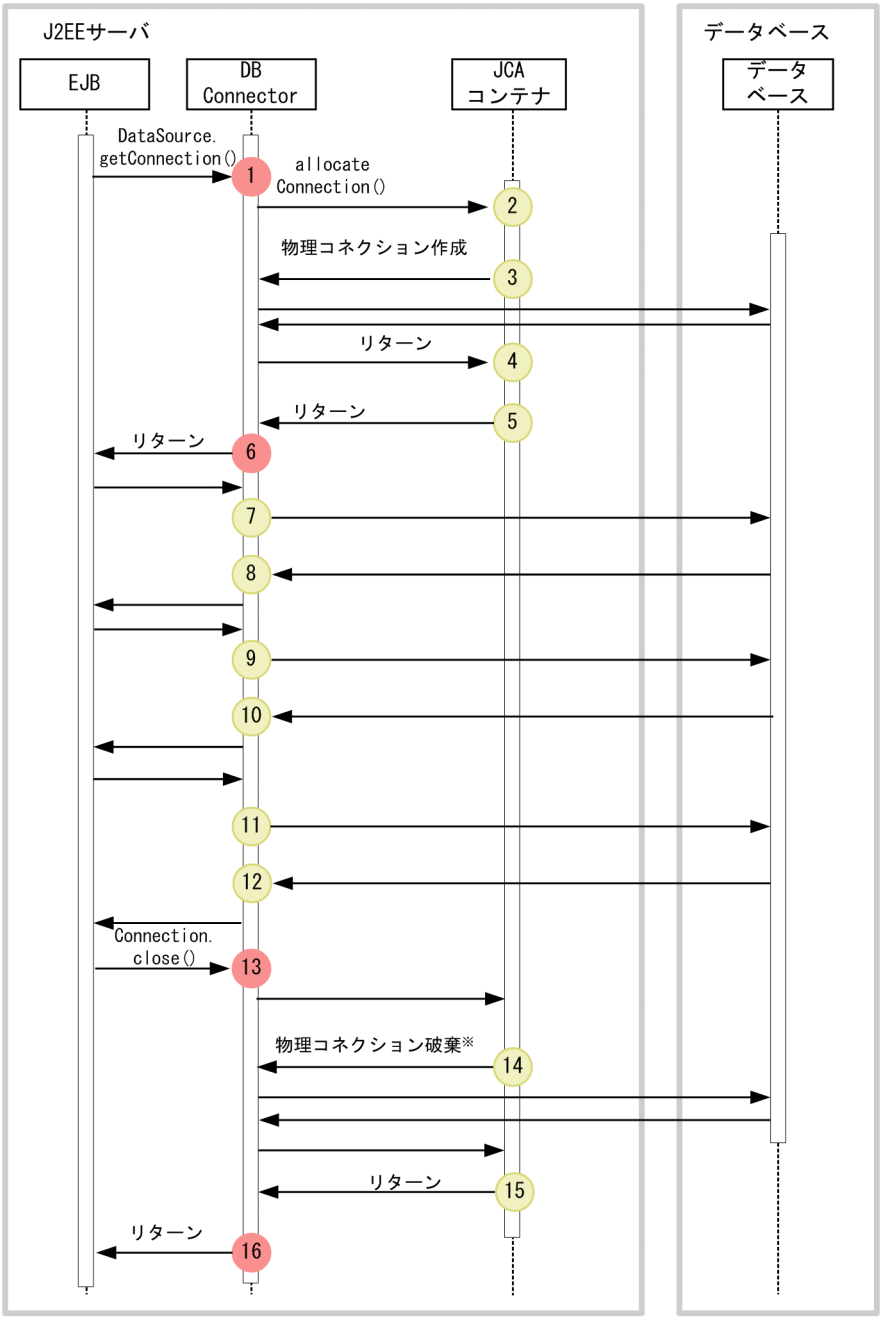
イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8D11	10		処理終了	B
0x8D12	9	executeUpdate(String sql)	処理開始	B
0x8D13	10		処理終了	B
0x8D14	9	executeUpdate(String sql, int autoGeneratedKeys)	処理開始	B
0x8D15	10		処理終了	B
0x8D16	9	executeUpdate(String sql, int[] columnIndexes)	処理開始	B
0x8D17	10		処理終了	B
0x8D18	9	executeUpdate(String sql, String[] columnNames)	処理開始	B
0x8D19	10		処理終了	B
0x8D1A	9	executeLargeBatch()	処理開始	B
0x8D1B	10		処理終了	B
0x8D1C	9	executeLargeUpdate(String sql)	処理開始	B
0x8D1D	10		処理終了	B
0x8D1E	9	executeLargeUpdate(String sql,int autoGeneratedKeys)	処理開始	B
0x8D1F	10		処理終了	B
0x8D20	9	executeLargeUpdate(String sql, int[] columnIndexes)	処理開始	B
0x8D21	10		処理終了	B
0x8D22	9	executeLargeUpdate(String sql, String[] columnNames)	処理開始	B
0x8D23	10		処理終了	B
0x8D24	9	executeLargeUpdate()	処理開始	B
0x8D25	10		処理終了	B

（凡例）B：詳細

注※　図 8-49 と対応しています。

DB Connector，JCA コンテナでのトレース取得ポイントを次の図に示します。

図 8-49　DB Connector，JCA コンテナのトレース取得ポイント（コネクション関連）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

注※　物理コネクションの破棄は，発生する場合と発生しない場合があります。

(2)　取得できるトレース情報

DB Connector，JCA コンテナで取得できるトレース情報について，次に示します。ここでは，図 8-49 の番号と対応づけて説明しています。

- 番号 1～6・11～16 に対応するトレース情報

  番号 1～6・11～16 に対応するイベント ID とトレース情報を，次の表に示します。

表 8-86　DB Connector，JCA コンテナで取得できるトレース情報（コネクション関連の処理実行時）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8C00	A	−	−	−
	0x8C02	A	−	−	−
	0x8C10	A	−	−	−
	0x8C12	A	−	−	−
2	0x8B00	B	−	−	−
3	0x8B80	B	−	−	−
4	0x8B81	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
5	0x8B01	B	−	−	
6	0x8C01	A	コネクション ID	−	
	0x8C03	A	コネクション ID	−	
	0x8C11	A	コネクション ID	−	
	0x8C13	A	コネクション ID	−	
11	0x8C22	B	−	−	−
	0x8C24	B	−	−	−
	0x8C26	B	−	−	−
12	0x8C23	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
	0x8C25	B	−	−	
	0x8C27	B	−	−	
13	0x8C20	A	コネクション ID	−	−
14	0x8B82	B	−	−	−
15	0x8B83	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
16	0x8C21	A	−	−	

（凡例）A：標準　B：詳細　−：該当なし

注※　図 8-49 中の番号と対応しています。

- 番号 7 に対応するトレース情報

  番号 7 に対応するイベント ID は次のとおりです。

0x8C28，0x8C2A，0x8C2C，0x8C2E，0x8C30，0x8C32，0x8C34，
0x8C36，0x8C38，0x8C3A，0x8C3C，0x8C3E

これらのイベント ID で取得できるトレース情報を示します。

- PRF トレース取得レベル
  すべて「詳細」です。
- インタフェース名，およびオペレーション名
  これらのイベント ID では出力されません。
- オプション
  メソッドの引数に sql を持つ場合，SQL 文が表示されます。

- 番号 8 に対応するトレース情報

  番号 8 に対応するイベント ID は次のとおりです。

  0x8C29，0x8C2B，0x8C2D，0x8C2F，0x8C31，0x8C33，0x8C35，
  0x8C37，0x8C39，0x8C3B，0x8C3D，0x8C3F

  これらのイベント ID で取得できるトレース情報を示します。

  - PRF トレース取得レベル
    すべて「詳細」です。
  - インタフェース名，オペレーション名
    これらのイベント ID では出力されません。
  - オプション
    これらのイベント ID では，正常に処理された場合は入り口時刻が表示されます。例外が発生した場合は入り口時刻および例外が表示されます。

- 番号 9 に対応するトレース情報

  番号 9 に対応するイベント ID は次のとおりです。

  0x8C80，0x8C82，0x8C84，0x8C86，0x8C88，0x8C8A，0x8C8C，0x8C8E，0x8C90，
  0x8C92，0x8CC0，0x8CC2，0x8CC4，0x8CC6，0x8CC8，0x8CCA，0x8CCC，0x8CCE，
  0x8CD0，0x8CD2，0x8CD4，0x8CD6，0x8CD8，0x8CDA，0x8CDC，0x8CDE，0x8CE0，
  0x8CE2，0x8D00，0x8D02，0x8D04，0x8D06，0x8D08，0x8D0A，0x8D0C，0x8D0E，
  0x8D10，0x8D12，0x8D14，0x8D16，0x8D18，0x8D1A，0x8D1C，0x8D1E，0x8D20，
  0x8D22，0x8D24，0x8D42，0x8D44，0x8D46，0x8D48，0x8D4A，0x8D4C

  これらのイベント ID で取得できるトレース情報を示します。

  - PRF トレース取得レベル
    すべて「詳細」です。
  - インタフェース名，およびオペレーション名
    これらのイベント ID では出力されません。
  - オプション
    メソッドの引数に sql を持つ場合，SQL 文が表示されます。

- 番号 10 に対応するトレース情報

  番号 10 に対応するイベント ID は次のとおりです。

  0x8C81，0x8C83，0x8C85，0x8C87，0x8C89，0x8C8B，0x8C8D，0x8C8F，0x8C91，0x8C93，0x8CC1，0x8CC3，0x8CC5，0x8CC7，0x8CC9，0x8CCB，0x8CCD，0x8CCF，0x8CD1，0x8CD3，0x8CD5，0x8CD7，0x8CD9，0x8CDB，0x8CDD，0x8CDF，0x8CE1，0x8CE3，0x8D01，0x8D03，0x8D05，0x8D07，0x8D09，0x8D0B，0x8D0D，0x8D0F，0x8D11，0x8D13，0x8D15，0x8D17，0x8D19，0x8D1B，0x8D1D，0x8D1F，0x8D21，0x8D23，0x8D25，0x8D43，0x8D45，0x8D47，0x8D49，0x8D4B，0x8D4D

  これらのイベント ID で取得できるトレース情報を示します。

  - PRF トレース取得レベル

    すべて「詳細」です。
  - インタフェース名，オペレーション名

    これらのイベント ID では出力されません。
  - オプション

    これらのイベント ID では，正常に処理された場合は入り口時刻が表示されます。例外が発生した場合は入り口時刻および例外が表示されます。

8.12.2 ローカルトランザクションを使用する場合のトレース取得ポイントと取得できるトレース情報

ローカルトランザクションを使用する場合のトレース取得ポイントと取得できるトレース情報トレース取得ポイントと取得できるトレース情報について説明します。

(1) トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-87 DB Connector，JCA コンテナでのトレース取得ポイントの詳細（ローカルトランザクションの処理実行時）

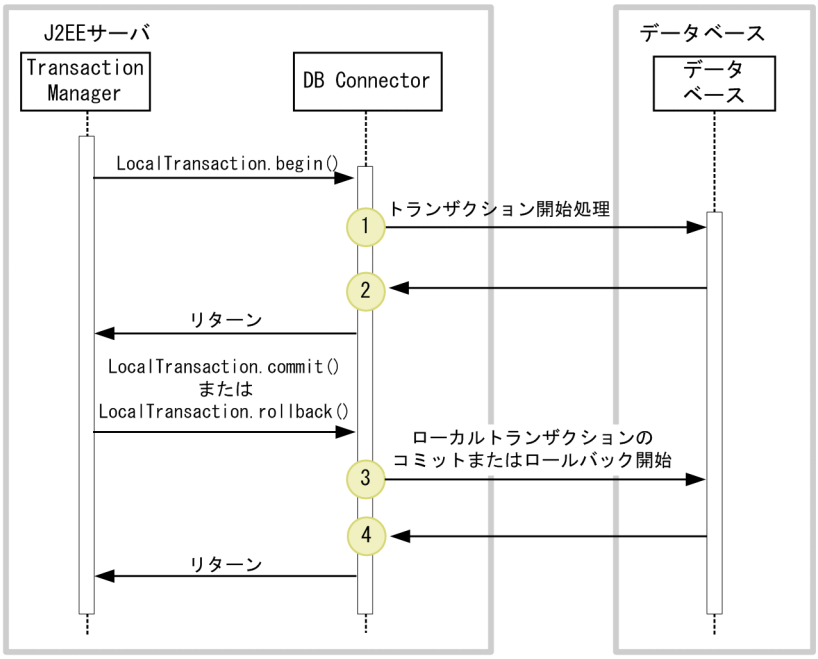
イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8C60	1	ローカルトランザクションの開始	処理開始	B
0x8C61	2		処理終了	B
0x8C62	3	ローカルトランザクションのコミット	処理開始	B
0x8C63	4		処理終了	B
0x8C64	3	ローカルトランザクションのロールバック	処理開始	B
0x8C65	4		処理終了	B

（凡例）B：詳細

注※　図 8-50 と対応しています。

トレース取得ポイントを次の図に示します。

図 8-50　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（ローカルトランザクション関連）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(2)　取得できるトレース情報

取得できるトレース情報を次の表に示します。

表 8-88　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（ローカルトランザクションの処理実行時）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8C60	B	−	−	−
2	0x8C61	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><><例外名>
3	0x8C62	B	−	−	−
	0x8C64	B	−	−	−
4	0x8C63	B	−	−	• 正常時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0x8C65	B	−	−	<入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）B：詳細　−：該当なし

注※　図 8-50 中の番号と対応しています。

8.12.3　コネクションアソシエーションを使用する場合のトレース取得ポイントと取得できるトレース情報

コネクションアソシエーションを使用する場合のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-89　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（コネクションアソシエーションの処理実行時）

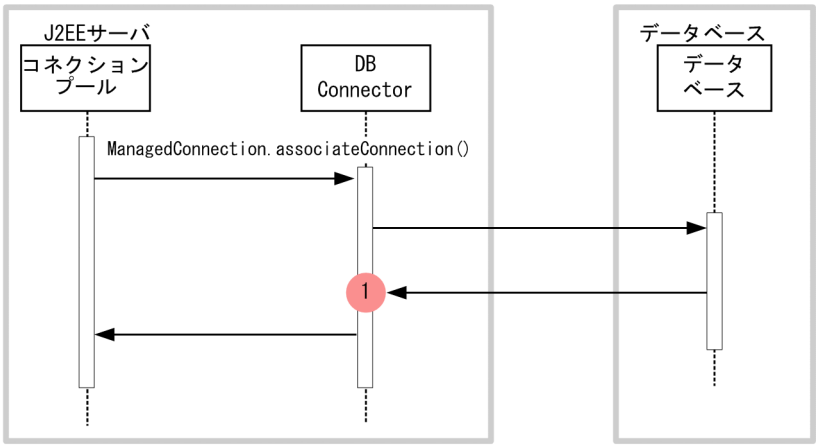
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8C40	1	コネクションアソシエーション機能によって，論理コネクションに対する物理コネクションが差し替えられた時	A

（凡例）A：標準

注※　図 8-51 の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-51　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（コネクションアソシエーション関連）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

取得できるトレース情報を次の表に示します。

表 8-90　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（コネクションアソシエーション処理実行時）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8C40	A	コネクション ID（コネクションアソシエーション機能によって差し替えられた先の物理コネクションのコネクション ID）	－	－

（凡例）A：標準　－：該当なし

注※　図 8-51 中の番号と対応しています。

8.12.4　コネクション自動クローズ機能を使用する場合のトレース取得ポイントと取得できるトレース情報

コネクション自動クローズ時のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-91　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（コネクション自動クローズ時）

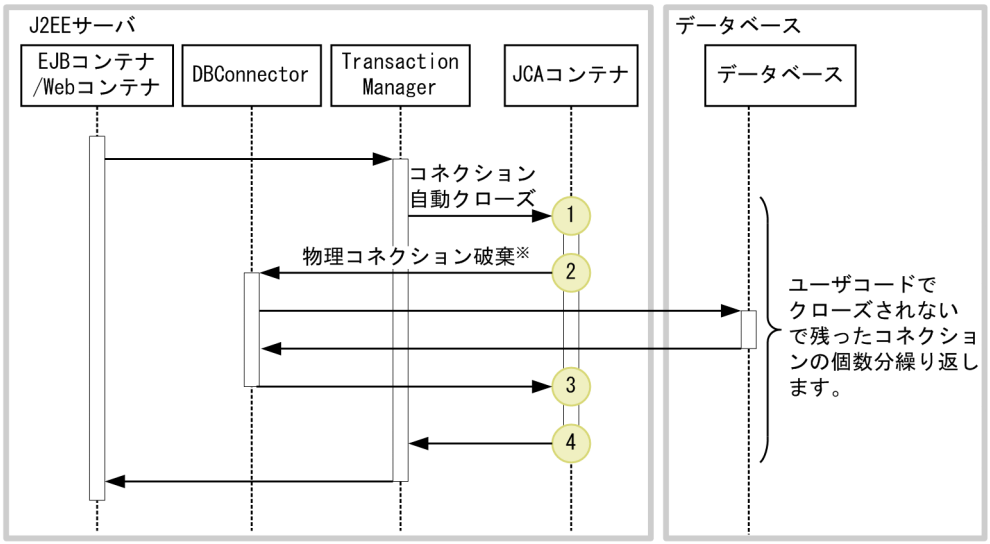
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8B84	1	コネクション自動クローズ機能によるクローズ処理呼び出し直後	B
0x8B82	2	物理コネクション破棄の呼び出し直前	B
0x8B83	3	物理コネクション破棄のリターン直後	B
0x8B85	4	コネクション自動クローズ機能によるクローズ処理リターン直前	B

（凡例）B：詳細

注※　図 8-52 の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-52　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（コネクション自動クローズ時）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

注※　コネクションのクローズ時に，常に物理コネクション破棄が起こるわけではありません。

(2)　取得できるトレース情報

取得できるトレース情報を次の表に示します。

表 8-92　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（コネクション自動クローズ時）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8B84	B	コネクション ID	−	−

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
2	0x8B82	B	−	−	−
3	0x8B83	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
4	0x8B85	B	−	−	<入り口時刻>

（凡例）B：詳細　−：該当なし

注※　図 8-52 中の番号と対応しています。

8.12.5　DB Connector for Reliable Messaging と連携する場合のトレース取得ポイントと取得できるトレース情報

DB Connector for Reliable Messaging と連携する場合のトレース取得ポイントと取得できるトレース情報について説明します。

なお，DB Connector for Reliable Messaging では，DB Connector を使用してデータベースに接続しています。そのため DB Connector for Reliable Messaging 利用時には DB Connector のトレースポイントも取得します。JDBC コネクションの生成物（java.sql.Statement など）は，DB Connector と同じトレース取得ポイントになります。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-93　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（DB Connector for Reliable Messaging 連携時）

イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8D60	1	DB Connector for Reliable Messaging 利用時の javax.sql.DataSource.getConnection()でのデータベース接続の確立	処理開始	A
0x8D61	2		処理終了	A
0x8D62	1	DB Connector for Reliable Messaging 利用時の javax.sql.DataSource.getConnection(String username, String password)での，データベース接続の確立	処理開始	A
0x8D63	2		処理終了	A

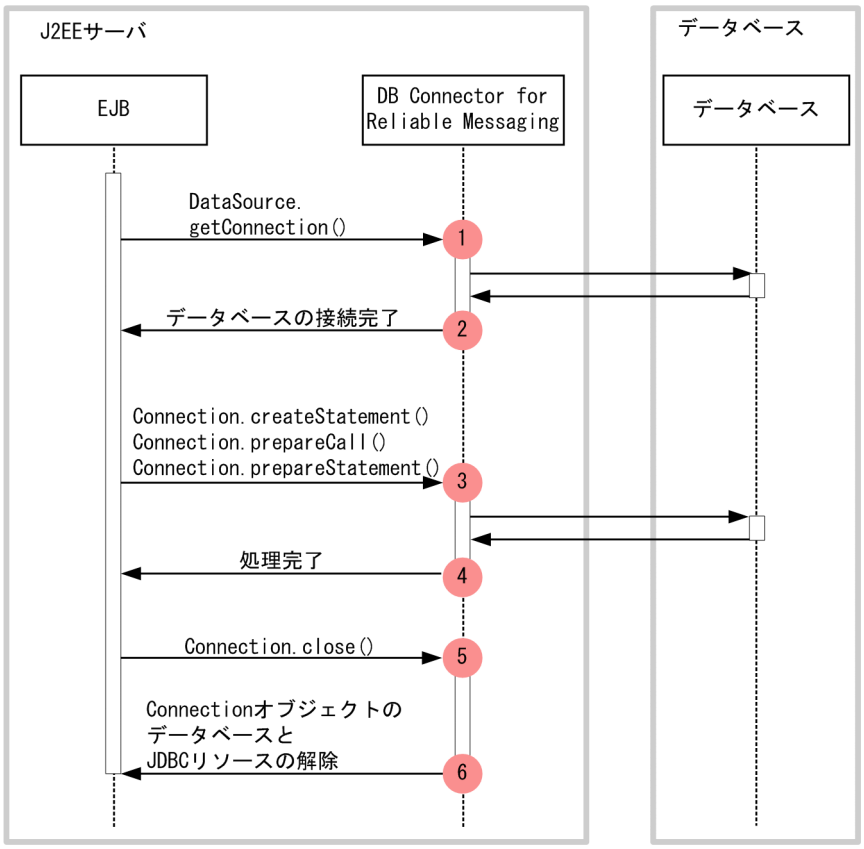
イベント ID	図中の番号※	トレース取得ポイント		レベル
0x8D80	5	DB Connector for Reliable Messaging 利用時の java.sql.Connection.close()での，Connection オブジェクトのデータベースと JDBC リソースの解除	処理開始	A
0x8D81	6		処理終了	A
0x8D82	3	DB Connector for Reliable Messaging 利用時の Connection.createStatement()	処理開始	B
0x8D83	4		処理終了	B
0x8D84	3	DB Connector for Reliable Messaging 利用時の Connection.createStatement(int resultSetType, int resultSetConcurrency)	処理開始	B
0x8D85	4		処理終了	B
0x8D86	3	DB Connector for Reliable Messaging 利用時の Connection.createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)	処理開始	B
0x8D87	4		処理終了	B
0x8D88	3	DB Connector for Reliable Messaging 利用時の Connection.prepareCall(String sql)	処理開始	B
0x8D89	4		処理終了	B
0x8D8A	3	DB Connector for Reliable Messaging 利用時の(String sql, int resultSetType, int resultSetConcurrency)	処理開始	B
0x8D8B	4		処理終了	B
0x8D8C	3	DB Connector for Reliable Messaging 利用時の Connection.prepareCall(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	処理開始	B
0x8D8D	4		処理終了	B
0x8D8E	3	DB Connector for Reliable Messaging 利用時の Connection.prepareStatement(String sql)	処理開始	B
0x8D8F	4		処理終了	B
0x8D90	3	DB Connector for Reliable Messaging 利用時の Connection.prepareStatement(String sql, int autoGeneratedKeys)	処理開始	B
0x8D91	4		処理終了	B
0x8D92	3	DB Connector for Reliable Messaging 利用時の Connection.prepareStatement(String sql, int[] columnIndexes)	処理開始	B
0x8D93	4		処理終了	B
0x8D94	3	DB Connector for Reliable Messaging 利用時の Connection.prepareStatement(String sql, int resultSetType, int resultSetConcurrency)	処理開始	B
0x8D95	4		処理終了	B
0x8D96	3	DB Connector for Reliable Messaging 利用時の Connection.prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	処理開始	B
0x8D97	4		処理終了	B
0x8D98	3	DB Connector for Reliable Messaging 利用時の Connection.prepareStatement(String sql, String[] columnNames)	処理開始	B
0x8D99	4		処理終了	B

（凡例）A：標準　B：詳細

注※　図 8-53 の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-53　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（DB Connector for Reliable Messaging 連携時）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

取得できるトレース情報を次の表に示します。

表 8-94　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（DB Connector for Reliable Messaging 連携時）

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8D60	A	－	－	－
2	0x8D61	A	コネクション ID	－	• 正常時 <入り口時刻> • 例外発生時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻><例外名>
1	0x8D62	A	－	－	－
2	0x8D63	A	コネクション ID	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
5	0x8D80	A	コネクション ID	－	－
6	0x8D81	A	－	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D82	B	－	－	－
4	0x8D83	B	－	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D84	B	－	－	－
4	0x8D85	B	－	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D86	B	－	－	－
4	0x8D87	B	－	－	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D88	B	－	－	SQL 文
4	0x8D89	B	－	－	• 正常時 <入り口時刻> • 例外発生時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻><例外名>
3	0x8D8A	B	−	−	SQL 文
4	0x8D8B	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D8C	B	−	−	SQL 文
4	0x8D8D	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D8E	B	−	−	SQL 文
4	0x8D8F	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D90	B	−	−	SQL 文
4	0x8D91	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D92	B	−	−	SQL 文
4	0x8D93	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D94	B	−	−	SQL 文
4	0x8D95	B	−	−	• 正常時 <入り口時刻> • 例外発生時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻><例外名>
3	0x8D96	B	−	−	SQL 文
4	0x8D97	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>
3	0x8D98	B	−	−	SQL 文
4	0x8D99	B	−	−	• 正常時 <入り口時刻> • 例外発生時 <入り口時刻><例外名>

（凡例）A：標準　B：詳細　−：該当なし

注※　図 8-53 中の番号と対応しています。

8.12.6　ワーク管理を使用する場合のトレース取得ポイントと取得できるトレース情報

Connector1.5 に準拠したリソースアダプタでワーク管理を使用する場合のトレース取得ポイントと取得できるトレース情報について説明します。

なお，製品のリソースアダプタ以外を使用している場合，0x8B86 を出力するときにルート AP 情報が新規に採番されます。DD（ra.xml）の<resourceadapter>-<resourceadapter-class>の値が"com.hitachi"または"com.cosminexus"で始まっている場合，製品のリソースアダプタであると判断されます。

Work から Message-driven Bean を呼び出した場合，MDB コンテナの入り口で設定されるルート AP 情報は，Work のルート AP 情報となります。個々の Message-driven Bean の処理は，スレッド ID で区別されます。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-95　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（ワーク管理を使用する場合）

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8B86	1	javax.resource.spi.work.WorkManager のメソッドの呼び出し直後	A
0x8B87	6，9，14	javax.resource.spi.work.WorkManager のメソッドのリターン直前	A
0x8B88	2，4，7，12	javax.resource.spi.work.WorkListener のメソッドの呼び出し直前	B
0x8B89	3，5，8，13	javax.resource.spi.work.WorkListener のメソッドのリターン直後	B
0x8B8A	10	javax.resource.spi.work.Work のメソッド呼び出し直前	A
0x8B8B	11	javax.resource.spi.work.Work のメソッドのリターン直後	A

（凡例）A：標準　B：詳細

注※　図 8-54～図 8-56 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-54　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（scheduleWork() 呼び出し時）

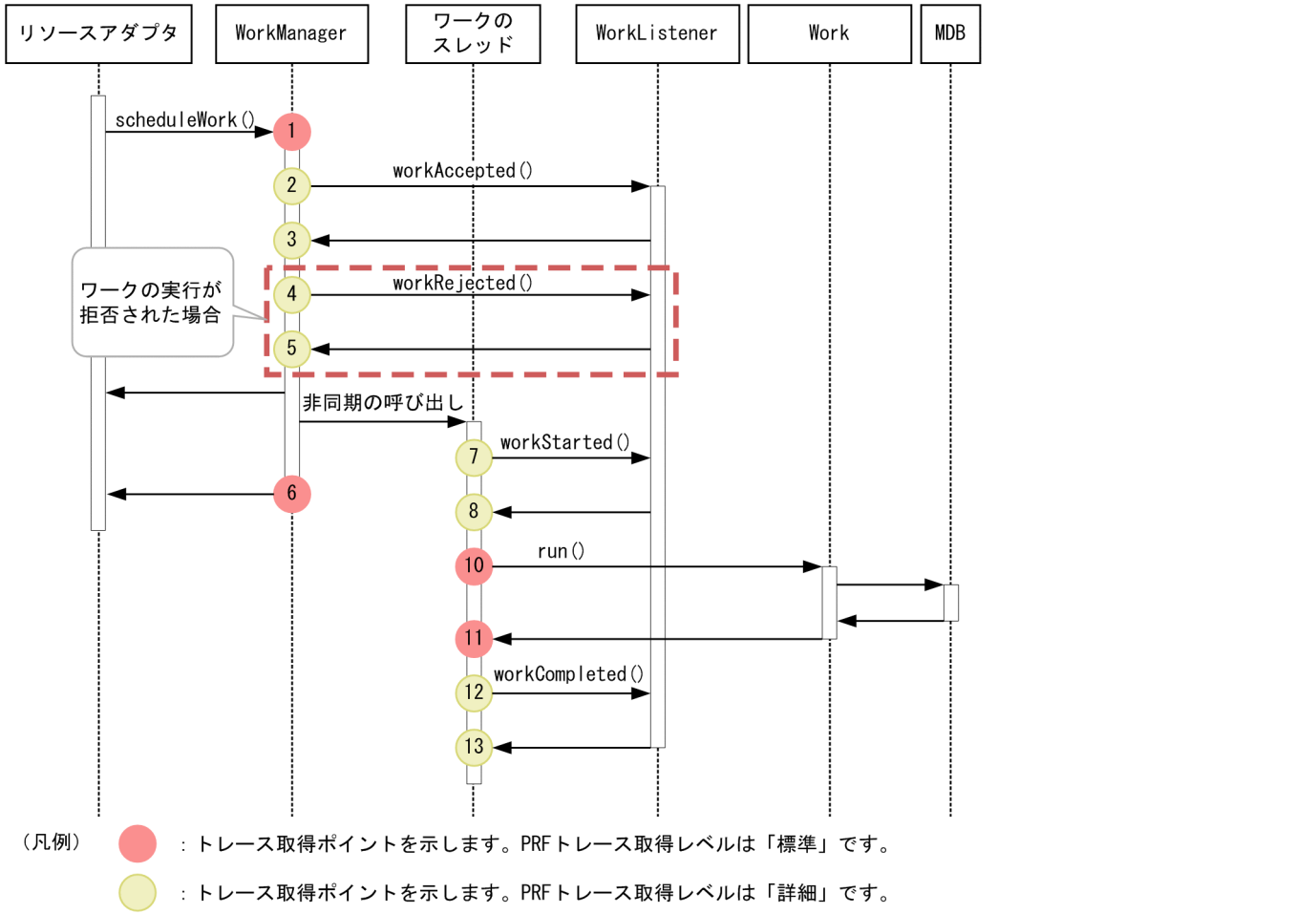
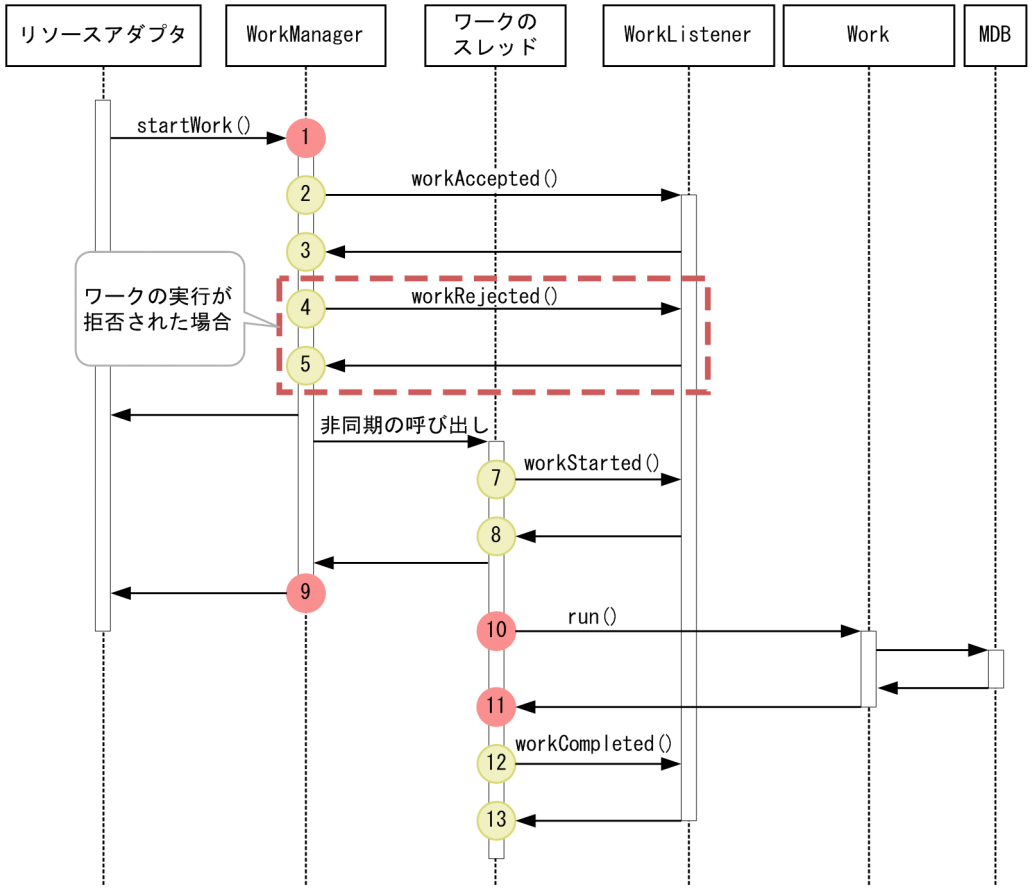


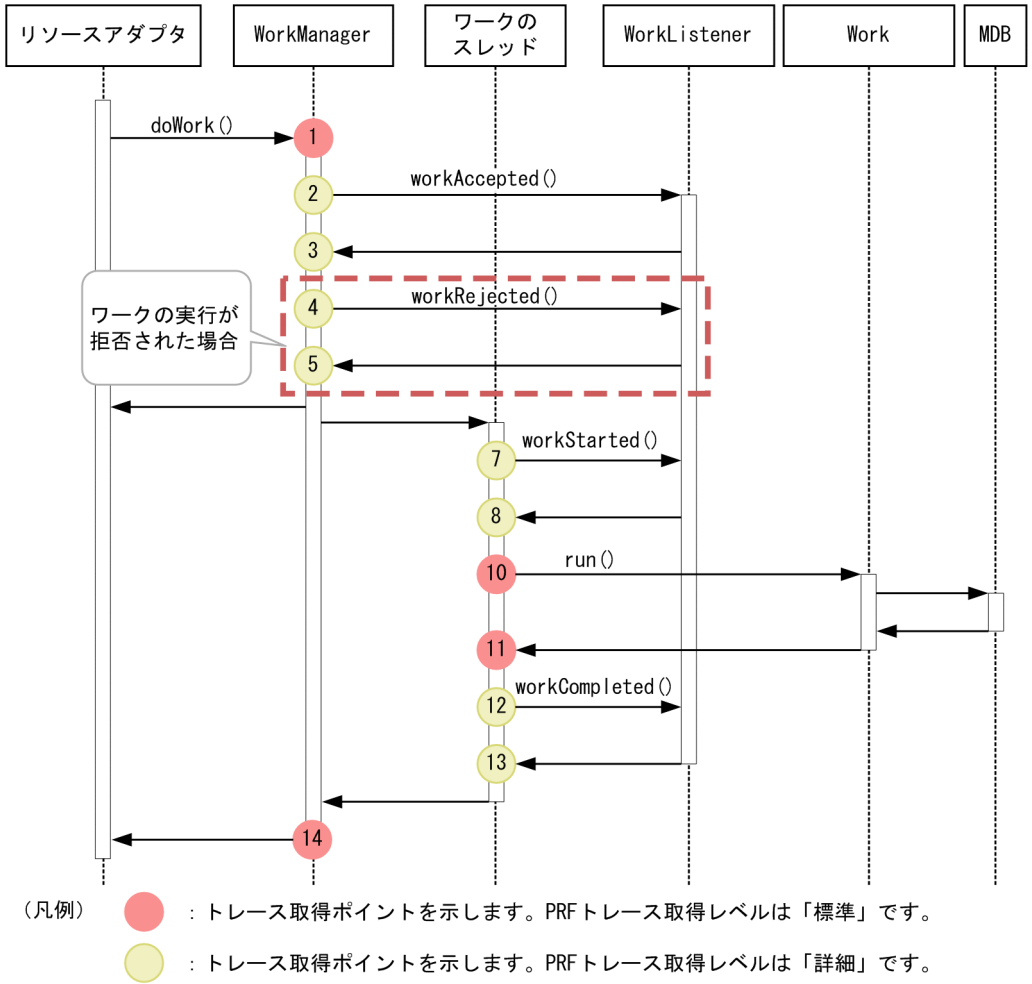
図 8-55　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（startWork()呼び出し時）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

図 8-56　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（doWork()呼び出し時）



(2)　取得できるトレース情報

取得できるトレース情報を次の表に示します。

表 8-96　DB Connector，JCA コンテナでのトレース取得ポイントの詳細（ワーク管理を使用する場合）

図中の番号 ※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8B86	A	Work のクラス名	メソッド名	リソースアダプタの表示名
2	0x8B88	B	Work のクラス名	メソッド名	リソースアダプタの表示名※2
3	0x8B89	B	Work のクラス名	メソッド名	• 正常時 <入り口時刻> • 例外発生時

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					<入り口時刻><例外名>
4	0x8B88	B	Work のクラス名	メソッド名	リソースアダプタの表示名※2
5	0x8B89	B	Work のクラス名	メソッド名	• 正常時<入り口時刻> • 例外発生時<入り口時刻><例外名>
6	0x8B87	A	Work のクラス名	メソッド名	
7	0x8B88	B	Work のクラス名	メソッド名	リソースアダプタの表示名※2
8	0x8B89	B	Work のクラス名	メソッド名	• 正常時<入り口時刻> • 例外発生時<入り口時刻><例外名>
9	0x8B87	A	Work のクラス名	メソッド名	
10	0x8B8A	A	Work のクラス名	メソッド名	リソースアダプタの表示名※2
11	0x8B8B	A	Work のクラス名	メソッド名	• 正常時<入り口時刻> • 例外発生時<入り口時刻><例外名>
12	0x8B88	B	Work のクラス名	メソッド名	リソースアダプタの表示名※2
13	0x8B89	B	Work のクラス名	メソッド名	• 正常時<入り口時刻> • 例外発生時<入り口時刻><例外名>
14	0x8B87	A	Work のクラス名	メソッド名	

（凡例）A：標準　B：詳細

注※1　図 8-54～図 8-56 中の番号と対応しています。

注※2　アプリケーションに含まれているリソースアダプタの場合，「アプリケーション名:リソースアダプタの表示名」と表示されます。

8.13　RMI のトレース取得ポイント

ここでは，RMI のトレース取得ポイントと，取得できるトレース情報について説明します。

8.13.1　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-97　RMI でのトレース取得ポイントの詳細

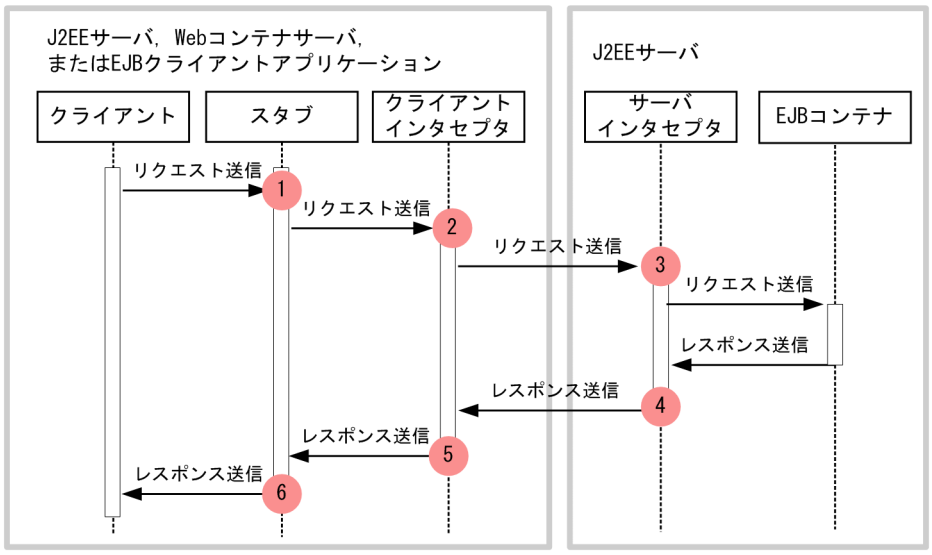
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8E01	1	スタブでのリクエスト送信処理の始まり	A
0x8E02	6	スタブでのレスポンス受信処理の終わり	A
0x8E03	2	クライアント側のリクエスト送信処理	A
0x8E04	5	クライアント側のレスポンス受信処理	A
0x8E05	3	サーバ側のリクエスト受信処理	A
0x8E06	4	サーバ側のレスポンス送信処理	A

（凡例）A：標準

注※　図 8-57 中の番号と対応しています。

RMI でのトレース取得ポイントを次の図に示します。

図 8-57　RMI のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

8.13.2　取得できるトレース情報

RMI で取得できるトレース情報を次の表に示します。

表 8-98　RMI で取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8E01	A	インタフェース名	オペレーション名	－
2	0x8E03	A	－	－	－
3	0x8E05	A	－	－	－
4	0x8E06	A	－	－	－
5	0x8E04	A	－	－	－
6	0x8E02	A	インタフェース名※2	オペレーション名※2	－※2

（凡例）A：標準　－：該当なし

注※1　図 8-57 中の番号と対応しています。

注※2　例外が発生した場合，インタフェース名およびオペレーション名は表示されません。また，オプションには発生した例外が表示されます。

8.14　OTS のトレース取得ポイント

ここでは，OTS のトレース取得ポイントと，取得できるトレース情報について説明します。

8.14.1　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-99　OTS でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x9400	1	トランザクション生成直後	A
0x9401	2	トランザクションの状態が MarkedRollback へ遷移した直後	A
0x9402	3	トランザクションの状態が RollingBack へ遷移した直後	A
0x9403	12	トランザクション決着直後	A
0x9404	4	javax.transaction.xa.XAResource への決着処理直前	B
0x9405	5	javax.transaction.xa.XAResource への決着処理直後	B
0x9406	6	サブオーディネートトランザクションへの決着処理直前	B
0x9407	9	サブオーディネートトランザクションへの決着処理直後	B
0x9408	7	スペリアトランザクションからの決着処理要求受信直後	B
0x9409	8	スペリアトランザクションからの決着処理要求返信直前	B
0x9410	13	javax.transaction.xa.XAResource オブジェクト取得処理直前	B
0x9411	14	javax.transaction.xa.XAResource オブジェクト取得処理直後	B
0x9412	10	ステータスファイルへの書き込み・読み込み直前	B
0x9413	11	ステータスファイルへの書き込み・読み込み直後	B

（凡例）A：標準　B：詳細

注※　図 8-58～図 8-63 中の番号と対応しています。

OTS でのトレース取得ポイントを次の図に示します。

図 8-58　トランザクション生成から決着までのトレース取得ポイント

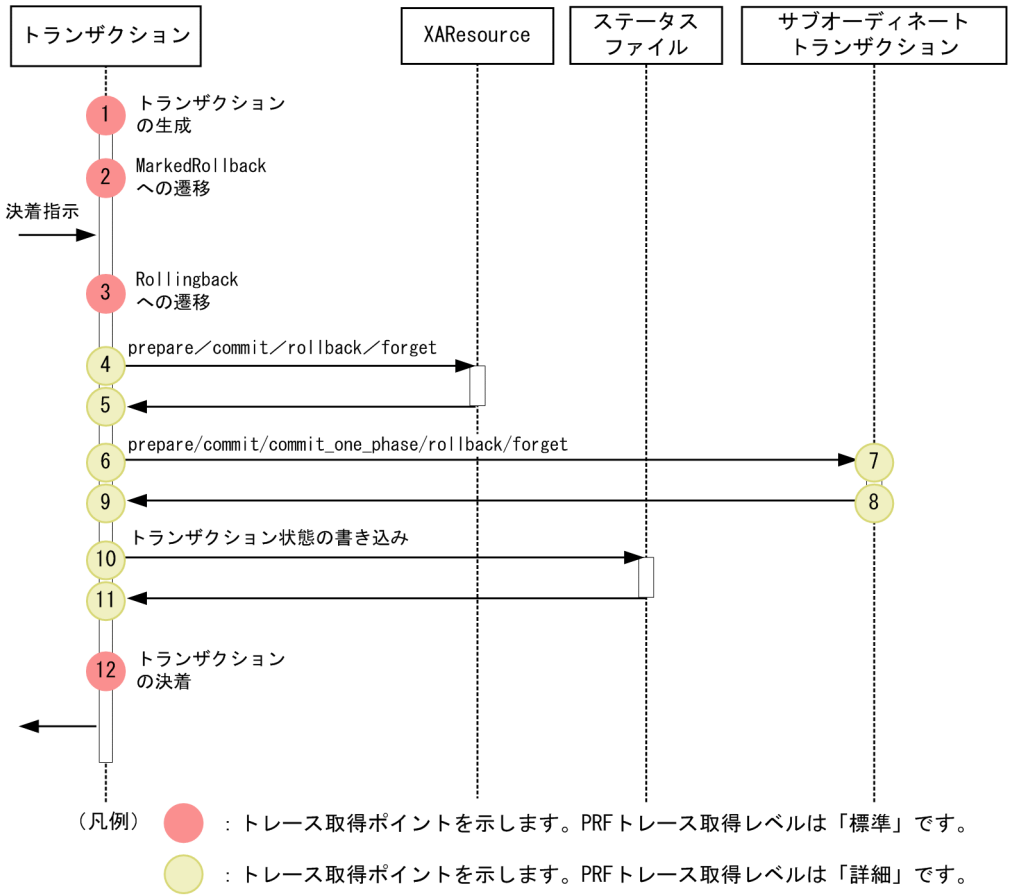


図 8-59　インプロセス OTS 初期化時のステータスファイルへの読み込み・書き込みに関連するトレース取得ポイント

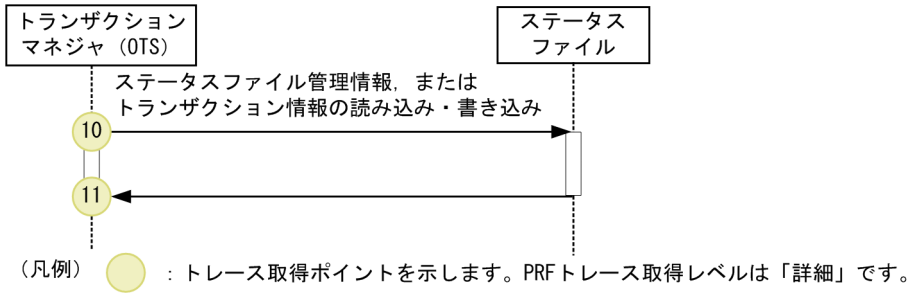


図 8-60　javax.transaction.xa.Xid の回復処理でのトレース取得ポイント

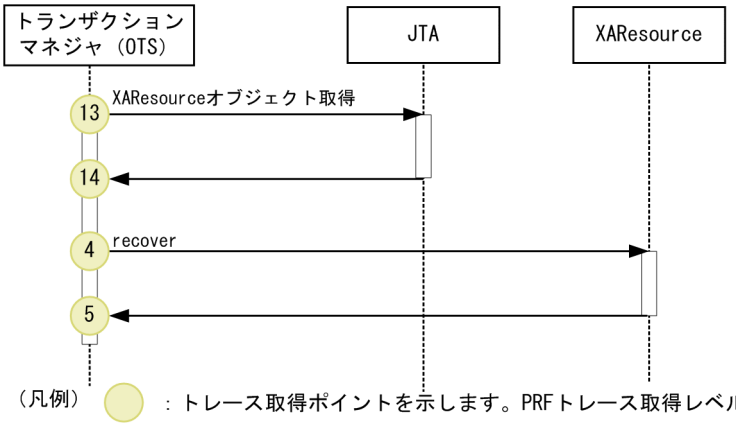


図 8-61　典型的な 1 フェーズコミットモデルでのトレース取得ポイント

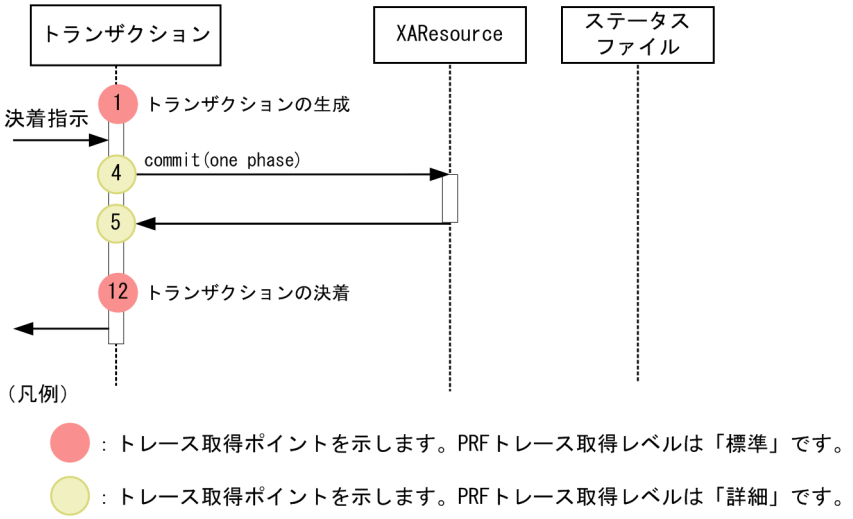


図 8-62　典型的な 2 フェーズコミットモデルでのトレース取得ポイント

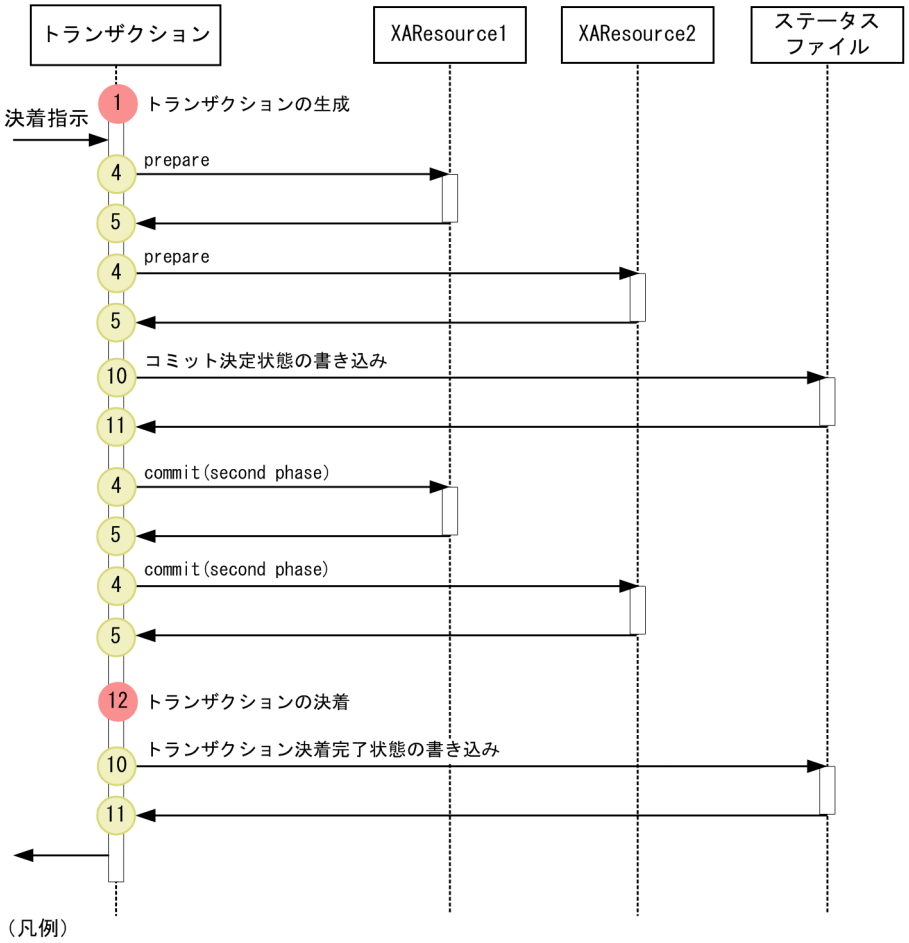
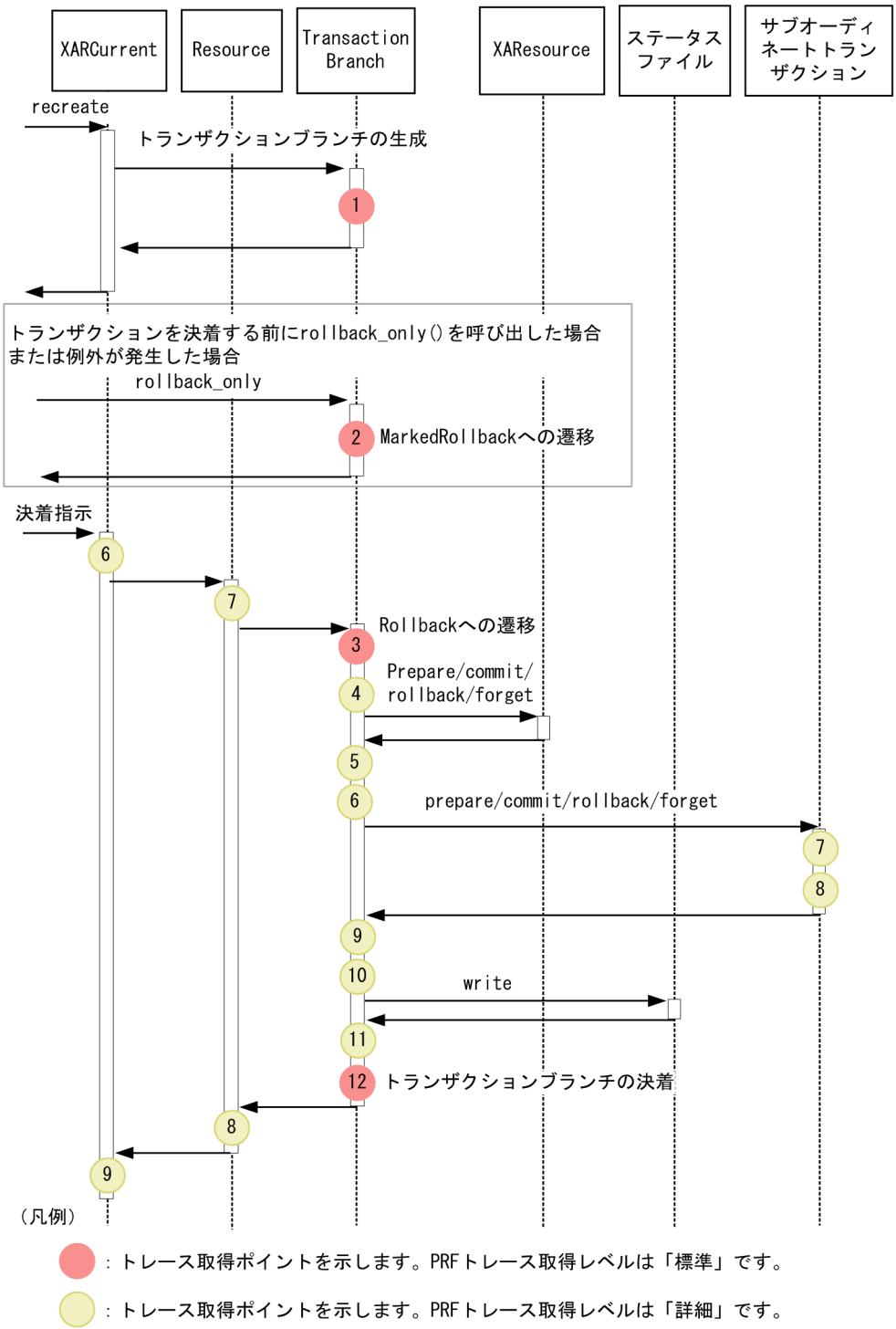


図 8-63　OpenTP1 とトランザクション連携した場合の PRF 取得情報



8.14.2　取得できるトレース情報

OTS で取得できるトレース情報を次の表に示します。なお，次の表の中で，一つの項目に複数の情報が個条書きで記載してある場合は，そのどれかが出力されることを意味します。

表 8-100　OTS で取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x9400	A	global transaction	• | created トランザクション開始指示を受け，生成されました。 • | recreated 別ノードからトランザクション処理への参加指示を受け，生成されました。 • | recovered ステータスファイルから回復されました。 • | recovered(orphan) javax.transaction.xa.XAResource から Xid が回復されましたが，これに対応するトランザクションが存在しないため，トランザクションが新たに生成されました。	<グローバルトランザクション ID>
2	0x9401	A	global transaction	| marked rollback(<遷移した理由>※2)	<グローバルトランザクション ID>
3	0x9402	A	global transaction	| rolling back(<遷移した理由>※3)	<グローバルトランザクション ID>
12	0x9403	A	global transaction	• | committed コミットしました。 • | rolled back ロールバックしました。 • | heuristic commit 強制コミットしました。 • | heuristic rollback 強制ロールバックしました。 • | heuristic mixed 部分的にコミットおよびロールバックされました。 • | heuristic hazard コミットしたかロールバックしたか不明です。 • | unknown コミットしたかロールバックしたか不明です。 • | invalid status	<グローバルトランザクション ID>

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
				上記以外の状態で決着しました。	
4	0x9404	B	• <XAResource 識別子> • なし	• > prepare prepare を発行します。 • > commit(second phase) 2 相目の commit を発行します。 • > commit(one phase) 1 相目の commit を発行します。 • > rollback rollback を発行します。 • > forget forget を発行します。 • > recover recover を発行します。	• <グローバルトランザクション ID> • なし
5	0x9405	B	• <XAResource 識別子> • なし	正常リターンの場合 • < prepare • < commit(second phase) • < commit(one phase) • < rollback • < forget • < recover 異常リターンの場合（想定外の値がリターンされた場合，または例外が発生した場合） • <!prepare • <!commit(second phase) • <!commit(one phase) • <!rollback • <!forget • <!recover	• <グローバルトランザクション ID>:<結果>※4 • なし
6	0x9406	B	resource	• > prepare prepare を発行します。 • > commit(second phase) 2 相目の commit を発行します。 • > commit(one phase) 1 相目の commit を発行します。 • > rollback rollback を発行します。 • > forget forget を発行します。	<グローバルトランザクション ID>

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
9	0x9407	B	resource	正常リターンの場合 • < prepare • < commit(second phase) • < commit(one phase) • < rollback • < forget 異常リターンの場合（想定外の値がリターンされた場合，または例外が発生した場合） • <!prepare • <!commit(second phase) • <!commit(one phase) • <!rollback • <!forget	• <グローバルトランザクション ID> • <グローバルトランザクション ID>:<結果> ※5
7	0x9408	B	subordinate transaction	• > prepare prepare 指示が入ってきました。 • > commit(second phase) 2 相目の commit 指示が入ってきました。 • > commit(one phase) 1 相目の commit 指示が入ってきました。 • > rollback rollback 指示が入ってきました。 • > forget forget 指示が入ってきました。	<グローバルトランザクション ID>
8	0x9409	B	subordinate transaction	正常リターンの場合 • < prepare • < commit(second phase) • < commit(one phase) • < rollback • < forget 異常リターンの場合（想定外の値がリターンされた場合，または例外が発生した場合） • <!prepare • <!commit(second phase) • <!commit(one phase) • <!rollback • <!forget	• <グローバルトランザクション ID> • <グローバルトランザクション ID>:<結果> ※6
13	0x9410	B	• なし	> get xaresource	• なし

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
			• <XAResource 識別子>		• <グローバルトランザクション ID>
14	0x9411	B	• なし • <XAResource 識別子>	正常リターンの場合 • < get xaresource 異常リターンの場合（想定外の値がリターンされた場合，または例外が発生した場合） • <!get xaresource	• なし • <結果>※7 • <グローバルトランザクション ID> • <グローバルトランザクション ID>:<結果>※7
10	0x9412	B	• <ステータスファイル名> • <ステータスファイル名>:<エントリ番号>※8	• > write(<書き込む内容>※9) • > read(<読み込む内容>※10)	• なし • <グローバルトランザクション ID>
11	0x9413	B	• <ステータスファイル名> • <ステータスファイル名>:<エントリ番号>※8	正常リターンの場合 • < write • < read 異常リターンの場合（想定外の値がリターンされた場合，または例外が発生した場合） • <!write • <!read	• なし • <結果>※11 • <グローバルトランザクション ID> • <グローバルトランザクション ID>:<結果>※11

（凡例）A：標準　B：詳細

注※1　図 8-58～図 8-63 中の番号と対応しています。

注※2　遷移した理由として，次のどれかが出力されます。

- operation：インプロセス OTS 外部から指示されました。
- server call：別ノードにあるサーバへの呼び出しに失敗しました。
- superior：別ノードからトランザクション処理への参加指示を受けましたが，すでにそのトランザクションが MarkedRollback 状態でした。
- sync before：決着処理中の JTA へのコールバック処理に失敗しました。

注※3　遷移した理由として，次のどれかが出力されます。

- operation：インプロセス OTS 外部から指示されました。
- timeout：グローバルトランザクションタイムアウトが発生しました。
- superior：スペリアトランザクション，または cjrollbacktrn コマンドから指示されました。
- forgotten：サブオーディネートトランザクション，または javax.transaction.xa.XAResource に対して決着指示を出すトランザクションが存在しないと判断しました。

- end：javax.transaction.xa.XAResource に対する end に失敗しました。
- prepare：javax.transaction.xa.XAResource に対する prepare に失敗しました。
- write prepared：ステータスファイルへの prepared 書き込みに失敗しました。
- write committing：ステータスファイルへの committing 書き込みに失敗しました。

注※4　prepare，commit，rollback または forget の場合，結果として次のどれかが出力されます。

- リターン値
- XAException のエラーコード
- 例外(=XAException 以外の例外)の toString()

recover の場合，結果として次のどれかが出力されます。

- 回復された Xid の個数
- null（Xid 配列自体が null の場合）
- XAException のエラーコード
- 例外(=XAException 以外の例外)の toString()

注※5　prepare の場合，結果として次のどちらかが出力されます。

- リターンされた値
- 例外の toString()

prepare 以外の場合，結果として例外の toString()が出力されます。

注※6　prepare の場合，結果として次のどちらかが出力されます。

- リターンする値
- 例外の toString()

prepare 以外の場合，結果として例外の toString()が出力されます。

注※7　結果として次のどちらかが出力されます。

- 例外の toString()
- null（リターン値が null の場合）

注※8　内部情報です。

注※9　書き込み内容として，次のどれかが出力されます。

- management info：ステータスファイル管理情報
- status file body：ステータスファイル本体
- prepared：プリペア完了状態
- committing：コミット決定状態
- heuristic commit：強制コミットされた状態
- heuristic rollback：強制ロールバックされた状態
- heuristic mixed：部分的にコミットおよびロールバックされた状態
- heuristic hazard：コミットしたかロールバックしたか不明な状態
- forgotten：トランザクション決着完了状態

注※10　読み込む内容として，次のどちらかが出力されます。

- management info：ステータスファイル管理情報
- status file body：ステータスファイル本体

注※11　結果として，次のどれかが出力されます。

- 書き込みサイズ（単位：バイト）
- 読み込みサイズ（単位：バイト）
- 例外の toString()

8.15　標準出力／標準エラー出力／ユーザログのトレース取得ポイント

ここでは，標準出力／標準エラー出力／ユーザログのトレース取得ポイントと，取得できるトレース情報について説明します。

8.15.1　標準出力／標準エラー出力のトレース取得ポイント

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-101　標準出力／標準エラー出力でのトレース取得ポイントの詳細

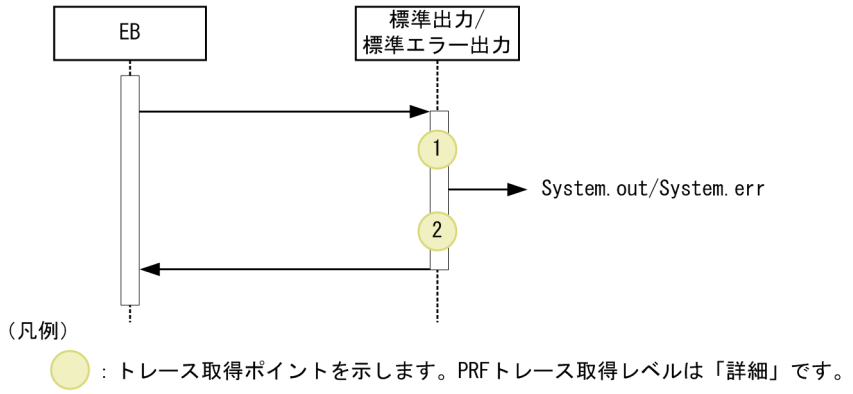
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x9C00	1	標準出力/標準エラー出力への出力開始時	B
0x9C01	2	標準出力/標準エラー出力への出力完了時	B

（凡例）B：詳細

注※　図 8-64 中の番号と対応しています。

標準出力／標準エラー出力のトレース取得ポイントを次の図に示します。

図 8-64　標準出力／標準エラー出力のトレース取得ポイント



(2)　取得できるトレース情報

標準出力／標準エラー出力で取得できるトレース情報を次の表に示します。

表 8-102　標準出力／標準エラー出力で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x9C00	B	ストリーム名（「out」／「err」）	－	－
2	0x9C01	B	ストリーム名（「out」／「err」）	－	－

（凡例）B：詳細　－：該当なし

注※　図 8-64 中の番号と対応しています。

8.15.2　ユーザログのトレース取得ポイント

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-103　ユーザログのトレース取得ポイントの詳細

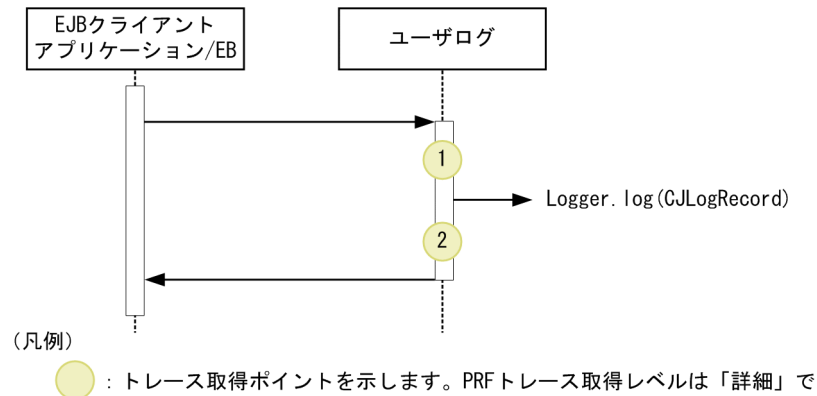
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x9C02	1	ユーザログを使用したメッセージ出力開始時	B
0x9C03	2	ユーザログを使用したメッセージ出力完了時	B

（凡例）B：詳細

注※　図 8-65 中の番号と対応しています。

ユーザログのトレース取得ポイントを次の図に示します。

図 8-65　ユーザログのトレース取得ポイント



(2)　取得できるトレース情報

ユーザログで取得できるトレース情報を次の表に示します。

表 8-104　ユーザログで取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x9C02	B	アプリケーション識別名	メッセージ ID	－
2	0x9C03	B	アプリケーション識別名	メッセージ ID	－

（凡例）B：詳細　－：該当なし

注※　図 8-65 中の番号と対応しています。

8.16　DI のトレース取得ポイント

ここでは，DI のトレース取得ポイントと，取得できるトレース情報について説明します。

8.16.1　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-105　DI でのトレース取得ポイントの詳細

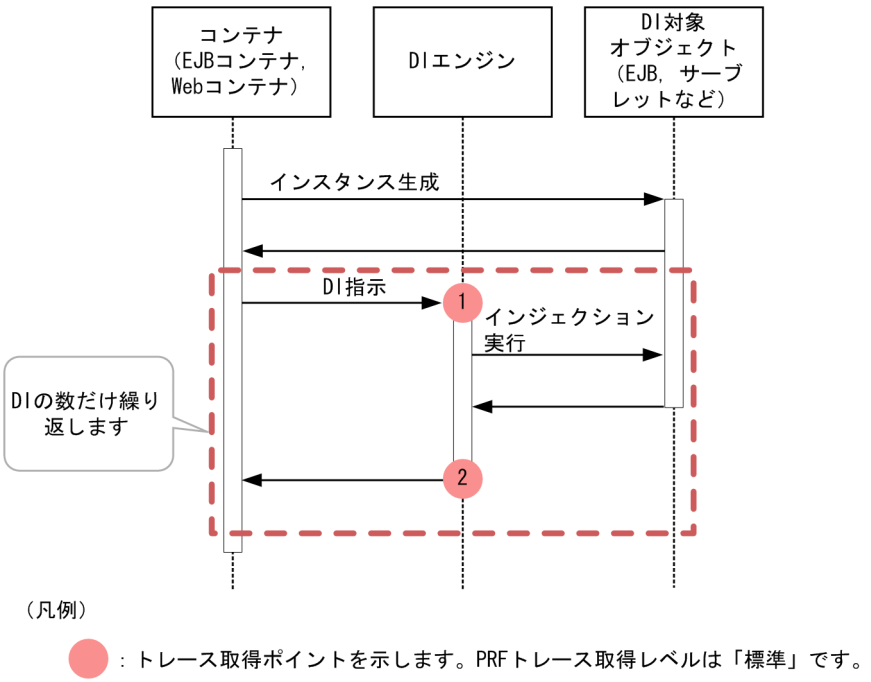
イベント ID	図中の番号※	トレース取得ポイント	レベル
0x9D00	1	依存性を注入する直前	A
0x9D01	2	依存性を注入した直後	A

（凡例）A：標準

注※　図 8-66 中の番号と対応しています。

DI のトレース取得ポイントを次の図に示します。

図 8-66　DI のトレース取得ポイント



8.16.2　取得できるトレース情報

DI で取得できるトレース情報を次の表に示します。

表 8-106　DI で取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x9D00	A	依存性を注入する先のターゲット名	注入するリファレンス名	－
2	0x9D01	A	依存性を注入した先のターゲット名	注入したリファレンス名	※2

（凡例）A：標準　－：該当なし

注※1　図 8-66 中の番号と対応しています。

注※2　正常に処理された場合，入り口時刻が表示されます。例外が発生した場合，入り口時刻および例外が表示されます。

8.17　バッチアプリケーション実行機能のトレース取得ポイント

ここでは，バッチアプリケーション実行機能のトレース取得ポイントと，取得できるトレース情報について説明します。

8.17.1　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-107　バッチアプリケーション実行機能でのトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0x8E05	1※2	バッチアプリケーションの実行リクエストを受信した直後	A
0xA100	2	バッチアプリケーションの実行直前	A
0xA101	3	バッチアプリケーションの終了直後	A
0x8E06	4※2	バッチアプリケーションの終了結果を送信する直前	A

（凡例）A：標準

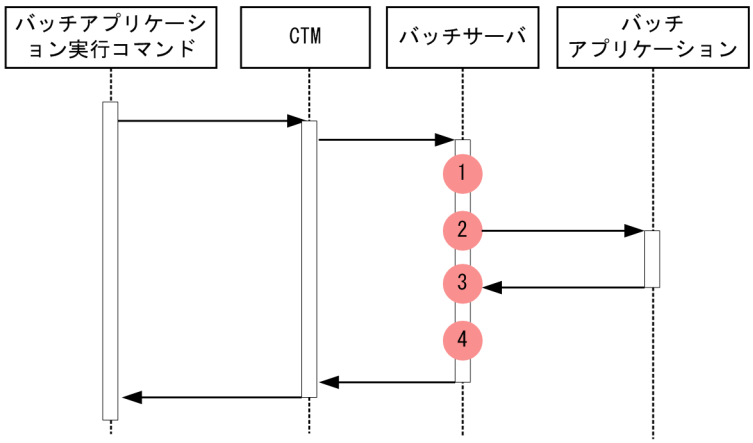
注※1　図 8-67 中の番号と対応しています。

注※2　トレースが取得されるのはスケジューリング機能使用時だけです。

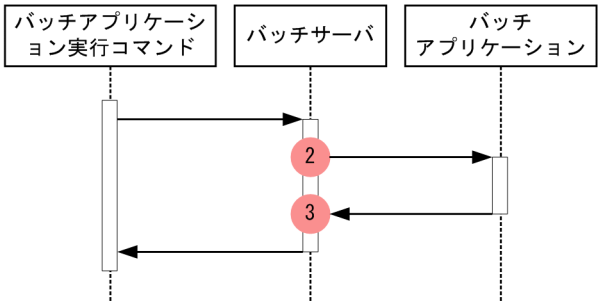
バッチアプリケーション実行機能のトレース取得ポイントを次の図に示します。

図 8-67　バッチアプリケーション実行機能のトレース取得ポイント

スケジューリング機能を使用する場合



スケジューリング機能を使用しない場合



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

8.17.2　取得できるトレース情報

バッチアプリケーション実行機能で取得できるトレース情報を次の表に示します。

表 8-108　バッチアプリケーション実行機能で取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0x8E05	A	－	－	－
2	0xA100	A	バッチアプリケーションのクラス名（パッケージ名を含む）	－	－
3	0xA101	A	バッチアプリケーションのクラス名（パッケージ名を含む）	－	※2
4	0x8E06	A	－	－	－

（凡例）A：標準　－：該当なし

注※1　図 8-67 中の番号と対応しています。

注※2　正常に処理された場合，入り口時刻が表示されます。例外が発生した場合，入り口時刻および例外が表示されます。

8.18　JPA でのトレース取得ポイント

ここでは，JPA のトレース取得ポイントと，取得できるトレース情報について説明します。

8.18.1　アプリケーション管理の永続化コンテキストを利用した場合のトレース取得ポイントと取得できるトレース情報

アプリケーション管理の永続化コンテキストを利用した場合のトレース取得ポイントと取得できるトレース情報について説明します。

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-109　アプリケーション管理の永続化コンテキストを利用した場合のトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD100	1	EntityManagerFactory#createEntityManager()の処理開始	A
0xD101	2	EntityManagerFactory#createEntityManager()の処理終了	A
0xD102	1	EntityManagerFactory#createEntityManager(Map map)の処理開始	A
0xD103	2	EntityManagerFactory#createEntityManager(Map map)の処理終了	A
0xD104	1	EntityManagerFactory#createEntityManager(SynchronizationType type)の処理開始	A
0xD105	2	EntityManagerFactory#createEntityManager(SynchronizationType type)の処理終了	A
0xD106	1	EntityManagerFactory#createEntityManager(SynchronizationType type, Map map)の処理開始	A
0xD107	2	EntityManagerFactory#createEntityManager(SynchronizationType type, Map map)の処理終了	A
0xD108	1	EntityManagerFactory#getCriteriaBuilder()の処理開始	B
0xD109	2	EntityManagerFactory#getCriteriaBuilder()の処理終了	B
0xD10A	1	EntityManagerFactory#getMetamodel()の処理開始	B
0xD10B	2	EntityManagerFactory#getMetamodel()の処理終了	B
0xD10C	1	EntityManagerFactory#getProperties()の処理開始	B
0xD10D	2	EntityManagerFactory#getProperties()の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD10E	1	EntityManagerFactory#getCache()の処理開始	B
0xD10F	2	EntityManagerFactory#getCache()の処理終了	B
0xD110	1	EntityManagerFactory#getPersistenceUnitUtil()の処理開始	B
0xD111	2	EntityManagerFactory#getPersistenceUnitUtil()の処理終了	B
0xD112	1	EntityManagerFactory#addNamedQuery(String name, Query query)の処理開始	B
0xD113	2	EntityManagerFactory#addNamedQuery(String name, Query query)の処理終了	B
0xD114	1	EntityManagerFactory#unwrap(Class<T> cls)の処理開始	B
0xD115	2	EntityManagerFactory#unwrap(Class<T> cls)の処理終了	B
0xD116	1	EntityManagerFactory#addNamedEntityGraph(String graphName, EntityGraph<T> entityGraph)の処理開始	B
0xD117	2	EntityManagerFactory#addNamedEntityGraph(String graphName, EntityGraph<T> entityGraph)の処理終了	B
0xD118	1	EntityManagerFactory#isOpen()の処理開始	B
0xD119	2	EntityManagerFactory#isOpen()の処理終了	B
0xD11A	1	EntityManagerFactory#close()の処理開始	A
0xD11B	2	EntityManagerFactory#close()の処理終了	A
0xD11C	1	EntityManager#find(Class<T> entityClass, Object primaryKey)の処理開始	B
0xD11D	2	EntityManager#find(Class<T> entityClass, Object primaryKey)の処理終了	B
0xD11E	1	EntityManager#find(Class<T> entityClass, Object primaryKey, Map<String, Object> properties)の処理開始	B
0xD11F	2	EntityManager#find(Class<T> entityClass, Object primaryKey, Map<String, Object> properties)の処理終了	B
0xD120	1	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode)の処理開始	B
0xD121	2	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode)の処理終了	B
0xD122	1	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode, Map<String, Object> properties)の処理開始	B
0xD123	2	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode, Map<String, Object> properties)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD124	1	EntityManager#getReference(Class<T> entityClass, Object primaryKey)の処理開始	B
0xD125	2	EntityManager#getReference(Class<T> entityClass, Object primaryKey)の処理終了	B
0xD126	1	EntityManager#contains(Object entity)の処理開始	B
0xD127	2	EntityManager#contains(Object entity)の処理終了	B
0xD128	1	EntityManager#lock(Object entity, LockModeType lockMode)の処理開始	B
0xD129	2	EntityManager#lock(Object entity, LockModeType lockMode)の処理終了	B
0xD12A	1	EntityManager#lock(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理開始	B
0xD12B	2	EntityManager#lock(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理終了	B
0xD12C	1	EntityManager#merge(T entity)の処理開始	B
0xD12D	2	EntityManager#merge(T entity)の処理終了	B
0xD12E	1	EntityManager#persist(Object entity)の処理開始	B
0xD12F	2	EntityManager#persist(Object entity)の処理終了	B
0xD130	1	EntityManager#refresh(Object entity)の処理開始	B
0xD131	2	EntityManager#refresh(Object entity)の処理終了	B
0xD132	1	EntityManager#refresh(Object entity, Map<String,Object> properties)の処理開始	B
0xD133	2	EntityManager#refresh(Object entity, Map<String,Object> properties)の処理終了	B
0xD134	1	EntityManager#refresh(Object entity, LockModeType lockMode)の処理開始	B
0xD135	2	EntityManager#refresh(Object entity, LockModeType lockMode)の処理終了	B
0xD136	1	EntityManager#refresh(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理開始	B
0xD137	2	EntityManager#refresh(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理終了	B
0xD138	1	EntityManager#remove(Object entity)の処理開始	B
0xD139	2	EntityManager#remove(Object entity)の処理終了	B
0xD13A	1	EntityManager#clear()の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD13B	2	EntityManager#clear()の処理終了	B
0xD13C	1	EntityManager#flush()の処理開始	B
0xD13D	2	EntityManager#flush()の処理終了	B
0xD13E	1	EntityManager#detach(Object entity)の処理開始	B
0xD13F	2	EntityManager#detach(Object entity)の処理終了	B
0xD140	1	EntityManager#createEntityGraph(Class<T> rootType)の処理開始	B
0xD141	2	EntityManager#createEntityGraph(Class<T> rootType)の処理終了	B
0xD142	1	EntityManager#createEntityGraph(String graphName)の処理開始	B
0xD143	2	EntityManager#createEntityGraph(String graphName)の処理終了	B
0xD144	1	EntityManager#createQuery(String qlString)の処理開始	A
0xD145	2	EntityManager#createQuery(String qlString)の処理終了	A
0xD146	1	EntityManager#createQuery(CriteriaQuery<T> criteriaQuery)の処理開始	A
0xD147	2	EntityManager#createQuery(CriteriaQuery<T> criteriaQuery)の処理終了	A
0xD148	1	EntityManager#createQuery(CriteriaUpdate updateQuery)の処理開始	A
0xD149	2	EntityManager#createQuery(CriteriaUpdate updateQuery)の処理終了	A
0xD14A	1	EntityManager#createQuery(CriteriaDelete deleteQuery)の処理開始	A
0xD14B	2	EntityManager#createQuery(CriteriaDelete deleteQuery)の処理終了	A
0xD14C	1	EntityManager#createQuery(String qlString, Class<T> resultClass)の処理開始	A
0xD14D	2	EntityManager#createQuery(String qlString, Class<T> resultClass)の処理終了	A
0xD14E	1	EntityManager#createNamedQuery(String name)の処理開始	A
0xD14F	2	EntityManager#createNamedQuery(String name)の処理終了	A
0xD150	1	EntityManager#createNamedQuery(String name, Class<T> resultClass)の処理開始	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD151	2	EntityManager#createNamedQuery(String name, Class<T> resultClass)の処理終了	A
0xD152	1	EntityManager#createNativeQuery(String sqlString)の処理開始	A
0xD153	2	EntityManager#createNativeQuery(String sqlString)の処理終了	A
0xD154	1	EntityManager#createNativeQuery(String sqlString, Class resultClass)の処理開始	A
0xD155	2	EntityManager#createNativeQuery(String sqlString, Class resultClass)の処理終了	A
0xD156	1	EntityManager#createNativeQuery(String sqlString, String resultSetMapping)の処理開始	A
0xD157	2	EntityManager#createNativeQuery(String sqlString, String resultSetMapping)の処理終了	A
0xD158	1	EntityManager#createNamedStoredProcedureQuery(String name)の処理開始	A
0xD159	2	EntityManager#createNamedStoredProcedureQuery(String name)の処理終了	A
0xD15A	1	EntityManager#createStoredProcedureQuery(String procedureName)の処理開始	A
0xD15B	2	EntityManager#createStoredProcedureQuery(String procedureName)の処理終了	A
0xD15C	1	EntityManager#createStoredProcedureQuery(String procedureName, Class... resultClasses)の処理開始	A
0xD15D	2	EntityManager#createStoredProcedureQuery(String procedureName, Class... resultClasses)の処理終了	A
0xD15E	1	EntityManager#createStoredProcedureQuery(String procedureName, String... resultSetMappings)の処理開始	A
0xD15F	2	EntityManager#createStoredProcedureQuery(String procedureName, String... resultSetMappings)の処理終了	A
0xD160	1	EntityManager#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD161	2	EntityManager#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD162	1	EntityManager#getFlushMode()の処理開始	B
0xD163	2	EntityManager#getFlushMode()の処理終了	B
0xD164	1	EntityManager#joinTransaction()の処理開始	B
0xD165	2	EntityManager#joinTransaction()の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD166	1	EntityManager#getTransaction()の処理開始	B
0xD167	2	EntityManager#getTransaction()の処理終了	B
0xD168	1	EntityManager#getDelegate()の処理開始	B
0xD169	2	EntityManager#getDelegate()の処理終了	B
0xD16A	1	EntityManager#unwrap(Class<T> cls)の処理開始	B
0xD16B	2	EntityManager#unwrap(Class<T> cls)の処理終了	B
0xD16C	1	EntityManager#getCriteriaBuilder()の処理開始	B
0xD16D	2	EntityManager#getCriteriaBuilder()の処理終了	B
0xD16E	1	EntityManager#getEntityGraph(String graphName)の処理開始	B
0xD16F	2	EntityManager#getEntityGraph(String graphName)の処理終了	B
0xD170	1	EntityManager#getEntityGraphs(Class<T> entityClass)の処理開始	B
0xD171	2	EntityManager#getEntityGraphs(Class<T> entityClass)の処理終了	B
0xD172	1	EntityManager#getEntityManagerFactory()の処理開始	B
0xD173	2	EntityManager#getEntityManagerFactory()の処理終了	B
0xD174	1	EntityManager#getLockMode(Object entity)の処理開始	B
0xD175	2	EntityManager#getLockMode(Object entity)の処理終了	B
0xD176	1	EntityManager#getMetamodel()の処理開始	B
0xD177	2	EntityManager#getMetamodel()の処理終了	B
0xD178	1	EntityManager#setProperty(String propertyName, Object value)の処理開始	B
0xD179	2	EntityManager#setProperty(String propertyName, Object value)の処理終了	B
0xD17A	1	EntityManager#getProperties()の処理開始	B
0xD17B	2	EntityManager#getProperties()の処理終了	B
0xD17C	1	EntityManager#isJoinedToTransaction()の処理開始	B
0xD17D	2	EntityManager#isJoinedToTransaction()の処理終了	B
0xD17E	1	EntityManager#isOpen()の処理開始	B
0xD17F	2	EntityManager#isOpen()の処理終了	B
0xD180	1	EntityManager#close()の処理開始	A
0xD181	2	EntityManager#close()の処理終了	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD182	1	EntityTransaction#begin()の処理開始	A
0xD183	2	EntityTransaction#begin()の処理終了	A
0xD184	1	EntityTransaction#commit()の処理開始	A
0xD185	2	EntityTransaction#commit()の処理終了	A
0xD186	1	EntityTransaction#rollback()の処理開始	A
0xD187	2	EntityTransaction#rollback()の処理終了	A
0xD188	1	EntityTransaction#getRollbackOnly()の処理開始	B
0xD189	2	EntityTransaction#getRollbackOnly()の処理終了	B
0xD18A	1	EntityTransaction#setRollbackOnly()の処理開始	B
0xD18B	2	EntityTransaction#setRollbackOnly()の処理終了	B
0xD18C	1	EntityTransaction#isActive()の処理開始	B
0xD18D	2	EntityTransaction#isActive()の処理終了	B
0xD18E	1	Query#executeUpdate()の処理開始	B
0xD18F	2	Query#executeUpdate()の処理終了	B
0xD190	1	Query#getResultList()の処理開始	B
0xD191	2	Query#getResultList()の処理終了	B
0xD192	1	Query#getSingleResult()の処理開始	B
0xD193	2	Query#getSingleResult()の処理終了	B
0xD194	1	Query#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD195	2	Query#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD196	1	Query#getFlushMode()の処理開始	B
0xD197	2	Query#getFlushMode()の処理終了	B
0xD198	1	Query#setFirstResult(int startPosition)の処理開始	B
0xD199	2	Query#setFirstResult(int startPosition)の処理終了	B
0xD19A	1	Query#getFirstResult()の処理開始	B
0xD19B	2	Query#getFirstResult()の処理終了	B
0xD19C	1	Query#setMaxResults(int maxResult)の処理開始	B
0xD19D	2	Query#setMaxResults(int maxResult)の処理終了	B
0xD19E	1	Query#getMaxResults()の処理開始	B
0xD19F	2	Query#getMaxResults()の処理終了	B
0xD1A0	1	Query#setHint(String hintName, Object value)の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1A1	2	Query#setHint(String hintName, Object value)の処理終了	B
0xD1A2	1	Query#getHints()の処理開始	B
0xD1A3	2	Query#getHints()の処理終了	B
0xD1A4	1	Query#setParameter(int position, Calendar value, TemporalType temporalType)の処理開始	B
0xD1A5	2	Query#setParameter(int position, Calendar value, TemporalType temporalType)の処理終了	B
0xD1A6	1	Query#setParameter(int position, Date value, TemporalType temporalType)の処理開始	B
0xD1A7	2	Query#setParameter(int position, Date value, TemporalType temporalType)の処理終了	B
0xD1A8	1	Query#setParameter(int position, Object value)の処理開始	B
0xD1A9	2	Query#setParameter(int position, Object value)の処理終了	B
0xD1AA	1	Query#setParameter(String name, Calendar value, TemporalType temporalType)の処理開始	B
0xD1AB	2	Query#setParameter(String name, Calendar value, TemporalType temporalType)の処理終了	B
0xD1AC	1	Query#setParameter(String name, Date value, TemporalType temporalType)の処理開始	B
0xD1AD	2	Query#setParameter(String name, Date value, TemporalType temporalType)の処理終了	B
0xD1AE	1	Query#setParameter(String name, Object value)の処理開始	B
0xD1AF	2	Query#setParameter(String name, Object value)の処理終了	B
0xD1B0	1	Query#setParameter(Parameter<T> param, T value)の処理開始	B
0xD1B1	2	Query#setParameter(Parameter<T> param, T value)の処理終了	B
0xD1B2	1	Query#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理開始	B
0xD1B3	2	Query#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理終了	B
0xD1B4	1	Query#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理開始	B
0xD1B5	2	Query#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理終了	B
0xD1B6	1	Query#getParameters()の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1B7	2	Query#getParameters()の処理終了	B
0xD1B8	1	Query#getParameter(String name)の処理開始	B
0xD1B9	2	Query#getParameter(String name)の処理終了	B
0xD1BA	1	Query#getParameter(String name, Class<T> type)の処理開始	B
0xD1BB	2	Query#getParameter(String name, Class<T> type)の処理終了	B
0xD1BC	1	Query#getParameter(int position)の処理開始	B
0xD1BD	2	Query#getParameter(int position)の処理終了	B
0xD1BE	1	Query#getParameter(int position, Class<T> type)の処理開始	B
0xD1BF	2	Query#getParameter(int position, Class<T> type)の処理終了	B
0xD1C0	1	Query#isBound(Parameter<?> param)の処理開始	B
0xD1C1	2	Query#isBound(Parameter<?> param)の処理終了	B
0xD1C2	1	Query#getParameterValue(Parameter<T> param)の処理開始	B
0xD1C3	2	Query#getParameterValue(Parameter<T> param)の処理終了	B
0xD1C4	1	Query#getParameterValue(String name)の処理開始	B
0xD1C5	2	Query#getParameterValue(String name)の処理終了	B
0xD1C6	1	Query#getParameterValue(int position)の処理開始	B
0xD1C7	2	Query#getParameterValue(int position)の処理終了	B
0xD1C8	1	Query#setLockMode(LockModeType lockMode)の処理開始	B
0xD1C9	2	Query#setLockMode(LockModeType lockMode)の処理終了	B
0xD1CA	1	Query#getLockMode()の処理開始	B
0xD1CB	2	Query#getLockMode()の処理終了	B
0xD1CC	1	Query#unwrap(Class<T> cls)の処理開始	B
0xD1CD	2	Query#unwrap(Class<T> cls)の処理終了	B
0xD1CE	1	TypedQuery#executeUpdate()の処理開始	B
0xD1CF	2	TypedQuery#executeUpdate()の処理終了	B
0xD1D0	1	TypedQuery#getResultList()の処理開始	B
0xD1D1	2	TypedQuery#getResultList()の処理終了	B
0xD1D2	1	TypedQuery#getSingleResult()の処理開始	B
0xD1D3	2	TypedQuery#getSingleResult()の処理終了	B
0xD1D4	1	TypedQuery#setMaxResults(int maxResult)の処理開始	B
0xD1D5	2	TypedQuery#setMaxResults(int maxResult)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1D6	1	TypedQuery#setFirstResult(int startPosition)の処理開始	B
0xD1D7	2	TypedQuery#setFirstResult(int startPosition)の処理終了	B
0xD1D8	1	TypedQuery#setHint(String hintName, Object value)の処理開始	B
0xD1D9	2	TypedQuery#setHint(String hintName, Object value)の処理終了	B
0xD1DA	1	TypedQuery#setParameter(Parameter<T> param, T value)の処理開始	B
0xD1DB	2	TypedQuery#setParameter(Parameter<T> param, T value)の処理終了	B
0xD1DC	1	TypedQuery#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理開始	B
0xD1DD	2	TypedQuery#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理終了	B
0xD1DE	1	TypedQuery#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理開始	B
0xD1DF	2	TypedQuery#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理終了	B
0xD1E0	1	TypedQuery#setParameter(String name, Object value)の処理開始	B
0xD1E1	2	TypedQuery#setParameter(String name, Object value)の処理終了	B
0xD1E2	1	TypedQuery#setParameter(String name, Calendar value, TemporalType temporalType)の処理開始	B
0xD1E3	2	TypedQuery#setParameter(String name, Calendar value, TemporalType temporalType)の処理終了	B
0xD1E4	1	TypedQuery#setParameter(String name, Date value, TemporalType temporalType)の処理開始	B
0xD1E5	2	TypedQuery#setParameter(String name, Date value, TemporalType temporalType)の処理終了	B
0xD1E6	1	TypedQuery#setParameter(int position, Object value)の処理開始	B
0xD1E7	2	TypedQuery#setParameter(int position, Object value)の処理終了	B
0xD1E8	1	TypedQuery#setParameter(int position, Calendar value, TemporalType temporalType)の処理開始	B
0xD1E9	2	TypedQuery#setParameter(int position, Calendar value, TemporalType temporalType)の処理終了	B

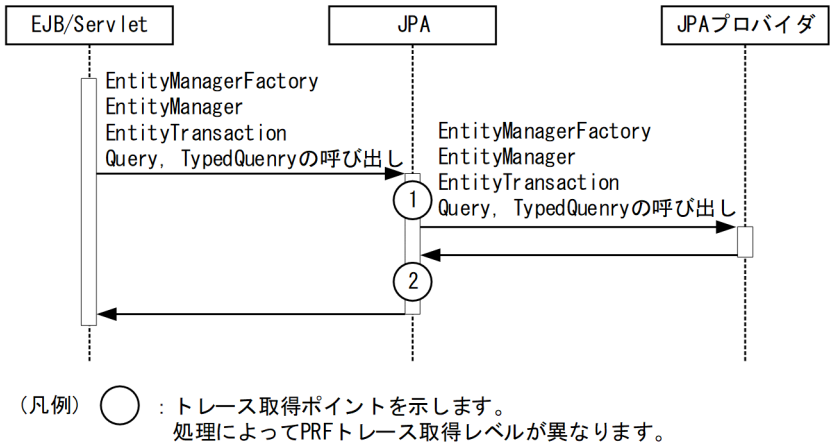
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1EA	1	TypedQuery#setParameter(int position, Date value, TemporalType temporalType)の処理開始	B
0xD1EB	2	TypedQuery#setParameter(int position, Date value, TemporalType temporalType)の処理終了	B
0xD1EC	1	TypedQuery#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD1ED	2	TypedQuery#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD1EE	1	TypedQuery#setLockMode(LockModeType lockMode)の処理開始	B
0xD1EF	2	TypedQuery#setLockMode(LockModeType lockMode)の処理終了	B

（凡例）A：標準　B：詳細

注※　図 8-68 の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-68　アプリケーション管理の永続化コンテキストを利用した場合のトレース取得ポイント



(2)　取得できるトレース情報

アプリケーション管理の永続化コンテキストを利用した場合に取得できるトレース情報を次の表に示します。

表 8-110　アプリケーション管理の永続化コンテキストを利用した場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
1	0xD100	A	−	−	−
	0xD102	A	−	−	−

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
	0xD104	A	SynchronizationType の値	−	−
	0xD106	A	SynchronizationType の値	−	−
	0xD108	B	−	−	−
	0xD10A	B	−	−	−
	0xD10C	B	−	−	−
	0xD10E	B	−	−	−
	0xD110	B	−	−	−
	0xD112	B	name の値	−	−
	0xD114	B	cls のクラス名	−	−
	0xD116	B	graphName の値	−	−
	0xD118	B	−	−	−
	0xD11A	A	−	−	−
	0xD11C	B	entity のクラス名	−	−
	0xD11E	B	entityClass のクラス名	−	−
	0xD120	B	entityClass のクラス名	−	−
	0xD122	B	entityClass のクラス名	−	−
	0xD124	B	entity のクラス名	−	−
	0xD126	B	entity のクラス名	−	−
	0xD128	B	entity のクラス名	lockMode の値	−
	0xD12A	B	entity のクラス名	lockMode の値	−
	0xD12C	B	entity のクラス名	−	−
	0xD12E	B	entity のクラス名	−	−
	0xD130	B	entity のクラス名	−	−
	0xD132	B	entity のクラス名	−	−
	0xD134	B	entity のクラス名	−	−
	0xD136	B	entity のクラス名	−	−
	0xD138	B	entity のクラス名	−	−
	0xD13A	B	−	−	−
	0xD13C	B	−	−	−
	0xD13E	B	entity のクラス名	−	−

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
	0xD140	B	rootTyoe のクラス名	−	−
	0xD142	B	graphName の値	−	−
	0xD144	A	−	−	−
	0xD146	A	−	−	−
	0xD148	A	−	−	−
	0xD14A	A	−	−	−
	0xD14C	A	−	−	−
	0xD14E	A	name の値	−	−
	0xD150	A	name の値	−	−
	0xD152	A	−	−	−
	0xD154	A	resultClass のクラス名	−	−
	0xD156	A	resultSetMapping	−	−
	0xD158	A	name の値	−	−
	0xD15A	A	procedureName の値	−	−
	0xD15C	A	procedureName の値	−	−
	0xD15E	A	procedureName の値	−	−
	0xD160	B	flushMode の値	−	−
	0xD162	B	−	−	−
	0xD164	B	−	−	−
	0xD166	B	−	−	−
	0xD168	B	−	−	−
	0xD16A	B	cls のクラス名	−	−
	0xD16C	B	−	−	−
	0xD16E	B	graphName の値	−	−
	0xD170	B	entityClass のクラス名	−	−
	0xD172	B	−	−	−
	0xD174	B	entity のクラス名	−	−
	0xD176	B	−	−	−
	0xD178	B	propertyName の値	−	−
	0xD17A	B	−	−	−

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
	0xD17C	B	−	−	−
	0xD17E	B	−	−	−
	0xD180	A	−	−	−
	0xD182	A	−	−	−
	0xD184	A	−	−	−
	0xD186	A	−	−	−
	0xD188	B	−	−	−
	0xD18A	B	−	−	−
	0xD18C	B	−	−	−
	0xD18E	B	−	−	−
	0xD190	B	−	−	−
	0xD192	B	−	−	−
	0xD194	B	flushMode の値	−	−
	0xD196	B	−	−	−
	0xD198	B	startPosition の値	−	−
	0xD19A	B	−	−	−
	0xD19C	B	maxResult の値	−	−
	0xD19E	B	−	−	−
	0xD1A0	B	hintName	value のクラス名	−
	0xD1A2	B	−	−	−
	0xD1A4	B	position の値	−	−
	0xD1A6	B	position の値	−	−
	0xD1A8	B	position の値	−	−
	0xD1AA	B	name の値	−	−
	0xD1AC	B	name の値	−	−
	0xD1AE	B	name の値	−	−
	0xD1B0	B	−	−	−
	0xD1B2	B	−	−	−
	0xD1B4	B	−	−	−
	0xD1B6	B	−	−	−

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
	0xD1B8	B	name の値	−	−
	0xD1BA	B	name の値	−	−
	0xD1BC	B	position の値	−	−
	0xD1BE	B	position の値	−	−
	0xD1C0	B	−	−	−
	0xD1C2	B	−	−	−
	0xD1C4	B	name の値	−	−
	0xD1C6	B	position の値	−	−
	0xD1C8	B	−	−	−
	0xD1CA	B	−	−	−
	0xD1CC	B	−	−	−
	0xD1CE	B	−	−	−
	0xD1D0	B	−	−	−
	0xD1D2	B	−	−	−
	0xD1D4	B	maxResult の値	−	−
	0xD1D6	B	startPosition の値	−	−
	0xD1D8	B	hintName	value のクラス名	−
	0xD1DA	B	−	−	−
	0xD1DC	B	−	−	−
	0xD1DE	B	−	−	−
	0xD1E0	B	name の値	−	−
	0xD1E2	B	name の値	−	−
	0xD1E4	B	name の値	−	−
	0xD1E6	B	position の値	−	−
	0xD1E8	B	position の値	−	−
	0xD1EA	B	position の値	−	−
	0xD1EC	B	flushMode の値	−	−
	0xD1EE	B	−	−	−
2	0xD101	A	−	−	• 正常時
	0xD103	A	−	−	

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
	0xD105	A	−	−	<入り口時刻> • 例外発生時 <入り口時刻><例外名>
	0xD107	A	−	−	
	0xD109	B	−	−	
	0xD10B	B	−	−	
	0xD10D	B	−	−	
	0xD10F	B	−	−	
	0xD111	B	−	−	
	0xD113	B	−	−	
	0xD115	B	−	−	
	0xD117	B	−	−	
	0xD119	B	−	−	
	0xD11B	A	−	−	
	0xD11D	B	−	−	
	0xD11F	B	−	−	
	0xD121	B	−	−	
	0xD123	B	−	−	
	0xD125	B	−	−	
	0xD127	B	−	−	
	0xD129	B	−	−	
	0xD12B	B	−	−	
	0xD12D	B	−	−	
	0xD12F	B	−	−	
	0xD131	B	−	−	
	0xD133	B	−	−	
	0xD135	B	−	−	
	0xD137	B	−	−	
	0xD139	B	−	−	
	0xD13B	B	−	−	
	0xD13D	B	−	−	
	0xD13F	B	−	−	

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
	0xD141	B	−	−	
	0xD143	B	−	−	
	0xD145	A	−	−	
	0xD147	A	−	−	
	0xD149	A	−	−	
	0xD14B	A	−	−	
	0xD14D	A	−	−	
	0xD14F	A	−	−	
	0xD151	A	−	−	
	0xD153	A	−	−	
	0xD155	A	−	−	
	0xD157	A	−	−	
	0xD159	A	−	−	
	0xD15B	A	−	−	
	0xD15D	A	−	−	
	0xD15F	A	−	−	
	0xD161	B	−	−	
	0xD163	B	−	−	
	0xD165	B	−	−	
	0xD167	B	−	−	
	0xD169	B	−	−	
	0xD16B	B	−	−	
	0xD16D	B	−	−	
	0xD16F	B	−	−	
	0xD171	B	−	−	
	0xD173	B	−	−	
	0xD175	B	−	−	
	0xD177	B	−	−	
	0xD179	B	−	−	
	0xD17B	B	−	−	

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
	0xD17D	B	−	−	
	0xD17F	B	−	−	
	0xD181	A	−	−	
	0xD183	A	−	−	
	0xD185	A	−	−	
	0xD187	A	−	−	
	0xD189	B	−	−	
	0xD18B	B	−	−	
	0xD18D	B	−	−	
	0xD18F	B	−	−	
	0xD191	B	−	−	
	0xD193	B	−	−	
	0xD195	B	−	−	
	0xD197	B	−	−	
	0xD199	B	−	−	
	0xD19B	B	−	−	
	0xD19D	B	−	−	
	0xD19F	B	−	−	
	0xD1A1	B	−	−	
	0xD1A3	B	−	−	
	0xD1A5	B	−	−	
	0xD1A7	B	−	−	
	0xD1A9	B	−	−	
	0xD1AB	B	−	−	
	0xD1AD	B	−	−	
	0xD1AF	B	−	−	
	0xD1B1	B	−	−	
	0xD1B3	B	−	−	
	0xD1B5	B	−	−	
	0xD1B7	B	−	−	

図中の番号※	イベント ID	レベル	取得できる情報		
			インターフェース名	オペレーション名	オプション
	0xD1B9	B	－	－	
	0xD1BB	B	－	－	
	0xD1BD	B	－	－	
	0xD1BF	B	－	－	
	0xD1C1	B	－	－	
	0xD1C3	B	－	－	
	0xD1C5	B	－	－	
	0xD1C7	B	－	－	
	0xD1C9	B	－	－	
	0xD1CB	B	－	－	
	0xD1CD	B	－	－	
	0xD1CF	B	－	－	
	0xD1D1	B	－	－	
	0xD1D3	B	－	－	
	0xD1D5	B	－	－	
	0xD1D7	B	－	－	
	0xD1D9	B	－	－	
	0xD1DB	B	－	－	
	0xD1DD	B	－	－	
	0xD1DF	B	－	－	
	0xD1E1	B	－	－	
	0xD1E3	B	－	－	
	0xD1E5	B	－	－	
	0xD1E7	B	－	－	
	0xD1E9	B	－	－	
	0xD1EB	B	－	－	
	0xD1ED	B	－	－	
	0xD1EF	B	－	－	

（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-68 中の番号と対応しています。

8.18.2 コンテナ管理の永続化コンテキストを利用した場合のトレース取得ポイントと取得できるトレース情報

コンテナ管理の永続化コンテキストを利用した場合のトレース取得ポイントと取得できるトレース情報について説明します。ここでは，次の四つの場合に分けて説明します。

- トランザクションスコープの永続化コンテキストをトランザクション内で利用した場合
- トランザクションスコープの永続化コンテキストに関連づいているエンティティマネージャをトランザクション外で利用した場合
- トランザクション外で生成された Query をトランザクション外で利用した場合
- 拡張永続化コンテキストを利用した場合

(1) トレース取得ポイントと PRF トレース取得レベル

(a) トランザクションスコープの永続化コンテキストをトランザクション内で利用した場合

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-111 トランザクションスコープの永続化コンテキストをトランザクション内で利用した場合のトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD11C	1，5	EntityManager#find(Class<T> entityClass, Object primaryKey)の処理開始	B
0xD11D	4，6	EntityManager#find(Class<T> entityClass, Object primaryKey)の処理終了	B
0xD11E	1，5	EntityManager#find(Class<T> entityClass, Object primaryKey, Map<String, Object> properties)の処理開始	B
0xD11F	4，6	EntityManager#find(Class<T> entityClass, Object primaryKey, Map<String, Object> properties)の処理終了	B
0xD120	1，5	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode)の処理開始	B
0xD121	4，6	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode)の処理終了	B
0xD122	1，5	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode, Map<String, Object> properties)の処理開始	B
0xD123	4，6	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode, Map<String, Object> properties)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD124	1，5	EntityManager#getReference(Class<T> entityClass, Object primaryKey)の処理開始	B
0xD125	4，6	EntityManager#getReference(Class<T> entityClass, Object primaryKey)の処理終了	B
0xD126	1，5	EntityManager#contains(Object entity)の処理開始	B
0xD127	4，6	EntityManager#contains(Object entity)の処理終了	B
0xD128	1，5	EntityManager#lock(Object entity, LockModeType lockMode)の処理開始	B
0xD129	4，6	EntityManager#lock(Object entity, LockModeType lockMode)の処理終了	B
0xD12A	1，5	EntityManager#lock(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理開始	B
0xD12B	4，6	EntityManager#lock(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理終了	B
0xD12C	1，5	EntityManager#merge(T entity)の処理開始	B
0xD12D	4，6	EntityManager#merge(T entity)の処理終了	B
0xD12E	1，5	EntityManager#persist(Object entity)の処理開始	B
0xD12F	4，6	EntityManager#persist(Object entity)の処理終了	B
0xD130	1，5	EntityManager#refresh(Object entity)の処理開始	B
0xD131	4，6	EntityManager#refresh(Object entity)の処理終了	B
0xD132	1，5	EntityManager#refresh(Object entity, Map<String,Object> properties)の処理開始	B
0xD133	4，6	EntityManager#refresh(Object entity, Map<String,Object> properties)の処理終了	B
0xD134	1，5	EntityManager#refresh(Object entity, LockModeType lockMode)の処理開始	B
0xD135	4，6	EntityManager#refresh(Object entity, LockModeType lockMode)の処理終了	B
0xD136	1，5	EntityManager#refresh(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理開始	B
0xD137	4，6	EntityManager#refresh(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理終了	B
0xD138	1，5	EntityManager#remove(Object entity)の処理開始	B
0xD139	4，6	EntityManager#remove(Object entity)の処理終了	B
0xD13A	1，5	EntityManager#clear()の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD13B	4，6	EntityManager#clear()の処理終了	B
0xD13C	1，5	EntityManager#flush()の処理開始	B
0xD13D	4，6	EntityManager#flush()の処理終了	B
0xD13E	1，5	EntityManager#detach(Object entity)の処理開始	B
0xD13F	4，6	EntityManager#detach(Object entity)の処理終了	B
0xD140	1，5	EntityManager#createEntityGraph(Class<T> rootType)の処理開始	B
0xD141	4，6	EntityManager#createEntityGraph(Class<T> rootType)の処理終了	B
0xD142	1，5	EntityManager#createEntityGraph(String graphName)の処理開始	B
0xD143	4，6	EntityManager#createEntityGraph(String graphName)の処理終了	B
0xD144	1，5	EntityManager#createQuery(String qlString)の処理開始	A
0xD145	4，6	EntityManager#createQuery(String qlString)の処理終了	A
0xD146	1，5	EntityManager#createQuery(CriteriaQuery<T> criteriaQuery)の処理開始	A
0xD147	4，6	EntityManager#createQuery(CriteriaQuery<T> criteriaQuery)の処理終了	A
0xD148	1，5	EntityManager#createQuery(CriteriaUpdate updateQuery)の処理開始	A
0xD149	4，6	EntityManager#createQuery(CriteriaUpdate updateQuery)の処理終了	A
0xD14A	1，5	EntityManager#createQuery(CriteriaDelete deleteQuery)の処理開始	A
0xD14B	4，6	EntityManager#createQuery(CriteriaDelete deleteQuery)の処理終了	A
0xD14C	1，5	EntityManager#createQuery(String qlString, Class<T> resultClass)の処理開始	A
0xD14D	4，6	EntityManager#createQuery(String qlString, Class<T> resultClass)の処理終了	A
0xD14E	1，5	EntityManager#createNamedQuery(String name)の処理開始	A
0xD14F	4，6	EntityManager#createNamedQuery(String name)の処理終了	A
0xD150	1，5	EntityManager#createNamedQuery(String name, Class<T> resultClass)の処理開始	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD151	4，6	EntityManager#createNamedQuery(String name, Class<T> resultClass)の処理終了	A
0xD152	1，5	EntityManager#createNativeQuery(String sqlString)の処理開始	A
0xD153	4，6	EntityManager#createNativeQuery(String sqlString)の処理終了	A
0xD154	1，5	EntityManager#createNativeQuery(String sqlString, Class resultClass)の処理開始	A
0xD155	4，6	EntityManager#createNativeQuery(String sqlString, Class resultClass)の処理終了	A
0xD156	1，5	EntityManager#createNativeQuery(String sqlString, String resultSetMapping)の処理開始	A
0xD157	4，6	EntityManager#createNativeQuery(String sqlString, String resultSetMapping)の処理終了	A
0xD158	1，5	EntityManager#createNamedStoredProcedureQuery(String name)の処理開始	A
0xD159	4，6	EntityManager#createNamedStoredProcedureQuery(String name)の処理終了	A
0xD15A	1，5	EntityManager#createStoredProcedureQuery(String procedureName)の処理開始	A
0xD15B	4，6	EntityManager#createStoredProcedureQuery(String procedureName)の処理終了	A
0xD15C	1，5	EntityManager#createStoredProcedureQuery(String procedureName, Class... resultClasses)の処理開始	A
0xD15D	4，6	EntityManager#createStoredProcedureQuery(String procedureName, Class... resultClasses)の処理終了	A
0xD15E	1，5	EntityManager#createStoredProcedureQuery(String procedureName, String... resultSetMappings)の処理開始	A
0xD15F	4，6	EntityManager#createStoredProcedureQuery(String procedureName, String... resultSetMappings)の処理終了	A
0xD160	1，5	EntityManager#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD161	4，6	EntityManager#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD162	1，5	EntityManager#getFlushMode()の処理開始	B
0xD163	4，6	EntityManager#getFlushMode()の処理終了	B
0xD164	1，5	EntityManager#joinTransaction()の処理開始	B
0xD165	4，6	EntityManager#joinTransaction()の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD166	1，5	EntityManager#getTransaction()の処理開始	B
0xD167	4，6	EntityManager#getTransaction()の処理終了	B
0xD168	1，5	EntityManager#getDelegate()の処理開始	B
0xD169	4，6	EntityManager#getDelegate()の処理終了	B
0xD16A	1，5	EntityManager#unwrap(Class<T> cls)の処理開始	B
0xD16B	4，6	EntityManager#unwrap(Class<T> cls)の処理終了	B
0xD16C	1，5	EntityManager#getCriteriaBuilder()の処理開始	B
0xD16D	4，6	EntityManager#getCriteriaBuilder()の処理終了	B
0xD16E	1，5	EntityManager#getEntityGraph(String graphName)の処理開始	B
0xD16F	4，6	EntityManager#getEntityGraph(String graphName)の処理終了	B
0xD170	1，5	EntityManager#getEntityGraphs(Class<T> entityClass)の処理開始	B
0xD171	4，6	EntityManager#getEntityGraphs(Class<T> entityClass)の処理終了	B
0xD172	1，5	EntityManager#getEntityManagerFactory()の処理開始	B
0xD173	4，6	EntityManager#getEntityManagerFactory()の処理終了	B
0xD174	1，5	EntityManager#getLockMode(Object entity)の処理開始	B
0xD175	4，6	EntityManager#getLockMode(Object entity)の処理終了	B
0xD176	1，5	EntityManager#getMetamodel()の処理開始	B
0xD177	4，6	EntityManager#getMetamodel()の処理終了	B
0xD178	1，5	EntityManager#setProperty(String propertyName, Object value)の処理開始	B
0xD179	4，6	EntityManager#setProperty(String propertyName, Object value)の処理終了	B
0xD17A	1，5	EntityManager#getProperties()の処理開始	B
0xD17B	4，6	EntityManager#getProperties()の処理終了	B
0xD17C	1，5	EntityManager#isJoinedToTransaction()の処理開始	B
0xD17D	4，6	EntityManager#isJoinedToTransaction()の処理終了	B
0xD17E	1，5	EntityManager#isOpen()の処理開始	B
0xD17F	4，6	EntityManager#isOpen()の処理終了	B
0xD180	1，5	EntityManager#close()の処理開始	A
0xD181	4，6	EntityManager#close()の処理終了	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD18E	1，5	Query#executeUpdate()の処理開始	B
0xD18F	4，6	Query#executeUpdate()の処理終了	B
0xD190	1，5	Query#getResultList()の処理開始	B
0xD191	4，6	Query#getResultList()の処理終了	B
0xD192	1，5	Query#getSingleResult()の処理開始	B
0xD193	4，6	Query#getSingleResult()の処理終了	B
0xD194	1，5	Query#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD195	4，6	Query#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD196	1，5	Query#getFlushMode()の処理開始	B
0xD197	4，6	Query#getFlushMode()の処理終了	B
0xD198	1，5	Query#setFirstResult(int startPosition)の処理開始	B
0xD199	4，6	Query#setFirstResult(int startPosition)の処理終了	B
0xD19A	1，5	Query#getFirstResult()の処理開始	B
0xD19B	4，6	Query#getFirstResult()の処理終了	B
0xD19C	1，5	Query#setMaxResults(int maxResult)の処理開始	B
0xD19D	4，6	Query#setMaxResults(int maxResult)の処理終了	B
0xD19E	1，5	Query#getMaxResults()の処理開始	B
0xD19F	4，6	Query#getMaxResults()の処理終了	B
0xD1A0	1，5	Query#setHint(String hintName, Object value)の処理開始	B
0xD1A1	4，6	Query#setHint(String hintName, Object value)の処理終了	B
0xD1A2	1，5	Query#getHints()の処理開始	B
0xD1A3	4，6	Query#getHints()の処理終了	B
0xD1A4	1，5	Query#setParameter(int position, Calendar value, TemporalType temporalType)の処理開始	B
0xD1A5	4，6	Query#setParameter(int position, Calendar value, TemporalType temporalType)の処理終了	B
0xD1A6	1，5	Query#setParameter(int position, Date value, TemporalType temporalType)の処理開始	B
0xD1A7	4，6	Query#setParameter(int position, Date value, TemporalType temporalType)の処理終了	B
0xD1A8	1，5	Query#setParameter(int position, Object value)の処理開始	B
0xD1A9	4，6	Query#setParameter(int position, Object value)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1AA	1，5	Query#setParameter(String name, Calendar value, TemporalType temporalType)の処理開始	B
0xD1AB	4，6	Query#setParameter(String name, Calendar value, TemporalType temporalType)の処理終了	B
0xD1AC	1，5	Query#setParameter(String name, Date value, TemporalType temporalType)の処理開始	B
0xD1AD	4，6	Query#setParameter(String name, Date value, TemporalType temporalType)の処理終了	B
0xD1AE	1，5	Query#setParameter(String name, Object value)の処理開始	B
0xD1AF	4，6	Query#setParameter(String name, Object value)の処理終了	B
0xD1B0	1，5	Query#setParameter(Parameter<T> param, T value)の処理開始	B
0xD1B1	4，6	Query#setParameter(Parameter<T> param, T value)の処理終了	B
0xD1B2	1，5	Query#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理開始	B
0xD1B3	4，6	Query#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理終了	B
0xD1B4	1，5	Query#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理開始	B
0xD1B5	4，6	Query#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理終了	B
0xD1B6	1，5	Query#getParameters()の処理開始	B
0xD1B7	4，6	Query#getParameters()の処理終了	B
0xD1B8	1，5	Query#getParameter(String name)の処理開始	B
0xD1B9	4，6	Query#getParameter(String name)の処理終了	B
0xD1BA	1，5	Query#getParameter(String name, Class<T> type)の処理開始	B
0xD1BB	4，6	Query#getParameter(String name, Class<T> type)の処理終了	B
0xD1BC	1，5	Query#getParameter(int position)の処理開始	B
0xD1BD	4，6	Query#getParameter(int position)の処理終了	B
0xD1BE	1，5	Query#getParameter(int position, Class<T> type)の処理開始	B
0xD1BF	4，6	Query#getParameter(int position, Class<T> type)の処理終了	B
0xD1C0	1，5	Query#isBound(Parameter<?> param)の処理開始	B
0xD1C1	4，6	Query#isBound(Parameter<?> param)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1C2	1，5	Query#getParameterValue(Parameter<T> param)の処理開始	B
0xD1C3	4，6	Query#getParameterValue(Parameter<T> param)の処理終了	B
0xD1C4	1，5	Query#getParameterValue(String name)の処理開始	B
0xD1C5	4，6	Query#getParameterValue(String name)の処理終了	B
0xD1C6	1，5	Query#getParameterValue(int position)の処理開始	B
0xD1C7	4，6	Query#getParameterValue(int position)の処理終了	B
0xD1C8	1，5	Query#setLockMode(LockModeType lockMode)の処理開始	B
0xD1C9	4，6	Query#setLockMode(LockModeType lockMode)の処理終了	B
0xD1CA	1，5	Query#getLockMode()の処理開始	B
0xD1CB	4，6	Query#getLockMode()の処理終了	B
0xD1CC	1，5	Query#unwrap(Class<T> cls)の処理開始	B
0xD1CD	4，6	Query#unwrap(Class<T> cls)の処理終了	B
0xD1CE	1，5	TypedQuery#executeUpdate()の処理開始	B
0xD1CF	4，6	TypedQuery#executeUpdate()の処理終了	B
0xD1D0	1，5	TypedQuery#getResultList()の処理開始	B
0xD1D1	4，6	TypedQuery#getResultList()の処理終了	B
0xD1D2	1，5	TypedQuery#getSingleResult()の処理開始	B
0xD1D3	4，6	TypedQuery#getSingleResult()の処理終了	B
0xD1D4	1，5	TypedQuery#setMaxResults(int maxResult)の処理開始	B
0xD1D5	4，6	TypedQuery#setMaxResults(int maxResult)の処理終了	B
0xD1D6	1，5	TypedQuery#setFirstResult(int startPosition)の処理開始	B
0xD1D7	4，6	TypedQuery#setFirstResult(int startPosition)の処理終了	B
0xD1D8	1，5	TypedQuery#setHint(String hintName, Object value)の処理開始	B
0xD1D9	4，6	TypedQuery#setHint(String hintName, Object value)の処理終了	B
0xD1DA	1，5	TypedQuery#setParameter(Parameter<T> param, T value)の処理開始	B
0xD1DB	4，6	TypedQuery#setParameter(Parameter<T> param, T value)の処理終了	B
0xD1DC	1，5	TypedQuery#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1DD	4，6	TypedQuery#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理終了	B
0xD1DE	1，5	TypedQuery#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理開始	B
0xD1DF	4，6	TypedQuery#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理終了	B
0xD1E0	1，5	TypedQuery#setParameter(String name, Object value)の処理開始	B
0xD1E1	4，6	TypedQuery#setParameter(String name, Object value)の処理終了	B
0xD1E2	1，5	TypedQuery#setParameter(String name, Calendar value, TemporalType temporalType)の処理開始	B
0xD1E3	4，6	TypedQuery#setParameter(String name, Calendar value, TemporalType temporalType)の処理終了	B
0xD1E4	1，5	TypedQuery#setParameter(String name, Date value, TemporalType temporalType)の処理開始	B
0xD1E5	4，6	TypedQuery#setParameter(String name, Date value, TemporalType temporalType)の処理終了	B
0xD1E6	1，5	TypedQuery#setParameter(int position, Object value)の処理開始	B
0xD1E7	4，6	TypedQuery#setParameter(int position, Object value)の処理終了	B
0xD1E8	1，5	TypedQuery#setParameter(int position, Calendar value, TemporalType temporalType)の処理開始	B
0xD1E9	4，6	TypedQuery#setParameter(int position, Calendar value, TemporalType temporalType)の処理終了	B
0xD1EA	1，5	TypedQuery#setParameter(int position, Date value, TemporalType temporalType)の処理開始	B
0xD1EB	4，6	TypedQuery#setParameter(int position, Date value, TemporalType temporalType)の処理終了	B
0xD1EC	1，5	TypedQuery#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD1ED	4，6	TypedQuery#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD1EE	1，5	TypedQuery#setLockMode(LockModeType lockMode)の処理開始	B
0xD1EF	4，6	TypedQuery#setLockMode(LockModeType lockMode)の処理終了	B

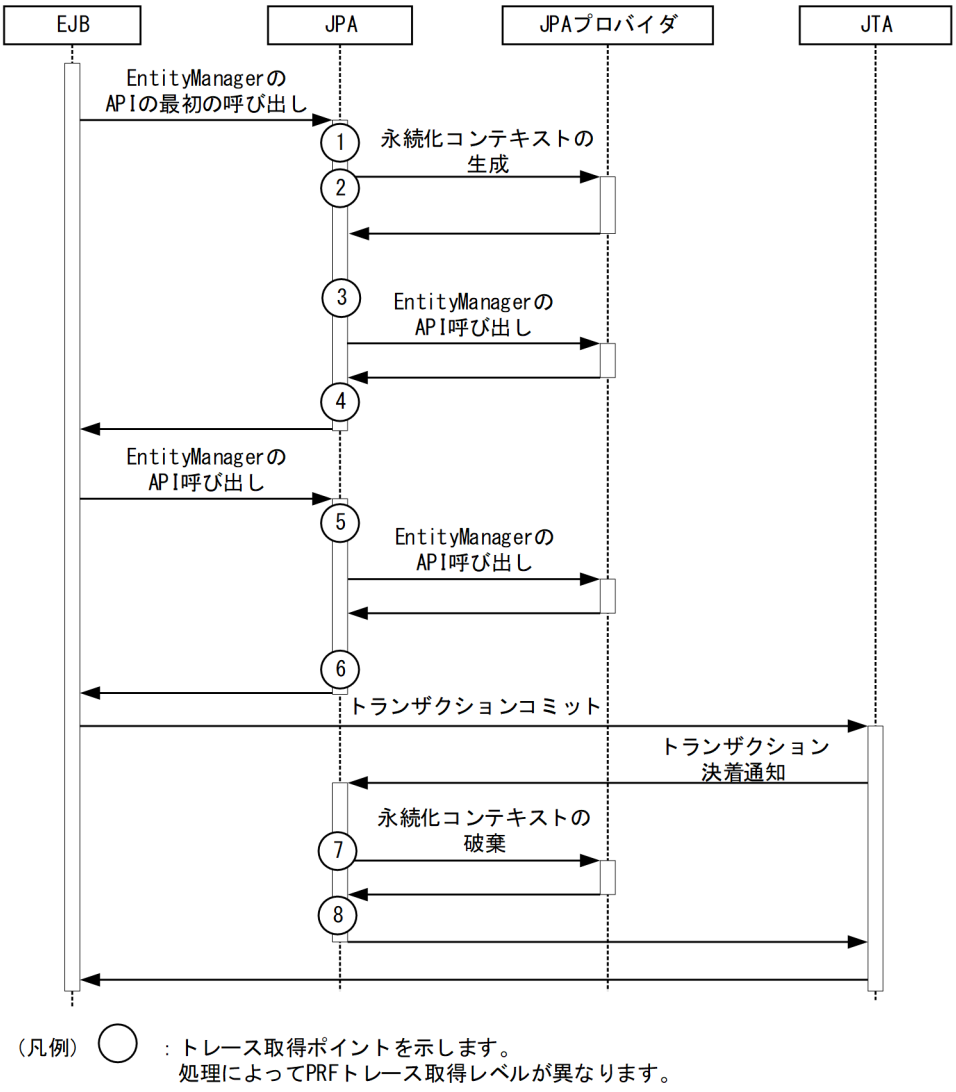
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1F0	2	トランザクションスコープの永続化コンテキストを利用した際の永続化コンテキストの生成処理開始	A
0xD1F1	3	トランザクションスコープの永続化コンテキストを利用した際の永続化コンテキストの生成処理終了	A
0xD1F2	7	トランザクションスコープの永続化コンテキストを利用した際の永続化コンテキストの破棄処理開始	A
0xD1F3	8	トランザクションスコープの永続化コンテキストを利用した際の永続化コンテキストの破棄処理終了	A

（凡例）A：標準　B：詳細

注※　図 8-69 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-69　トランザクションスコープの永続化コンテキストをトランザクション内で利用した場合のトレース取得ポイント



（凡例）◯　：トレース取得ポイントを示します。
処理によってPRFトレース取得レベルが異なります。

(b) トランザクションスコープの永続化コンテキストに関連づいているエンティティマネージャをトランザクション外で利用した場合

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-112 トランザクションスコープの永続化コンテキストに関連づいているエンティティマネージャをトランザクション外で利用した場合のトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD11C	1	EntityManager#find(Class<T> entityClass, Object primaryKey)の処理開始	B
0xD11D	4	EntityManager#find(Class<T> entityClass, Object primaryKey)の処理終了	B
0xD11E	1	EntityManager#find(Class<T> entityClass, Object primaryKey, Map<String, Object> properties)の処理開始	B
0xD11F	4	EntityManager#find(Class<T> entityClass, Object primaryKey, Map<String, Object> properties)の処理終了	B
0xD120	1	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode)の処理開始	B
0xD121	4	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode)の処理終了	B
0xD122	1	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode, Map<String, Object> properties)の処理開始	B
0xD123	4	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode, Map<String, Object> properties)の処理終了	B
0xD124	1	EntityManager#getReference(Class<T> entityClass, Object primaryKey)の処理開始	B
0xD125	4	EntityManager#getReference(Class<T> entityClass, Object primaryKey)の処理終了	B
0xD126	1	EntityManager#contains(Object entity)の処理開始	B
0xD127	4	EntityManager#contains(Object entity)の処理終了	B
0xD128	1	EntityManager#lock(Object entity, LockModeType lockMode)の処理開始	B
0xD129	4	EntityManager#lock(Object entity, LockModeType lockMode)の処理終了	B
0xD12A	1	EntityManager#lock(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理開始	B
0xD12B	4	EntityManager#lock(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD12C	1	EntityManager#merge(T entity)の処理開始	B
0xD12D	4	EntityManager#merge(T entity)の処理終了	B
0xD12E	1	EntityManager#persist(Object entity)の処理開始	B
0xD12F	4	EntityManager#persist(Object entity)の処理終了	B
0xD130	1	EntityManager#refresh(Object entity)の処理開始	B
0xD131	4	EntityManager#refresh(Object entity)の処理終了	B
0xD132	1	EntityManager#refresh(Object entity, Map<String,Object> properties)の処理開始	B
0xD133	4	EntityManager#refresh(Object entity, Map<String,Object> properties)の処理終了	B
0xD134	1	EntityManager#refresh(Object entity, LockModeType lockMode)の処理開始	B
0xD135	4	EntityManager#refresh(Object entity, LockModeType lockMode)の処理終了	B
0xD136	1	EntityManager#refresh(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理開始	B
0xD137	4	EntityManager#refresh(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理終了	B
0xD138	1	EntityManager#remove(Object entity)の処理開始	B
0xD139	4	EntityManager#remove(Object entity)の処理終了	B
0xD13A	1	EntityManager#clear()の処理開始	B
0xD13B	4	EntityManager#clear()の処理終了	B
0xD13C	1	EntityManager#flush()の処理開始	B
0xD13D	4	EntityManager#flush()の処理終了	B
0xD13E	1	EntityManager#detach(Object entity)の処理開始	B
0xD13F	4	EntityManager#detach(Object entity)の処理終了	B
0xD140	1	EntityManager#createEntityGraph(Class<T> rootType)の処理開始	B
0xD141	4	EntityManager#createEntityGraph(Class<T> rootType)の処理終了	B
0xD142	1	EntityManager#createEntityGraph(String graphName)の処理開始	B
0xD143	4	EntityManager#createEntityGraph(String graphName)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD160	1	EntityManager#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD161	4	EntityManager#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD162	1	EntityManager#getFlushMode()の処理開始	B
0xD163	4	EntityManager#getFlushMode()の処理終了	B
0xD164	1	EntityManager#joinTransaction()の処理開始	B
0xD165	4	EntityManager#joinTransaction()の処理終了	B
0xD166	1	EntityManager#getTransaction()の処理開始	B
0xD167	4	EntityManager#getTransaction()の処理終了	B
0xD168	1	EntityManager#getDelegate()の処理開始	B
0xD169	4	EntityManager#getDelegate()の処理終了	B
0xD16A	1	EntityManager#unwrap(Class<T> cls)の処理開始	B
0xD16B	4	EntityManager#unwrap(Class<T> cls)の処理終了	B
0xD16C	1	EntityManager#getCriteriaBuilder()の処理開始	B
0xD16D	4	EntityManager#getCriteriaBuilder()の処理終了	B
0xD16E	1	EntityManager#getEntityGraph(String graphName)の処理開始	B
0xD16F	4	EntityManager#getEntityGraph(String graphName)の処理終了	B
0xD170	1	EntityManager#getEntityGraphs(Class<T> entityClass)の処理開始	B
0xD171	4	EntityManager#getEntityGraphs(Class<T> entityClass)の処理終了	B
0xD172	1	EntityManager#getEntityManagerFactory()の処理開始	B
0xD173	4	EntityManager#getEntityManagerFactory()の処理終了	B
0xD174	1	EntityManager#getLockMode(Object entity)の処理開始	B
0xD175	4	EntityManager#getLockMode(Object entity)の処理終了	B
0xD176	1	EntityManager#getMetamodel()の処理開始	B
0xD177	4	EntityManager#getMetamodel()の処理終了	B
0xD178	1	EntityManager#setProperty(String propertyName, Object value)の処理開始	B
0xD179	4	EntityManager#setProperty(String propertyName, Object value)の処理終了	B
0xD17A	1	EntityManager#getProperties()の処理開始	B

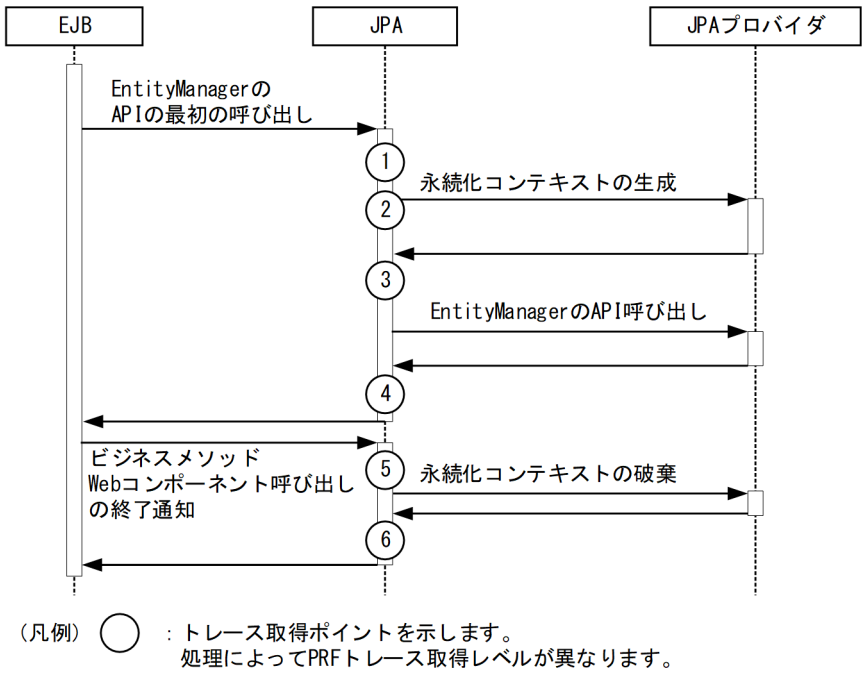
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD17B	4	EntityManager#getProperties()の処理終了	B
0xD17C	1	EntityManager#isJoinedToTransaction()の処理開始	B
0xD17D	4	EntityManager#isJoinedToTransaction()の処理終了	B
0xD17E	1	EntityManager#isOpen()の処理開始	B
0xD17F	4	EntityManager#isOpen()の処理終了	B
0xD180	1	EntityManager#close()の処理開始	A
0xD181	4	EntityManager#close()の処理終了	A
0xD1F4	2	トランザクションスコープの EntityManager の API を，トランザクションの存在しない状況で利用した際の永続化コンテキストの生成処理開始	A
0xD1F5	3	トランザクションスコープの EntityManager の API を，トランザクションの存在しない状況で利用した際の永続化コンテキストの生成処理終了	A
0xD1F6	5	トランザクションスコープの EntityManager の API を，トランザクションの存在しない状況で利用した際に生成した永続化コンテキストの破棄処理開始	A
0xD1F7	6	トランザクションスコープの EntityManager の API を，トランザクションの存在しない状況で利用した際に生成した永続化コンテキストの破棄処理終了	A

（凡例）A：標準　B：詳細

注※　図 8-70 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-70　トランザクションスコープの永続化コンテキストに関連づいているエンティティマネージャをトランザクション外で利用した場合のトレース取得ポイント



(c)　トランザクション外で生成された Query をトランザクション外で利用した場合

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-113　トランザクション外で生成された Query をトランザクション外で利用した場合のトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD144	1	EntityManager#createQuery(String qlString)の処理開始	A
0xD145	4	EntityManager#createQuery(String qlString)の処理終了	A
0xD146	1	EntityManager#createQuery(CriteriaQuery<T> criteriaQuery)の処理開始	A
0xD147	4	EntityManager#createQuery(CriteriaQuery<T> criteriaQuery)の処理終了	A
0xD148	1	EntityManager#createQuery(CriteriaUpdate updateQuery)の処理開始	A
0xD149	4	EntityManager#createQuery(CriteriaUpdate updateQuery)の処理終了	A
0xD14A	1	EntityManager#createQuery(CriteriaDelete deleteQuery)の処理開始	A
0xD14B	4	EntityManager#createQuery(CriteriaDelete deleteQuery)の処理終了	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD14C	1	EntityManager#createQuery(String qlString, Class<T> resultClass)の処理開始	A
0xD14D	4	EntityManager#createQuery(String qlString, Class<T> resultClass)の処理終了	A
0xD14E	1	EntityManager#createNamedQuery(String name)の処理開始	A
0xD14F	4	EntityManager#createNamedQuery(String name)の処理終了	A
0xD150	1	EntityManager#createNamedQuery(String name, Class<T> resultClass)の処理開始	A
0xD151	4	EntityManager#createNamedQuery(String name, Class<T> resultClass)の処理終了	A
0xD152	1	EntityManager#createNativeQuery(String sqlString)の処理開始	A
0xD153	4	EntityManager#createNativeQuery(String sqlString)の処理終了	A
0xD154	1	EntityManager#createNativeQuery(String sqlString, Class resultClass)の処理開始	A
0xD155	4	EntityManager#createNativeQuery(String sqlString, Class resultClass)の処理終了	A
0xD156	1	EntityManager#createNativeQuery(String sqlString, String resultSetMapping)の処理開始	A
0xD157	4	EntityManager#createNativeQuery(String sqlString, String resultSetMapping)の処理終了	A
0xD158	1	EntityManager#createNamedStoredProcedureQuery(String name)の処理開始	A
0xD159	4	EntityManager#createNamedStoredProcedureQuery(String name)の処理終了	A
0xD15A	1	EntityManager#createStoredProcedureQuery(String procedureName)の処理開始	A
0xD15B	4	EntityManager#createStoredProcedureQuery(String procedureName)の処理終了	A
0xD15C	1	EntityManager#createStoredProcedureQuery(String procedureName, Class... resultClasses)の処理開始	A
0xD15D	4	EntityManager#createStoredProcedureQuery(String procedureName, Class... resultClasses)の処理終了	A
0xD15E	1	EntityManager#createStoredProcedureQuery(String procedureName, String... resultSetMappings)の処理開始	A
0xD15F	4	EntityManager#createStoredProcedureQuery(String procedureName, String... resultSetMappings)の処理終了	A
0xD18E	1，5	Query#executeUpdate()の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD18F	4，6	Query#executeUpdate()の処理終了	B
0xD190	1，5	Query#getResultList()の処理開始	B
0xD191	4，6	Query#getResultList()の処理終了	B
0xD192	1，5	Query#getSingleResult()の処理開始	B
0xD193	4，6	Query#getSingleResult()の処理終了	B
0xD194	1，5	Query#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD195	4，6	Query#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD196	1，5	Query#getFlushMode()の処理開始	B
0xD197	4，6	Query#getFlushMode()の処理終了	B
0xD198	1，5	Query#setFirstResult(int startPosition)の処理開始	B
0xD199	4，6	Query#setFirstResult(int startPosition)の処理終了	B
0xD19A	1，5	Query#getFirstResult()の処理開始	B
0xD19B	4，6	Query#getFirstResult()の処理終了	B
0xD19C	1，5	Query#setMaxResults(int maxResult)の処理開始	B
0xD19D	4，6	Query#setMaxResults(int maxResult)の処理終了	B
0xD19E	1，5	Query#getMaxResults()の処理開始	B
0xD19F	4，6	Query#getMaxResults()の処理終了	B
0xD1A0	1，5	Query#setHint(String hintName, Object value)の処理開始	B
0xD1A1	4，6	Query#setHint(String hintName, Object value)の処理終了	B
0xD1A2	1，5	Query#getHints()の処理開始	B
0xD1A3	4，6	Query#getHints()の処理終了	B
0xD1A4	1，5	Query#setParameter(int position, Calendar value, TemporalType temporalType)の処理開始	B
0xD1A5	4，6	Query#setParameter(int position, Calendar value, TemporalType temporalType)の処理終了	B
0xD1A6	1，5	Query#setParameter(int position, Date value, TemporalType temporalType)の処理開始	B
0xD1A7	4，6	Query#setParameter(int position, Date value, TemporalType temporalType)の処理終了	B
0xD1A8	1，5	Query#setParameter(int position, Object value)の処理開始	B
0xD1A9	4，6	Query#setParameter(int position, Object value)の処理終了	B
0xD1AA	1，5	Query#setParameter(String name, Calendar value, TemporalType temporalType)の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1AB	4，6	Query#setParameter(String name, Calendar value, TemporalType temporalType)の処理終了	B
0xD1AC	1，5	Query#setParameter(String name, Date value, TemporalType temporalType)の処理開始	B
0xD1AD	4，6	Query#setParameter(String name, Date value, TemporalType temporalType)の処理終了	B
0xD1AE	1，5	Query#setParameter(String name, Object value)の処理開始	B
0xD1AF	4，6	Query#setParameter(String name, Object value)の処理終了	B
0xD1B0	1，5	Query#setParameter(Parameter<T> param, T value)の処理開始	B
0xD1B1	4，6	Query#setParameter(Parameter<T> param, T value)の処理終了	B
0xD1B2	1，5	Query#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理開始	B
0xD1B3	4，6	Query#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理終了	B
0xD1B4	1，5	Query#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理開始	B
0xD1B5	4，6	Query#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理終了	B
0xD1B6	1，5	Query#getParameters()の処理開始	B
0xD1B7	4，6	Query#getParameters()の処理終了	B
0xD1B8	1，5	Query#getParameter(String name)の処理開始	B
0xD1B9	4，6	Query#getParameter(String name)の処理終了	B
0xD1BA	1，5	Query#getParameter(String name, Class<T> type)の処理開始	B
0xD1BB	4，6	Query#getParameter(String name, Class<T> type)の処理終了	B
0xD1BC	1，5	Query#getParameter(int position)の処理開始	B
0xD1BD	4，6	Query#getParameter(int position)の処理終了	B
0xD1BE	1，5	Query#getParameter(int position, Class<T> type)の処理開始	B
0xD1BF	4，6	Query#getParameter(int position, Class<T> type)の処理終了	B
0xD1C0	1，5	Query#isBound(Parameter<?> param)の処理開始	B
0xD1C1	4，6	Query#isBound(Parameter<?> param)の処理終了	B
0xD1C2	1，5	Query#getParameterValue(Parameter<T> param)の処理開始	B
0xD1C3	4，6	Query#getParameterValue(Parameter<T> param)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1C4	1，5	Query#getParameterValue(String name)の処理開始	B
0xD1C5	4，6	Query#getParameterValue(String name)の処理終了	B
0xD1C6	1，5	Query#getParameterValue(int position)の処理開始	B
0xD1C7	4，6	Query#getParameterValue(int position)の処理終了	B
0xD1C8	1，5	Query#setLockMode(LockModeType lockMode)の処理開始	B
0xD1C9	4，6	Query#setLockMode(LockModeType lockMode)の処理終了	B
0xD1CA	1，5	Query#getLockMode()の処理開始	B
0xD1CB	4，6	Query#getLockMode()の処理終了	B
0xD1CC	1，5	Query#unwrap(Class<T> cls)の処理開始	B
0xD1CD	4，6	Query#unwrap(Class<T> cls)の処理終了	B
0xD1CE	1，5	TypedQuery#executeUpdate()の処理開始	B
0xD1CF	4，6	TypedQuery#executeUpdate()の処理終了	B
0xD1D0	1，5	TypedQuery#getResultList()の処理開始	B
0xD1D1	4，6	TypedQuery#getResultList()の処理終了	B
0xD1D2	1，5	TypedQuery#getSingleResult()の処理開始	B
0xD1D3	4，6	TypedQuery#getSingleResult()の処理終了	B
0xD1D4	1，5	TypedQuery#setMaxResults(int maxResult)の処理開始	B
0xD1D5	4，6	TypedQuery#setMaxResults(int maxResult)の処理終了	B
0xD1D6	1，5	TypedQuery#setFirstResult(int startPosition)の処理開始	B
0xD1D7	4，6	TypedQuery#setFirstResult(int startPosition)の処理終了	B
0xD1D8	1，5	TypedQuery#setHint(String hintName, Object value)の処理開始	B
0xD1D9	4，6	TypedQuery#setHint(String hintName, Object value)の処理終了	B
0xD1DA	1，5	TypedQuery#setParameter(Parameter<T> param, T value)の処理開始	B
0xD1DB	4，6	TypedQuery#setParameter(Parameter<T> param, T value)の処理終了	B
0xD1DC	1，5	TypedQuery#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理開始	B
0xD1DD	4，6	TypedQuery#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1DE	1，5	TypedQuery#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理開始	B
0xD1DF	4，6	TypedQuery#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理終了	B
0xD1E0	1，5	TypedQuery#setParameter(String name, Object value)の処理開始	B
0xD1E1	4，6	TypedQuery#setParameter(String name, Object value)の処理終了	B
0xD1E2	1，5	TypedQuery#setParameter(String name, Calendar value, TemporalType temporalType)の処理開始	B
0xD1E3	4，6	TypedQuery#setParameter(String name, Calendar value, TemporalType temporalType)の処理終了	B
0xD1E4	1，5	TypedQuery#setParameter(String name, Date value, TemporalType temporalType)の処理開始	B
0xD1E5	4，6	TypedQuery#setParameter(String name, Date value, TemporalType temporalType)の処理終了	B
0xD1E6	1，5	TypedQuery#setParameter(int position, Object value)の処理開始	B
0xD1E7	4，6	TypedQuery#setParameter(int position, Object value)の処理終了	B
0xD1E8	1，5	TypedQuery#setParameter(int position, Calendar value, TemporalType temporalType)の処理開始	B
0xD1E9	4，6	TypedQuery#setParameter(int position, Calendar value, TemporalType temporalType)の処理終了	B
0xD1EA	1，5	TypedQuery#setParameter(int position, Date value, TemporalType temporalType)の処理開始	B
0xD1EB	4，6	TypedQuery#setParameter(int position, Date value, TemporalType temporalType)の処理終了	B
0xD1EC	1，5	TypedQuery#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD1ED	4，6	TypedQuery#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD1EE	1，5	TypedQuery#setLockMode(LockModeType lockMode)の処理開始	B
0xD1EF	4，6	TypedQuery#setLockMode(LockModeType lockMode)の処理終了	B

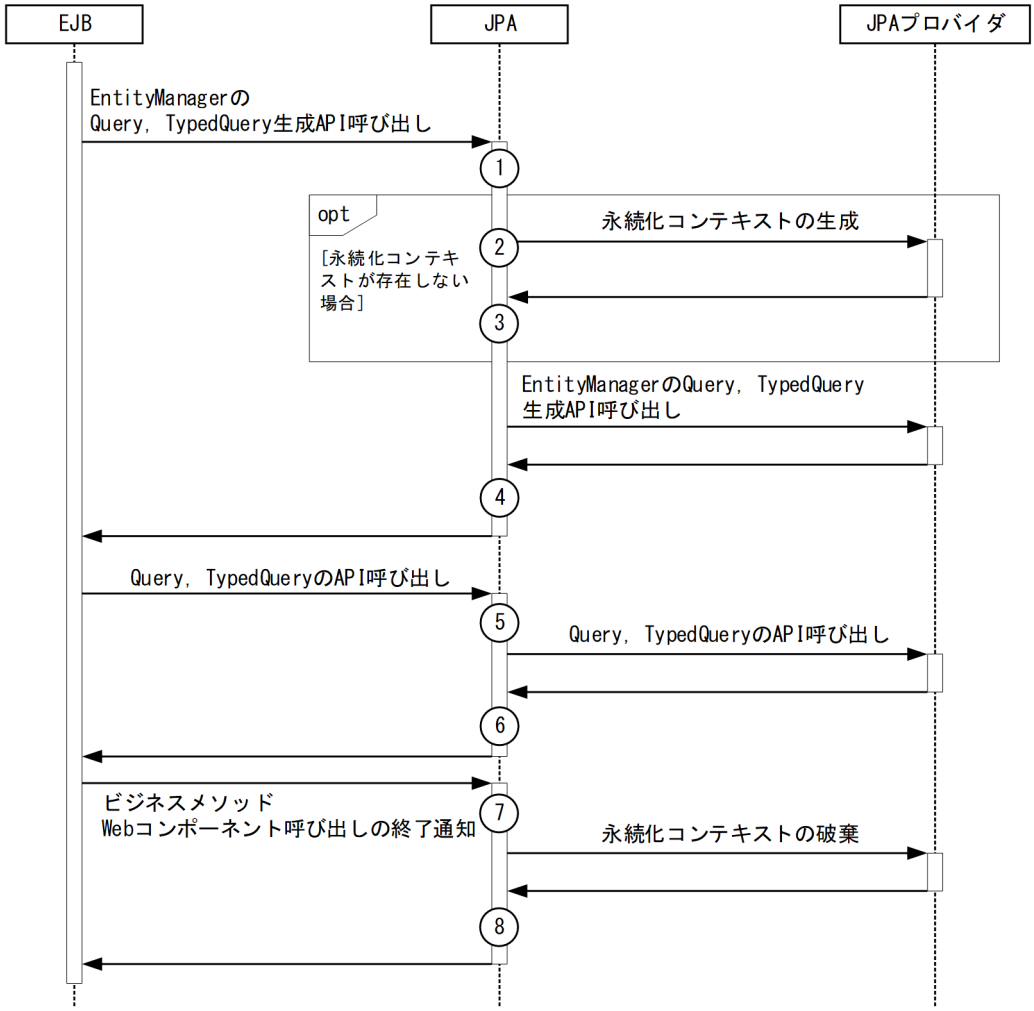
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1F4	2	トランザクションスコープの EntityManager の API を，トランザクションの存在しない状況で利用した際の永続化コンテキストの生成処理開始	A
0xD1F5	3	トランザクションスコープの EntityManager の API を，トランザクションの存在しない状況で利用した際の永続化コンテキストの生成処理終了	A
0xD1F6	7	トランザクションスコープの EntityManager の API を，トランザクションの存在しない状況で利用した際の永続化コンテキストの破棄処理開始	A
0xD1F7	8	トランザクションスコープの EntityManager の API を，トランザクションの存在しない状況で利用した際の永続化コンテキストの破棄処理終了	A

（凡例）A：標準　B：詳細

注※　図 8-71 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-71　トランザクション外で生成された Query をトランザクション外で利用した場合のトレース取得ポイント



（凡例）◯：トレース取得ポイントを示します。
処理によってPRFトレース取得レベルが異なります。

(d)　拡張永続化コンテキストを利用した場合

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-114　拡張永続化コンテキストを利用した場合のトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD11C	3	EntityManager#find(Class<T> entityClass, Object primaryKey)の処理開始	B
0xD11D	4	EntityManager#find(Class<T> entityClass, Object primaryKey)の処理終了	B
0xD11E	3	EntityManager#find(Class<T> entityClass, Object primaryKey, Map<String, Object> properties)の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD11F	4	EntityManager#find(Class<T> entityClass, Object primaryKey, Map<String, Object> properties)の処理終了	B
0xD120	3	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode)の処理開始	B
0xD121	4	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode)の処理終了	B
0xD122	3	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode, Map<String, Object> properties)の処理開始	B
0xD123	4	EntityManager#find(Class<T> entityClass, Object primaryKey, LockModeType lockMode, Map<String, Object> properties)の処理終了	B
0xD124	3	EntityManager#getReference(Class<T> entityClass, Object primaryKey)の処理開始	B
0xD125	4	EntityManager#getReference(Class<T> entityClass, Object primaryKey)の処理終了	B
0xD126	3	EntityManager#contains(Object entity)の処理開始	B
0xD127	4	EntityManager#contains(Object entity)の処理終了	B
0xD128	3	EntityManager#lock(Object entity, LockModeType lockMode)の処理開始	B
0xD129	4	EntityManager#lock(Object entity, LockModeType lockMode)の処理終了	B
0xD12A	3	EntityManager#lock(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理開始	B
0xD12B	4	EntityManager#lock(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理終了	B
0xD12C	3	EntityManager#merge(T entity)の処理開始	B
0xD12D	4	EntityManager#merge(T entity)の処理終了	B
0xD12E	3	EntityManager#persist(Object entity)の処理開始	B
0xD12F	4	EntityManager#persist(Object entity)の処理終了	B
0xD130	3	EntityManager#refresh(Object entity)の処理開始	B
0xD131	4	EntityManager#refresh(Object entity)の処理終了	B
0xD132	3	EntityManager#refresh(Object entity, Map<String,Object> properties)の処理開始	B
0xD133	4	EntityManager#refresh(Object entity, Map<String,Object> properties)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD134	3	EntityManager#refresh(Object entity, LockModeType lockMode)の処理開始	B
0xD135	4	EntityManager#refresh(Object entity, LockModeType lockMode)の処理終了	B
0xD136	3	EntityManager#refresh(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理開始	B
0xD137	4	EntityManager#refresh(Object entity, LockModeType lockMode, Map<String,Object> properties)の処理終了	B
0xD138	3	EntityManager#remove(Object entity)の処理開始	B
0xD139	4	EntityManager#remove(Object entity)の処理終了	B
0xD13A	3	EntityManager#clear()の処理開始	B
0xD13B	4	EntityManager#clear()の処理終了	B
0xD13C	3	EntityManager#flush()の処理開始	B
0xD13D	4	EntityManager#flush()の処理終了	B
0xD13E	3	EntityManager#detach(Object entity)の処理開始	B
0xD13F	4	EntityManager#detach(Object entity)の処理終了	B
0xD140	3	EntityManager#createEntityGraph(Class<T> rootType)の処理開始	B
0xD141	4	EntityManager#createEntityGraph(Class<T> rootType)の処理終了	B
0xD142	3	EntityManager#createEntityGraph(String graphName)の処理開始	B
0xD143	4	EntityManager#createEntityGraph(String graphName)の処理終了	B
0xD144	3	EntityManager#createQuery(String qlString)の処理開始	A
0xD145	4	EntityManager#createQuery(String qlString)の処理終了	A
0xD146	3	EntityManager#createQuery(CriteriaQuery<T> criteriaQuery)の処理開始	A
0xD147	4	EntityManager#createQuery(CriteriaQuery<T> criteriaQuery)の処理終了	A
0xD148	3	EntityManager#createQuery(CriteriaUpdate updateQuery)の処理開始	A
0xD149	4	EntityManager#createQuery(CriteriaUpdate updateQuery)の処理終了	A
0xD14A	3	EntityManager#createQuery(CriteriaDelete deleteQuery)の処理開始	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD14B	4	EntityManager#createQuery(CriteriaDelete deleteQuery)の処理終了	A
0xD14C	3	EntityManager#createQuery(String qlString, Class<T> resultClass)の処理開始	A
0xD14D	4	EntityManager#createQuery(String qlString, Class<T> resultClass)の処理終了	A
0xD14E	3	EntityManager#createNamedQuery(String name)の処理開始	A
0xD14F	4	EntityManager#createNamedQuery(String name)の処理終了	A
0xD150	3	EntityManager#createNamedQuery(String name, Class<T> resultClass)の処理開始	A
0xD151	4	EntityManager#createNamedQuery(String name, Class<T> resultClass)の処理終了	A
0xD152	3	EntityManager#createNativeQuery(String sqlString)の処理開始	A
0xD153	4	EntityManager#createNativeQuery(String sqlString)の処理終了	A
0xD154	3	EntityManager#createNativeQuery(String sqlString, Class resultClass)の処理開始	A
0xD155	4	EntityManager#createNativeQuery(String sqlString, Class resultClass)の処理終了	A
0xD156	3	EntityManager#createNativeQuery(String sqlString, String resultSetMapping)の処理開始	A
0xD157	4	EntityManager#createNativeQuery(String sqlString, String resultSetMapping)の処理終了	A
0xD158	3	EntityManager#createNamedStoredProcedureQuery(String name)の処理開始	A
0xD159	4	EntityManager#createNamedStoredProcedureQuery(String name)の処理終了	A
0xD15A	3	EntityManager#createStoredProcedureQuery(String procedureName)の処理開始	A
0xD15B	4	EntityManager#createStoredProcedureQuery(String procedureName)の処理終了	A
0xD15C	3	EntityManager#createStoredProcedureQuery(String procedureName, Class... resultClasses)の処理開始	A
0xD15D	4	EntityManager#createStoredProcedureQuery(String procedureName, Class... resultClasses)の処理終了	A
0xD15E	3	EntityManager#createStoredProcedureQuery(String procedureName, String... resultSetMappings)の処理開始	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD15F	4	EntityManager#createStoredProcedureQuery(String procedureName, String... resultSetMappings)の処理終了	A
0xD160	3	EntityManager#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD161	4	EntityManager#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD162	3	EntityManager#getFlushMode()の処理開始	B
0xD163	4	EntityManager#getFlushMode()の処理終了	B
0xD164	3	EntityManager#joinTransaction()の処理開始	B
0xD165	4	EntityManager#joinTransaction()の処理終了	B
0xD166	3	EntityManager#getTransaction()の処理開始	B
0xD167	4	EntityManager#getTransaction()の処理終了	B
0xD168	3	EntityManager#getDelegate()の処理開始	B
0xD169	4	EntityManager#getDelegate()の処理終了	B
0xD16A	3	EntityManager#unwrap(Class<T> cls)の処理開始	B
0xD16B	4	EntityManager#unwrap(Class<T> cls)の処理終了	B
0xD16C	3	EntityManager#getCriteriaBuilder()の処理開始	B
0xD16D	4	EntityManager#getCriteriaBuilder()の処理終了	B
0xD16E	3	EntityManager#getEntityGraph(String graphName)の処理開始	B
0xD16F	4	EntityManager#getEntityGraph(String graphName)の処理終了	B
0xD170	3	EntityManager#getEntityGraphs(Class<T> entityClass)の処理開始	B
0xD171	4	EntityManager#getEntityGraphs(Class<T> entityClass)の処理終了	B
0xD172	3	EntityManager#getEntityManagerFactory()の処理開始	B
0xD173	4	EntityManager#getEntityManagerFactory()の処理終了	B
0xD174	3	EntityManager#getLockMode(Object entity)の処理開始	B
0xD175	4	EntityManager#getLockMode(Object entity)の処理終了	B
0xD176	3	EntityManager#getMetamodel()の処理開始	B
0xD177	4	EntityManager#getMetamodel()の処理終了	B
0xD178	3	EntityManager#setProperty(String propertyName, Object value)の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD179	4	EntityManager#setProperty(String propertyName, Object value)の処理終了	B
0xD17A	3	EntityManager#getProperties()の処理開始	B
0xD17B	4	EntityManager#getProperties()の処理終了	B
0xD17C	3	EntityManager#isJoinedToTransaction()の処理開始	B
0xD17D	4	EntityManager#isJoinedToTransaction()の処理終了	B
0xD17E	3	EntityManager#isOpen()の処理開始	B
0xD17F	4	EntityManager#isOpen()の処理終了	B
0xD180	3	EntityManager#close()の処理開始	A
0xD181	4	EntityManager#close()の処理終了	A
0xD18E	3	Query#executeUpdate()の処理開始	B
0xD18F	4	Query#executeUpdate()の処理終了	B
0xD190	3	Query#getResultList()の処理開始	B
0xD191	4	Query#getResultList()の処理終了	B
0xD192	3	Query#getSingleResult()の処理開始	B
0xD193	4	Query#getSingleResult()の処理終了	B
0xD194	3	Query#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD195	4	Query#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD196	3	Query#getFlushMode()の処理開始	B
0xD197	4	Query#getFlushMode()の処理終了	B
0xD198	3	Query#setFirstResult(int startPosition)の処理開始	B
0xD199	4	Query#setFirstResult(int startPosition)の処理終了	B
0xD19A	3	Query#getFirstResult()の処理開始	B
0xD19B	4	Query#getFirstResult()の処理終了	B
0xD19C	3	Query#setMaxResults(int maxResult)の処理開始	B
0xD19D	4	Query#setMaxResults(int maxResult)の処理終了	B
0xD19E	3	Query#getMaxResults()の処理開始	B
0xD19F	4	Query#getMaxResults()の処理終了	B
0xD1A0	3	Query#setHint(String hintName, Object value)の処理開始	B
0xD1A1	4	Query#setHint(String hintName, Object value)の処理終了	B
0xD1A2	3	Query#getHints()の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1A3	4	Query#getHints()の処理終了	B
0xD1A4	3	Query#setParameter(int position, Calendar value, TemporalType temporalType)の処理開始	B
0xD1A5	4	Query#setParameter(int position, Calendar value, TemporalType temporalType)の処理終了	B
0xD1A6	3	Query#setParameter(int position, Date value, TemporalType temporalType)の処理開始	B
0xD1A7	4	Query#setParameter(int position, Date value, TemporalType temporalType)の処理終了	B
0xD1A8	3	Query#setParameter(int position, Object value)の処理開始	B
0xD1A9	4	Query#setParameter(int position, Object value)の処理終了	B
0xD1AA	3	Query#setParameter(String name, Calendar value, TemporalType temporalType)の処理開始	B
0xD1AB	4	Query#setParameter(String name, Calendar value, TemporalType temporalType)の処理終了	B
0xD1AC	3	Query#setParameter(String name, Date value, TemporalType temporalType)の処理開始	B
0xD1AD	4	Query#setParameter(String name, Date value, TemporalType temporalType)の処理終了	B
0xD1AE	3	Query#setParameter(String name, Object value)の処理開始	B
0xD1AF	4	Query#setParameter(String name, Object value)の処理終了	B
0xD1B0	3	Query#setParameter(Parameter<T> param, T value)の処理開始	B
0xD1B1	4	Query#setParameter(Parameter<T> param, T value)の処理終了	B
0xD1B2	3	Query#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理開始	B
0xD1B3	4	Query#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理終了	B
0xD1B4	3	Query#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理開始	B
0xD1B5	4	Query#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理終了	B
0xD1B6	3	Query#getParameters()の処理開始	B
0xD1B7	4	Query#getParameters()の処理終了	B
0xD1B8	3	Query#getParameter(String name)の処理開始	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1B9	4	Query#getParameter(String name)の処理終了	B
0xD1BA	3	Query#getParameter(String name, Class<T> type)の処理開始	B
0xD1BB	4	Query#getParameter(String name, Class<T> type)の処理終了	B
0xD1BC	3	Query#getParameter(int position)の処理開始	B
0xD1BD	4	Query#getParameter(int position)の処理終了	B
0xD1BE	3	Query#getParameter(int position, Class<T> type)の処理開始	B
0xD1BF	4	Query#getParameter(int position, Class<T> type)の処理終了	B
0xD1C0	3	Query#isBound(Parameter<?> param)の処理開始	B
0xD1C1	4	Query#isBound(Parameter<?> param)の処理終了	B
0xD1C2	3	Query#getParameterValue(Parameter<T> param)の処理開始	B
0xD1C3	4	Query#getParameterValue(Parameter<T> param)の処理終了	B
0xD1C4	3	Query#getParameterValue(String name)の処理開始	B
0xD1C5	4	Query#getParameterValue(String name)の処理終了	B
0xD1C6	3	Query#getParameterValue(int position)の処理開始	B
0xD1C7	4	Query#getParameterValue(int position)の処理終了	B
0xD1C8	3	Query#setLockMode(LockModeType lockMode)の処理開始	B
0xD1C9	4	Query#setLockMode(LockModeType lockMode)の処理終了	B
0xD1CA	3	Query#getLockMode()の処理開始	B
0xD1CB	4	Query#getLockMode()の処理終了	B
0xD1CC	3	Query#unwrap(Class<T> cls)の処理開始	B
0xD1CD	4	Query#unwrap(Class<T> cls)の処理終了	B
0xD1CE	3	TypedQuery#executeUpdate()の処理開始	B
0xD1CF	4	TypedQuery#executeUpdate()の処理終了	B
0xD1D0	3	TypedQuery#getResultList()の処理開始	B
0xD1D1	4	TypedQuery#getResultList()の処理終了	B
0xD1D2	3	TypedQuery#getSingleResult()の処理開始	B
0xD1D3	4	TypedQuery#getSingleResult()の処理終了	B
0xD1D4	3	TypedQuery#setMaxResults(int maxResult)の処理開始	B
0xD1D5	4	TypedQuery#setMaxResults(int maxResult)の処理終了	B
0xD1D6	3	TypedQuery#setFirstResult(int startPosition)の処理開始	B
0xD1D7	4	TypedQuery#setFirstResult(int startPosition)の処理終了	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1D8	3	TypedQuery#setHint(String hintName, Object value)の処理開始	B
0xD1D9	4	TypedQuery#setHint(String hintName, Object value)の処理終了	B
0xD1DA	3	TypedQuery#setParameter(Parameter<T> param, T value)の処理開始	B
0xD1DB	4	TypedQuery#setParameter(Parameter<T> param, T value)の処理終了	B
0xD1DC	3	TypedQuery#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理開始	B
0xD1DD	4	TypedQuery#setParameter(Parameter<Calendar> param, Calendar value, TemporalType temporalType)の処理終了	B
0xD1DE	3	TypedQuery#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理開始	B
0xD1DF	4	TypedQuery#setParameter(Parameter<Date> param, Date value, TemporalType temporalType)の処理終了	B
0xD1E0	3	TypedQuery#setParameter(String name, Object value)の処理開始	B
0xD1E1	4	TypedQuery#setParameter(String name, Object value)の処理終了	B
0xD1E2	3	TypedQuery#setParameter(String name, Calendar value, TemporalType temporalType)の処理開始	B
0xD1E3	4	TypedQuery#setParameter(String name, Calendar value, TemporalType temporalType)の処理終了	B
0xD1E4	3	TypedQuery#setParameter(String name, Date value, TemporalType temporalType)の処理開始	B
0xD1E5	4	TypedQuery#setParameter(String name, Date value, TemporalType temporalType)の処理終了	B
0xD1E6	3	TypedQuery#setParameter(int position, Object value)の処理開始	B
0xD1E7	4	TypedQuery#setParameter(int position, Object value)の処理終了	B
0xD1E8	3	TypedQuery#setParameter(int position, Calendar value, TemporalType temporalType)の処理開始	B
0xD1E9	4	TypedQuery#setParameter(int position, Calendar value, TemporalType temporalType)の処理終了	B
0xD1EA	3	TypedQuery#setParameter(int position, Date value, TemporalType temporalType)の処理開始	B

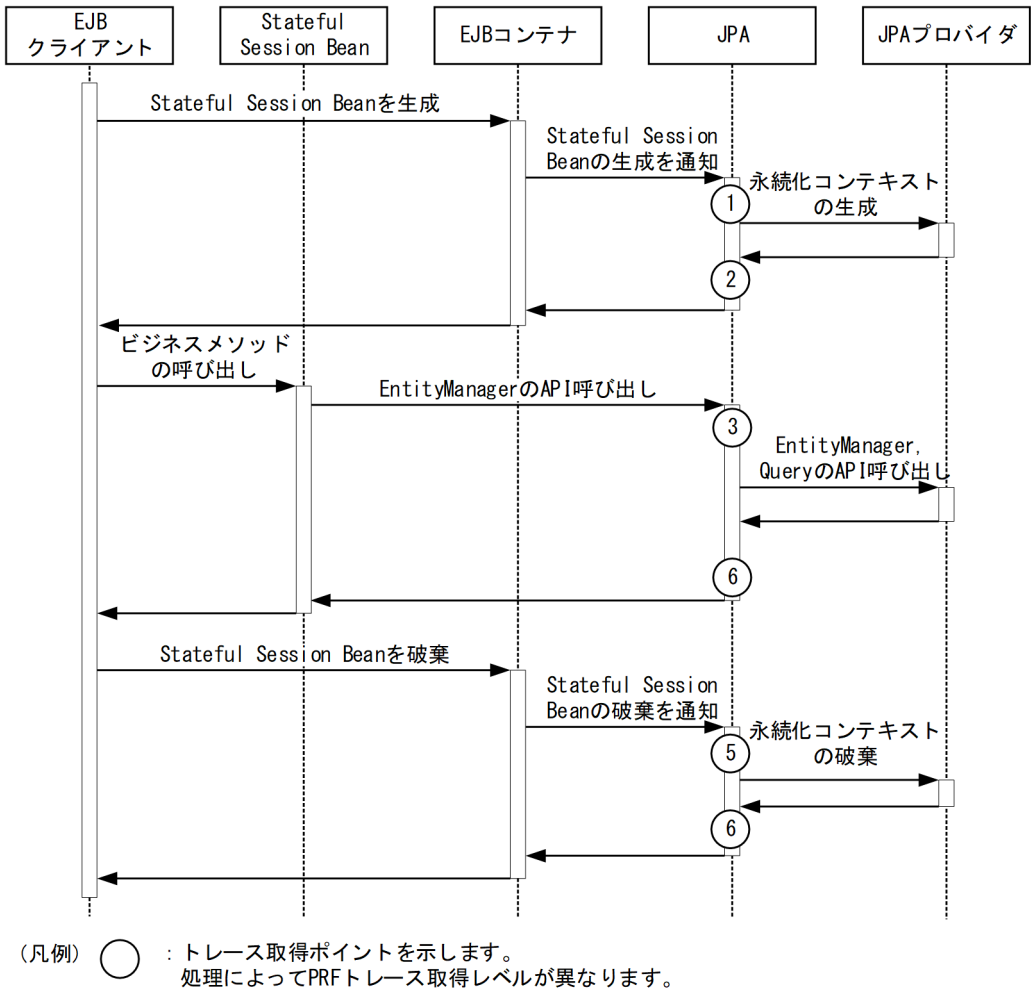
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD1EB	4	TypedQuery#setParameter(int position, Date value, TemporalType temporalType)の処理終了	B
0xD1EC	3	TypedQuery#setFlushMode(FlushModeType flushMode)の処理開始	B
0xD1ED	4	TypedQuery#setFlushMode(FlushModeType flushMode)の処理終了	B
0xD1EE	3	TypedQuery#setLockMode(LockModeType lockMode)の処理開始	B
0xD1EF	4	TypedQuery#setLockMode(LockModeType lockMode)の処理終了	B
0xD1FC	1	拡張スコープの永続化コンテキストを利用した際の永続化コンテキストの生成処理開始	A
0xD1FD	2	拡張スコープの永続化コンテキストを利用した際の永続化コンテキストの生成処理終了	A
0xD1FE	5	拡張スコープの永続化コンテキストを利用した際のEntityManager.close()の処理開始	A
0xD1FF	6	拡張スコープの永続化コンテキストを利用した際のEntityManager.close()の処理終了	A

（凡例）A：標準　B：詳細

注※　図 8-72 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-72　拡張永続化コンテキストを利用した場合のトレース取得ポイント



（凡例）◯：トレース取得ポイントを示します。
処理によってPRFトレース取得レベルが異なります。

(2)　取得できるトレース情報

(a)　トランザクションスコープの永続化コンテキストをトランザクション内で利用した場合

トランザクションスコープの永続化コンテキストをトランザクション内で利用した場合に取得できるトレース情報を次の表に示します。

表 8-115　トランザクションスコープの永続化コンテキストをトランザクション内で利用した場合に取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1，5	0xD11C	B	entity のクラス名	−	−
	0xD11E	B	entityClass のクラス名	−	−
	0xD120	B	entityClass のクラス名	−	−
	0xD122	B	entityClass のクラス名	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD124	B	entity のクラス名	−	−
	0xD126	B	entity のクラス名	−	−
	0xD128	B	entity のクラス名	lockMode の値	−
	0xD12A	B	entity のクラス名	lockMode の値	−
	0xD12C	B	entity のクラス名	−	−
	0xD12E	B	entity のクラス名	−	−
	0xD130	B	entity のクラス名	−	−
	0xD132	B	entity のクラス名	−	−
	0xD134	B	entity のクラス名	−	−
	0xD136	B	entity のクラス名	−	−
	0xD138	B	entity のクラス名	−	−
	0xD13A	B	−	−	−
	0xD13C	B	−	−	−
	0xD13E	B	entity のクラス名	−	−
	0xD140	B	rootTyoe のクラス名	−	−
	0xD142	B	graphName の値	−	−
	0xD144	A	−	−	−
	0xD146	A	−	−	−
	0xD148	A	−	−	−
	0xD14A	A	−	−	−
	0xD14C	A	−	−	−
	0xD14E	A	name の値	−	−
	0xD150	A	name の値	−	−
	0xD152	A	−	−	−
	0xD154	A	resultClass のクラス名	−	−
	0xD156	A	resultSetMapping	−	−
	0xD158	A	name の値	−	−
	0xD15A	A	procedureName の値	−	−
	0xD15C	A	procedureName の値	−	−
	0xD15E	A	procedureName の値	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD160	B	flushMode の値	−	−
	0xD162	B	−	−	−
	0xD164	B	−	−	−
	0xD166	B	−	−	−
	0xD168	B	−	−	−
	0xD16A	B	cls のクラス名	−	−
	0xD16C	B	−	−	−
	0xD16E	B	graphName の値	−	−
	0xD170	B	entityClass のクラス名	−	−
	0xD172	B	−	−	−
	0xD174	B	entity のクラス名	−	−
	0xD176	B	−	−	−
	0xD178	B	propertyName の値	−	−
	0xD17A	B	−	−	−
	0xD17C	B	−	−	−
	0xD17E	B	−	−	−
	0xD180	A	−	−	−
	0xD18E	B	−	−	−
	0xD190	B	−	−	−
	0xD192	B	−	−	−
	0xD194	B	flushMode の値	−	−
	0xD196	B	−	−	−
	0xD198	B	startPosition の値	−	−
	0xD19A	B	−	−	−
	0xD19C	B	maxResult の値	−	−
	0xD19E	B	−	−	−
	0xD1A0	B	hintName	value のクラス名	−
	0xD1A2	B	−	−	−
	0xD1A4	B	position の値	−	−
	0xD1A6	B	position の値	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1A8	B	position の値	−	−
	0xD1AA	B	name の値	−	−
	0xD1AC	B	name の値	−	−
	0xD1AE	B	name の値	−	−
	0xD1B0	B	−	−	−
	0xD1B2	B	−	−	−
	0xD1B4	B	−	−	−
	0xD1B6	B	−	−	−
	0xD1B8	B	name の値	−	−
	0xD1BA	B	name の値	−	−
	0xD1BC	B	position の値	−	−
	0xD1BE	B	position の値	−	−
	0xD1C0	B	−	−	−
	0xD1C2	B	−	−	−
	0xD1C4	B	name の値	−	−
	0xD1C6	B	position の値	−	−
	0xD1C8	B	−	−	−
	0xD1CA	B	−	−	−
	0xD1CC	B	−	−	−
	0xD1CE	B	−	−	−
	0xD1D0	B	−	−	−
	0xD1D2	B	−	−	−
	0xD1D4	B	maxResult の値	−	−
	0xD1D6	B	startPosition の値	−	−
	0xD1D8	B	hintName	value のクラス名	−
	0xD1DA	B	−	−	−
	0xD1DC	B	−	−	−
	0xD1DE	B	−	−	−
	0xD1E0	B	name の値	−	−
	0xD1E2	B	name の値	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1E4	B	name の値	−	−
	0xD1E6	B	position の値	−	−
	0xD1E8	B	position の値	−	−
	0xD1EA	B	position の値	−	−
	0xD1EC	B	flushMode の値	−	−
	0xD1EE	B	−	−	−
2	0xD1F0	A	−	−	−
3	0xD1F1	A	−	−	※2
4，6	0xD11D	B	−	−	※2
	0xD11F	B	−	−	※2
	0xD121	B	−	−	※2
	0xD123	B	−	−	※2
	0xD125	B	−	−	※2
	0xD127	B	−	−	※2
	0xD129	B	−	−	※2
	0xD12B	B	−	−	※2
	0xD12D	B	−	−	※2
	0xD12F	B	−	−	※2
	0xD131	B	−	−	※2
	0xD133	B	−	−	※2
	0xD135	B	−	−	※2
	0xD137	B	−	−	※2
	0xD139	B	−	−	※2
	0xD13B	B	−	−	※2
	0xD13D	B	−	−	※2
	0xD13F	B	−	−	※2
	0xD141	B	−	−	※2
	0xD143	B	−	−	※2
	0xD145	A	−	−	※2
	0xD147	A	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD149	A	−	−	※2
	0xD14B	A	−	−	※2
	0xD14D	A	−	−	※2
	0xD14F	A	−	−	※2
	0xD151	A	−	−	※2
	0xD153	A	−	−	※2
	0xD155	A	−	−	※2
	0xD157	A	−	−	※2
	0xD159	A	−	−	※2
	0xD15B	A	−	−	※2
	0xD15D	A	−	−	※2
	0xD15F	A	−	−	※2
	0xD161	B	−	−	※2
	0xD163	B	−	−	※2
	0xD165	B	−	−	※2
	0xD167	B	−	−	※2
	0xD169	B	−	−	※2
	0xD16B	B	−	−	※2
	0xD16D	B	−	−	※2
	0xD16F	B	−	−	※2
	0xD171	B	−	−	※2
	0xD173	B	−	−	※2
	0xD175	B	−	−	※2
	0xD177	B	−	−	※2
	0xD179	B	−	−	※2
	0xD17B	B	−	−	※2
	0xD17D	B	−	−	※2
	0xD17F	B	−	−	※2
	0xD181	A	−	−	※2
	0xD18F	B	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD191	B	−	−	※2
	0xD193	B	−	−	※2
	0xD195	B	−	−	※2
	0xD197	B	−	−	※2
	0xD199	B	−	−	※2
	0xD19B	B	−	−	※2
	0xD19D	B	−	−	※2
	0xD19F	B	−	−	※2
	0xD1A1	B	−	−	※2
	0xD1A3	B	−	−	※2
	0xD1A5	B	−	−	※2
	0xD1A7	B	−	−	※2
	0xD1A9	B	−	−	※2
	0xD1AB	B	−	−	※2
	0xD1AD	B	−	−	※2
	0xD1AF	B	−	−	※2
	0xD1B1	B	−	−	※2
	0xD1B3	B	−	−	※2
	0xD1B5	B	−	−	※2
	0xD1B7	B	−	−	※2
	0xD1B9	B	−	−	※2
	0xD1BB	B	−	−	※2
	0xD1BD	B	−	−	※2
	0xD1BF	B	−	−	※2
	0xD1C1	B	−	−	※2
	0xD1C3	B	−	−	※2
	0xD1C5	B	−	−	※2
	0xD1C7	B	−	−	※2
	0xD1C9	B	−	−	※2
	0xD1CB	B	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1CD	B	−	−	※2
	0xD1CF	B	−	−	※2
	0xD1D1	B	−	−	※2
	0xD1D3	B	−	−	※2
	0xD1D5	B	−	−	※2
	0xD1D7	B	−	−	※2
	0xD1D9	B	−	−	※2
	0xD1DB	B	−	−	※2
	0xD1DD	B	−	−	※2
	0xD1DF	B	−	−	※2
	0xD1E1	B	−	−	※2
	0xD1E3	B	−	−	※2
	0xD1E5	B	−	−	※2
	0xD1E7	B	−	−	※2
	0xD1E9	B	−	−	※2
	0xD1EB	B	−	−	※2
	0xD1ED	B	−	−	※2
	0xD1EF	B	−	−	※2
7	0xD1F2	A	−	−	−
8	0xD1F3	A	−	−	※2

（凡例）A：標準　B：詳細　−：該当なし

注※1　図 8-69 中の番号と対応しています。

注※2　正常に処理された場合，入り口時刻が表示されます。例外が発生した場合，入り口時刻と例外が表示されます。

(b)　トランザクションスコープの永続化コンテキストに関連づいているエンティティマネージャをトランザクション外で利用した場合

トランザクションスコープの永続化コンテキストに関連づいているエンティティマネージャをトランザクション外で利用した場合に取得できるトレース情報を次の表に示します。

表 8-116　トランザクションスコープの永続化コンテキストに関連づいているエンティティマネージャをトランザクション外で利用した場合に取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xD11C	B	entity のクラス名	−	−
	0xD11E	B	entityClass のクラス名	−	−
	0xD120	B	entityClass のクラス名	−	−
	0xD122	B	entityClass のクラス名	−	−
	0xD124	B	entity のクラス名	−	−
	0xD126	B	entity のクラス名	−	−
	0xD128	B	entity のクラス名	lockMode の値	−
	0xD12A	B	entity のクラス名	lockMode の値	−
	0xD12C	B	entity のクラス名	−	−
	0xD12E	B	entity のクラス名	−	−
	0xD130	B	entity のクラス名	−	−
	0xD132	B	entity のクラス名	−	−
	0xD134	B	entity のクラス名	−	−
	0xD136	B	entity のクラス名	−	−
	0xD138	B	entity のクラス名	−	−
	0xD13A	B	−	−	−
	0xD13C	B	−	−	−
	0xD13E	B	entity のクラス名	−	−
	0xD140	B	rootTyoe のクラス名	−	−
	0xD142	B	graphName の値	−	−
	0xD160	B	flushMode の値	−	−
	0xD162	B	−	−	−
	0xD164	B	−	−	−
	0xD166	B	−	−	−
	0xD168	B	−	−	−
	0xD16A	B	cls のクラス名	−	−
	0xD16C	B	−	−	−
	0xD16E	B	graphName の値	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD170	B	entityClass のクラス名	−	−
	0xD172	B	−	−	−
	0xD174	B	entity のクラス名	−	−
	0xD176	B	−	−	−
	0xD178	B	propertyName の値	−	−
	0xD17A	B	−	−	−
	0xD17C	B	−	−	−
	0xD17E	B	−	−	−
	0xD180	A	−	−	−
2	0xD1F4	A	−	−	−
3	0xD1F5	A	−	−	※2
4	0xD11D	B	−	−	※2
	0xD11F	B	−	−	※2
	0xD121	B	−	−	※2
	0xD123	B	−	−	※2
	0xD125	B	−	−	※2
	0xD127	B	−	−	※2
	0xD129	B	−	−	※2
	0xD12B	B	−	−	※2
	0xD12D	B	−	−	※2
	0xD12F	B	−	−	※2
	0xD131	B	−	−	※2
	0xD133	B	−	−	※2
	0xD135	B	−	−	※2
	0xD137	B	−	−	※2
	0xD139	B	−	−	※2
	0xD13B	B	−	−	※2
	0xD13D	B	−	−	※2
	0xD13F	B	−	−	※2
	0xD141	B	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD143	B	−	−	※2
	0xD161	B	−	−	※2
	0xD163	B	−	−	※2
	0xD165	B	−	−	※2
	0xD167	B	−	−	※2
	0xD169	B	−	−	※2
	0xD16B	B	−	−	※2
	0xD16D	B	−	−	※2
	0xD16F	B	−	−	※2
	0xD171	B	−	−	※2
	0xD173	B	−	−	※2
	0xD175	B	−	−	※2
	0xD177	B	−	−	※2
	0xD179	B	−	−	※2
	0xD17B	B	−	−	※2
	0xD17D	B	−	−	※2
	0xD17F	B	−	−	※2
	0xD181	A	−	−	※2
5	0xD1F6	A	−	−	−
6	0xD1F7	A	−	−	※2

（凡例）A：標準　B：詳細　−：該当なし

注※1　図 8-70 中の番号と対応しています。

注※2　正常に処理された場合，入り口時刻が表示されます。例外が発生した場合，入り口時刻と例外が表示されます。

(c)　トランザクション外で生成された Query をトランザクション外で利用した場合

トランザクション外で生成された Query をトランザクション外で利用した場合に取得できるトレース情報を次の表に示します。

表 8-117　トランザクション外で生成された Query をトランザクション外で利用した場合に取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xD144	A	−	−	−
	0xD146	A	−	−	−
	0xD148	A	−	−	−
	0xD14A	A	−	−	−
	0xD14C	A	−	−	−
	0xD14E	A	name の値	−	−
	0xD150	A	name の値	−	−
	0xD152	A	−	−	−
	0xD154	A	resultClass のクラス名	−	−
	0xD156	A	resultSetMapping	−	−
	0xD158	A	name の値	−	−
	0xD15A	A	procedureName の値	−	−
	0xD15C	A	procedureName の値	−	−
	0xD15E	A	procedureName の値	−	−
1，5	0xD18E	B	−	−	−
	0xD190	B	−	−	−
	0xD192	B	−	−	−
	0xD194	B	flushMode の値	−	−
	0xD196	B	−	−	−
	0xD198	B	startPosition の値	−	−
	0xD19A	B	−	−	−
	0xD19C	B	maxResult の値	−	−
	0xD19E	B	−	−	−
	0xD1A0	B	hintName	value のクラス名	−
	0xD1A2	B	−	−	−
	0xD1A4	B	position の値	−	−
	0xD1A6	B	position の値	−	−
	0xD1A8	B	position の値	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1AA	B	name の値	−	−
	0xD1AC	B	name の値	−	−
	0xD1AE	B	name の値	−	−
	0xD1B0	B	−	−	−
	0xD1B2	B	−	−	−
	0xD1B4	B	−	−	−
	0xD1B6	B	−	−	−
	0xD1B8	B	name の値	−	−
	0xD1BA	B	name の値	−	−
	0xD1BC	B	position の値	−	−
	0xD1BE	B	position の値	−	−
	0xD1C0	B	−	−	−
	0xD1C2	B	−	−	−
	0xD1C4	B	name の値	−	−
	0xD1C6	B	position の値	−	−
	0xD1C8	B	−	−	−
	0xD1CA	B	−	−	−
	0xD1CC	B	−	−	−
	0xD1CE	B	−	−	−
	0xD1D0	B	−	−	−
	0xD1D2	B	−	−	−
	0xD1D4	B	maxResult の値	−	−
	0xD1D6	B	startPosition の値	−	−
	0xD1D8	B	hintName	value のクラス名	−
	0xD1DA	B	−	−	−
	0xD1DC	B	−	−	−
	0xD1DE	B	−	−	−
	0xD1E0	B	name の値	−	−
	0xD1E2	B	name の値	−	−
	0xD1E4	B	name の値	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1E6	B	position の値	−	−
	0xD1E8	B	position の値	−	−
	0xD1EA	B	position の値	−	−
	0xD1EC	B	flushMode の値	−	−
	0xD1EE	B	−	−	−
2	0xD1F4	A	−	−	−
3	0xD1F5	A	−	−	※2
4	0xD145	A	−	−	※2
	0xD147	A	−	−	※2
	0xD149	A	−	−	※2
	0xD14B	A	−	−	※2
	0xD14D	A	−	−	※2
	0xD14F	A	−	−	※2
	0xD151	A	−	−	※2
	0xD153	A	−	−	※2
	0xD155	A	−	−	※2
	0xD157	A	−	−	※2
	0xD159	A	−	−	※2
	0xD15B	A	−	−	※2
	0xD15D	A	−	−	※2
	0xD15F	A	−	−	※2
4，6	0xD18F	B	−	−	※2
	0xD191	B	−	−	※2
	0xD193	B	−	−	※2
	0xD195	B	−	−	※2
	0xD197	B	−	−	※2
	0xD199	B	−	−	※2
	0xD19B	B	−	−	※2
	0xD19D	B	−	−	※2
	0xD19F	B	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1A1	B	−	−	※2
	0xD1A3	B	−	−	※2
	0xD1A5	B	−	−	※2
	0xD1A7	B	−	−	※2
	0xD1A9	B	−	−	※2
	0xD1AB	B	−	−	※2
	0xD1AD	B	−	−	※2
	0xD1AF	B	−	−	※2
	0xD1B1	B	−	−	※2
	0xD1B3	B	−	−	※2
	0xD1B5	B	−	−	※2
	0xD1B7	B	−	−	※2
	0xD1B9	B	−	−	※2
	0xD1BB	B	−	−	※2
	0xD1BD	B	−	−	※2
	0xD1BF	B	−	−	※2
	0xD1C1	B	−	−	※2
	0xD1C3	B	−	−	※2
	0xD1C5	B	−	−	※2
	0xD1C7	B	−	−	※2
	0xD1C9	B	−	−	※2
	0xD1CB	B	−	−	※2
	0xD1CD	B	−	−	※2
	0xD1CF	B	−	−	※2
	0xD1D1	B	−	−	※2
	0xD1D3	B	−	−	※2
	0xD1D5	B	−	−	※2
	0xD1D7	B	−	−	※2
	0xD1D9	B	−	−	※2
	0xD1DB	B	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1DD	B	−	−	※2
	0xD1DF	B	−	−	※2
	0xD1E1	B	−	−	※2
	0xD1E3	B	−	−	※2
	0xD1E5	B	−	−	※2
	0xD1E7	B	−	−	※2
	0xD1E9	B	−	−	※2
	0xD1EB	B	−	−	※2
	0xD1ED	B	−	−	※2
	0xD1EF	B	−	−	※2
7	0xD1F6	A	−	−	−
8	0xD1F7	A	−	−	※2

（凡例）A：標準　B：詳細　−：該当なし

注※1　図 8-71 中の番号と対応しています。

注※2　正常に処理された場合，入り口時刻が表示されます。例外が発生した場合，入り口時刻と例外が表示されます。

(d)　拡張永続化コンテキストを利用した場合

拡張永続化コンテキストを利用した場合に取得できるトレース情報を次の表に示します。

表 8-118　拡張永続化コンテキストを利用した場合に取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xD1FC	A	−	−	−
2	0xD1FD	A	−	−	※2
3	0xD11C	B	entity のクラス名	−	−
	0xD11E	B	entityClass のクラス名	−	−
	0xD120	B	entityClass のクラス名	−	−
	0xD122	B	entityClass のクラス名	−	−
	0xD124	B	entity のクラス名	−	−
	0xD126	B	entity のクラス名	−	−
	0xD128	B	entity のクラス名	lockMode の値	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD12A	B	entity のクラス名	lockMode の値	－
	0xD12C	B	entity のクラス名	－	－
	0xD12E	B	entity のクラス名	－	－
	0xD130	B	entity のクラス名	－	－
	0xD132	B	entity のクラス名	－	－
	0xD134	B	entity のクラス名	－	－
	0xD136	B	entity のクラス名	－	－
	0xD138	B	entity のクラス名	－	－
	0xD13A	B	－	－	－
	0xD13C	B	－	－	－
	0xD13E	B	entity のクラス名	－	－
	0xD140	B	rootTyoe のクラス名	－	－
	0xD142	B	graphName の値	－	－
	0xD144	A	－	－	－
	0xD146	A	－	－	－
	0xD148	A	－	－	－
	0xD14A	A	－	－	－
	0xD14C	A	－	－	－
	0xD14E	A	name の値	－	－
	0xD150	A	name の値	－	－
	0xD152	A	－	－	－
	0xD154	A	resultClass のクラス名	－	－
	0xD156	A	resultSetMapping	－	－
	0xD158	A	name の値	－	－
	0xD15A	A	procedureName の値	－	－
	0xD15C	A	procedureName の値	－	－
	0xD15E	A	procedureName の値	－	－
	0xD160	B	flushMode の値	－	－
	0xD162	B	－	－	－
	0xD164	B	－	－	－

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD166	B	−	−	−
	0xD168	B	−	−	−
	0xD16A	B	cls のクラス名	−	−
	0xD16C	B	−	−	−
	0xD16E	B	graphName の値	−	−
	0xD170	B	entityClass のクラス名	−	−
	0xD172	B	−	−	−
	0xD174	B	entity のクラス名	−	−
	0xD176	B	−	−	−
	0xD178	B	propertyName の値	−	−
	0xD17A	B	−	−	−
	0xD17C	B	−	−	−
	0xD17E	B	−	−	−
	0xD180	A	−	−	−
	0xD18E	B	−	−	−
	0xD190	B	−	−	−
	0xD192	B	−	−	−
	0xD194	B	flushMode の値	−	−
	0xD196	B	−	−	−
	0xD198	B	startPosition の値	−	−
	0xD19A	B	−	−	−
	0xD19C	B	maxResult の値	−	−
	0xD19E	B	−	−	−
	0xD1A0	B	hintName	value のクラス名	−
	0xD1A2	B	−	−	−
	0xD1A4	B	position の値	−	−
	0xD1A6	B	position の値	−	−
	0xD1A8	B	position の値	−	−
	0xD1AA	B	name の値	−	−
	0xD1AC	B	name の値	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1AE	B	name の値	−	−
	0xD1B0	B	−	−	−
	0xD1B2	B	−	−	−
	0xD1B4	B	−	−	−
	0xD1B6	B	−	−	−
	0xD1B8	B	name の値	−	−
	0xD1BA	B	name の値	−	−
	0xD1BC	B	position の値	−	−
	0xD1BE	B	position の値	−	−
	0xD1C0	B	−	−	−
	0xD1C2	B	−	−	−
	0xD1C4	B	name の値	−	−
	0xD1C6	B	position の値	−	−
	0xD1C8	B	−	−	−
	0xD1CA	B	−	−	−
	0xD1CC	B	−	−	−
	0xD1CE	B	−	−	−
	0xD1D0	B	−	−	−
	0xD1D2	B	−	−	−
	0xD1D4	B	maxResult の値	−	−
	0xD1D6	B	startPosition の値	−	−
	0xD1D8	B	hintName	value のクラス名	−
	0xD1DA	B	−	−	−
	0xD1DC	B	−	−	−
	0xD1DE	B	−	−	−
	0xD1E0	B	name の値	−	−
	0xD1E2	B	name の値	−	−
	0xD1E4	B	name の値	−	−
	0xD1E6	B	position の値	−	−
	0xD1E8	B	position の値	−	−

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1EA	B	position の値	−	−
	0xD1EC	B	flushMode の値	−	−
	0xD1EE	B	−	−	−
4	0xD11D	B	−	−	※2
	0xD11F	B	−	−	※2
	0xD121	B	−	−	※2
	0xD123	B	−	−	※2
	0xD125	B	−	−	※2
	0xD127	B	−	−	※2
	0xD129	B	−	−	※2
	0xD12B	B	−	−	※2
	0xD12D	B	−	−	※2
	0xD12F	B	−	−	※2
	0xD131	B	−	−	※2
	0xD133	B	−	−	※2
	0xD135	B	−	−	※2
	0xD137	B	−	−	※2
	0xD139	B	−	−	※2
	0xD13B	B	−	−	※2
	0xD13D	B	−	−	※2
	0xD13F	B	−	−	※2
	0xD141	B	−	−	※2
	0xD143	B	−	−	※2
	0xD145	A	−	−	※2
	0xD147	A	−	−	※2
	0xD149	A	−	−	※2
	0xD14B	A	−	−	※2
	0xD14D	A	−	−	※2
	0xD14F	A	−	−	※2
	0xD151	A	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD153	A	−	−	※2
	0xD155	A	−	−	※2
	0xD157	A	−	−	※2
	0xD159	A	−	−	※2
	0xD15B	A	−	−	※2
	0xD15D	A	−	−	※2
	0xD15F	A	−	−	※2
	0xD161	B	−	−	※2
	0xD163	B	−	−	※2
	0xD165	B	−	−	※2
	0xD167	B	−	−	※2
	0xD169	B	−	−	※2
	0xD16B	B	−	−	※2
	0xD16D	B	−	−	※2
	0xD16F	B	−	−	※2
	0xD171	B	−	−	※2
	0xD173	B	−	−	※2
	0xD175	B	−	−	※2
	0xD177	B	−	−	※2
	0xD179	B	−	−	※2
	0xD17B	B	−	−	※2
	0xD17D	B	−	−	※2
	0xD17F	B	−	−	※2
	0xD181	A	−	−	※2
	0xD18F	B	−	−	※2
	0xD191	B	−	−	※2
	0xD193	B	−	−	※2
	0xD195	B	−	−	※2
	0xD197	B	−	−	※2
	0xD199	B	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD19B	B	−	−	※2
	0xD19D	B	−	−	※2
	0xD19F	B	−	−	※2
	0xD1A1	B	−	−	※2
	0xD1A3	B	−	−	※2
	0xD1A5	B	−	−	※2
	0xD1A7	B	−	−	※2
	0xD1A9	B	−	−	※2
	0xD1AB	B	−	−	※2
	0xD1AD	B	−	−	※2
	0xD1AF	B	−	−	※2
	0xD1B1	B	−	−	※2
	0xD1B3	B	−	−	※2
	0xD1B5	B	−	−	※2
	0xD1B7	B	−	−	※2
	0xD1B9	B	−	−	※2
	0xD1BB	B	−	−	※2
	0xD1BD	B	−	−	※2
	0xD1BF	B	−	−	※2
	0xD1C1	B	−	−	※2
	0xD1C3	B	−	−	※2
	0xD1C5	B	−	−	※2
	0xD1C7	B	−	−	※2
	0xD1C9	B	−	−	※2
	0xD1CB	B	−	−	※2
	0xD1CD	B	−	−	※2
	0xD1CF	B	−	−	※2
	0xD1D1	B	−	−	※2
	0xD1D3	B	−	−	※2
	0xD1D5	B	−	−	※2

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xD1D7	B	－	－	※2
	0xD1D9	B	－	－	※2
	0xD1DB	B	－	－	※2
	0xD1DD	B	－	－	※2
	0xD1DF	B	－	－	※2
	0xD1E1	B	－	－	※2
	0xD1E3	B	－	－	※2
	0xD1E5	B	－	－	※2
	0xD1E7	B	－	－	※2
	0xD1E9	B	－	－	※2
	0xD1EB	B	－	－	※2
	0xD1ED	B	－	－	※2
	0xD1EF	B	－	－	※2
5	0xD1FE	A	－	－	－
6	0xD1FF	A	－	－	※2

（凡例）A：標準　B：詳細　－：該当なし

注※1　図 8-72 中の番号と対応しています。

注※2　正常に処理された場合，入り口時刻が表示されます。例外が発生した場合，入り口時刻と例外が表示されます。

8.19　TP1 インバウンド連携機能のトレース取得ポイント

ここでは，TP1 インバウンド連携機能のトレース取得ポイントと，取得できるトレース情報について説明します。

8.19.1　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

(1)　TP1 インバウンド連携機能でのトレース取得ポイントの詳細

TP1 インバウンド連携機能でのトレース取得ポイントの詳細について次の表に示します。

表 8-119　TP1 インバウンド連携機能でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x842F	19	リソースアダプタから Message-driven Bean のメッセージリスナのメソッドを呼び出した直後	A
0x8430	22	リソースアダプタから呼び出した Message-driven Bean のメッセージリスナのメソッドのリターン直前	A
0x8431	20	EJB コンテナが Message-driven Bean のメッセージリスナのメソッドをコールバックする直前	A
0x8432	21	Message-driven Bean のメッセージリスナのメソッドのコールバックからリターンした直後	A
0x8825	17	トランザクションの recreate 処理直前	B
0x8826	18	トランザクションの recreate 処理直後	B
0x8B86	14	javax.resource.spi.work.WorkManager のメソッドの呼び出し直後	A
0x8B87	15	javax.resource.spi.work.WorkManager のメソッドのリターン直前	A
0x8B8A	16	javax.resource.spi.work.Work のメソッド呼び出し直前	A
0x8B8B	29	javax.resource.spi.work.Work のメソッドのリターン直後	A
0x8B8C	33	javax.resource.spi.XATerminator.prepare（Xid xid）呼び出し直後	B
0x8B8D	34	javax.resource.spi.XATerminator.prepare（Xid xid）リターン直前	B
0x8B8E	−	javax.resource.spi.XATerminator.commit（Xid xid，boolean onePhase）呼び出し直後	B
0x8B8F	−	javax.resource.spi.XATerminator.commit（Xid xid，boolean onePhase）リターン直前	B

イベント ID	図中の番号※	トレース取得ポイント	レベル
0x8B90	−	javax.resource.spi.XATerminator.rollback（Xid xid）呼び出し直後	B
0x8B91	−	javax.resource.spi.XATerminator.rollback（Xid xid）リターン直前	B
0x8B92	−	javax.resource.spi.XATerminator.forget（Xid xid）呼び出し直後	B
0x8B93	−	javax.resource.spi.XATerminator.forget（Xid xid）リターン直前	B
0xAA00	8	電文受信開始直後	A
0xAA01	12	RPC 応答送信完了直後	A
0xAA02	4	OpenTP1 からの電文読み込み開始直後	A
0xAA03	10	OpenTP1 とのコネクション切断直前（OpenTP1 の定義に rpc_close_after_send=Y が指定されている場合）	A
0xAA04	5	ソケットの read 呼び出し直前	B
0xAA05	6	ソケットの read 完了直後	B
0xAA06	13	電文受信が完了し，スケジュールキュー投入直前	A
0xAA08	23	RPC 応答電文の送信開始直前	A
0xAA09	27	応答用のコネクション切断直前（TP1 インバウンドアダプタのプロパティに rpc_close_after_send=true が指定されている場合）	A
0xAA0A	25	ソケットの write 呼び出し直前	B
0xAA0B	26	ソケットの write 完了直後	B
0xAA0C	30	送信失敗。リトライ待機開始直前	A
0xAA0D	31	リトライ開始直前	A
0xAA10	11	• 不正電文を受信してコネクションを切断した直後 • 受信タイムアウトが発生してコネクションを切断した直後 • 受信タイマ監視スレッドで RPC 要求の受信タイムアウトが発生し電文が破棄された直後	A
0xAA11	28	RPC 応答送信失敗直後	A
0xAA12	2	OpenTP1 とのコネクションを受信監視対象に追加する直前	B
0xAA13	3	OpenTP1 とのコネクションで電文の受信を検知した時	B
0xAA14	32	同期点処理での OpenTP1 からの受信完了直後	A
0xAA15	35	同期点処理での OpenTP1 への送信完了直後，または送信できないエラーが発生した直後	A
0xAA16	1	OpenTP1 とのコネクション確立直後	A
0xAA17	9	OpenTP1 とのコネクション切断直前，または通信エラー発生による OpenTP1 とのコネクション切断検知直後	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xAA18	7	受信した電文を解析した直後	B
0xAA19	24	電文を送信する直前	B

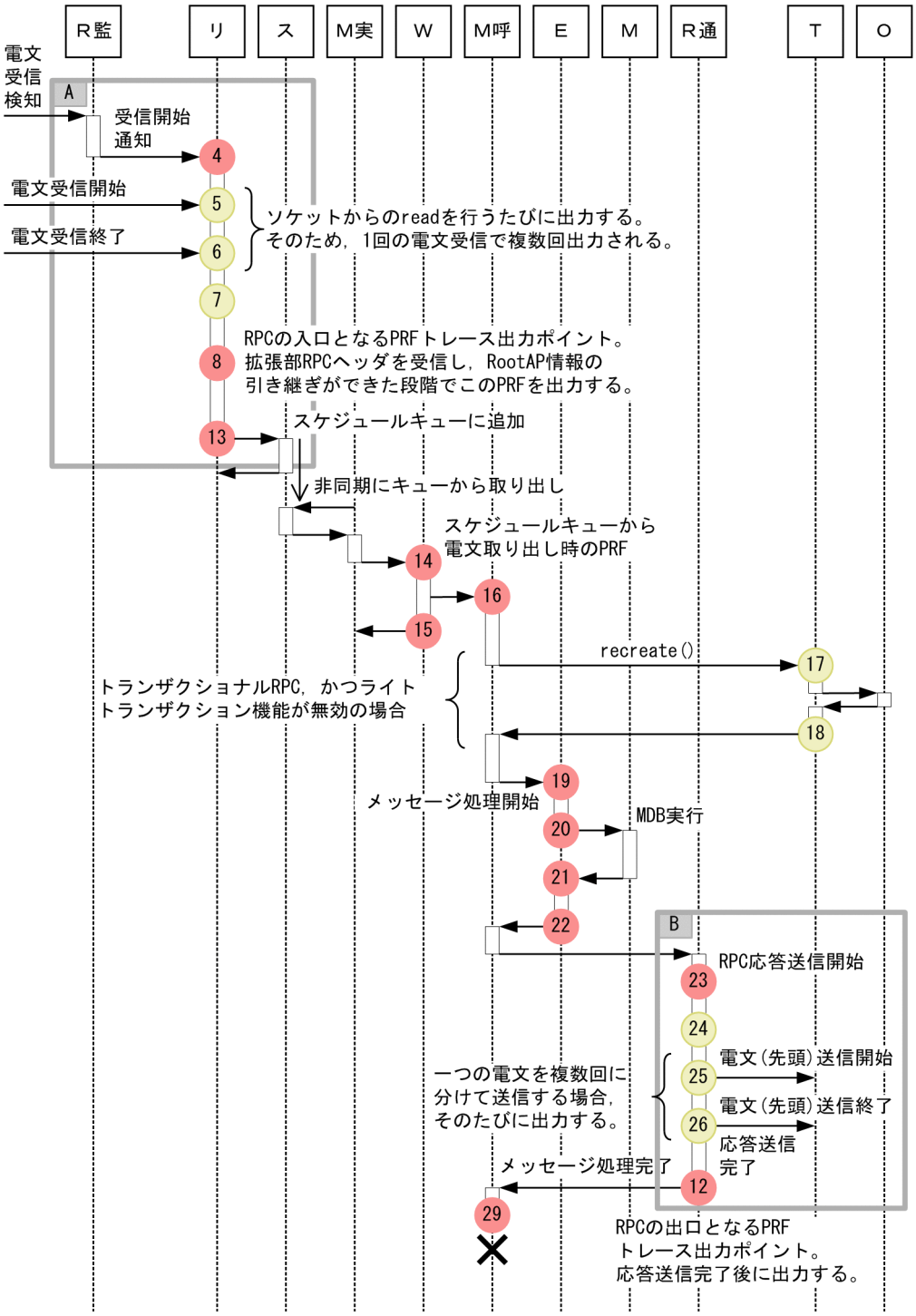
（凡例）A：標準　B：詳細　－：該当なし

注※　図 8-73～図 8-79 の番号と対応しています。

(2)　TP1 インバウンド連携機能でのトレース取得ポイント

TP1 インバウンド連携機能でのトレース取得ポイントの全体を次の図に示します。なお，図中の A，および B については，さらに詳細なトレース取得ポイントがあります。

図 8-73　TP1 インバウンド連携機能のトレース取得ポイントの全体



（凡例）

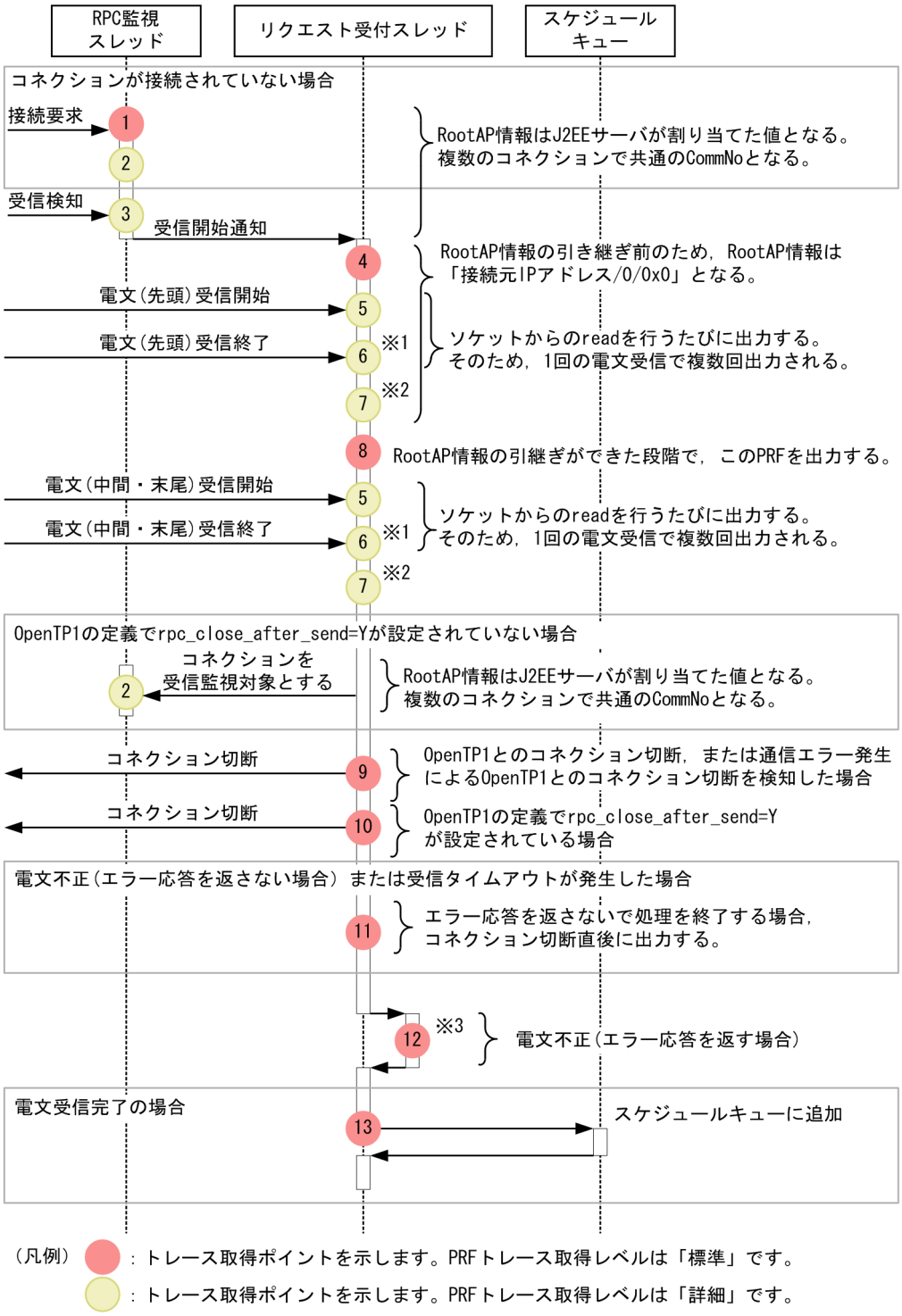
R監	：RPC監視スレッドを表します。
リ	：リクエスト受付スレッドを表します。
ス	：スケジュールキューを表します。
M実	：MDB（サービス）実行制御スレッドを表します。
W	：WorkManagerを表します。
M呼	：MDB（サービス）呼び出しスレッドを表します。
E	：EJBコンテナを表します。
M	：MDBを表します。
R通	：RPC通信機能を表します。
T	：TransactionManagerを表します。
O	：OTSを表します。

：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

：トレース取得ポイントを示します。PRFトレース取得レベルは「詳細」です。

図中のＡおよびＢの詳細なトレース取得ポイントについて，次の二つの図で示します。

図 8-74　A 部分のトレース取得ポイント



注※1　read で例外が発生した場合，出力しません。

注※2　電文受信後に出力します。分割された電文の場合，分割された電文ごとに出力します。ただし，電文不正時は出力しません。

注※3　RPC 応答送信処理（0xAA01 以外の PRF トレース取得ポイント）については，図 8-75 を参照してください。

RPC 受信時の PRF トレース取得ポイントは，図 8-74 に加えて，受信タイマ監視スレッドが RPC 要求の受信タイムアウトを検知し，電文を破棄した直後に出力する 0xAA10 があります。

図 8-75　B 部分のトレース取得ポイント

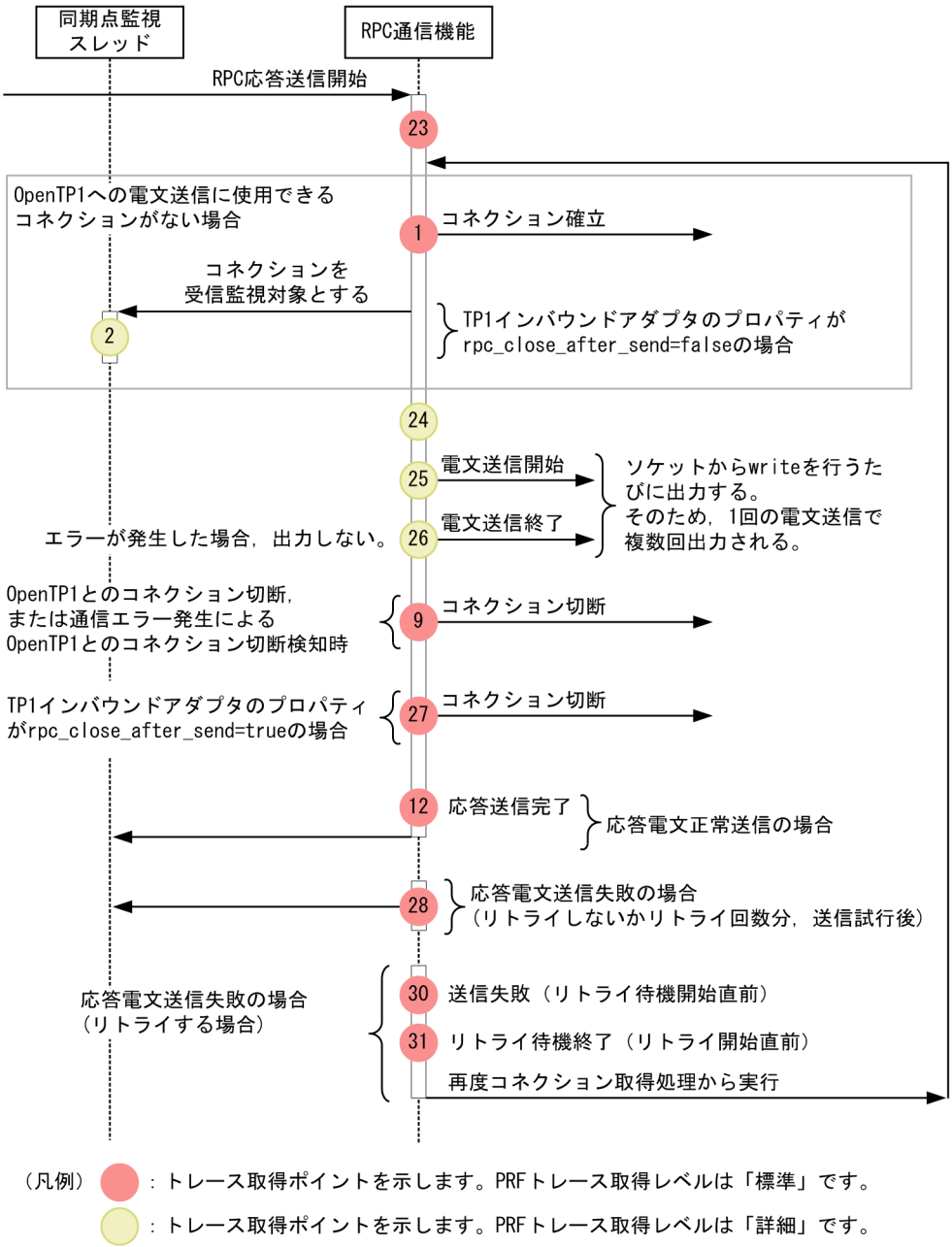
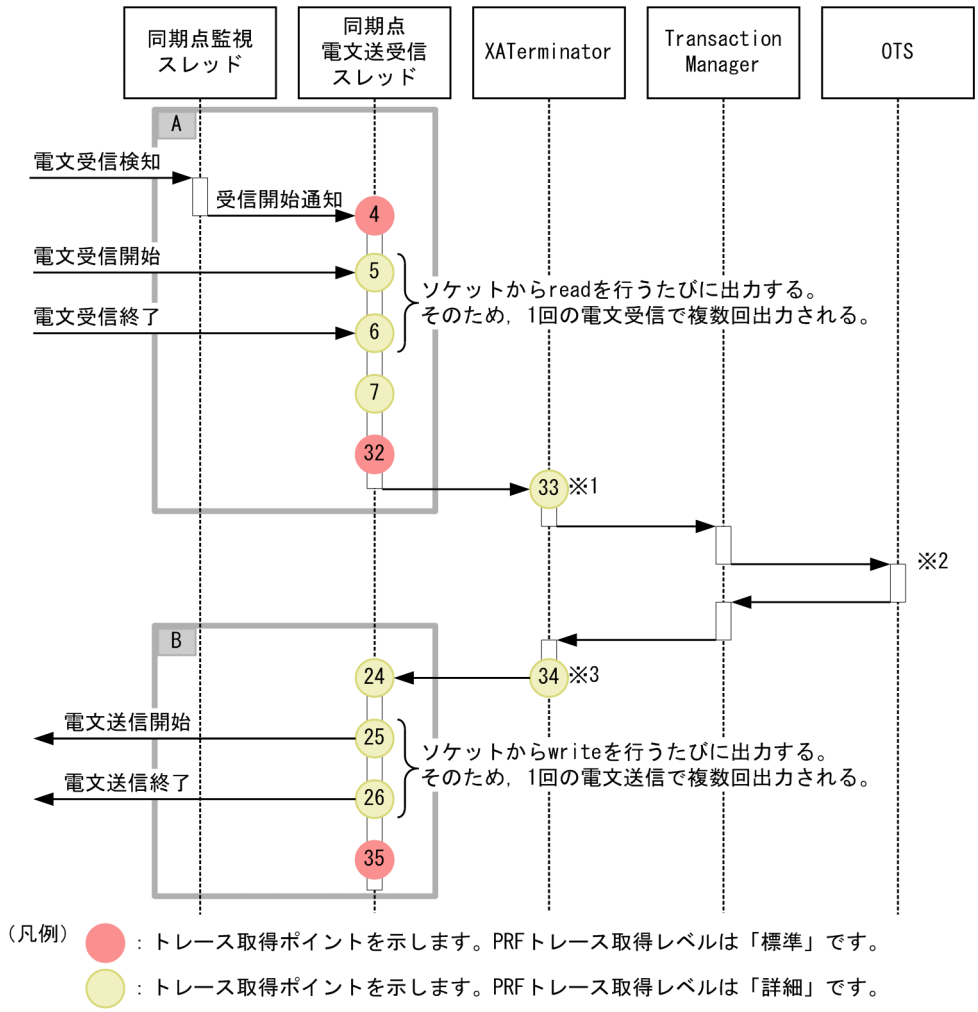


図 8-75 の応答送信時の PRF トレース出力ポイントは，RPC 通信時だけでなく，J2EE アプリケーション停止時にも出力します。このときのルートアプリケーション情報は，OpenTP1 から引き継いだルートアプリケーション情報ではなく，J2EE サーバによって生成されたルートアプリケーション情報です。そのため，J2EE アプリケーション停止時の PRF トレース出力ポイントと RPC 通信時のほかの PRF トレース出力ポイントとの対応づけは，J2EE アプリケーション停止時の 0xAA08 のインタフェース名に出力するルートアプリケーション情報を使用します。

(3)　同期点処理でのトレース取得ポイント

同期点処理でのトレース取得ポイントの全体を次の図に示します。なお，図中の A，および B については，さらに詳細なトレース取得ポイントがあります。

図 8-76　同期点処理のトレース取得ポイントの全体



注※1

処理要求が XATerminator に移った時点で，メソッドごとに出力されるイベント ID が異なります。各メソッドと出力されるイベント ID の対応を次の表に示します。

表 8-120　メソッドとイベント ID の対応

メソッド	イベント ID	図中の番号
prepare メソッド	0x8B8C	33
commit メソッド	0x8B8E	－
rollback メソッド	0x8B90	－
forget メソッド	0x8B92	－

（凡例）－：図中では使用していません。

注※2

OTS の出力ポイントについては，「8.14　OTS のトレース取得ポイント」を参照してください。

注※3

処理要求が XATerminator から同期点電文送受信スレッドへ移る時点で，メソッドごとに出力されるイベント ID が異なります。各メソッドと出力されるイベント ID の対応を次の表に示します。

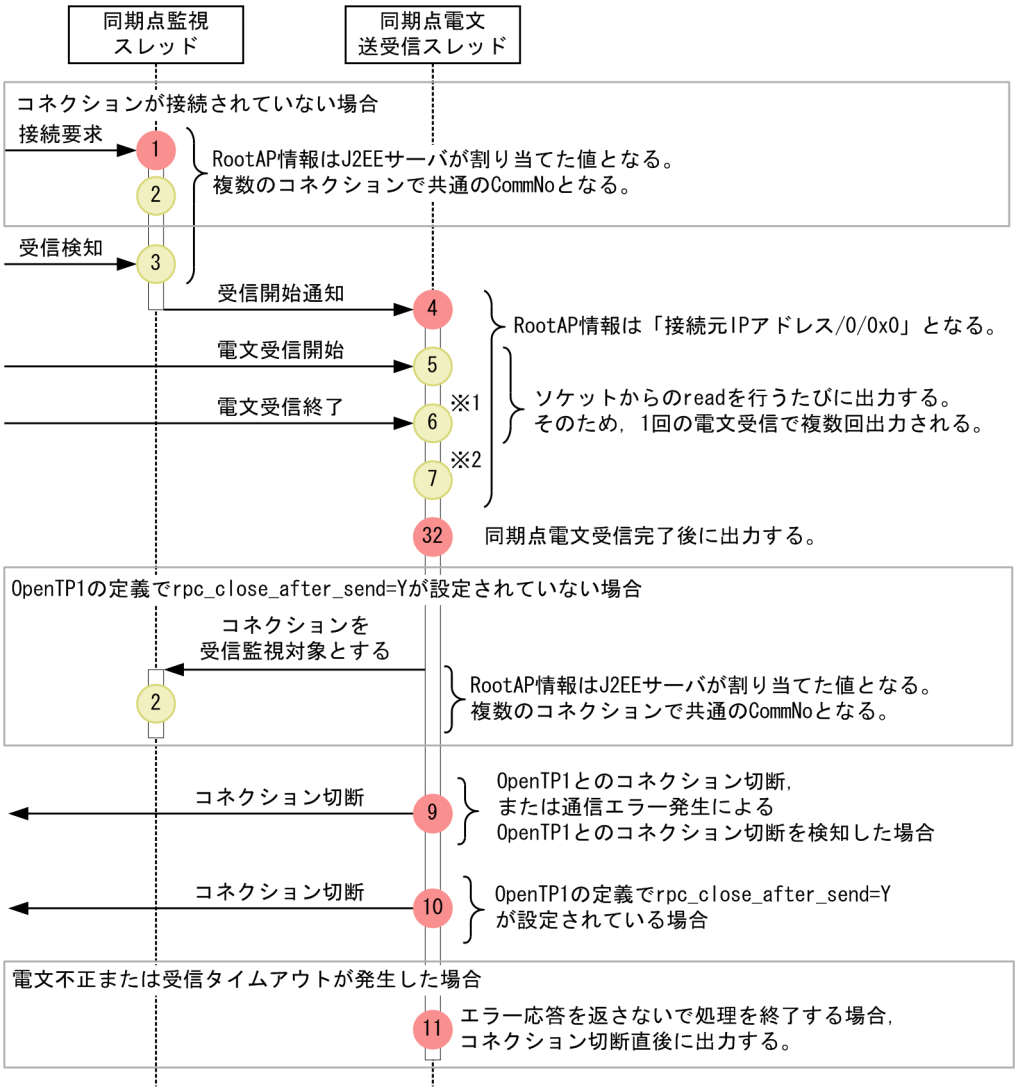
表 8-121　メソッドとイベント ID の対応

メソッド	イベント ID	図中の番号
prepare メソッド	0x8B8D	34
commit メソッド	0x8B8F	－
rollback メソッド	0x8B91	－
forget メソッド	0x8B93	－

（凡例）－：図中では使用していません。

図中の A および B の詳細なトレース取得ポイントについて，次の二つの図で示します。

図 8-77　同期点処理の A 部分のトレース取得ポイント



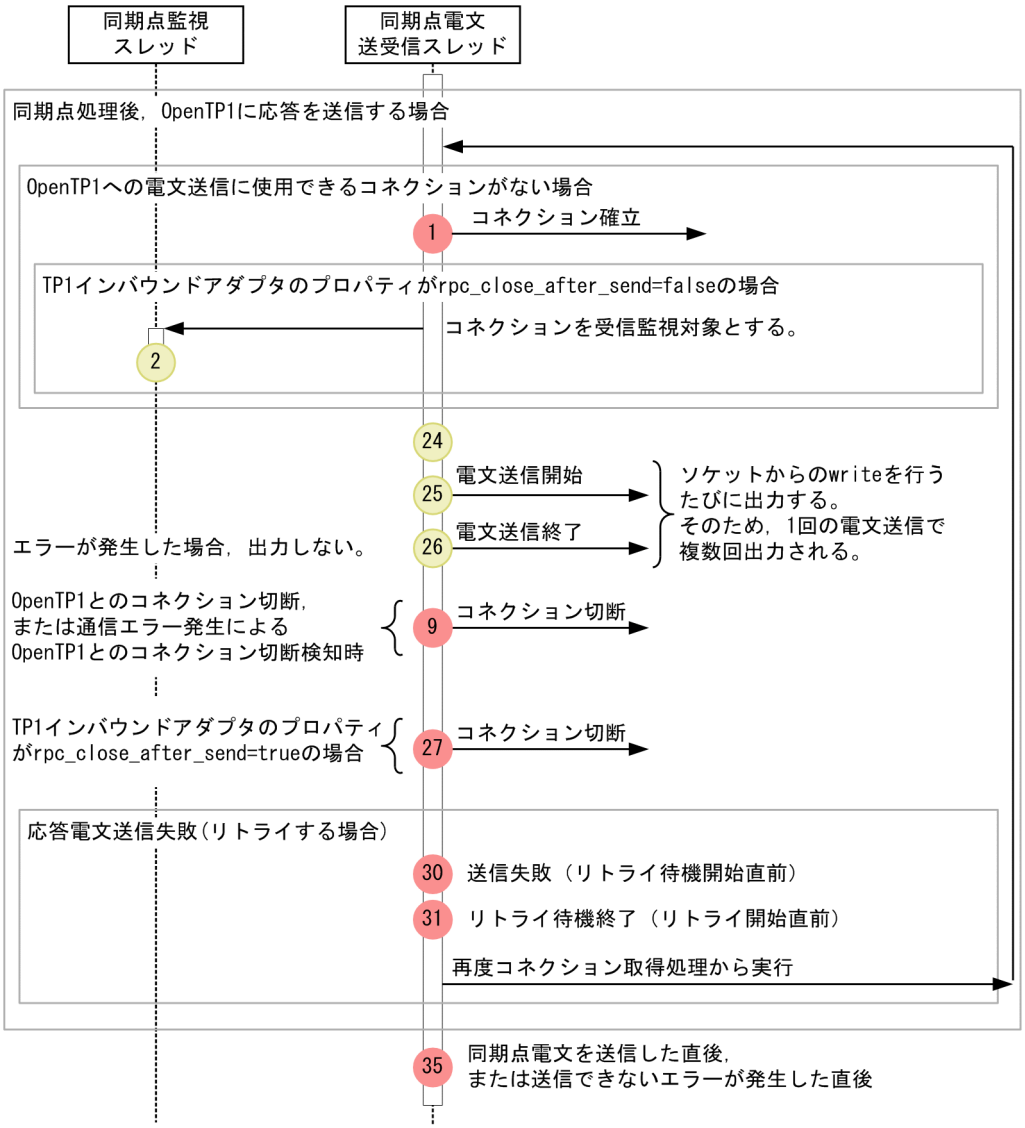
（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

　　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

注※1　read で例外が発生した場合，出力しません。

注※2　電文受信後に出力します。分割された電文の場合，分割された電文ごとに出力します。ただし，電文不正時は出力しません。

図 8-78　同期点処理の B 部分のトレース取得ポイント

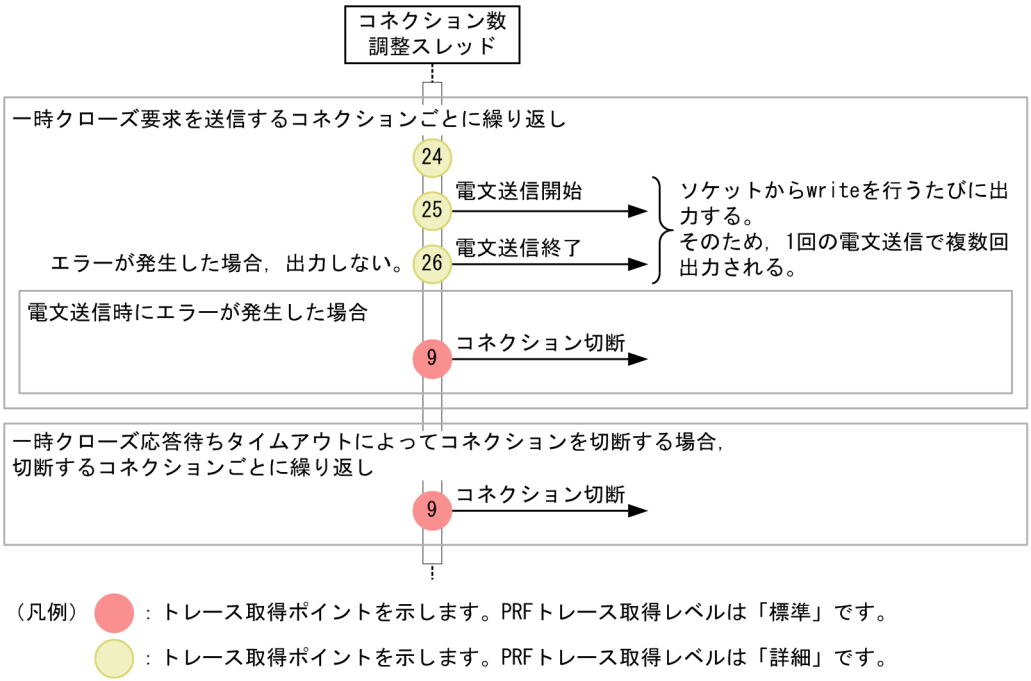


（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

　　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

コネクション数調整スレッドのトレース取得ポイントを次の図で示します。

図 8-79　コネクション数調整スレッドのトレース取得ポイント



8.19.2　取得できるトレース情報

TP1 インバウンド連携機能で取得できるトレース情報を次の表に示します。

表 8-122　TP1 インバウンド連携機能で取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xAA16	A	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	−	TP1 インバウンドアダプタ側の Listen ポート番号
2	0xAA12	B	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	−	TP1 インバウンドアダプタ側の Listen ポート番号
3	0xAA13	B	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	−	TP1 インバウンドアダプタ側の Listen ポート番号
4	0xAA02	A	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	−	TP1 インバウンドアダプタ側の Listen ポート番号

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
5	0xAA04	B	要求サイズ	−	−
6	0xAA05	B	読み込めたサイズ	−	−
7	0xAA18	B	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	電文の種類	TP1 インバウンドアダプタ側の Listen ポート番号
8	0xAA00	A	サービスグループ名	サービス名	呼び出し元の OpenTP1 のノード識別子※3，トランザクショングローバル識別子(トランザクショナル RPC の場合)
9	0xAA17	A	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	−	TP1 インバウンドアダプタ側の Listen ポート番号
10	0xAA03	A	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	−	TP1 インバウンドアダプタ側の Listen ポート番号
11	0xAA10	A	サービスグループ名※4	サービス名※5	入り口時刻
12	0xAA01	A	サービスグループ名	サービス名	正常時：入り口時刻 異常時：入り口時刻およびエラー応答コード
13	0xAA06	A	サービスグループ名	サービス名	呼び出し元の OpenTP1 のノード識別子※3，キュー内の電文数
14	0x8B86	A	Work のクラス名	メソッド名	リソースアダプタの表示名
15	0x8B87	A	Work のクラス名	メソッド名	正常時：入り口時刻 異常時：入り口時刻および例外
16	0x8B8A	A	Work のクラス名	メソッド名	リソースアダプタの表示名

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
17	0x8825	B	－	－	OpenTP1 のグローバルトランザクション ID（バイト列）
18	0x8826	B	－	－	正常時：入り口時刻 異常時：入り口時刻および例外
19	0x842F	A	Bean 名	メソッド名	－
20	0x8431	A	Bean 名	メソッド名	－
21	0x8432	A	Bean 名	メソッド名	正常時：入り口時刻 異常時：入り口時刻および例外
22	0x8430	A	Bean 名	メソッド名	正常時：入り口時刻 異常時：入り口時刻および例外
23	0xAA08	A	ルートアプリケーション情報※6	－	－
24	0xAA19	B	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	電文の種類	TP1 インバウンドアダプタ側の Listen ポート番号
25	0xAA0A	B	書き込みサイズ	－	－
26	0xAA0B	B	書き込めたサイズ	－	－
27	0xAA09	A	OpenTP1 の IP アドレス：OpenTP1 の listen ポート番号※2	－	TP1 インバウンドアダプタ側の Listen ポート番号
28	0xAA11	A	サービスグループ名	サービス名	入り口時刻
29	0x8B8B	A	Work のクラス名	メソッド名	正常時：入り口時刻 異常時：入り口時刻および例外
30	0xAA0C	A	－	－	－
31	0xAA0D	A	－	－	－

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
32	0xAA14	A	OpenTP1 のノード識別子	次のどれかを出力します。 • prepare • commit • rollback	トランザクショングローバル識別子
33	0x8B8C	B	−	−	XATerminator の引数 xid のグローバルトランザクション ID（バイト列）
34	0x8B8D	B	−	−	正常時：入り口時刻 異常時：入り口時刻および例外
35	0xAA15	A	−	正常時は次のどれかを出力します※7。 • prepared • read only • committed • rolled back 異常時は何も出力しません。	入口時刻
−	0x8B8E	B	−	−	XATerminator の引数 xid のグローバルトランザクション ID（バイト列）
−	0x8B8F	B	−	−	正常時：入り口時刻 異常時：入り口時刻および例外
−	0x8B90	B	−	−	XATerminator の引数 xid のグローバルトランザクション ID（バイト列）
−	0x8B91	B	−	−	正常時：入り口時刻 異常時：入り口時刻および例外
−	0x8B92	B	−	−	XATerminator の引数 xid のグローバルトランザ

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					クション ID（バイト列）
−	0x8B93	B	−	−	正常時：入り口時刻 異常時：入り口時刻および例外

（凡例）A：標準　B：詳細　−：該当なし

注※1　図 8-73～図 8-79 中の番号と対応しています。

注※2　OpenTP1 の listen ポートが判別できない場合は「OpenTP1 の IP アドレス：−」と出力します。

注※3　呼び出し元の OpenTP1 から PRF 情報を引き継いでいる場合は，「root RPC ノード識別子」が出力されます。引き継いでいない場合は，「呼出元ノード ID」が出力されます。

注※4　インタフェース名に出力されるサービスグループ名が判別できない場合は「−」が出力されます。

注※5　オペレーション名に出力されるサービス名が判別できない場合は「−」が出力されます。

注※6　J2EE アプリケーションの停止時だけ出力します。出力する内容を以下に示します。

- 呼び出し元の OpenTP1 から PRF 情報を引き継いでいる場合は，OpenTP1 から引き継いだルートアプリケーション情報を出力します。
- 呼び出し元の OpenTP1 から PRF 情報を引き継いでいない場合は，TP1 インバウンドアダプタが採番したルートアプリケーション情報を出力します。

注※7　それぞれ次の条件の場合に出力します。

- prepared：プリペアが完了した場合
- read only：リソースへのアクセスが参照の場合などプリペアの結果がリードオンリーだった場合。または実行した Message-driven Bean（サービス）のトランザクションの設定が CMT NotSupported，または BMT の場合
- committed：コミットが完了した場合
- rolled back：ロールバックが完了した場合。プリペアやコミットの結果がロールバックとなった場合も含みます。

8.20　CJMS プロバイダのトレース取得ポイント

ここでは，CJMS プロバイダのトレース取得ポイントと，取得できるトレース情報について説明します。

8.20.1　JMS ConnectionFactory インタフェースのトレース取得ポイントと取得できるトレース情報

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-123　JMS ConnectionFactory インタフェースでのトレース取得ポイントの詳細

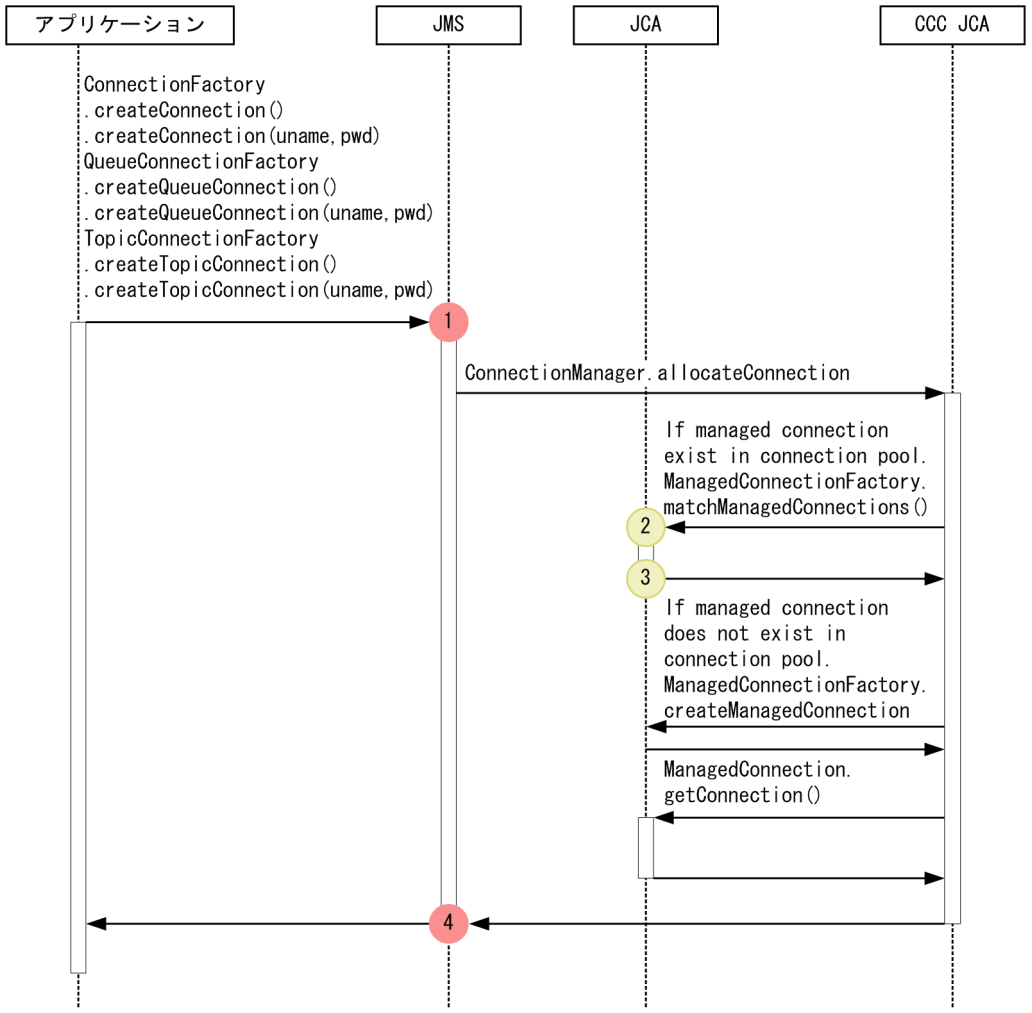
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA600	1	ConnectionFactory.createConnection()の処理開始	A
0xA601	4	ConnectionFactory.createConnection()の処理終了	A
0xA602	1	ConnectionFactory.createConnection(uname,pwd)の処理開始	A
0xA603	4	ConnectionFactory.createConnection(uname,pwd)の処理終了	A
0xA604	1	QueueConnectionFactory.createQueueConnection()の処理開始	A
0xA605	4	QueueConnectionFactory.createQueueConnection()の処理終了	A
0xA606	1	QueueConnectionFactory.createQueueConnection(uname,pwd)の処理開始	A
0xA607	4	QueueConnectionFactory.createQueueConnection(uname,pwd)の処理終了	A
0xA608	1	TopicConnectionFactory.createTopicConnection()の処理開始	A
0xA609	4	TopicConnectionFactory.createTopicConnection()の処理終了	A
0xA60A	1	TopicConnectionFactory.createTopicConnection(uname,pwd)の処理開始	A
0xA60B	4	TopicConnectionFactory.createTopicConnection(uname,pwd)の処理終了	A
0xA60C	2	ManagedConnectionFactory.matchManagedConnections()の処理開始	B
0xA60D	3	ManagedConnectionFactory.matchManagedConnections()の処理終了	B

（凡例）A：標準　B：詳細

注※　図 8-80 の番号と対応しています。

JMS ConnectionFactory インタフェースでのトレース取得ポイントを次の図に示します。

図 8-80　JMS ConnectionFactory インタフェースのトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

　　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(2)　取得できるトレース情報

JMS ConnectionFactory インタフェースで取得できるトレース情報（インタフェース名，オペレーション名，およびオプション）はありません。

8.20.2　JMS Connection インタフェースのトレース取得ポイントと取得できるトレース情報

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-124　JMS Connection インタフェースでのトレース取得ポイントの詳細

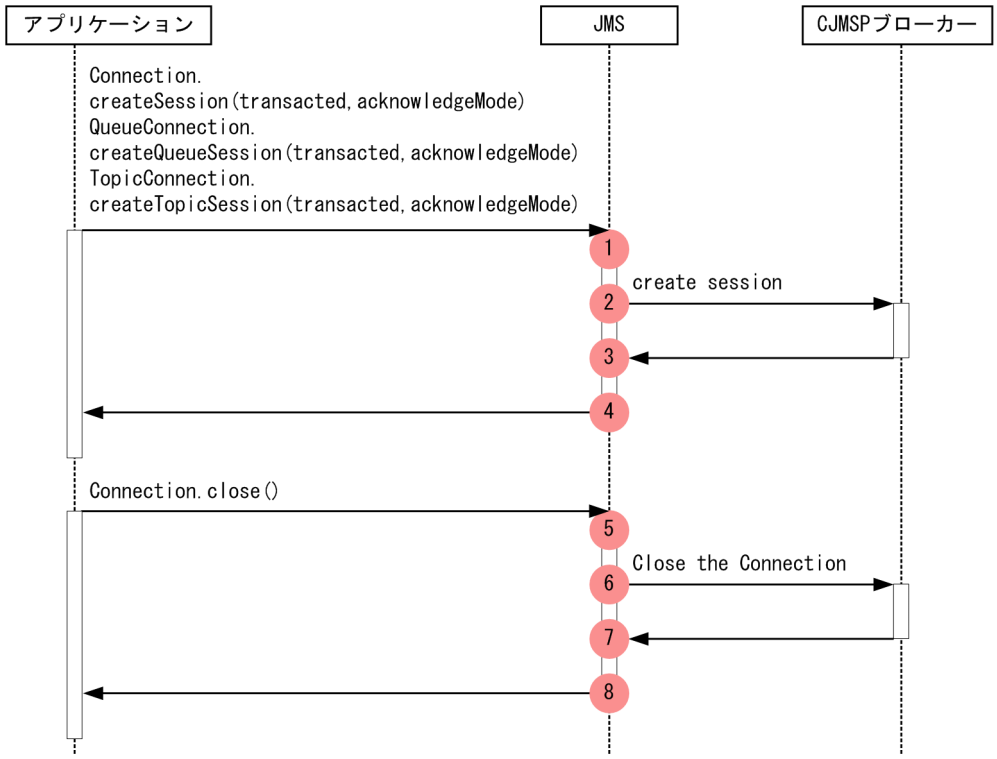
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA60E	1	Connection.createSession(transacted,acknowledgeMode)の処理開始	A
0xA60F	4	Connection.createSession(transacted,acknowledgeMode)の処理終了	A
0xA610	1	QueueConnection.createQueueSession(transacted,acknowledgeMode)の処理開始	A
0xA611	4	QueueConnection.createQueueSession(transacted,acknowledgeMode)の処理終了	A
0xA612	1	TopicConnection.createTopicSession(transacted,acknowledgeMode)の処理開始	A
0xA613	4	TopicConnection.createTopicSession(transacted,acknowledgeMode)の処理終了	A
0xA614	5	Connection.close()処理の開始	A
0xA615	8	Connection.close()処理の終了	A
0xA616	2	セッション作成直前の CJMSP ブローカーによる呼び出し	A
0xA617	3	セッション作成直後の CJMSP ブローカーによる呼び出し	A
0xA618	6	コネクションの切断直前の CJMSP ブローカーによる呼び出し	A
0xA619	7	コネクションの切断直後の CJMSP ブローカーによる呼び出し	A

（凡例）A：標準

注※　図 8-81 の番号と対応しています。

JMS Connection インタフェースでのトレース取得ポイントを次の図に示します。

図 8-81　JMS Connection インタフェースのトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

JMS Connection インタフェースで取得できるトレース情報を次の表に示します。

表 8-125　JMS Connection インタフェースで取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xA60E	A	トランザクション	Ack モード	−
	0xA610	A			
	0xA612	A			
2	0xA616	A	−	−	−
3	0xA617	A	−	−	−
4	0xA60F	A	−	−	−
	0xA611	A			
	0xA613	A			
5	0xA614	A	−	−	−
6	0xA618	A	−	−	−

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
7	0xA619	A	−	−	−
8	0xA615	A	−	−	−

（凡例）A：標準　−：該当なし

注※　図 8-81 の番号と対応しています。

注意事項

イベント ID の 0xA618 と 0xA619 は Message-driven Bean アプリケーションを停止している間にも出力されます。

参考

Acknowledgement モードは数値で出力されます。数値は，Acknowledgement モードの種類ごとに対応しています。対応を次に示します。

- AUTO_ACKNOWLEDGE = 1
- CLIENT_ACKNOWLEDGE = 2
- DUPS_OK_ACKNOWLEDGE = 3
- SESSION_TRANSACTED = 0

それぞれのモードの数値は JMS 仕様で定義されています。

8.20.3　JMS セッションインタフェースのトレース取得ポイントと取得できるトレース情報

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-126　JMS セッションインタフェースでのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA61E	5	Session.createSubscriber(topic)の処理開始	A
0xA61F	8	Session.createSubscriber(topic)の処理終了	A
0xA620	1	Session.createBrowser(queue)の処理開始	A
0xA621	4	Session.createBrowser(queue)の処理終了	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA622	1	Session.createBrowser(queue,selector)の処理開始	A
0xA623	4	Session.createBrowser(queue,selector)の処理終了	A
0xA624	5	Session.createSubscriber(topic,selector,noLocal)の処理開始	A
0xA625	8	Session.createSubscriber(topic,selector,noLocal)の処理終了	A
0xA626	5	Session.createDurableSubscriber(topic,name)の処理開始	A
0xA627	8	Session.createDurableSubscriber(topic,name)の処理終了	A
0xA628	5	Session.createDurableSubscriber(topic,name,selector,noLocal)の処理開始	A
0xA629	8	Session.createDurableSubscriber(topic,name,selector,noLocal)の処理終了	A
0xA62A	5	Session.createConsumer(destination)の処理開始	A
0xA62B	8	Session.createConsumer(destination)の処理終了	A
0xA62C	5	Session.createConsumer(destination,selector)の処理開始	A
0xA62D	8	Session.createConsumer(destination,selector)の処理終了	A
0xA62E	5	Session.createConsumer(destination,selector,noLocal)の処理開始	A
0xA62F	8	Session.createConsumer(destination,selector,noLocal)の処理終了	A
0xA630	5	Session.createReceiver(queue)の処理開始	A
0xA631	8	Session.createReceiver(queue)の処理終了	A
0xA632	5	Session.createReceiver(queue,selector)の処理開始	A
0xA633	8	Session.createReceiver(queue,selector)の処理終了	A
0xA634	9	Session.createPublisher(topic)の処理開始	A
0xA635	12	Session.createPublisher(topic)の処理終了	A
0xA636	9	Session.createProducer(destination)の処理開始	A
0xA637	12	Session.createProducer(destination)の処理終了	A
0xA638	9	Session.createSender(queue)の処理開始	A
0xA639	12	Session.createSender(queue)の処理終了	A
0xA63A	13	Session.createQueue()の処理開始	A
0xA63B	16	Session.createQueue()の処理終了	A
0xA63C	13	Session.createTopic()の処理開始	A
0xA63D	16	Session.createTopic()の処理終了	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA63E	13	Session.createTemporaryQueue()の処理開始	A
0xA63F	16	Session.createTemporaryQueue()の処理終了	A
0xA640	13	Session.createTemporaryTopic()の処理開始	A
0xA641	16	Session.createTemporaryTopic()の処理終了	A
0xA642	17	Session.unsubscribe()の処理開始	A
0xA643	20	Session.unsubscribe()の処理終了	A
0xA644	21	Session.commit()の処理開始	A
0xA645	24	Session.commit()の処理終了	A
0xA646	21	Session.rollback()の処理開始	A
0xA647	24	Session.rollback()の処理終了	A
0xA648	25	Session.recover()の処理開始	A
0xA649	26	Session.recover()の処理終了	A
0xA64A	25	Session.close()の処理開始	A
0xA64B	26	Session.close()の処理終了	A
0xA64C	2	ブラウザーの送信先認証を呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA64D	3	ブラウザーの送信先認証を呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA64E	6	コンシューマーを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA64F	7	コンシューマーを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA650	10	プロデューサーを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA651	11	プロデューサーを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA652	14	送信先作成を呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA653	15	送信先作成を呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA654	18	永続化サブスクライバー作成を呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA655	19	永続化サブスクライバー作成を呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA656	22	セッションコミットを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA657	23	セッションコミットを呼び出した直後の CJMSP ブローカーによる呼び出し	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA658	22	セッションロールバックを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA659	23	セッションロールバックを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA65A	22	セッション回復を呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA65B	23	セッション回復を呼び出した直後の CJMSP ブローカーによる呼び出し	A

（凡例）A：標準

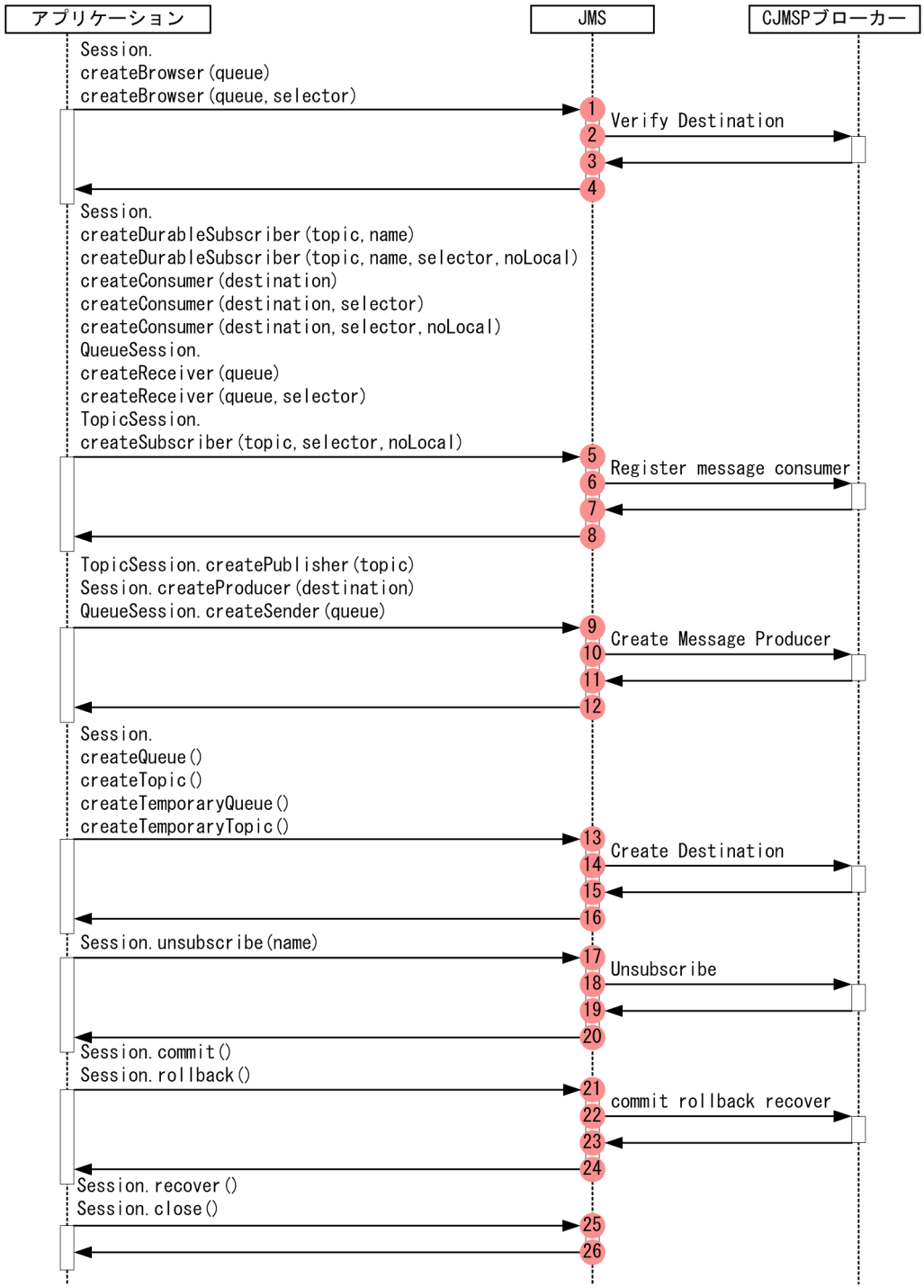
注※　図 8-82 の番号と対応しています。

注意事項

- イベント ID の 0xA644 と 0xA645，0xA646 と 0x647 に対応するメソッドである Session.commit()および Session.rollback()はサポートされていません。
- イベント ID の 0xA65A と 0xA65B は出力されません。XA トランザクションを使用したとき出力されます。

JMS セッションインタフェースでのトレース取得ポイントを次の図に示します。

図 8-82　JMS セッションインタフェースのトレース取得ポイント



（凡例）　●　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

参考

プロデューサーまたはコンシューマーの作成の間，destination が作成されます。このため，createProducer()メソッドおよび createConsumer()メソッドでは，イベント ID 0xa652 および 0xa653 が呼び出される可能性があります。

(2)　取得できるトレース情報

JMS セッションインタフェースで取得できるトレース情報を次の表に示します。

表 8-127　JMS セッションインタフェースで取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xA620	A	キュー名	－	－
	0xA622	A		セレクタ文字列	
2	0xA64C	A	－	－	－
3	0xA64D	A	－	－	－
4	0xA621	A	－	－	－
	0xA623	A			
5	0xA61E	A	トピック名	－	－
	0xA624	A		セレクタ文字列	
	0xA626	A		－	
	0xA628	A		セレクタ文字列	
	0xA62A	A	送信先名	－	
	0xA62C	A		セレクタ文字列	
	0xA62E	A			
	0xA630	A	キュー名	－	
	0xA632	A		セレクタ文字列	
6	0xA64E	A	－	－	－
7	0xA64F	A	－	－	－
8	0xA61F	A	－	－	－
	0xA625	A			
	0xA627	A			
	0xA629	A			
	0xA62B	A			
	0xA62D	A			
	0xA62F	A			
	0xA631	A			
	0xA633	A			

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
9	0xA634	A	トピック名	－	－
	0xA636	A	送信先名	－	－
	0xA638	A	キュー名	－	－
10	0xA650	A	－	－	－
11	0xA651	A	－	－	－
12	0xA635	A	－	－	－
	0xA637	A			
	0xA639	A			
13	0xA63A	A	キュー名	－	－
	0xA63C	A	トピック名		
	0xA63E	A	－		
	0xA640	A			
14	0xA652	A	－	－	－
15	0xA653	A	－	－	－
16	0xA63B	A	－	－	－
	0xA63D	A			
	0xA63F	A			
	0xA641	A			
17	0xA642	A	－	－	－
18	0xA654	A	－	－	－
19	0xA655	A	－	－	－
20	0xA643	A	－	－	－
21	0xA644	A	－	－	－
	0xA646	A			
	0xA648	A			
22	0xA656	A	－	－	－
	0xA658	A			
	0xA65A	A			
23	0xA657	A	－	－	－
	0xA659	A			

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xA65B	A			
24	0xA645	A	−	−	−
	0xA647	A			
	0xA649	A			
25	0xA64A	A	−	−	−
26	0xA64B	A	−	−	−

（凡例）A：標準　−：該当なし

注※　図 8-82 の番号と対応しています。

8.20.4　JMS メッセージ，プロデューサー，コンシューマーとキューブラウザーのトレース取得ポイントと取得できるトレース情報

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-128　JMS メッセージ，プロデューサー，コンシューマーとキューブラウザーでのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA65C	1	MessageProducer.send(msg)の処理開始	A
0xA65D	4	MessageProducer.send(msg)の処理終了	A
0xA65E	1	MessageProducer.send(msg,deliveryMode,priority,timeToLive)の処理開始	A
0xA65F	4	MessageProducer.send(msg,deliveryMode,priority,timeToLive)の処理終了	A
0xA660	1	MessageProducer.send(destination,msg)の処理開始	A
0xA661	4	MessageProducer.send(destination,msg)の処理終了	A
0xA662	1	MessageProducer.send(destination,msg,deliveryMode,priority,timeToLive)の処理開始	A
0xA663	4	MessageProducer.send(destination,msg,deliveryMode,priority,timeToLive)の処理終了	A
0xA664	9	MessageProducer.close()の処理開始	A
0xA665	10	MessageProducer.close()の処理終了	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA666	5	MessageConsumer.receive()の処理開始	A
0xA667	8	MessageConsumer.receive()の処理終了	A
0xA61A	5	MessageConsumer.receive(timeout)の処理開始	A
0xA61B	8	MessageConsumer.receive(timeout)の処理終了	A
0xA668	5	MessageConsumer.receiveNoWait()の処理開始	A
0xA669	8	MessageConsumer.receiveNoWait()の処理終了	A
0xA66A	9	MessageConsumer.close()の処理開始	A
0xA66B	10	MessageConsumel.close()の処理終了	A
0xA66C	11	QueueBrowser.getEnumeration()の処理開始	A
0xA66D	14	QueueBrowser.getEnumeration()の処理終了	A
0xA66E	15	Message.acknowledge()の処理開始	A
0xA66F	18	Message.acknowledge()の処理終了	A
0xA670	2	JMS メッセージの書き込みを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA671	3	JMS メッセージの書き込みを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA672	6	JMS メッセージの受信を呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA673	7	JMS メッセージの受信を呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA674	12	全 JMS メッセージの配信を呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA675	13	全 JMS メッセージの配信を呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA676	16	JMS メッセージの承認を呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA677	17	JMS メッセージの承認を呼び出した直後の CJMSP ブローカーによる呼び出し	A

（凡例）A：標準

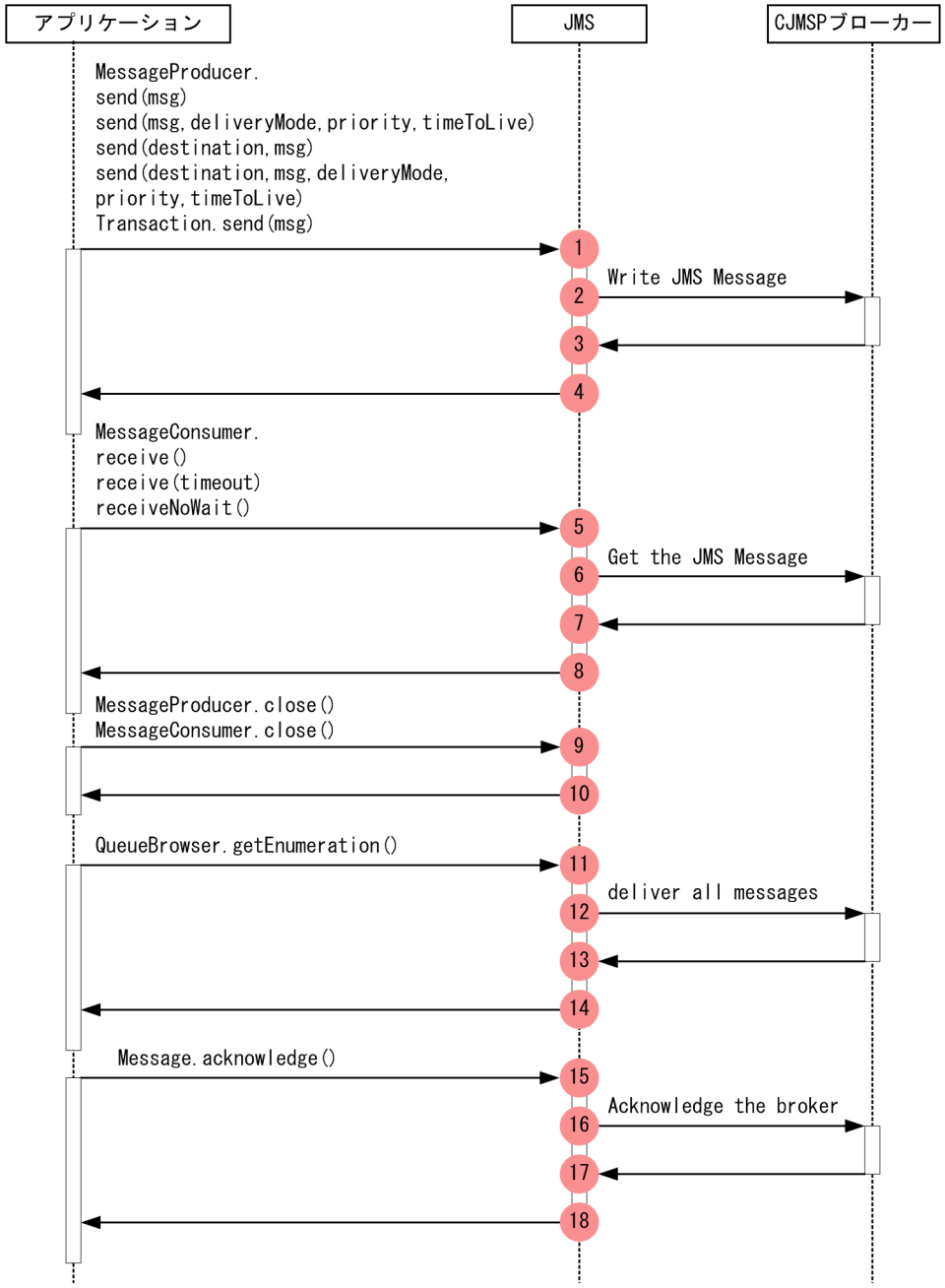
注※　図 8-83 の番号と対応しています。

注意事項

- トランザクションを使用した場合，または acknowledgement モードを CLIENT_ACKNOWLEDGE 以外に設定した場合は，メッセージを受信するとイベント ID の 0xA676 と 0xA677 が出力されます。
- Message-driven Bean を実行した場合は，イベント ID の 0xA672 と 0xA673 のルートアプリケーション情報が不正な状態になります。

JMS メッセージ，プロデューサー，コンシューマーとキューブラウザーでのトレース取得ポイントを次の図に示します。

図 8-83　JMS メッセージ，プロデューサー，コンシューマーとキューブラウザーのトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

JMS メッセージ，プロデューサー，コンシューマーとキューブラウザーで取得できるトレース情報を次の表に示します。

表 8-129　JMS メッセージ，プロデューサー，コンシューマーとキューブラウザーで取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xA65C	A	送信先名	－	－
	0xA65E	A		優先	
	0xA660	A		－	
	0xA662	A		優先	
2	0xA670	A	－	－	－
3	0xA671	A	－	－	－
4	0xA65D	A	－	－	－
	0xA65F	A			
	0xA661	A			
	0xA663	A			
5	0xA666	A	送信先名	0	－
	0xA61A	A		タイムアウト	
	0xA668	A		－	
6	0xA672	A	－	－	－
7	0xA673	A	－	－	－
8	0xA667	A	－	－	－
	0xA61B	A			
	0xA669	A			
9	0xA664	A	－	－	－
	0xA66A	A			
10	0xA665	A	－	－	－
	0xA66B	A			
11	0xA66C	A	－	－	－
12	0xA674	A	－	－	－
13	0xA675	A	－	－	－
14	0xA66D	A	－	－	－
15	0xA66E	A	－	－	－
16	0xA676	A	－	－	－

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
17	0xA677	A	−	−	−
18	0xA66F	A	−	−	−

（凡例）A：標準　−：該当なし

注※　図 8-83 の番号と対応しています。

8.20.5　CJMSP リソースアダプタと接続するときの CJMSP ブローカーのトレース取得ポイントと取得できるトレース情報

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-130　CJMSP リソースアダプタと接続するときの CJMSP ブローカーでのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA678	1	ResourceAdapter.start(ctx)の処理開始	A
0xA679	4	ResourceAdapter.start(ctx)の処理終了	A
0xA67A	5	ResourceAdapter.stop()の処理開始	A
0xA67B	8	ResourceAdapter.stop()の処理終了	A
0xA67C	2	hello メソッドを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA67D	3	hello メソッドを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA67E	6	good bye メソッドを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA67F	7	good bye メソッドを呼び出した直後の CJMSP ブローカーによる呼び出し	A

（凡例）A：標準

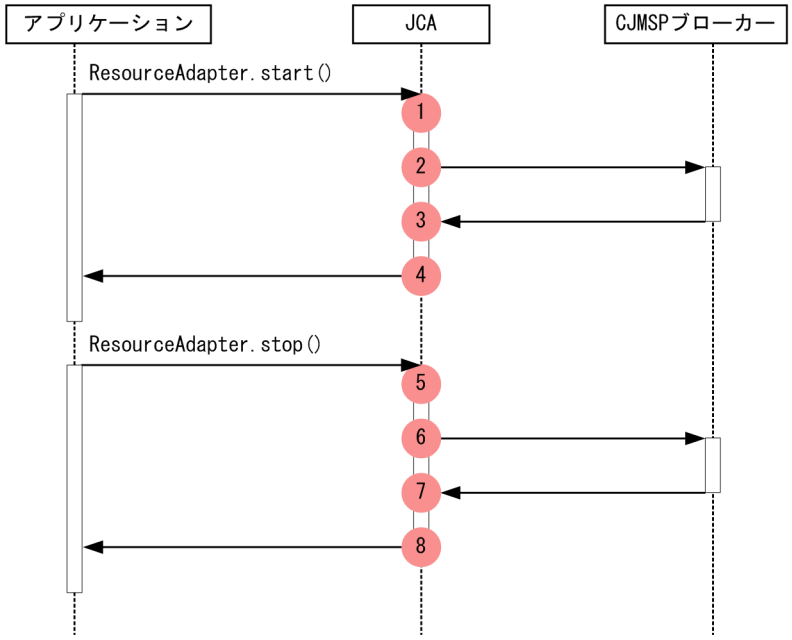
注※　図 8-84 の番号と対応しています。

注意事項

イベント ID の 0xA67C と 0xA67D は CJMSP ブローカーと CJMSP リソースアダプタがコネクション接続をするための通信をしている間にも出力されます。

CJMSP リソースアダプタと接続するときの CJMSP ブローカーでのトレース取得ポイントを次の図に示します。

図 8-84　CJMSP リソースアダプタと接続するときの CJMSP ブローカーのトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

CJMSP リソースアダプタと接続するときの CJMSP ブローカーで取得できるトレース情報（インタフェース名，オペレーション名，およびオプション）はありません。

8.20.6　CJMSP リソースアダプタでのトランザクション管理のトレース取得ポイントと取得できるトレース情報

(1)　CJMSP リソースアダプタでのトランザクション管理（LocalTransaction の場合）

(a)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-131　CJMSP リソースアダプタでのトランザクション管理（LocalTransaction の場合）のトレース取得ポイントの詳細

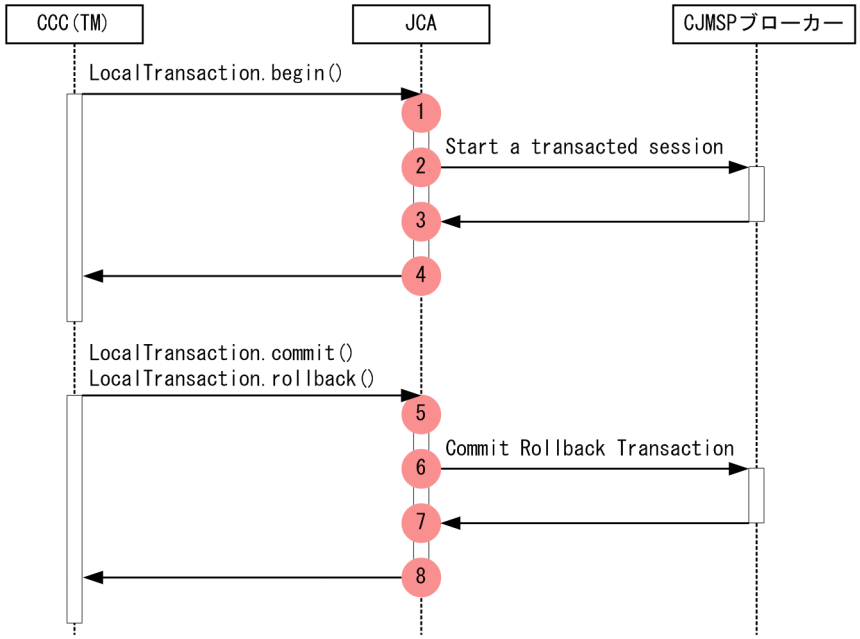
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA656	6	セッションコミットを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA657	7	セッションコミットを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA658	6	セッションロールバックを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA659	7	セッションロールバックを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA686	1	LocalTransaction.begin()の処理開始	A
0xA687	4	LocalTransaction.begin()の処理終了	A
0xA688	5	LocalTransaction.commit()の処理開始	A
0xA689	8	LocalTransaction.commit()の処理終了	A
0xA68A	5	LocalTransaction.rollback()の処理開始	A
0xA68B	8	LocalTransaction.rollback()の処理終了	A
0xA68C	2	トランザクション開始直前の CJMSP ブローカーによる呼び出し	A
0xA68D	3	トランザクション開始直後の CJMSP ブローカーによる呼び出し	A

（凡例）A：標準

注※　図 8-85 の番号と対応しています。

CJMSP リソースアダプタでのトランザクション管理（LocalTransaction の場合）のトレース取得ポイントを次の図に示します。

図 8-85　CJMSP リソースアダプタでのトランザクション管理（LocalTransaction の場合）のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b)　取得できるトレース情報

CJMSP リソースアダプタでのトランザクション管理（LocalTransaction の場合）で取得できるトレース情報（インタフェース名，オペレーション名，およびオプション）はありません。

(2)　CJMSP リソースアダプタでのトランザクション管理（XAResource の場合）

(a)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-132　CJMSP リソースアダプタでのトランザクション管理（XAResource の場合）でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA656	2	セッションコミットを呼び出す直前の CJMSP ブローカーによる呼び出し	A
0xA657	3	セッションコミットを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA658	2	セッションロールバックを呼び出す直前の CJMSP ブローカーによる呼び出し	A

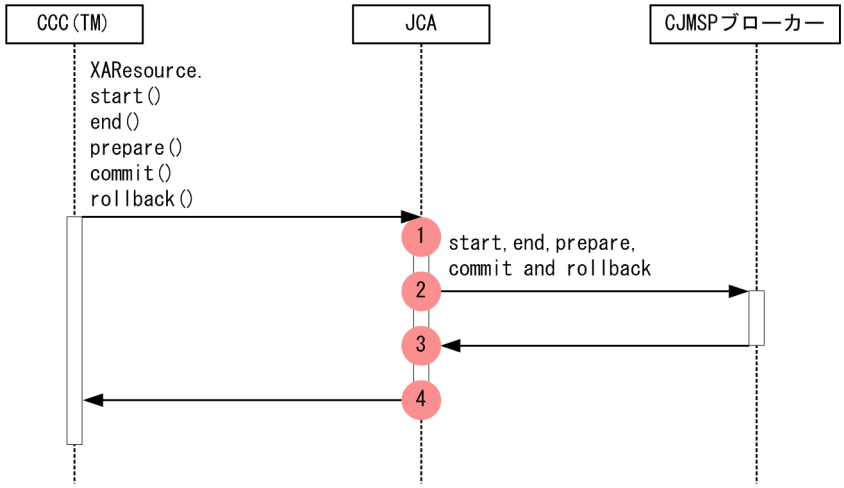
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA659	3	セッションロールバックを呼び出した直後の CJMSP ブローカーによる呼び出し	A
0xA68C	2	トランザクション開始直前の CJMSP ブローカーによる呼び出し	A
0xA68D	3	トランザクション開始直後の CJMSP ブローカーによる呼び出し	A
0xA692	1	XAResource.start()の処理開始	A
0xA693	4	XAResource.start()の処理終了	A
0xA694	1	XAResource.end()の処理開始	A
0xA695	4	XAResource.end()の処理終了	A
0xA696	1	XAResource.prepare()の処理開始	A
0xA697	4	XAResource.prepare()の処理終了	A
0xA698	1	XAResource.commit()の処理開始	A
0xA699	4	XAResource.commit()の処理終了	A
0xA69A	1	XAResource.rollback()の処理開始	A
0xA69B	4	XAResource.rollback()の処理終了	A
0xA69E	2	XAResource 停止を呼び出す直前のブローカーによる呼び出し	A
0xA69F	3	XAResource 停止を呼び出した直後のブローカーによる呼び出し	A
0xA6A0	2	XAResource 準備の直前のブローカーによる呼び出し	A
0xA6A1	3	XAResource 準備の直後のブローカーによる呼び出し	A

（凡例）A：標準

注※　図 8-86 の番号と対応しています。

CJMSP リソースアダプタでのトランザクション管理（XAResource の場合）でのトレース取得ポイントを次の図に示します。

図 8-86　CJMSP リソースアダプタでのトランザクション管理（XAResource の場合）のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(b)　取得できるトレース情報

CJMSP リソースアダプタでのトランザクション管理（XAResource の場合）で取得できるトレース情報を次の表に示します。

表 8-133　CJMSP リソースアダプタでのトランザクション管理（XAResource の場合）で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xA692	A	フラグ	−	−
	0xA694	A	フラグ		
	0xA696	A	−		
	0xA698	A	フラグ		
	0xA69A	A	−		
2	0xA656	A	−	−	−
	0xA658	A			
	0xA68C	A			
	0xA69E	A			
	0xA6A0	A			
3	0xA657	A	−	−	−
	0xA659	A			
	0xA68D	A			

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
	0xA69F	A			
	0xA6A1	A			
4	0xA693	A	－	－	－
	0xA695	A			
	0xA697	A			
	0xA699	A			
	0xA69B	A			

（凡例）A：標準　－：該当なし

注※　図 8-86 の番号と対応しています。

8.20.7　CJMSP リソースアダプタからの Message-driven Bean のデプロイ時のトレース取得ポイントと取得できるトレース情報

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-134　CJMSP リソースアダプタからの Message-driven Bean のデプロイ時でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xA6A6	1	ResourceAdapter.endpointActivation(endpointFactory,spec)の処理開始	A
0xA6A7	2	ResourceAdapter.endpointActivation(endpointFactory,spec)の処理終了	A
0xA6A8	3	ResourceAdapter.endpointDeactivation(endpointFactory,spec)の処理開始	A
0xA6A9	4	ResourceAdapter.endpointDeactivation(endpointFactory,spec)の処理終了	A
0xA6AA	5	MessageListener.onMessage()の処理開始	A
0xA6AB	6	MessageListener.onMessage()の処理終了	A

（凡例）A：標準

注※　図 8-87 の番号と対応しています。

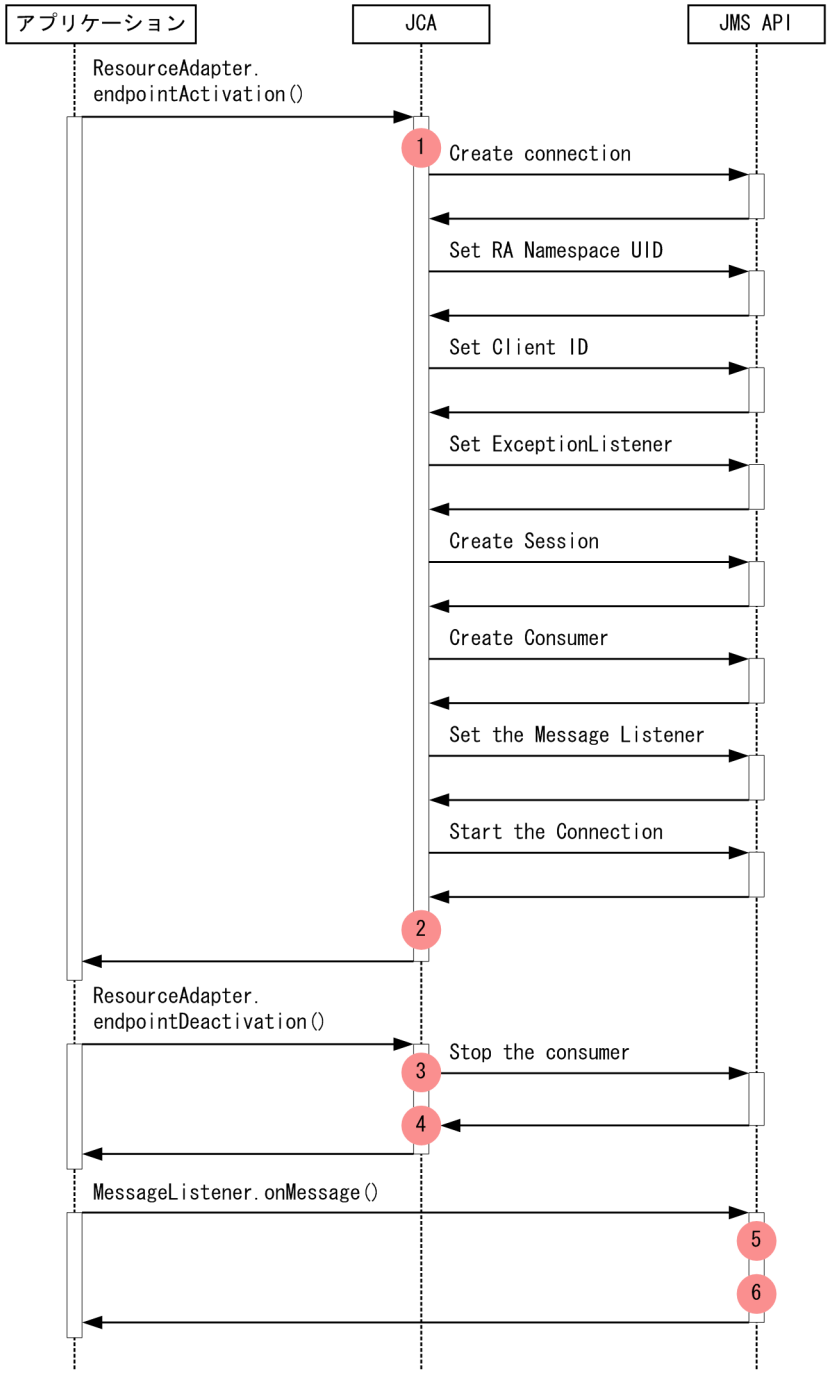
注意事項

endpointDeactivation()を呼び出すと 0xA67E と 0xA67F が出力されます。

endpointActivation()を呼び出すと 0xA67C と 0xA67D が出力されます。

CJMSP リソースアダプタからの Message-driven Bean のデプロイ時でのトレース取得ポイントを次の図に示します。

図 8-87　CJMSP リソースアダプタからの Message-driven Bean のデプロイ時のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

CJMSP リソースアダプタからの Message-driven Bean のデプロイ時で取得できるトレース情報（インタフェース名，オペレーション名，およびオプション）はありません。

8.21　JavaMail のトレース取得ポイント

ここでは，JavaMail のトレース取得ポイントと，取得できるトレース情報について説明します。

8.21.1　JavaMail 送信時のトレース取得ポイントと取得できるトレース情報

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-135　JavaMail 送信時のトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0xAD00	1	javax.mail.Transport クラスの connect(String host, int port, String user, String password)メソッドの入り口	A
0xAD01	22	javax.mail.Transport クラスの connect(String host, int port, String user, String password)メソッドの出口	A
0xAD02	23	javax.mail.Transport クラスの sendMessage(Message message, Address[] addresses)メソッドの入り口	A
0xAD03	38	javax.mail.Transport クラスの sendMessage(Message message, Address[] addresses)メソッドの出口	A
0xAD04	39	javax.mail.Transport クラスの close メソッドの入り口	A
0xAD05	44	javax.mail.Transport クラスの close メソッドの出口	A
0xAD06	2	コネクション取得処理の開始直前	A
0xAD07	3	コネクション取得処理の終了直後	A
0xAD08	26※2	すべての送信先情報の送信開始直前	A
0xAD09	29※2	すべての送信先情報の送信終了直後	A
0xAD0A	34※3	メール本体の送信開始直前	A
0xAD0B	35※3	メール本体の送信完了直後	A
0xAD0C	42	コネクション切断処理の開始直前	B
	6	コネクション切断処理の開始直前	B
0xAD0D	43	コネクション切断処理の終了直後	B
	7	コネクション切断処理の終了直後	B
0xAD0E	45	javax.mail.Transport クラスの send(Message msg, Address[] addresses)メソッド，javax.mail.Transport クラスの send(Message msg)メソッドの入り口	A

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0xAD0F	46	javax.mail.Transport クラスの send(Message msg, Address[] addresses)メソッド，javax.mail.Transport クラスの send(Message msg)メソッドの出口	A
0xAD10	8※4	EHLO または HELO コマンド発行の開始直前	A
0xAD11	9※4	EHELO または HELO コマンドのレスポンスを受信した直後	A
0xAD12	10	AUTH コマンド発行の開始直前	B
	14	ユーザ名，パスワードの送信の開始直前	B
	16	認証の終了通知開始直前	B
	18	ユーザ名の送信の開始直前	B
	20	パスワードの送信の開始直前	B
	24	MAIL コマンド発行の開始直前	B
	27※5	RCPT コマンド発行の開始直前	B
	30	RSET コマンド発行の開始直前	B
	32	DATA コマンド発行の開始直前	B
	36	メール本文送信終了通知の発行の開始直前	B
	40	QUIT コマンド発行の開始直前	B
	47※6	NOOP コマンドのレスポンスを受信した直後	B
	12	STARTTLS コマンド発行の開始直後	B
0xAD13	11	AUTH のレスポンスを受信した直後	B
	15	ユーザ名，パスワード送信のレスポンスを受信した直後	B
	17	認証の終了通知完了直後	B
	19	ユーザ名送信のレスポンスを受信した直後	B
	21	パスワード送信のレスポンスを受信した直後	B
	25	MAIL コマンドのレスポンスを受信した直後	B
	28※5	RCPT コマンドのレスポンスを受信した直後	B
	31	RSET コマンドのレスポンスを受信した直後	B
	33	DATA コマンドのレスポンスを受信した直後	B
	37	メール本文送信終了通知のレスポンスを受信した直後	B
	41	QUIT コマンドのレスポンスを受信した直後	B
	48※6	NOOP コマンド発行の開始直前	B
	13	STARTTLS コマンドのレスポンスを受信した直後	B

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0xAD14	4	接続時のサーバレスポンス取得を行う直前	B
0xAD15	5	接続時のサーバレスポンス取得直後	B

（凡例）A：標準　B：詳細

注※1

図 8-88，図 8-89，または図 8-90 中の番号と対応しています。

注※2

すべての送信先情報の送信開始～終了で取得します。

注※3

32 または 33 の取得ポイントのデータ送信処理では，送信するデータの読み込みとメールサーバへの送信を行います。例えば，添付ファイルが存在する場合は，添付ファイルのデータの読み込みとメールサーバへの送信を行います。

注※4

EHLO に失敗した場合，HELO で再度接続を試みるため，1 回のサーバ接続で EHLO と HELO の 2 回分のログが出力される場合があります。

注※5

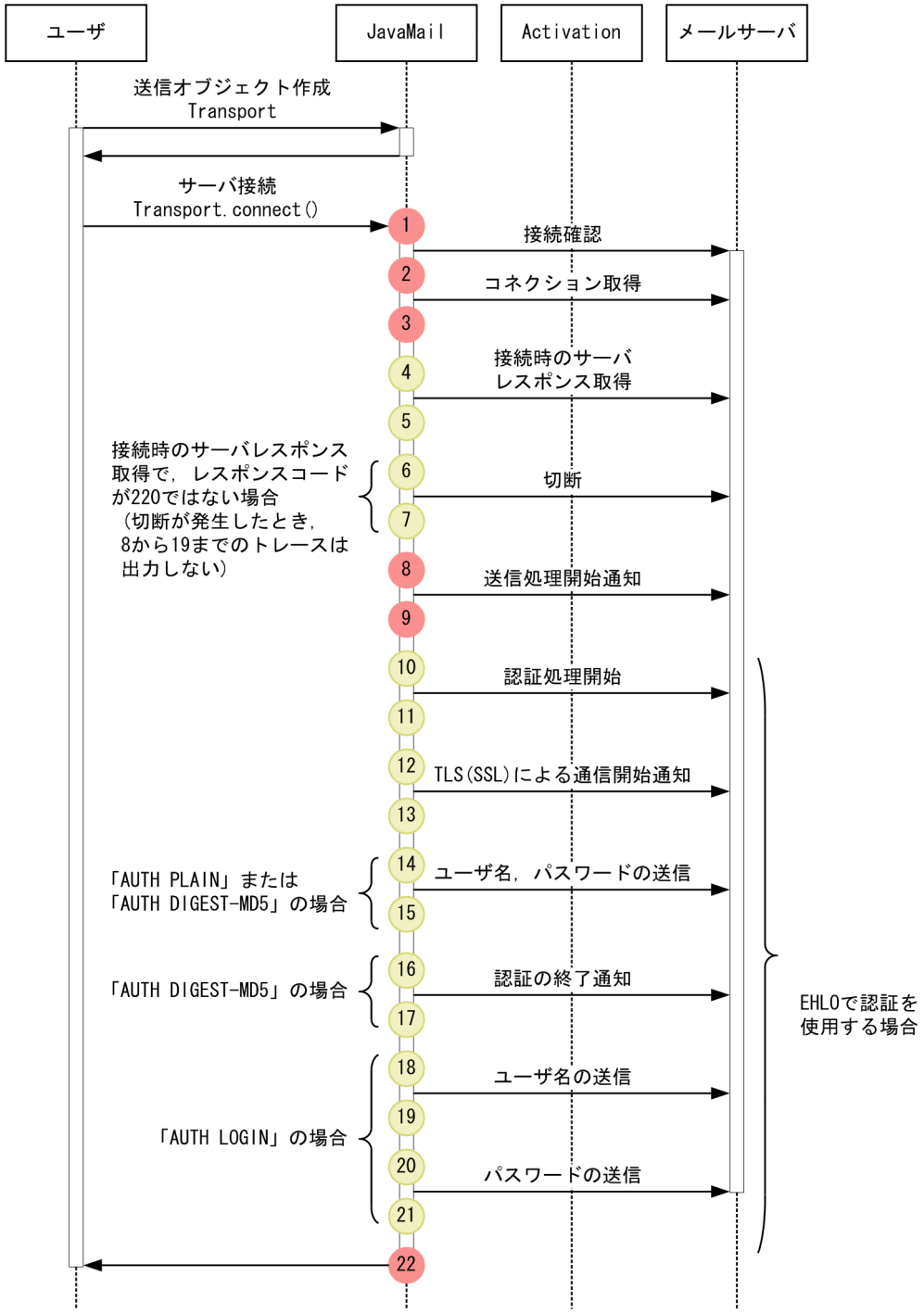
個々の送信先情報の送信開始～終了で取得します。

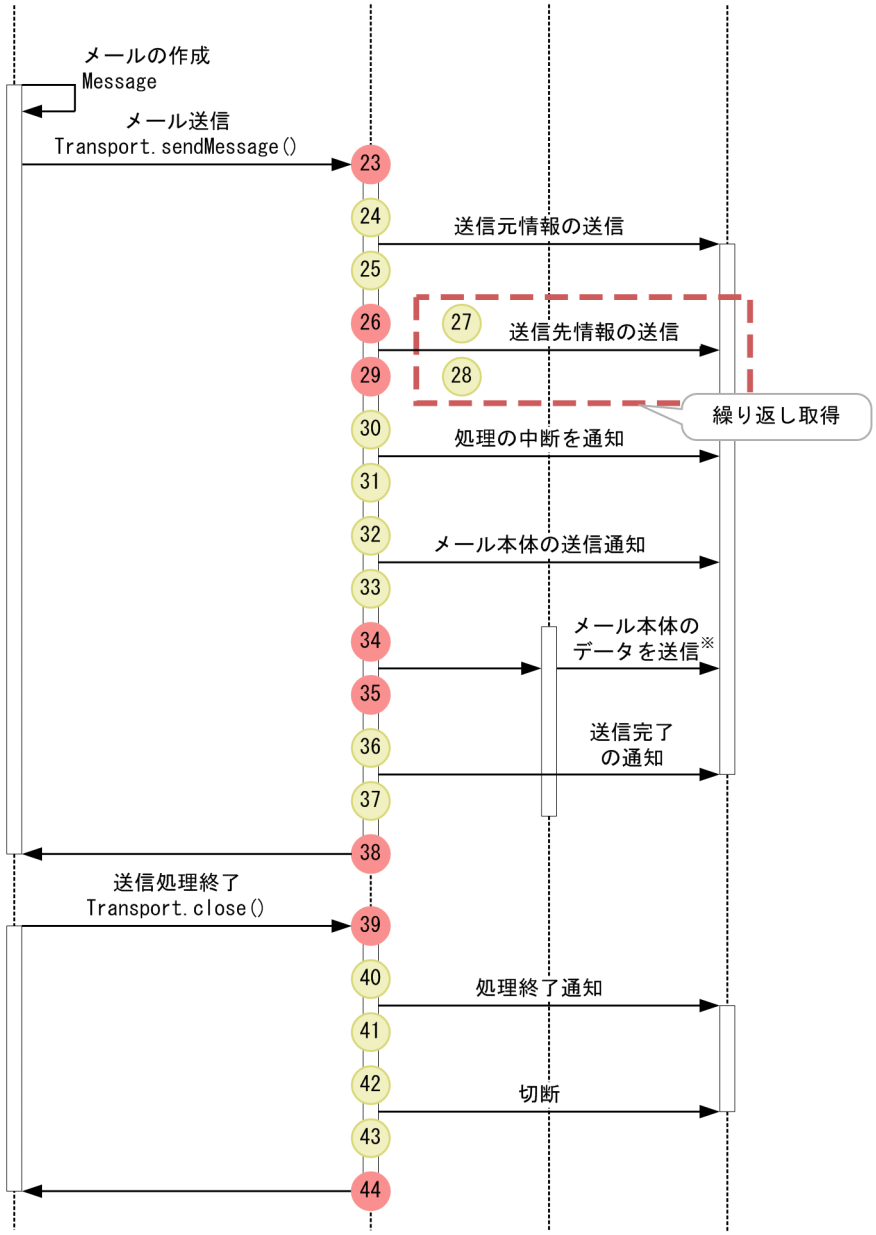
注※6

サーバと接続中の場合にだけ取得します。

JavaMail 送信時のトレース取得ポイントを以降の図に示します。

図 8-88　JavaMail の送信処理のトレース取得ポイント（javax.mail.Transport クラスの sendMessage(Message message, Address[] addresses)メソッドを使用する場合）



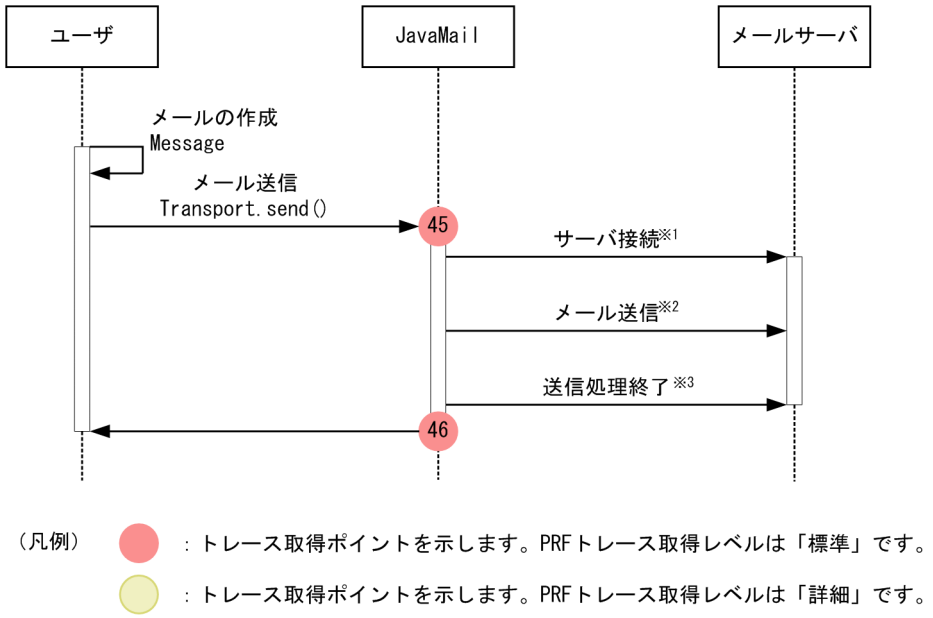


（凡例）　：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

　　：トレース取得ポイントを示します。PRFトレース取得レベルは「詳細」です。

注※　添付ファイルを使用する場合など，データ送信処理はActivationが行います。

図 8-89　JavaMail の送信処理のトレース取得ポイント（javax.mail.Transport クラスの send(Message msg, Address[] addresses)メソッド，javax.mail.Transport クラスの send(Message msg)メソッドを使用する場合）

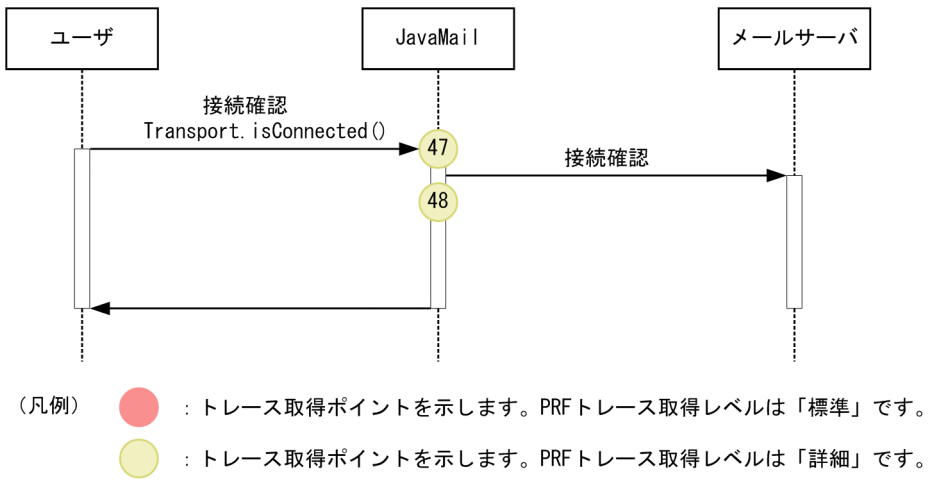


注※1　内部で javax.mail.Transport クラスの connect メソッドを呼び出すため，図 8-88 で示した 1～20 の取得ポイントも出力されます。

注※2　内部で javax.mail.Transport クラスの sendMessage メソッドを呼び出すため，図 8-88 で示した 21～36 の取得ポイントも出力されます。

注※3　内部で javax.mail.Transport クラスの close メソッドを呼び出すため，図 8-88 で示した 37～42 の取得ポイントも出力されます。

図 8-90　JavaMail のコネクション接続確認時のトレース取得ポイント



(2)　取得できるトレース情報

JavaMail 送信時に取得できるトレース情報を次の表に示します。

表 8-136　JavaMail 送信時に取得できるトレース情報

図中の番号 ※1	イベント ID	レベル	インタフェース名	オペレーション名	オプション
1	0xAD00	A	－	－	－
2	0xAD06	A	－	－	※2
3	0xAD07	A	－	－	異常時は例外のクラス名
4	0xAD14	B	通信処理を示す文字列※3	－	－
5	0xAD15	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
6	0xAD0C	B	－	－	－
7	0xAD0D	B	－	－	異常時は例外のクラス名
8	0xAD10	A	通信処理を示す文字列※3	－	－
9	0xAD11	A	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※6	異常時は例外のクラス名※7
10	0xAD12	B	通信処理を示す文字列※3	－	－
11	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
12	0xAD12	B	通信処理を示す文字列※3	－	－
13	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
14	0xAD12	B	通信処理を示す文字列※3	－	－
15	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
16	0xAD12	B	通信処理を示す文字列※3	－	－
17	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7

図中の番号 ※1	イベント ID	レベル	インタフェース名	オペレーション名	オプション
18	0xAD12	B	通信処理を示す文字列※3	－	－
19	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
20	0xAD12	B	通信処理を示す文字列※3	－	－
21	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
22	0xAD01	A	－	－	異常時は例外のクラス名
23	0xAD02	A	－	－	－
24	0xAD12	B	通信処理を示す文字列※3	－	－
25	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
26	0xAD08	A	－	－	－
27	0xAD12	B	通信処理を示す文字列※3	－	－
28	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
29	0xAD09	A	－	－	異常時は例外のクラス名
30	0xAD12	B	通信処理を示す文字列※3	－	－
31	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
31	0xAD12	B	通信処理を示す文字列※3	－	－
32	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
33	0xAD0A	A	－	－	－
34	0xAD0B	A	－	－	異常時は例外のクラス名

図中の番号※1	イベント ID	レベル	インタフェース名	オペレーション名	オプション
35	0xAD12	B	通信処理を示す文字列※3	－	－
36	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
37	0xAD03	A	－	－	異常時は例外のクラス名
38	0xAD04	A	－	－	－
39	0xAD12	B	通信処理を示す文字列※3	－	－
40	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7
41	0xAD0C	B	－	－	－
42	0xAD0D	B	－	－	異常時は例外のクラス名
43	0xAD05	A	－	－	異常時は例外のクラス名
44	0xAD0E	A	メソッド名(send(Message, Address[])またはsend(Message))	－	－
45	0xAD0F	A	メソッド名(send(Message, Address[])またはsend(Message))	－	異常時は例外のクラス名
46	0xAD12	B	通信処理を示す文字列※3	－	－
47	0xAD13	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4，※5，※6	異常時は例外のクラス名※7

（凡例）A：標準　B：詳細　－：該当なし

注※1

図 8-88，図 8-89，または図 8-90 中の番号と対応しています。

注※2

次の出力項目を出力します。

「ホスト名 ポート番号 コネクション取得時のタイムアウト値 通信のタイムアウト値」

各出力項目は半角スペースで区切って出力します。

タイムアウト値が無限に設定されている場合は 0 を出力します。また，ユーザがタイムアウト値を設定しなかった場合は-1 を出力します。

（例）「localhost 25 10000 10000」

注※3

通信処理を示す文字列出力内容は，次の表に示すとおりです。

項番	通信処理を示す文字列	通信処理
1	－（該当なし）	コネクション取得
2	connect-mail-server	接続時のサーバレスポンス取得
3	EHLO コマンド引数	EHLO コマンド発行
4	HELO コマンド引数	HELO コマンド発行
5	AUTH LOGIN	AUTH コマンド発行 (引数に LOGIN)
6	AUTH PLAIN	AUTH コマンド発行 (引数に PLAIN)
7	AUTH DIGEST-MD5	AUTH コマンド発行 (引数に DIGEST-MD5)
8	SEND USER	ユーザ名の送信
9	SEND PASS	パスワードの送信
10	SEND USER PASS	ユーザ名とパスワードの送信
11	AUTH END	認証の完了通知
12	QUIT	QUIT コマンド発行
13	MAIL コマンド引数	MAIL コマンド発行
14	RCPT コマンド引数	RCPT コマンド発行
15	RSET	RSET コマンド発行
16	NOOP	NOOP コマンド発行
17	DATA	DATA コマンド発行
18	SEND MAIL	メール本体のデータ送信
19	.	メール本文送信終了通知
20	STARTTLS	STARTTLS コマンド発行

注※4

レスポンスの出力内容は，次の表に示すとおりです。

レスポンス	出力内容
レスポンスコードが不正ではない（RFC 仕様内）	レスポンスコード
レスポンスコードが不正（RFC 仕様外）	レスポンス 1 行目の先頭 4 文字（詳細については，KDJE59111-E メッセージを参照してください） • 1 行目が 4 文字未満の場合 1 行目のすべての文字を出力

レスポンス	出力内容
	• レスポンスがない場合 空文字列を出力

注※5

QUIT コマンドでは mail.smtp.quitwait または mail.smtps.quitwait が ture の場合にだけレスポンスコードを出力します。

注※6

レスポンスが取得できた場合にだけレスポンスコードを出力します。通信で IOException が発生した場合など，異常時にレスポンスが取得できなかったとき，レスポンスコードは出力しません。

注※7

レスポンスコードが RFC 仕様外の場合，および EOF を検知した場合には例外のクラス名は出力しません。

8.21.2　JavaMail 受信時のトレース取得ポイントと取得できるトレース情報

(1)　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-137　JavaMail 受信時のトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0xAD80	1	javax.mail.Store クラスの connect(String host, int port, String user, String password)メソッドの入り口	A
0xAD81	14	javax.mail.Store クラスの connect(String host, int port, String user, String password)メソッドの出口	A
0xAD82	2	コネクション取得処理の開始直前	A
0xAD83	3	コネクション取得処理の終了直後	A
0xAD84	15	javax.mail.Folder クラスの open(int)メソッドの入り口	A
0xAD85	18	javax.mail.Folder クラスの open(int)メソッドの出口	A
0xAD86	25	javax.mail.Folder クラスの close(boolean)メソッドの入り口	A
0xAD87	36	javax.mail.Folder クラスの close(boolean)メソッドの出口	A
0xAD88	34	コネクション切断処理の開始直前	B
	6	コネクション切断処理の開始直前	B
0xAD89	35	コネクション切断処理の終了直後	B
	7	コネクション切断処理の終了直後	B
0xAD8A	19※2	全メッセージ情報取得処理の開始直前	A
0xAD8B	22※2	全メッセージ情報取得処理の終了直後	A

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0xAD8C	28※2	すべての削除対象のメッセージに対する DELE コマンド発行の開始直前	A
0xAD8D	31※2	すべての削除対象のメッセージに対する DELE コマンドのレスポンスを受信した直後	A
0xAD8E	10	USER コマンド発行の開始直前	A
	23※3	LIST，UIDL，RETR，または TOP コマンド発行の開始直前	A
	26	RSET コマンド発行の開始直前	A
	8	CAPA コマンド発行の開始直前	A
0xAD8F	11	USER コマンドのレスポンスを受信した直後	A
	24※3	LIST，UIDL，RETR，または TOP コマンドのレスポンスを受信した直後	A
	27	RSET コマンドのレスポンスを受信した直後	A
	9	CAPA コマンドのレスポンスを受信した直後	A
0xAD90	12	PASS コマンド発行の開始直前	B
	16	STAT コマンド発行の開始直前	B
	20※4，※5	TOP または LIST コマンド発行の開始直前	B
	29※5	DELE コマンド発行の開始直前	B
	32	QUIT コマンド発行の開始直前	B
	37※6	NOOP コマンド発行の開始直前	B
0xAD91	13	PASS コマンドのレスポンスを受信した直後	B
	17	STAT コマンドのレスポンスを受信した直後	B
	21※4，※5	TOP または LIST のレスポンスを受信した直後	B
	30※5	DELE コマンドのレスポンスを受信した直後	B
	33	QUIT コマンドのレスポンスを受信した直後	B
	38※6	NOOP コマンドのレスポンスを受信した直後	B
0xAD92	4	接続時のサーバレスポンス取得を行う直前	B
0xAD93	5	接続時のサーバレスポンス取得直後	B

（凡例）A：標準　B：詳細

注※1

図 8-91，または図 8-92 中の番号と対応しています。

注※2

すべてのメールの情報の取得開始～終了で取得します。

注※3

javax.mail.Folder クラスの fetch メソッドの第 2 引数に ENVELOPE が指定されて呼び出された場合以外です。

注※4

javax.mail.Folder クラスの fetch メソッドの第 2 引数に ENVELOPE が指定されて呼び出された場合だけです。

注※5

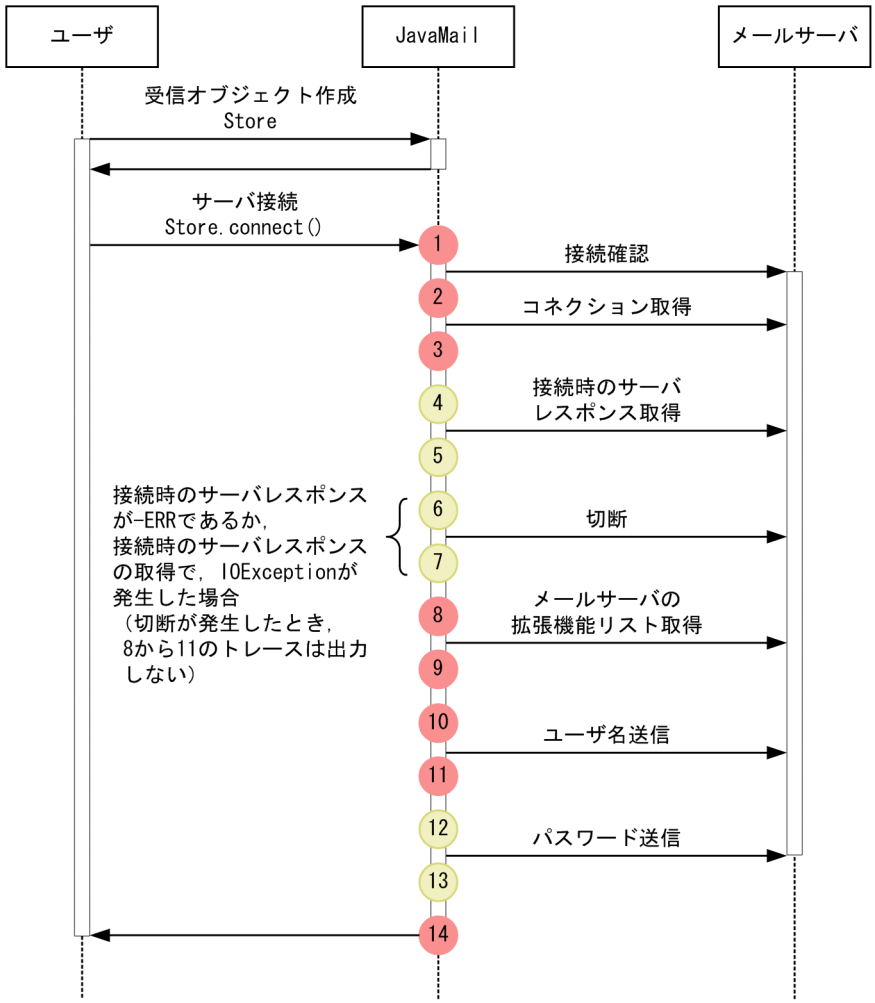
個々のメールの情報の取得開始～終了で取得します。

注※6

サーバと接続中の場合にだけ取得します。

JavaMail 受信時のトレース取得ポイントを以降の図に示します。

図 8-91　JavaMail の受信処理でのトレース取得ポイント



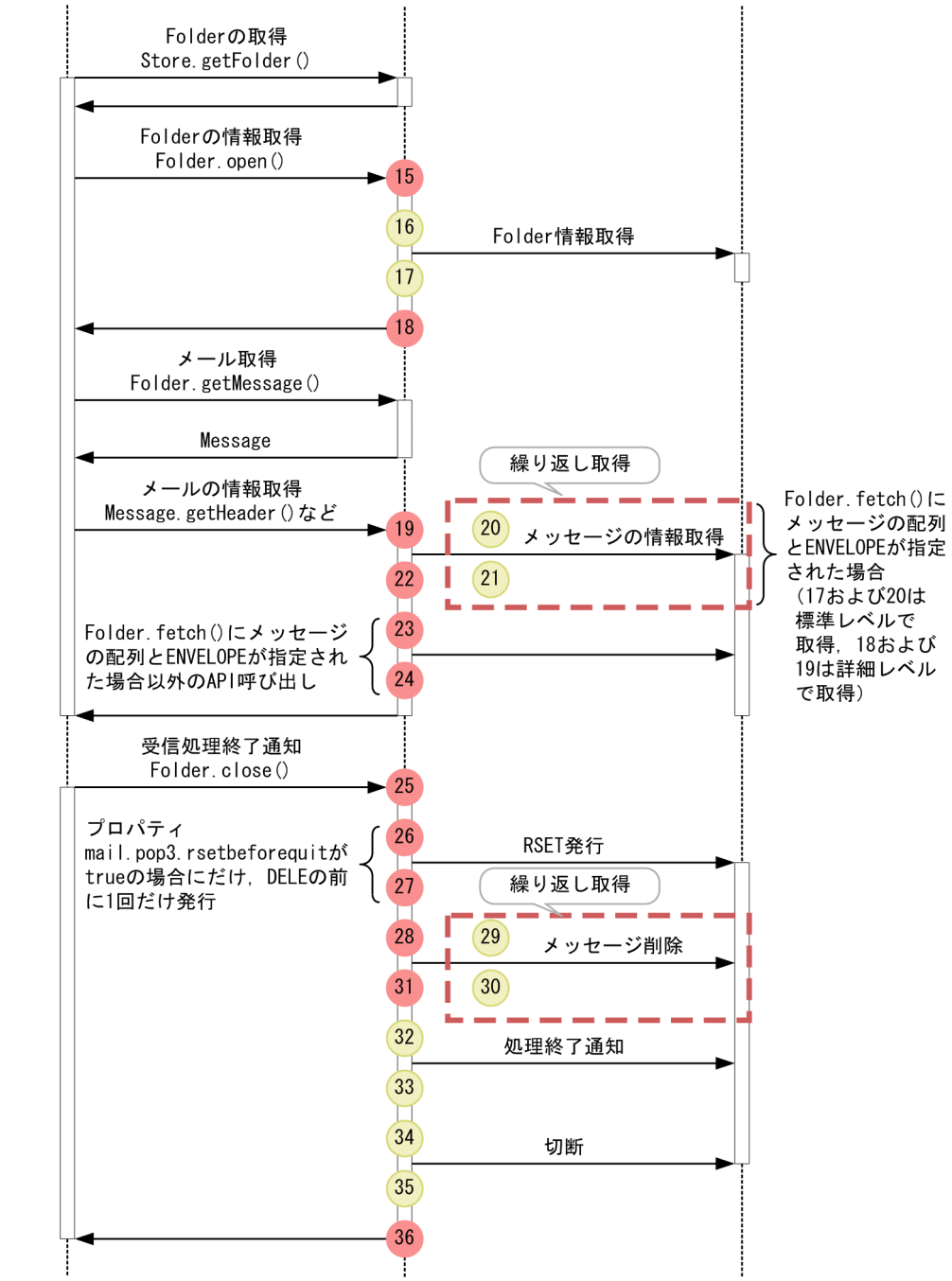
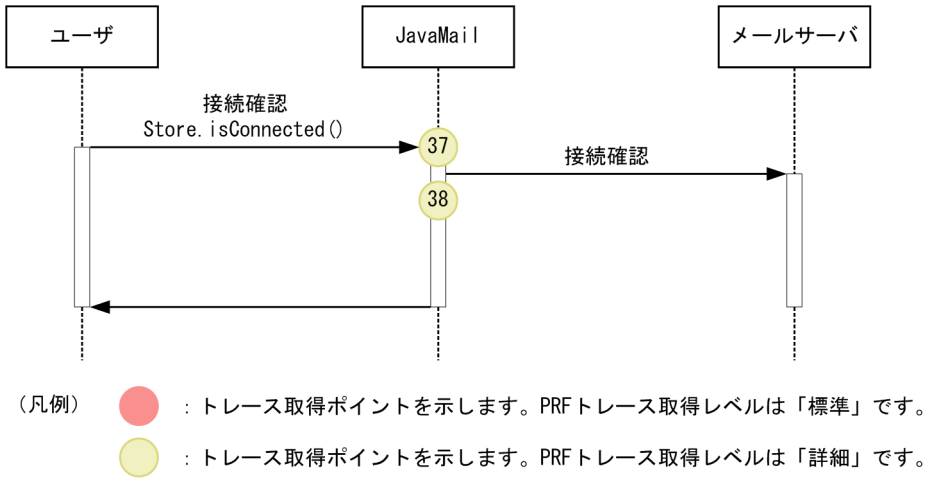

Folderの取得
Store.getFolder()
Folderの情報取得
Folder.open()
15
16
Folder情報取得
17
18
メール取得
Folder.getMessage()
Message
繰り返し取得
メールの情報取得
Message.getHeader()など
19
20
メッセージの情報取得
21
22
Folder.fetch()にメッセージの配列とENVELOPEが指定された場合以外のAPI呼び出し
23
24
Folder.fetch()にメッセージの配列とENVELOPEが指定された場合
（17および20は標準レベルで取得，18および19は詳細レベルで取得）
受信処理終了通知
Folder.close()
25
プロパティ
mail.pop3.rsetbeforequitがtrueの場合にだけ，DELEの前に1回だけ発行
26
RSET発行
27
繰り返し取得
28
29
メッセージ削除
30
31
32
処理終了通知
33
34
切断
35
36
（凡例）
：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。
：トレース取得ポイントを示します。PRFトレース取得レベルは「詳細」です。

図 8-92　JavaMail のコネクション接続確認時の取得ポイント



(2)　取得できるトレース情報

JavaMail 受信時に取得できるトレース情報を次の表に示します。

表 8-138　JavaMail 受信時に取得できるトレース情報

図中の番号 ※1	イベント ID	レベル	インタフェース名	オペレーション名	オプション
1	0xAD80	A	－	－	－
2	0xAD82	A	－	－	※2
3	0xAD83	A	－	－	異常時は例外のクラス名
4	0xAD92	B	通信処理を示す文字列※3	－	－
5	0xAD93	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
6	0xAD88	B	－	－	－
7	0xAD89	B	－	－	異常時は例外のクラス名
8	0xAD8E	A	通信処理を示す文字列※3	－	－
9	0xAD8F	A	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
10	0xAD8E	A	通信処理を示す文字列※3	－	－

図中の番号※1	イベント ID	レベル	インタフェース名	オペレーション名	オプション
11	0xAD8F	A	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
12	0xAD90	B	通信処理を示す文字列※3	－	－
13	0xAD91	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
14	0xAD81	A	－	－	異常時は例外のクラス名
15	0xAD84	A	－	－	－
16	0xAD90	B	通信処理を示す文字列※3	－	－
17	0xAD91	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
18	0xAD85	A	－	－	異常時は例外のクラス名
19※5	0xAD8A	A	－	－	－
20	0xAD90	B	通信処理を示す文字列※3	－	－
21※6	0xAD91	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
22※5	0xAD8B	A	－	－	異常時は例外のクラス名
23	0xAD8E	A	通信処理を示す文字列※3	－	－
24※6	0xAD8F	A	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
25	0xAD86	A	－	－	－
26	0xAD8E	A	通信処理を示す文字列※3	－	－
27	0xAD8F	A	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
28	0xAD8C	A	－	－	－

図中の番号※1	イベント ID	レベル	インタフェース名	オペレーション名	オプション
29	0xAD90	B	通信処理を示す文字列※3	－	－
30	0xAD91	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
31	0xAD8D	A	－	－	異常時は例外のクラス名
32	0xAD90	B	通信処理を示す文字列※3	－	－
33	0xAD91	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名
34	0xAD88	B	－	－	－
35	0xAD89	B	－	－	異常時は例外のクラス名
36	0xAD87	A	－	－	異常時は例外のクラス名
37	0xAD90	B	通信処理を示す文字列※3	－	－
38	0xAD91	B	通信処理を示す文字列※3	メールサーバからのレスポンスコード※4	異常時は例外のクラス名

（凡例）A：標準　B：詳細　－：該当なし

注※1

図 8-91，または図 8-92 中の番号と対応しています。

注※2

次の出力項目を出力します。

「ホスト名 ポート番号 コネクション取得時のタイムアウト値 通信のタイムアウト値」

各出力項目は半角スペースで区切って出力します。

タイムアウト値が無限に設定されている場合は 0 を出力します。また，ユーザがタイムアウト値を設定しなかった場合は-1 を出力します。

（例）「localhost 25 10000 10000」

注※3

通信処理を示す文字列出力内容は，次の表に示すとおりです。

項番	通信処理を示す文字列	通信処理
1	－（該当なし）	コネクション取得
2	connect-mail-server	接続時のサーバレスポンス取得
3	USER	USER コマンド発行
4	PASS	PASS コマンド発行

項番	通信処理を示す文字列	通信処理
5	QUIT	QUIT コマンド発行
6	STAT	STAT コマンド発行
7	LIST コマンド引数	LIST コマンド発行
8	UIDL コマンド引数	UIDL コマンド発行
9	RETR コマンド引数	RETR コマンド発行
10	TOP コマンド引数	TOP コマンド発行
11	DELE コマンド引数	DELE コマンド発行
12	NOOP	NOOP コマンド発行
13	RSET	RSET コマンド発行
14	CAPA	CAPA コマンド発行

注※4

レスポンス 1 行目が取得できた場合にだけレスポンスコードを出力します。通信で IOException が発生した場合など，異常時にレスポンスが取得できなかったとき，レスポンスコードは出力しません。

注※5

レスポンスの出力内容は，次の表に示すとおりです。

レスポンス	出力内容
レスポンスコードが不正ではない（RFC 仕様内）	レスポンスコード
レスポンスコードが不正（RFC 仕様外）	レスポンス 1 行目の先頭 4 文字（詳細については，KDJE59112-E メッセージを参照してください） • 1 行目が 4 文字未満の場合 1 行目のすべての文字を出力 • レスポンスがない場合 空文字列を出力

注※6

複数行のレスポンスを受け取るコマンド（RETR コマンド，TOP コマンド，および UIDL コマンドの発行時）では，レスポンスの 1 行目が取得できた場合，リターンコード 0 を出力します。

8.22　JSF 2.3 のトレース取得ポイント

ここでは，JSF 2.3 のトレース取得ポイントと，取得できるトレース情報について説明します。

8.22.1　トレース取得ポイントおよび取得できるトレース情報

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-139　JSF 2.3 でのトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	レベル
0xAF20	1	カスタムコンバータが呼び出される直前	A
0xAF21	2	カスタムコンバータの処理が終了した直後	A
0xAF22	3	カスタムバリデータが呼び出される直前	A
0xAF23	4	カスタムバリデータの処理が終了した直後	A
0xAF24	5	ValueChangeListener が呼び出される直前	A
0xAF25	6	ValueChangeListener の処理が終了した直後	A
0xAF26	7	ActionListener が呼び出される直前	A
0xAF27	8	ActionListener の処理が終了した直後	A
0xAF28	9	AjaxBehaviorListener が呼び出される直前	A
0xAF29	10	AjaxBehaviorListener の処理が終了した直後	A
0xAF2A	11	Action Method が呼び出される直前	A
0xAF2B	12	Action Method の処理が終了した直後	A
0xAF2C	13	ComponentSystemEventListener が呼び出される直前	A
0xAF2D	14	ComponentSystemEventListener の処理が終了した直後	A
0xAF2E	15	Websocket プッシュ接続の開始時（javax.websocket.RemoteEndpoint.Async.sendText(String)の呼び出し開始時）※2	A
0xAF2F	16	Websocket プッシュ接続の終了時（javax.websocket.RemoteEndpoint.Async.sendText(String)の呼び出し終了時）※2	A

（凡例）A：標準

注※1

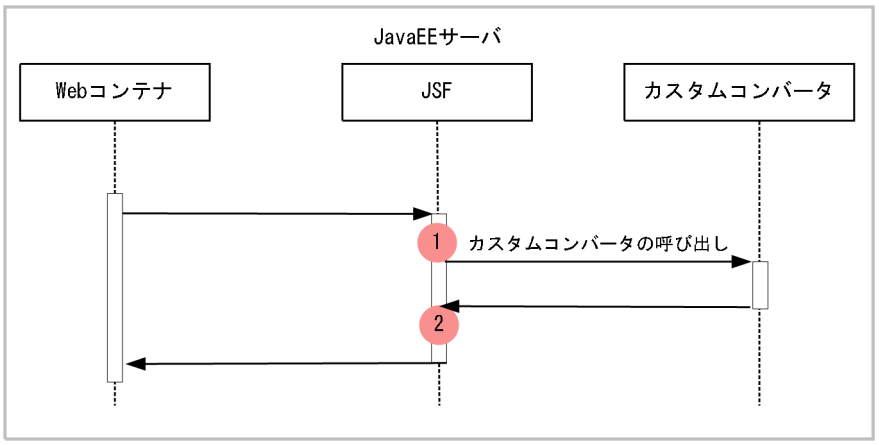
図 8-93～図 8-100 中の番号と対応しています。

注※2

Websocket 側で出力する PRF トレースの内容については，「8.28.3　データ送信時」を参照してください。

JSF 2.3 でのトレース取得ポイントを次の図に示します。

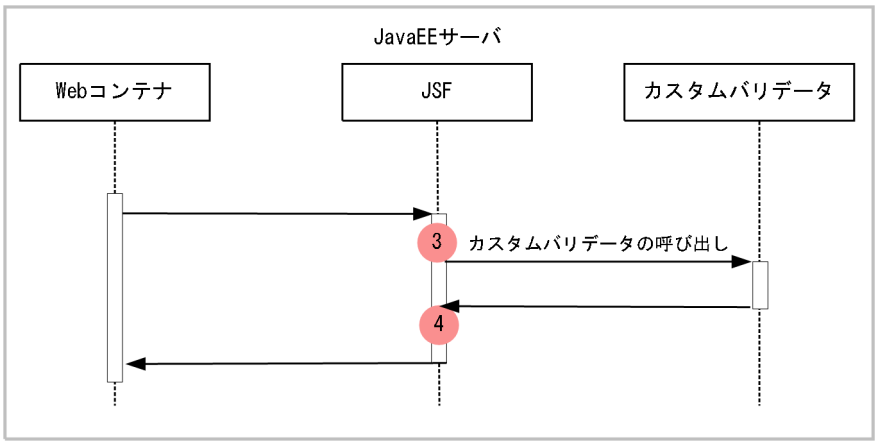
図 8-93　JSF 2.3 の性能解析トレース取得ポイント（カスタムコンバータ）



（凡例）

●：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

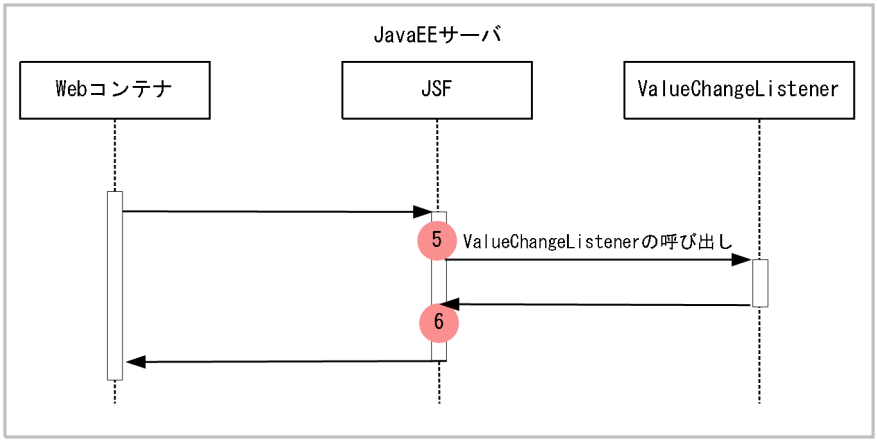
図 8-94　JSF 2.3 の性能解析トレース取得ポイント（カスタムバリデータ）



（凡例）

●：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

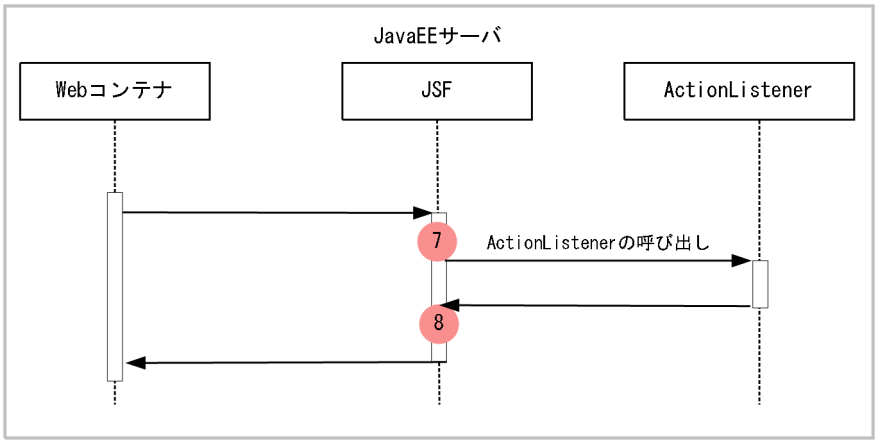
図 8-95　JSF 2.3 の性能解析トレース取得ポイント（ValueChangeListener）



（凡例）

●：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

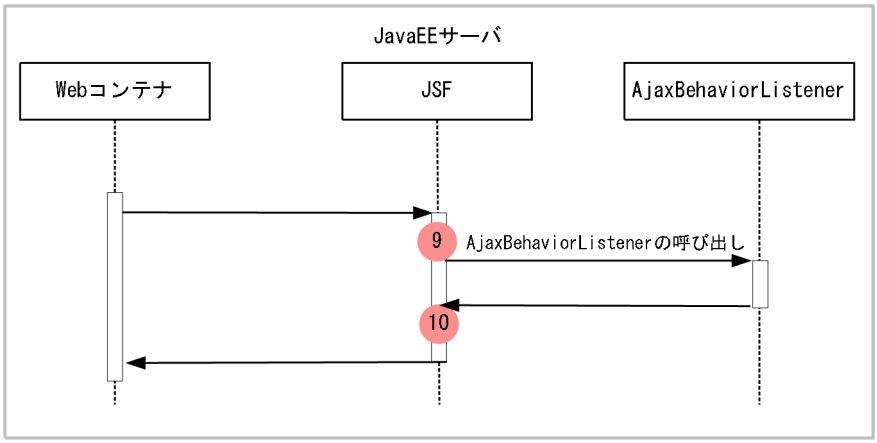
図 8-96　JSF 2.3 の性能解析トレース取得ポイント（ActionListener）



（凡例）

●：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

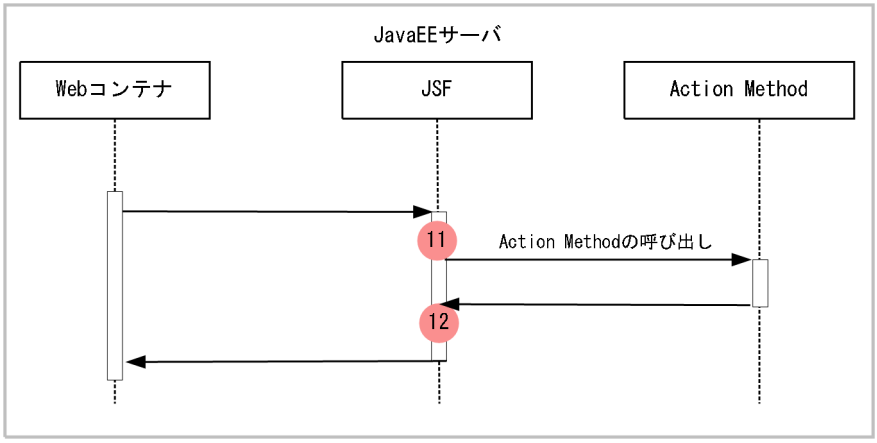
図 8-97　JSF 2.3 の性能解析トレース取得ポイント（AjaxBehaviorListener）



（凡例）

●：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

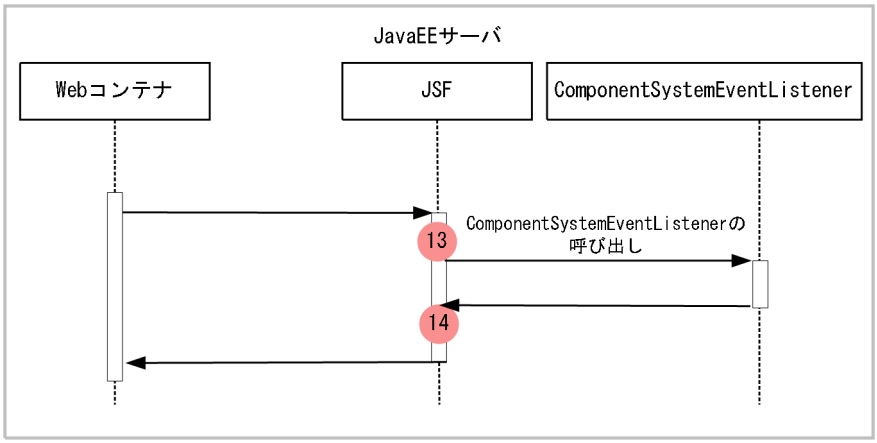
図 8-98　JSF 2.3 のトレース取得ポイント（Action Method）



（凡例）

●：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

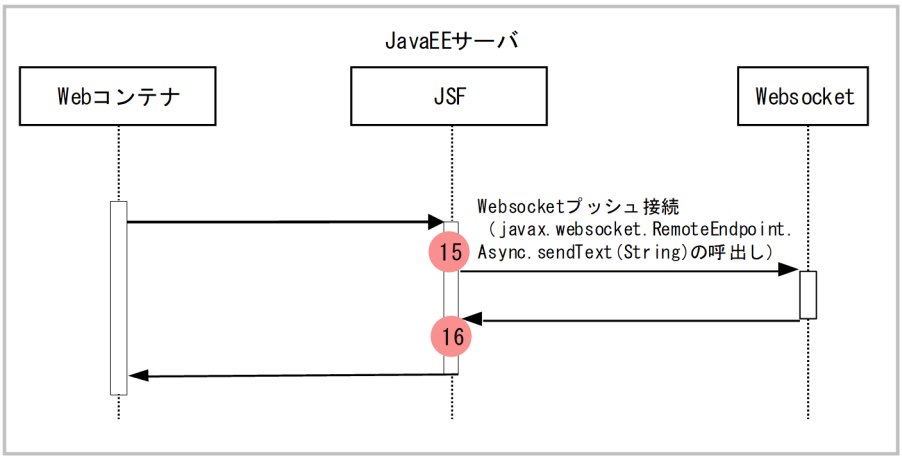
図 8-99　JSF 2.3 のトレース取得ポイント（ComponentSystemEventListener）



（凡例）

●：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

図 8-100　JSF 2.3 の性能解析トレース取得ポイント（Websocket プッシュ接続）



（凡例）

●：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

8.22.2　取得できるトレース情報

JSF 2.3 で取得できるトレース情報を次の表に示します。

表 8–140　JSF 2.3 で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xAF20	A	クライアント ID	カスタムコンバータのクラス名とメソッド名	−
2	0xAF21	A	クライアント ID	カスタムコンバータのクラス名とメソッド名	• 正常時 なし • 異常時 例外名
3	0xAF22	A	クライアント ID	カスタムバリデータのクラス名。MethodExpression を利用してカスタムバリデータを呼び出す場合は MethodExpression。 注 引数を複数記述した MethodExpression を使用した場合，性能解析トレースファイル（CSV 形式）の列がずれることがあります。	−
4	0xAF23	A	クライアント ID	カスタムバリデータのクラス名。MethodExpression を利用してカスタムバリデータを呼び出す場合は MethodExpression。 注 引数を複数記述した MethodExpression を使用した場合，性能解析トレースファイル（CSV 形式）の列がずれることがあります。	• 正常時 なし • 異常時 例外名
5	0xAF24	A	クライアント ID	ValueChangeListener のクラス名。 MethodExpression を利用して ValueChangeListener を呼び出す場合，MethodExpression。	−
6	0xAF25	A	クライアント ID	• 正常時 MethodExpression で ValueChangeListener が呼び出されて正常終了した場合，一つの引数を持つメソッドが	• 正常時 なし • 異常時 例外名

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
				呼び出された時は「one argument」を出力します。 引数のないメソッドが呼ばれた時は「no argument」を出力します。 • 異常時 なし。	
7	0xAF26	A	クライアント ID	ActionListener のクラス名。 MethodExpression を利用して ActionListener を呼び出す場合，MethodExpression。	−
8	0xAF27	A	クライアント ID	• 正常時 MethodExpression で ActionListener が呼び出されて正常終了した場合，一つの引数を持つメソッドが呼び出された時は「one argument」を出力します。 引数のないメソッドが呼ばれた時は「no argument」を出力します。 • 異常時 なし。	• 正常時 なし • 異常時 例外名
9	0xAF28	A	クライアント ID	AjaxBehaviorListener のクラス名。 MethodExpression を利用して AjaxBehaviorListener を呼び出す場合，MethodExpression。	−
10	0xAF29	A	クライアント ID	• 正常時 MethodExpression で AjaxBehaviorListener が呼び出されて正常終了した場合，一つの引数を持つメソッドが呼び出された時は「one argument」を出力します。 引数のないメソッドが呼ばれた時は「no argument」を出力します。 • 異常時 なし。	• 正常時 なし • 異常時 例外名
11	0xAF2A	A	クライアント ID	−	−
12	0xAF2B	A	クライアント ID	−	• 正常時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					なし • 異常時 例外名
13	0xAF2C	A	クライアント ID	ComponentSystemEventListener のクラス名。 MethodExpression を利用して ComponentSystemEventListener を呼び出す場合，MethodExpression。	−
14	0xAF2D	A	クライアント ID	• 正常時 MethodExpression で ComponentSystemEventListener が呼び出されて正常終了した場合，一つの引数を持つメソッドが呼び出された時は「one argument」を出力します。 引数のないメソッドが呼ばれた時は「no argument」を出力します。 • 異常時 なし。	• 正常時 なし • 異常時 例外名
15	0xAF2E	A	WebSocket の接続を開いたリクエストのクエリ文字列	−	−
16	0xAF2F	A	WebSocket の接続を開いたリクエストのクエリ文字列	−	• 正常時 なし • 異常時 例外名

（凡例）A：標準　−：該当なし

注※　図 8-93～図 8-100 中の番号と対応しています。

8.22.3　例外ログの出力

性能解析トレースの取得ポイントで付加情報（OPT）列に例外名を出力した場合は，同時に例外ログに例外の詳細情報を出力します。障害解析時には性能解析トレースで出力した例外名と例外ログを照合して障害解析を行うことができます。ただし，バッチアプリケーションなどで意図的に例外を発生させてジョブの処理を制御する目的で使用している場合，例外を発生させるたびに例外ログに詳細情報が出力されるため，ログ出力量が増加することがあります。このようなアプリケーションを実行する場合は，あらかじめ例外ログのサイズを大きく設定してください。

8.23　CDI のトレース取得ポイント

ここでは，CDI のトレース取得ポイントと，取得できるトレース情報について説明します。

8.23.1　CDI のトレース取得ポイントと取得できるトレース情報

CDI のトレース取得ポイントと取得できるトレース情報について説明します。ここでは，次の二つの場合に分けて説明します。

- JSF 2.3 と CDI を組み合わせて利用する場合
- サーブレットと CDI を組み合わせて利用する場合

(1)　トレース取得ポイントと PRF トレース取得レベル

(a)　JSF 2.3 と CDI を組み合わせて利用する場合

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-141　JSF 2.3 と CDI を組み合わせて利用する場合のトレース取得ポイントの詳細

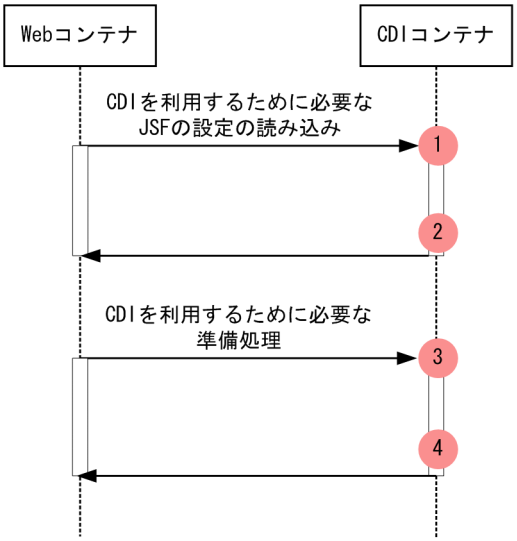
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xb002	1	CDI を利用するために必要な JSF 2.3 の設定の読み込み処理開始	A
0xb003	2	CDI を利用するために必要な JSF 2.3 の設定の読み込み処理終了（正常終了）	A
0xb004	3	CDI を利用するために必要な JSF 2.3 の準備の処理開始	A
0xb005	4	CDI を利用するために必要な JSF 2.3 の準備の処理終了（正常終了）	A
0xb006	5	EL 評価準備の処理開始	B
0xb007	6	EL 評価準備の処理終了（正常終了）	B

（凡例）A：標準　B：詳細

注※　図 8-101 および図 8-102 中の番号と対応しています。

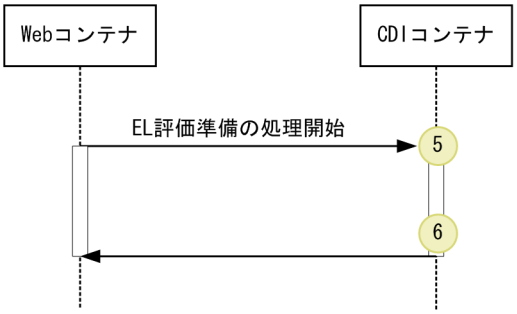
トレース取得ポイントを次の図に示します。

図 8-101　JSF 2.3 と CDI を組み合わせて利用する場合のトレース取得ポイント（JSF 2.3 の設定の読み込みと準備処理時）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

図 8-102　JSF 2.3 と CDI を組み合わせて利用する場合のトレース取得ポイント（EL 評価処理時）



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「詳細」です。

(b)　サーブレット・フィルタ・リスナと CDI を組み合わせて利用する場合

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-142　サーブレット・フィルタ・リスナと CDI を組み合わせて利用する場合のトレース取得ポイントの詳細

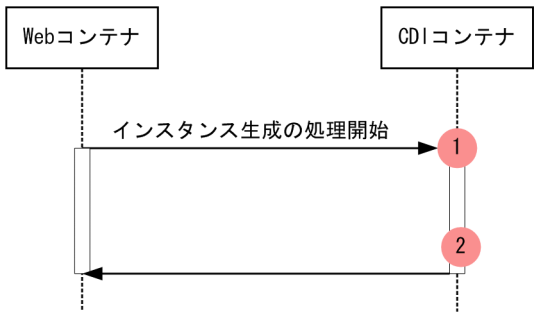
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xb008	1	サーブレット・フィルタ・リスナのインスタンス生成の処理開始	A
0xb009	2	サーブレット・フィルタ・リスナのインスタンス生成の処理終了（正常終了）	A

（凡例）A：標準

注※　図 8-103 中の番号と対応しています。

トレース取得ポイントを次の図に示します。

図 8-103　サーブレット・フィルタ・リスナと CDI を組み合わせて利用する場合のトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

(2)　取得できるトレース情報

(a)　JSF 2.3 と CDI を組み合わせて利用する場合

JSF 2.3 と CDI を組み合わせて利用する場合に取得できるトレース情報を次の表に示します。

表 8-143　JSF 2.3 と CDI を組み合わせて利用する場合に取得できる場合に取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xb002※2	A	WeldFacesConfigProvider	−	Web コンテナのコンテキスト情報
2	0xb003※2	A	WeldFacesConfigProvider	−	入口時刻
3	0xb004※2	A	WeldApplicationFactory	−	−
4	0xb005※2	A	WeldApplicationFactory	−	入口時刻
5	0xb006※3	B	WeldApplication	−	−
6	0xb007※3	B	WeldApplication	−	入口時刻

（凡例）A：標準　B：詳細　−：該当なし

注※1　図 8-101 および図 8-102 中の番号と対応しています。

注※2　CDI を利用するために必要な JSF 2.3 の設定の読み込みと，CDI を利用するために必要な JSF 2.3 の準備のトレース情報はアプリケーション開始時に取得します。

注※3　EL 評価準備のトレース情報は，FacesServlet が初期化処理時と JSF 2.3 で指定されている EL 式の評価実行時に取得します。

(b)　CDI からサーブレットを呼び出す場合

CDI からサーブレットを呼び出す場合に取得できるトレース情報を次の表に示します。

表 8-144　CDI からサーブレットを呼び出す場合に取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xb008	A	CDIServiceImpl	－	次の情報が出力されます。 • Managed クラス • クラス名 • コンテキストルート
2	0xb009	A	CDIServiceImpl	－	入口時刻

（凡例）A：標準　－：該当なし

注※　図 8-103 中の番号と対応しています。

なお，トレース情報は，サーブレット・フィルタ・リスナのインタフェース生成時に取得します。

8.24　J2EE サーバの開始・終了時のトレース取得ポイント

J2EE サーバの開始処理の完了時，および J2EE サーバの終了処理の開始時に，トレース情報を取得できます。

8.24.1　トレース取得ポイントと PRF トレース取得レベル

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8–145　J2EE サーバでのトレース取得ポイントの詳細

イベント ID	トレース取得ポイント	レベル
0x8FFE	J2EE サーバの開始処理完了（KDJE30028-I メッセージ出力）時	A
0x8FFF	J2EE サーバのシャットダウン開始（KDJE30031-I メッセージ出力）時	A

（凡例）A：標準

8.24.2　取得できるトレース情報

J2EE サーバの開始・終了時に取得できるトレース情報を次に示します。

- イベント ID
  0x8FFE，0x8FFF
- PRF トレース取得レベル
  すべて「標準」になります。
- インタフェース名，オペレーション名，およびオプション
  情報は出力されません。

8.25　アプリケーションのトレース取得ポイント

アプリケーションのトレースはユーザ拡張性能解析トレースで出力されます。ユーザ拡張性能解析トレースはユーザ拡張性能解析トレース設定ファイルで指定したメソッドが呼ばれるポイントでトレース情報を出力します。

ここでは，ユーザ拡張性能解析トレースのトレースポイントとトレース情報について説明します。

8.25.1　トレース取得ポイントと PRF トレース取得レベル

ユーザ拡張性能解析トレースがトレースを出力するポイントを次に示します。

表 8-146　ユーザ拡張性能解析トレースのトレース取得ポイント

トレース取得ポイント	図中の番号 ※1	説明	レベル
メソッドの正常入口	1	メソッドが呼び出された直後。	ユーザ拡張性能解析トレース設定ファイルで指定したトレース取得レベル※2
メソッドの正常出口	2	メソッドが正常に終了する直前。	
メソッドの異常出口	3	メソッドが例外またはエラーで異常終了する直前。	

（凡例）A：標準

注※1　図 8-104 中の番号と対応しています。

注※2　ユーザ拡張性能解析トレース設定ファイルのトレース取得レベルの指定については，「7.5.3　ユーザ拡張性能解析トレースのトレース対象メソッドの設定」を参照してください。

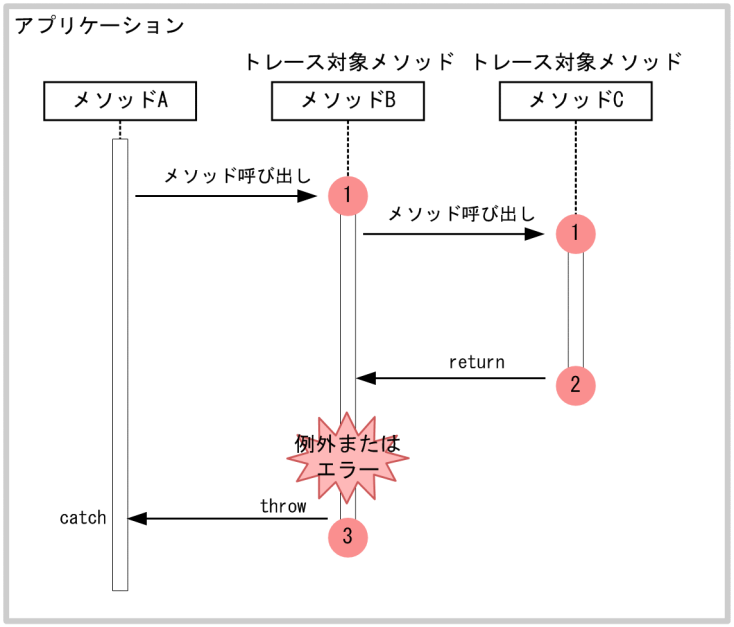
参考

トレース対象のメソッド，およびトレースの対象メソッドの呼び出し先メソッドで発生した例外やエラーでトレース対象メソッドが終了する場合，メソッドの異常出口としてトレースを取得します。

トレース対象メソッド内の try-catch 構文で例外やエラーを処理していて，トレース対象メソッドの呼び出し元に例外やエラーが throw されない場合は，メソッドの正常出口としてトレースを取得します。

ユーザ拡張性能解析トレースでのトレース取得ポイントを次の図に示します。

図 8-104　ユーザ拡張性能解析トレーストレース取得ポイント



（凡例）n　：トレース取得ポイントを示します。
n=1 メソッドの入口
n=2 メソッドの正常出口
n=3 メソッドの異常出口
PRF取得レベルは常にA（標準）です。
→：処理の流れ

8.25.2　取得できるトレース情報

ユーザ拡張性能解析トレースのトレース情報について説明します。

(1)　ユーザ拡張性能解析トレースのトレース情報

ユーザ拡張性能解析トレースのトレース情報を次の表に示します。

表 8-147　ユーザ拡張性能解析トレースのトレース情報

図中の番号※1	取得できる情報				
	イベント ID (Event)	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報（OPR）	付加情報※2 (OPT／ASCII※3)
1	ユーザ拡張性能解析トレース設定ファイルで指定したイベント ID。 指定しない場合は，0xae00。	0	ユーザ拡張性能解析トレース設定ファイルでユーザが指定した識別 ID。	−	次の情報が出力されます。 • パッケージ名 • クラス名 • メソッド名

図中の番号※1	取得できる情報				
	イベント ID (Event)	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報（OPR）	付加情報※2 (OPT／ASCII※3)
2	ユーザ拡張性能解析トレース設定ファイルで指定したイベント ID に+ 1 した値。指定しない場合は，0xae01。	0	ユーザ拡張性能解析トレース設定ファイルでユーザが指定した識別 ID。	メソッドの最後に実行した文の行番号※4。	次の情報が出力されます。 • パッケージ名 • クラス名 • メソッド名
3	ユーザ拡張性能解析トレース設定ファイルで指定したイベント ID に+1 した値。指定しない場合は，0xae01。	1	ユーザ拡張性能解析トレース設定ファイルでユーザが指定した識別 ID。	例外またはエラーのクラス名※5。	次の情報が出力されます。 • パッケージ名 • クラス名 • メソッド名

（凡例）－：出力なし。

注※1　図 8-104 中の番号と対応しています。

注※2　ユーザ拡張性能解析トレースは，jvm.userprf.LogLevel プロパティに指定した出力レベルでメソッド名を出力します。出力レベルと各レベルで出力される情報については，「(2)　出力レベル」を参照してください。

注※3　ASCII 領域に出力され付加情報が，ASCII 文字で 256 文字を超えていた場合，前から 256 文字が出力されます。

注※4　jvm.userprf.LineNumber プロパティに true を指定した場合に出力されます。jvm.userprf.LineNumber プロパティについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.3　JavaVM で使用するプロパティ」を参照してください。

注※5　jvm.userprf.ThrowableName プロパティに true を指定した場合，jvm.userprf.LogLevel プロパティに指定した出力レベル，および jvm.userprf.ThrowableNameEditMethod プロパティに指定した編集方法でクラス名が出力されます。jvm.userprf.ThrowableName プロパティおよび jvm.userprf.ThrowableNameEditMethod プロパティについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.3　JavaVM で使用するプロパティ」を参照してください。

(2)　出力レベル

jvm.userprf.LogLevel プロパティで，ユーザ拡張性能解析トレースのトレース情報の出力レベルを指定します。

出力レベルと出力されるトレース情報を次の表に示します。

表 8-148　出力レベルの指定とトレース出力情報

jvm.userprf.LogLevel プロパティの指定	付加情報（OPT／ASCII）	メソッドの異常出口のオペレーション情報（OPR）
class	クラス名	例外またはエラーのクラス名
package	完全修飾クラス名	例外またはエラーの完全修飾クラス名
method	完全修飾クラス名＋メソッド名	

jvm.userprf.LogLevel プロパティの指定	付加情報（OPT／ASCII）	メソッドの異常出口のオペレーション情報（OPR）
signature	完全修飾クラス名＋メソッド名＋メソッドの引数の型	

jvm.userprf.LogLevel プロパティについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.3　JavaVM で使用するプロパティ」を参照してください。

(3)　ユーザ拡張性能解析トレースの出力例

ユーザ拡張性能解析トレースの出力例を次に示します。

(a)　トレース情報の出力内容を変更しない場合の例

トレース情報の出力内容を変更しない場合の設定例を次に示します。

簡易構築定義ファイルの設定例

```
 :
<param>
<param-name>UserPrfText</param-name>
<param-value>
<![CDATA[
com.sample.ClassA.method1(int),test00,false
]]>
</param-value>
</param>
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
</param>
 :
```

出力内容

Event	Rc	INT	OPR	ASCII
0xae00	0	test00	（空白）	ClassA
0xae01	0	test00	（空白）	ClassA

この例では，次のようにトレース情報が出力されます。

- トレース対象メソッドに対してイベント ID の設定を省略しているため，トレース対象となったメソッドが呼び出されると，イベント ID としてメソッドの入口で 0xae00，メソッドの出口で 0xae01 のデフォルト値が出力されます。

(b)　オペレーション情報に行番号を出力する場合の例

ユーザ拡張性能解析トレース設定ファイル（/test/setting.txt）を使用して，オペレーション情報に行番号を出力する場合の設定例を次に示します。

簡易構築定義ファイルの設定例

```
 :
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
<param-value>-Djvm.userprf.File=/test/setting.txt</param-value>
<param-value>-Djvm.userprf.LineNumber=true</param-value>
</param>
 :
```

ユーザ拡張性能解析トレース設定ファイル（setting.txt）の設定例

```
com.sample.ClassA.method1(java.lang.String),test00,false,0xae77
com.sample.ClassB.method2(boolean),test01.false
```

出力内容

この例では jvm.userprf.LineNumber=true を指定しているため，メソッド正常出口でオペレーション情報（OPR 領域）に，各メソッドの最後に実行された行番号が出力されます。

Event	Rc	INT	OPR	ASCII
0xae77	0	test00	（空白）	ClassA
0xae78	0	test00	324	ClassA
0xae00	0	test01	（空白）	ClassB
0xae01	0	test01	15	ClassB

この例では，次のようにトレース情報が出力されます。

- ユーザ拡張性能解析トレース設定ファイルの 2 行目の設定では，トレース対象メソッドに対してイベント ID の設定を省略しているため，トレース対象となったメソッドが呼び出されると，イベント ID としてメソッドの入口で 0xae00，メソッドの出口で 0xae01 のデフォルト値が出力されます。

(c) 発生した例外またはエラーのクラス名を出力する場合の例

例外のクラス名をメソッド異常出口のオペレーション情報（OPR 領域）に出力する場合の設定例を次に示します。なお，この例では ClassA のサブクラスに ClassC が存在し，ClassC で ClassA.method1 をオーバーライドしている場合について説明します。

簡易構築定義ファイルの設定例

```
 :
<param>
<param-name>UserPrfText</param-name>
<param-value>
<![CDATA[
com.sample.ClassA.method1(),test00,true,0xae0a
]]>
</param-value>
</param>
<param>
<param-name>add.jvm.arg</param-name>
```

```
<param-value>-Djvm.userprf.Enable=true</param-value>
<param-value>-Djvm.userprf.LineNumber=true</param-value>
<param-value>-Djvm.userprf.ThrowableName=true</param-value>
</param>
 :
```

出力内容

Event	Rc	INT	OPR	ASCII
0xae0a	0	test00	（空白）	ClassA
0xae0b	0	test00	324	ClassA
0xae0a	0	test00	（空白）	ClassC
0xae0b	1	test00	IOException	ClassC

この例では，次のようにトレース情報が出力されます。

- サブクラスフラグに true を設定しているため，ClassA.method1 をオーバーライドしている ClassC.method1 の情報も出力されます。
- jvm.userprf.LineNumber=true を指定しているため，メソッド正常出口のオペレーション情報（OPR 領域）に，各メソッドの最後に実行された行番号が出力されます。
- jvm.userprf.ThrowableName=true を設定しているため，発生した例外のクラス名がメソッド異常出口のオペレーション情報（OPR 領域）に出力されます。ただし，jvm.userprf.LogLevel プロパティを指定していないため，発生した例外のクラス名だけが出力されます。

(d) 発生した例外またはエラーのクラス名の出力方式を変更した例

発生した例外またはエラーのクラス名の出力方式を変更した場合の設定例を次に示します。

簡易構築定義ファイルの設定例

```
 :
<param>
<param-name>UserPrfText</param-name>
<param-value>
<![CDATA[
com.sample.ClassA.method1(),test00,false
]]>
</param-value>
</param>
<param>
<param-name>add.jvm.arg</param-name>
<param-value>-Djvm.userprf.Enable=true</param-value>
<param-value>-Djvm.userprf.ThrowableName=true</param-value>
<param-value>-Djvm.userprf.ThrowableNameEditMethod=FRONT_CUT</param-value>
<param-value>-Djvm.userprf.LogLevel=method</param-value>
</param>
 :
```

出力内容

Event	Rc	INT	OPR	ASCII
0xae00	0	test00	（空白）	com.sample.ClassA.method1
0xae01	1	test00	*ment.IllegalClassFormatException	com.sample.ClassA.method1

この例では，次のようにトレース情報が出力されます。

- トレース対象メソッドに対してイベント ID の設定を省略しているため，トレース対象となったメソッドが呼び出されると，イベント ID としてメソッドの入口で 0xae00，メソッドの出口で 0xae01 のデフォルト値が出力されます。
- jvm.userprf.ThrowableName=true を指定しているため，発生した例外またはエラーのクラス名がメソッドの異常出口のオペレーション情報（OPR 領域）に出力されます。ただし，発生した例外名が 33 文字以上で，jvm.userprf.ThrowableNameEditMethod=FRONT_CUT を指定しているため，前方が省略されて出力されます。省略された部分は「*」で表示されます。
- jvm.userprf.LogLevel=method を指定しているため，付加情報（ASCII 領域）に完全修飾クラス名＋メソッド名が出力されます。

8.26　JAX-RS のトレース取得ポイント

ここでは，JAX-RS のトレース取得ポイントと，取得できるトレース情報について説明します。

8.26.1　トレース取得ポイントおよび取得できるトレース情報

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8–149　JAX-RS でのトレース取得ポイントの詳細

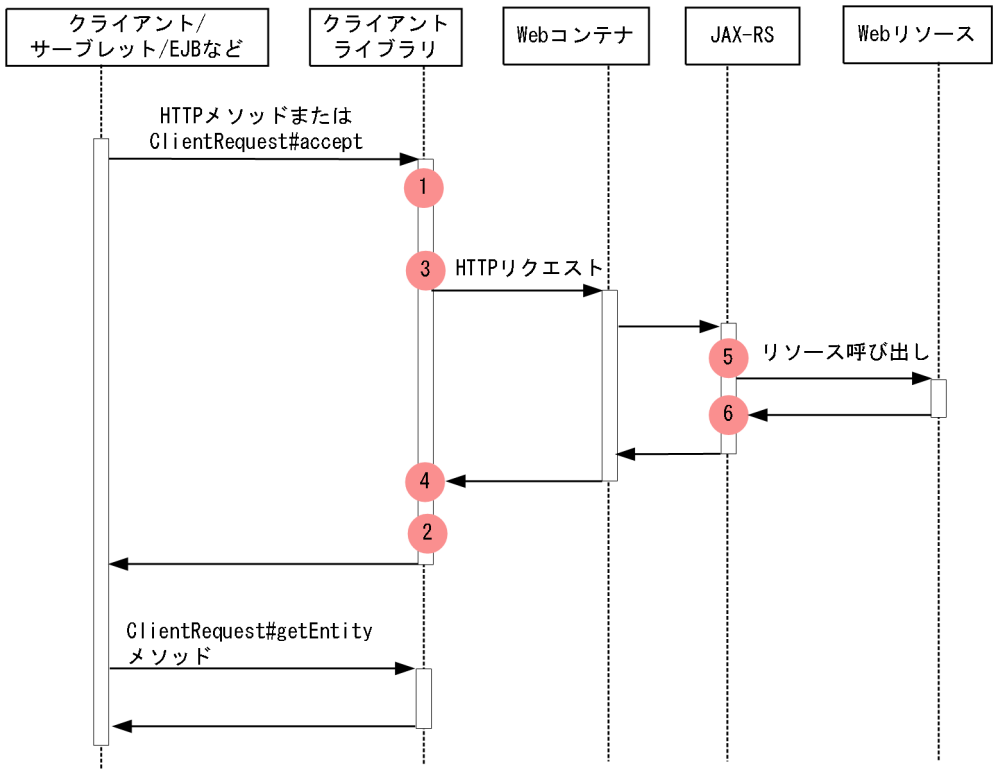
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD000	1	HTTP メソッド呼び出しの開始時	A
0xD001	2	HTTP メソッド呼び出しの終了時	A
0xD002	3	クライアントライブラリの HTTP メッセージ送信前	A
0xD003	4	クライアントライブラリの HTTP メッセージ受信後	A
0xD004	5	Web リソース呼び出し前	A
0xD005	6	Web リソース呼び出し後	A

（凡例）A：標準

注※　図 8-105 中の番号と対応しています。

JAX-RS でのトレース取得ポイントを次の図に示します。

図 8-105　JAX-RS のトレース取得ポイント



（凡例）

●：トレース取得ポイントを示します。PRF トレース取得レベルは「標準」です。

8.26.2　取得できるトレース情報

JAX-RS で取得できるトレース情報を次の表に示します。

表 8-150　JAX-RS で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xD000	A	クラス名	メソッド名	−
2	0xD001	A	クラス名	メソッド名	• 正常時 なし • 異常時 例外名
3	0xD002	A	クラス名	メソッド名	エンドポイント URI
4	0xD003	A	クラス名	メソッド名	• 正常時 なし • 異常時 例外名

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
5	0xD004	A	クラス名	メソッド名	呼び出し種別は次のどれか • ObjectOut Invoker • ResponseOu t Invoker • TypeOutInv oker • VoidOutInv oker • SseEventSin kInvoker • VoidToVoid Dispatcher
6	0xD005	A	クラス名	メソッド名	• 正常時 なし • 異常時 例外名

（凡例）A：標準　－：該当なし

注※　図 8-105 中の番号と対応しています。

8.26.3　例外ログの出力

性能解析トレースの取得ポイントで付加情報（OPT）列に例外名を出力した場合は，同時に例外ログに例外の詳細情報を出力します。障害解析時には性能解析トレースで出力した例外名と例外ログを照合して障害解析を行うことができます。ただし，バッチアプリケーションなどで意図的に例外を発生させてジョブの処理を制御する目的で使用している場合，例外を発生させるたびに例外ログに詳細情報が出力されるため，ログ出力量が増加することがあります。このようなアプリケーションを実行する場合は，あらかじめ例外ログのサイズを大きく設定してください。

8.27　Java Batch のトレース取得ポイント

ここでは，Java Batch のトレース取得ポイントと，取得できるトレース情報について説明します。

8.27.1　トレース取得ポイントおよび取得できるトレース情報

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8–151　Java Batch でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD020	1	javax.batch.runtime.BatchRuntime.getJobOperator()メソッド開始直後	A
0xD021	2	javax.batch.runtime.BatchRuntime.getJobOperator()メソッド終了直前	A
0xD022	3	javax.batch.operations.JobOperator.start(String jobXMLName, Properties jobParameters)メソッド開始直後	A
0xD023	4	javax.batch.operations.JobOperator.start(String jobXMLName, Properties jobParameters)メソッド終了直前	A
0xD024	5	javax.batch.operations.JobOperator.restart(long executionId, Properties restartParameters)メソッド開始直後	A
0xD025	6	javax.batch.operations.JobOperator.restart(long executionId, Properties restartParameters)メソッド終了直前	A
0xD026	7	javax.batch.operations.JobOperator.stop(long executionId)メソッド開始直後	A
0xD027	8	javax.batch.operations.JobOperator.stop(long executionId)メソッド終了直前	A
0xD028	9	javax.batch.operations.JobOperator.abandon(long executionId)メソッド開始直後	A
0xD029	10	javax.batch.operations.JobOperator.abandon(long executionId)メソッド終了直前	A
0xD02A	11	ジョブの開始直後	A
0xD02B	12	ジョブの終了直後	A
0xD02C	13	ステップの処理開始直後	A
0xD02D	14	ステップの処理終了直前	A
0xD02E	15	javax.batch.api.chunk.ItemReader の実装クラスの open(Serializable checkpoint)メソッドの呼び出し直前	A

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD02F	16	javax.batch.api.chunk.ItemReader の実装クラスの open(Serializable checkpoint)メソッドの処理完了直後	A
0xD030	17	javax.batch.api.chunk.ItemReader の実装クラスの close()メソッドの呼び出し直前	A
0xD031	18	javax.batch.api.chunk.ItemReader の実装クラスの close()メソッドの処理完了直後	A
0xD032	19	javax.batch.api.chunk.ItemReader の実装クラスの readItem()メソッドの呼び出し直前	A
0xD033	20	javax.batch.api.chunk.ItemReader の実装クラスの readItem()メソッドの処理完了直後	A
0xD034	21	javax.batch.api.chunk.ItemReader の実装クラスの checkpointInfo()メソッドの呼び出し直前	A
0xD035	22	javax.batch.api.chunk.ItemReader の実装クラスの checkpointInfo()メソッドの処理完了直後	A
0xD036	23	javax.batch.api.chunk.ItemProcessor の実装クラスの processItem(Object item)メソッドの呼び出し直前	A
0xD037	24	javax.batch.api.chunk.ItemProcessor の実装クラスの processItem(Object item)メソッドの処理完了直後	A
0xD038	25	javax.batch.api.chunk.ItemWriter の実装クラスの open(Serializable checkpoint)メソッドの呼び出し直前	A
0xD039	26	javax.batch.api.chunk.ItemWriter の実装クラスの open(Serializable checkpoint)メソッドの処理完了直後	A
0xD03A	27	javax.batch.api.chunk.ItemWriter の実装クラスの close()メソッドの呼び出し直前	A
0xD03B	28	javax.batch.api.chunk.ItemWriter の実装クラスの close()メソッドの処理完了直後	A
0xD03C	29	javax.batch.api.chunk.ItemWriter の実装クラスの writeItems(List<Object> items)メソッドの呼び出し直前	A
0xD03D	30	javax.batch.api.chunk.ItemWriter の実装クラスの writeItems(List<Object> items)メソッドの処理完了直後	A
0xD03E	31	javax.batch.api.chunk.ItemWriter の実装クラスの checkpointInfo()メソッドの呼び出し直前	A
0xD03F0	32	javax.batch.api.chunk.ItemWriter の実装クラスの checkpointInfo()メソッドの処理完了直後	A
0xD040	33	javax.batch.api.listener.JobListener の実装クラスの beforeJob()メソッドの呼び出し直前	A
0xD041	34	javax.batch.api.listener.JobListener の実装クラスの beforeJob()メソッドの処理完了直後	A

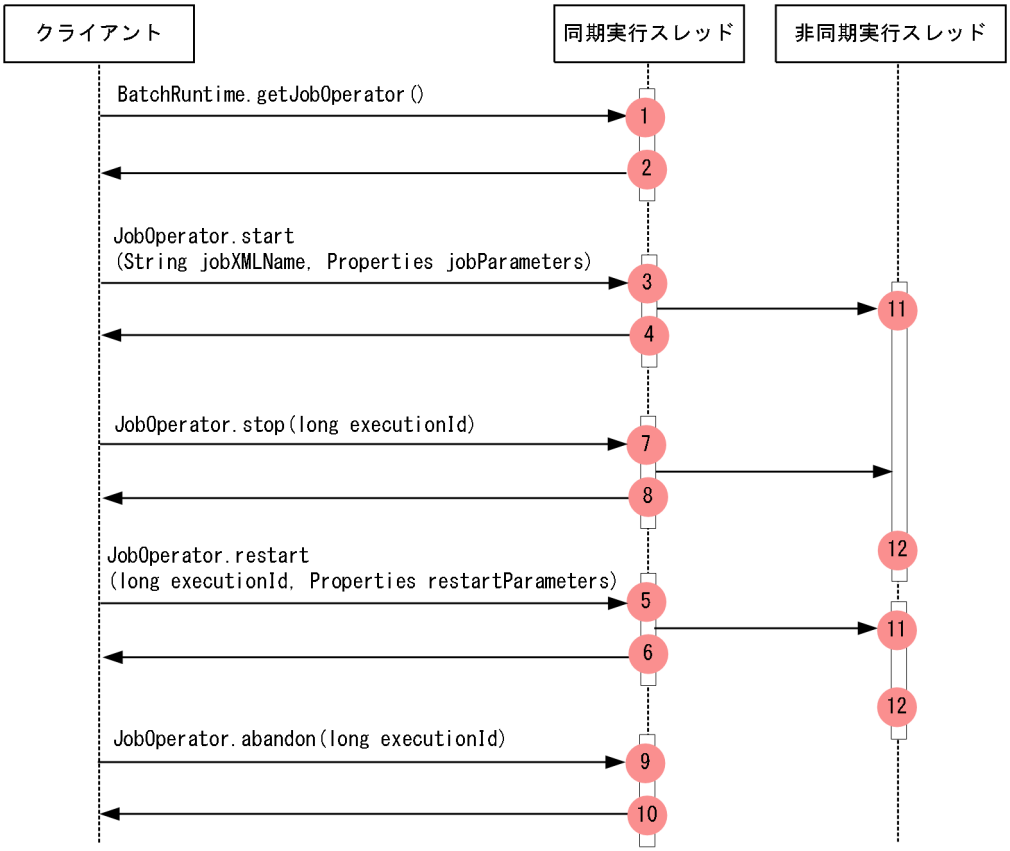
イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD042	35	javax.batch.api.listener.JobListener の実装クラスの afterJob()メソッドの呼び出し直前	A
0xD043	36	javax.batch.api.listener.JobListener の実装クラスの afterJob()メソッドの処理完了直後	A
0xD044	37	javax.batch.api.listener.StepListener の実装クラスの beforeStep()メソッドの呼び出し直前	A
0xD045	38	javax.batch.api.listener.StepListener の実装クラスの beforeStep()メソッドの処理完了直後	A
0xD046	39	javax.batch.api.listener.StepListener の実装クラスの afterStep()メソッドの呼び出し直前	A
0xD047	40	javax.batch.api.listener.StepListener の実装クラスの afterStep()メソッドの処理完了直後	A
0xD048	41	javax.batch.api.chunk.listener.ChunkListener の実装クラスの beforeChunk()メソッドの呼び出し直前	A
0xD049	42	javax.batch.api.chunk.listener.ChunkListener の実装クラスの beforeChunk()メソッドの処理完了直後	A
0xD04A	43	javax.batch.api.chunk.listener.ChunkListener の実装クラスの afterChunk()メソッドの呼び出し直前	A
0xD04B	44	javax.batch.api.chunk.listener.ChunkListener の実装クラスの afterChunk()メソッドの処理完了直後	A
0xD04C	45	javax.batch.api.chunk.listener.ChunkListener の実装クラスの onError(Exception ex)メソッドの呼び出し直前	A
0xD04D	46	javax.batch.api.chunk.listener.ChunkListener の実装クラスの onError(Exception ex)メソッドの処理完了直後	A

（凡例）A：標準

注※　図 8-106～図 8-109 中の番号と対応しています。

Java Batch でのトレース取得ポイントを次の図に示します。

図 8-106　Java Batch のトレース取得ポイント（クライアント・同期実行スレッド・非同期実行スレッド間で出力されるポイント）



（凡例）

●：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

図 8-107　Java Batch のトレース取得ポイント（ジョブの実行中に出力されるポイント）

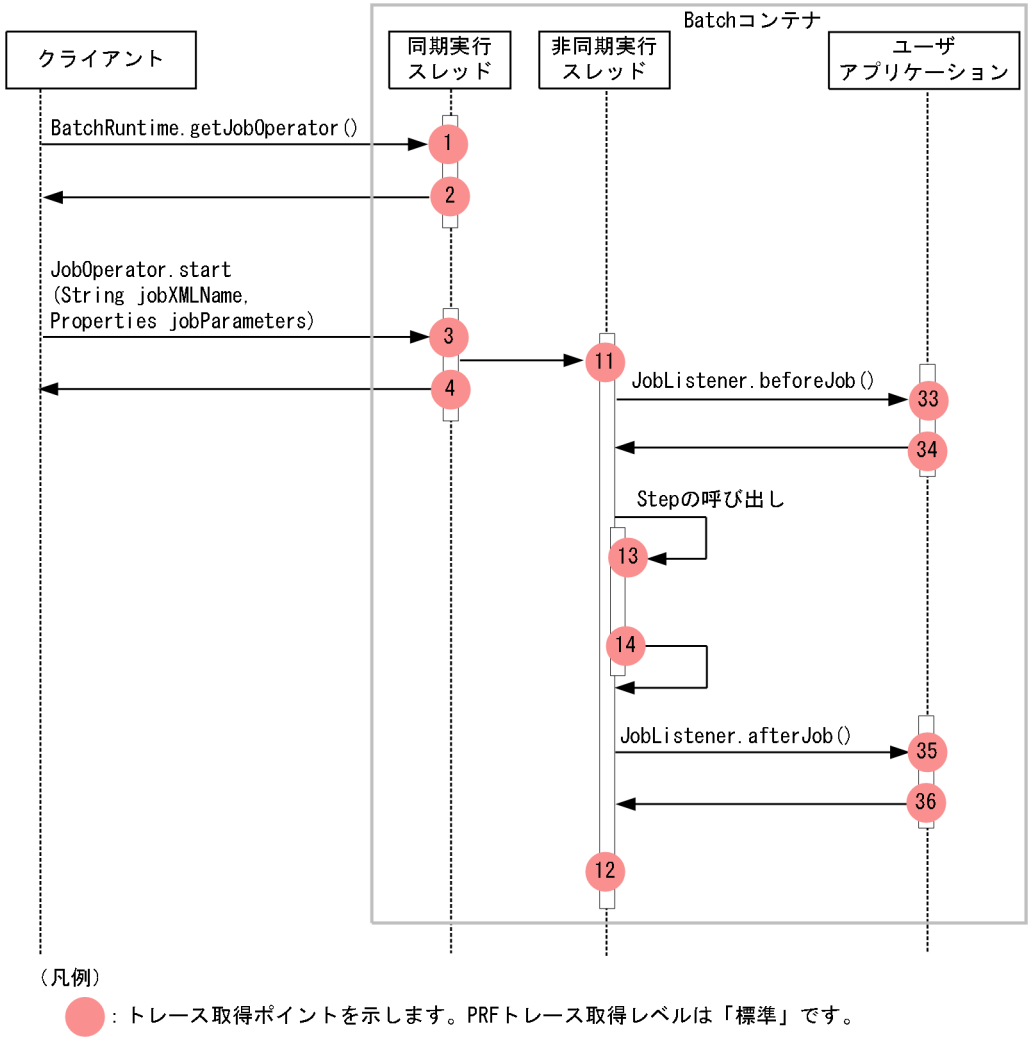


図 8-108　Java Batch のトレース取得ポイント（Step の処理中に出力されるポイント）

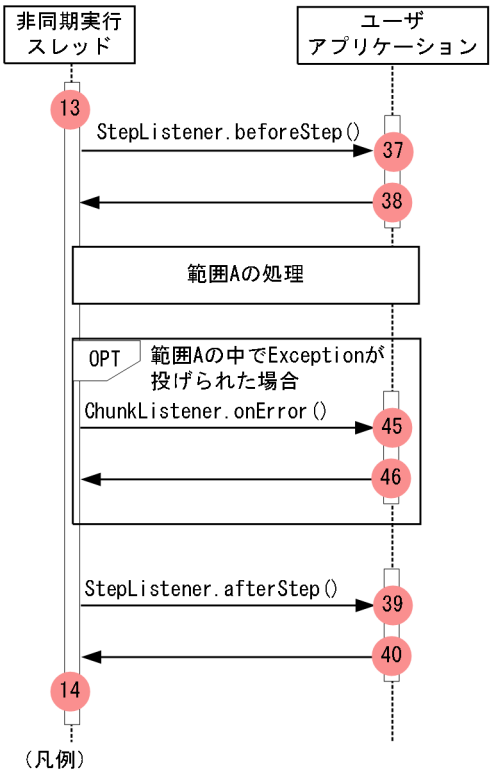
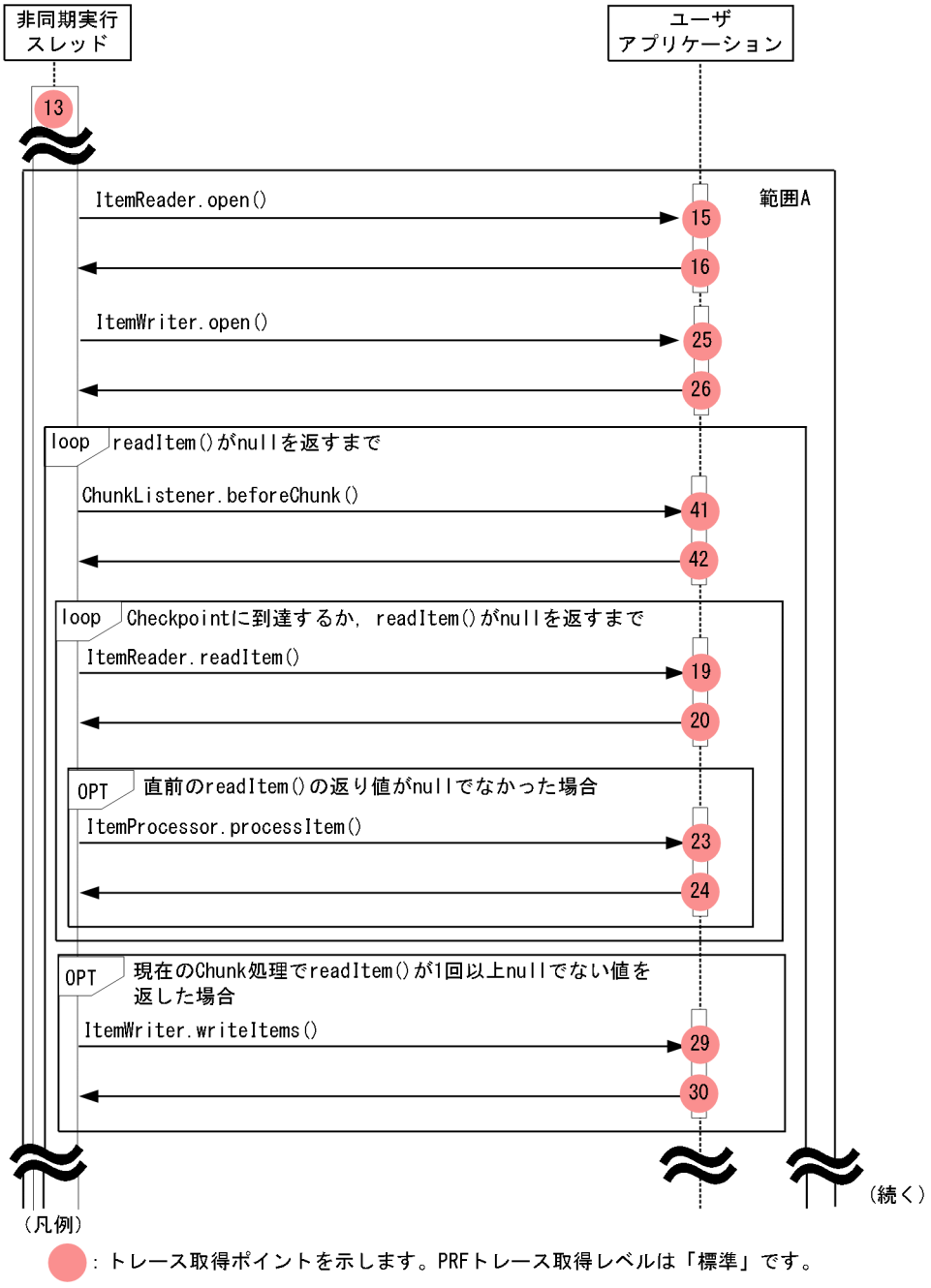
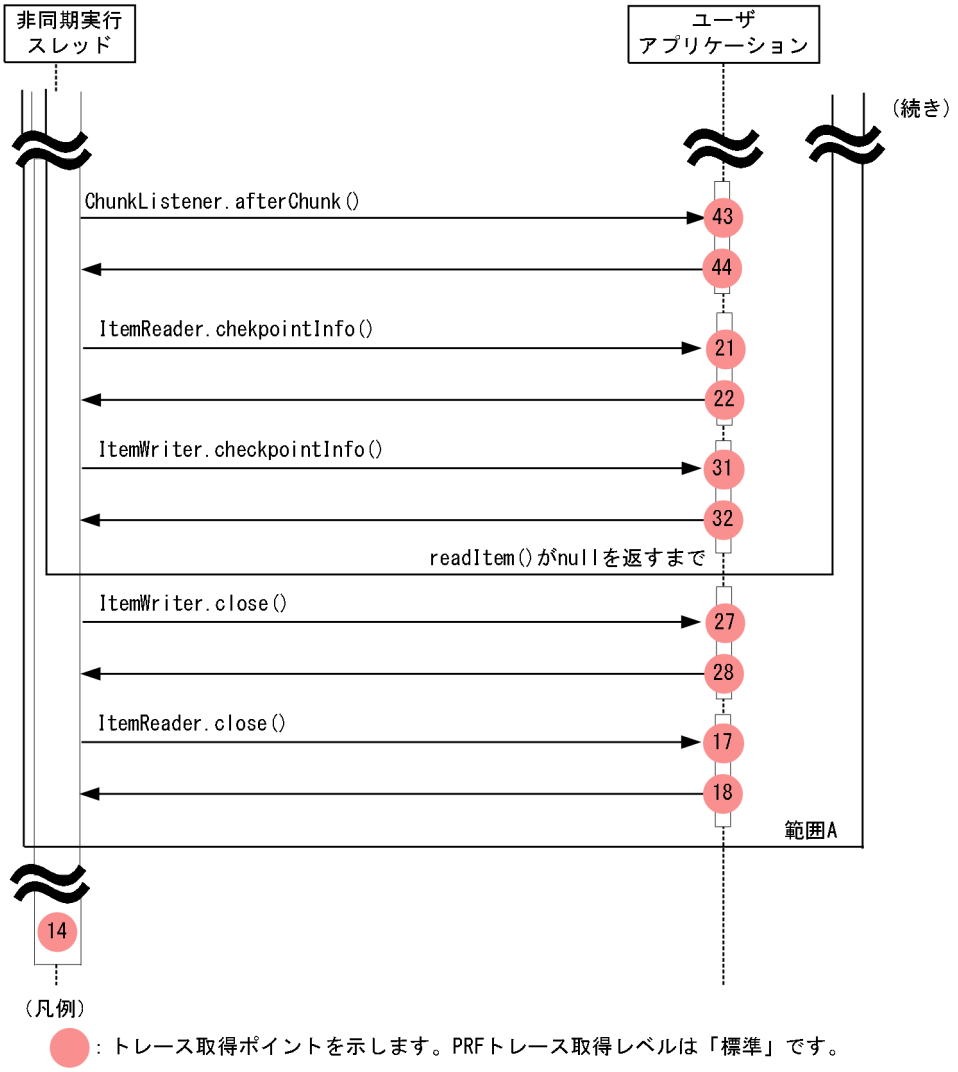


図 8-109　Java Batch のトレース取得ポイント（Step の処理中に出力されるポイント（範囲 A の処理））





8.27.2　取得できるトレース情報

Java Batch で取得できるトレース情報を次の表に示します。

表 8-152　Java Batch で取得できるトレース情報

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xD020	A	−	−	−
2	0xD021	A	−	−	• 正常時 なし • 異常時 例外名

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
3	0xD022	A	−	−	JobOperator.start()の引数の jobXMLName
4	0xD023	A	instanceId	executionId	• 正常時 なし • 異常時 例外名
5	0xD024	A	−	JobOperator.restart()の引数の executionId	−
6	0xD025	A	instanceId	新たに割り当てられた executionId	• 正常時 なし • 異常時 例外名
7	0xD026	A	−	JobOperator.stop()の引数の executionId	−
8	0xD027	A	−	−	• 正常時 なし • 異常時 例外名
9	0xD028	A	−	JobOperator.abandon()の引数の executionId	−
10	0xD029	A	−	−	• 正常時 なし • 異常時 例外名
11	0xD02A	A	−	−	−
12	0xD02B	A	Job batch status	−	• 正常時 Job exit status • 異常時 例外名
13	0xD02C	A	−	Step の ID	−
14	0xD02D	A	• TransitionElement が適用された場合 適用された TransitionElement 名 (next, fail, end, stop のどれか)	• 次の ExecutionElement が存在する場合 次に実行する ExecutionElement の ID • 上記以外の場合 なし	• 正常時 Step exit status • 異常時 例外名

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
			• next 属性が適用された場合 next • 上記以外の場合 Step batch status		
15	0xD02E	A	呼び出されたクラス名	−	−
16	0xD02F	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
17	0xD030	A	呼び出されたクラス名	−	−
18	0xD031	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
19	0xD032	A	呼び出されたクラス名	−	−
20	0xD033	A	呼び出されたクラス名	• 返り値が null null • 返り値が null 以外 not null • 異常時 なし	• 正常時 なし • 異常時 UP から発生した例外名
21	0xD034	A	呼び出されたクラス名	−	−
22	0xD035	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
23	0xD036	A	呼び出されたクラス名	−	−
24	0xD037	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
25	0xD038	A	呼び出されたクラス名	−	−
26	0xD039	A	呼び出されたクラス名	−	• 正常時

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					なし • 異常時 UP から発生した例外名
27	0xD03A	A	呼び出されたクラス名	−	−
28	0xD03B	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
29	0xD03C	A	呼び出されたクラス名	−	−
30	0xD03D	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
31	0xD03E	A	呼び出されたクラス名	−	−
32	0xD03F0	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
33	0xD040	A	呼び出されたクラス名	−	−
34	0xD041	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
35	0xD042	A	呼び出されたクラス名	−	−
36	0xD043	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
37	0xD044	A	呼び出されたクラス名	−	−
38	0xD045	A	呼び出されたクラス名	−	• 正常時 なし

図中の番号※	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
					• 異常時 UP から発生した例外名
39	0xD046	A	呼び出されたクラス名	−	−
40	0xD047	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
41	0xD048	A	呼び出されたクラス名	−	−
42	0xD049	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
43	0xD04A	A	呼び出されたクラス名	−	−
44	0xD04B	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名
45	0xD04C	A	呼び出されたクラス名	−	ChunkListener.onError(Exception)の引数の Exception
46	0xD04D	A	呼び出されたクラス名	−	• 正常時 なし • 異常時 UP から発生した例外名

（凡例）A：標準　−：該当なし

注※　図 8-106～図 8-109 中の番号と対応しています。

8.27.3　例外ログの出力

性能解析トレースの取得ポイントで付加情報（OPT）列に例外名を出力した場合は，同時に例外ログに例外の詳細情報を出力します。障害解析時には性能解析トレースで出力した例外名と例外ログを照合して障害解析を行うことができます。ただし，バッチアプリケーションなどで意図的に例外を発生させてジョブ

の処理を制御する目的で使用している場合，例外を発生させるたびに例外ログに詳細情報が出力されるため，ログ出力量が増加することがあります。このようなアプリケーションを実行する場合は，あらかじめ例外ログのサイズを大きく設定してください。

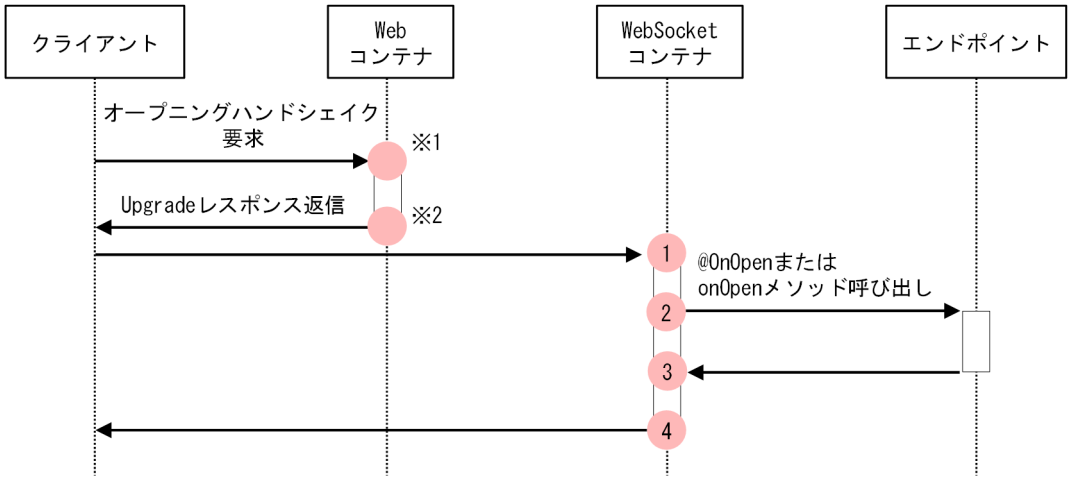
8.28　WebSocket のトレース取得ポイント

WebSocket でのトレース取得ポイントの詳細について説明します。

8.28.1　オープニングハンドシェイク要求受信時

WebSocket でのトレースの取得ポイントを次に示します。

図 8-110　WebSocket のトレース取得ポイント（オープニングハンドシェイク要求受信時）



（凡例）

：トレース取得ポイントを示します。

注※1　Web コンテナの取得ポイント（0x8236）を示します。
注※2　Web コンテナの取得ポイント（0x8336）を示します。

イベント ID，トレースレベル，トレース取得ポイント，取得できる情報について次の表に示します。

表 8-153　WebSocket のトレース取得ポイントの詳細（オープニングハンドシェイク要求受信時）

イベント ID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE100	1	A	オープニングハンドシェイク要求受信後処理開始直前	CONNECT	ハンドシェイク時の URI	−
0xE120	2	A	@OnOpen アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onOpen メソッドの呼び出し直前	エンドポイントのクラス名	メソッド名	−

イベントID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE121	3	A	@OnOpen アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onOpen メソッドの処理完了直後	エンドポイントのクラス名	メソッド名	• 正常時 なし • 異常時 例外名
0xE101	4	A	オープニングハンドシェイク要求受信後処理完了直後	CONNECT	ハンドシェイク時の URI	• 正常時 セッション ID • 異常時 例外名

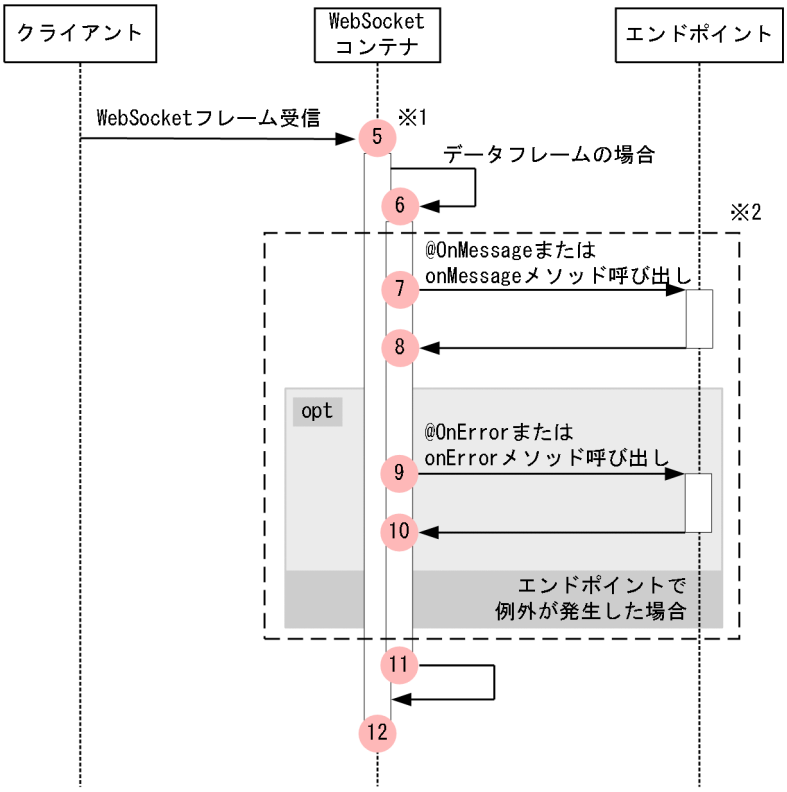
（凡例）A：標準　－：該当なし

注※　図 8-110 中の番号と対応しています。

8.28.2　メッセージ受信時

WebSocket でのトレースの取得ポイントを次に示します。

図 8-111　WebSocket のトレース取得ポイント（メッセージ（データフレーム）受信時）



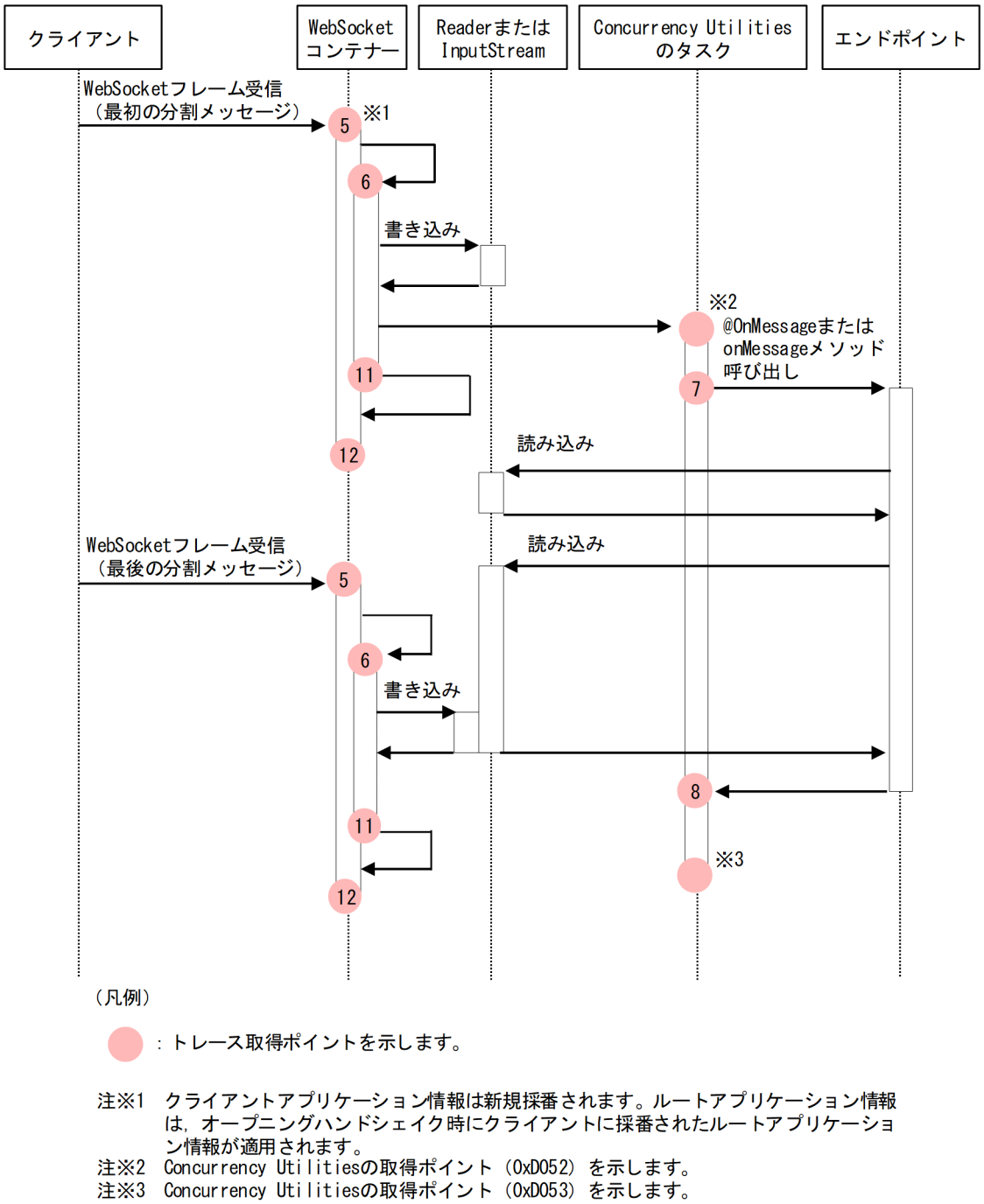
（凡例）

●：トレース取得ポイントを示します。

注※1　クライアントアプリケーション情報は新規採番されます。ルートアプリケーション情報は，オープニングハンドシェイク時にクライアントに採番されたルートアプリケーション情報が適用されます。

注※2　エンドポイントで@OnMessageが付与されたメソッドまたは javax.websocket.MessageHandler.Wholeの実装クラスが使用されていて，分割されたメッセージを受信した場合，枠内の取得ポイントは最後のフレームを受信したときだけ呼ばれます。

図 8-112　WebSocket のトレース取得ポイント（Reader または InputStream 型の引数を持つメッセージハンドラによる分割メッセージ受信時）



注※1　クライアントアプリケーション情報は新規採番されます。ルートアプリケーション情報は，オープニングハンドシェイク時にクライアントに採番されたルートアプリケーション情報が適用されます。
注※2　Concurrency Utilitiesの取得ポイント（0xD052）を示します。
注※3　Concurrency Utilitiesの取得ポイント（0xD053）を示します。

イベント ID，トレースレベル，トレース取得ポイント，取得できる情報について次の表に示します。

表 8-154　WebSocket のトレース取得ポイントの詳細（メッセージ受信時）

イベント ID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE102	5	A	受信した WebSocket フレームに対する処理開始直前	READ	−	−
0xE103	12	A	受信した WebSocket フレームに対する処理完了直後	READ	−	• 正常時 なし • 異常時 例外名
0xE140	6	A	データフレーム受信後処理開始直前	MESSAGE	次のどれか • onMessage(String) • onMessage(ByteBuffer) • onPartialMessage(String) • onPartialMessage(ByteBuffer)	−
0xE141	11	A	データフレーム受信後処理完了直後	MESSAGE	次のどれか • onMessage(String) • onMessage(ByteBuffer) • onPartialMessage(String) • onPartialMessage(ByteBuffer)	• 正常時 なし • 異常時 例外名
0xE122	7	A	@OnMessage アノテーションが付与されたメソッドの呼び出し直前	エンドポイントのクラス名	メソッド名	−
0xE123	8	A	@OnMessage アノテーションが付与されたメソッド処理完了直後	エンドポイントのクラス名	メソッド名	• 正常時 なし • 異常時 例外名

イベントID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE124	7	A	javax.websocket.MessageHandler.Whole の実装クラスの onMessage メソッド呼び出し直前	javax.websocket.MessageHandler.Whole の実装クラス名	メソッド名	−
0xE125	8	A	javax.websocket.MessageHandler.Whole の実装クラスの onMessage メソッド処理完了直後	javax.websocket.MessageHandler.Whole の実装クラス名	メソッド名	• 正常時 なし • 異常時 例外名
0xE126	7	A	javax.websocket.MessageHandler.Partial の実装クラスの onMessage メソッド呼び出し直前	javax.websocket.MessageHandler.Partial の実装クラス名	メソッド名	−
0xE127	8	A	javax.websocket.MessageHandler.Partial の実装クラスの onMessage メソッド処理完了直後	javax.websocket.MessageHandler.Partial の実装クラス名	メソッド名	• 正常時 なし • 異常時 例外名
0xE128	9	A	@OnError アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onError メソッドの呼び出し直前	エンドポイントのクラス名	メソッド名	−
0xE129	10	A	@OnError アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onError メソッドの処理完了直後	エンドポイントのクラス名	メソッド名	• 正常時 なし • 異常時 例外名

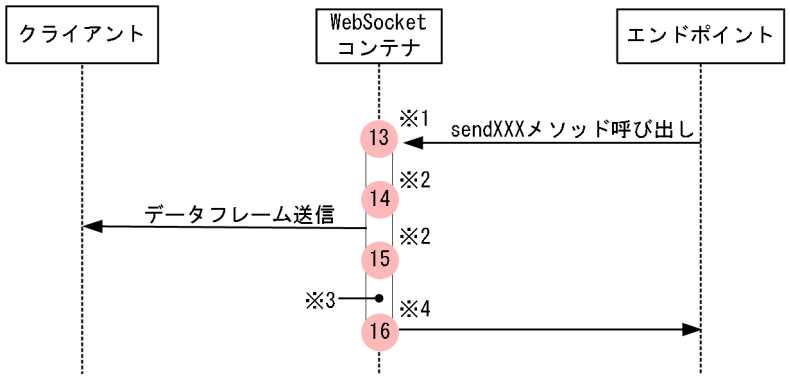
（凡例）A：標準　−：該当なし

注※　図 8-111，図 8-112 中の番号と対応しています。

8.28.3　データ送信時

WebSocket でのトレースの取得ポイントを次に示します。

図 8-113　WebSocket のトレース取得ポイント（データ送信時）



（凡例）

：トレース取得ポイントを示します。

注※1　クライアントアプリケーション情報は新規採番されます。ルートアプリケーション情報は，sendXXXメソッドの呼び出し元のルートアプリケーション情報が適用されます。

注※2　クライアントアプリケーション情報は，項番13で採番されたクライアントアプリケーション情報が適用されます。ルートアプリケーション情報は，データフレーム送信先のルートアプリケーション情報が適用されます。

注※3　ここでWebSocketアクセスログが出力されます。ルートアプリケーション情報は項番15と同じ値が適用されます。

注※4　クライアントアプリケーション情報は，項番13で採番されたクライアントアプリケーション情報が適用されます。ルートアプリケーション情報は，sendXXXメソッドの呼び出し元のルートアプリケーション情報が適用されます。

イベント ID，トレースレベル，トレース取得ポイント，取得できる情報について次の表に示します。

表 8-155　WebSocket のトレース取得ポイントの詳細（データ送信時）

イベント ID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE112	14	A	WebSocket フレームの送信処理開始直前	WRITE	－	－
0xE113	15	A	WebSocket フレームの送信処理完了直後	WRITE	－	• 正常時 なし • 異常時 例外名
0xE130	13	A	javax.websocket.RemoteEndpoint.Async.sendXXX メソッドが呼び出された直後	javax.websocket.RemoteEndpoint.Async	メソッド名（引数の型）	送信先のセッション ID
0xE131	16	A	javax.websocket.RemoteEndpoint.Async.sendXXX メソッドのリターン直前	javax.websocket.RemoteEndpoint.Async	メソッド名（引数の型）	• 正常時 なし • 異常時 例外名

イベントID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE132	13	A	javax.websocket.RemoteEndpoint.Basic.sendXXX メソッドが呼び出された直後	javax.websocket.RemoteEndpoint.Basic	メソッド名（引数の型）	送信先のセッション ID
0xE133	16	A	javax.websocket.RemoteEndpoint.Basic.sendXXX メソッドのリターン直前	javax.websocket.RemoteEndpoint.Basic	メソッド名（引数の型）	• 正常時 なし • 異常時 例外名
0xE134	13	A	javax.websocket.RemoteEndpoint.sendXXX メソッドが呼び出された直後	javax.websocket.RemoteEndpoint	メソッド名（引数の型）	送信先のセッション ID
0xE135	16	A	javax.websocket.RemoteEndpoint.sendXXX メソッドのリターン直前	javax.websocket.RemoteEndpoint	メソッド名（引数の型）	• 正常時 なし • 異常時 例外名

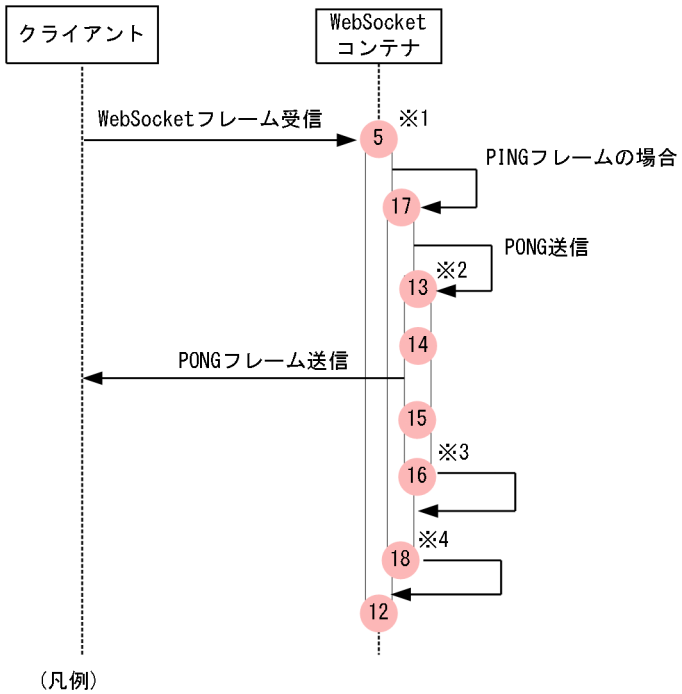
（凡例）A：標準　－：該当なし

注※　図 8-113 中の番号と対応しています。

8.28.4　PING 受信時

WebSocket でのトレースの取得ポイントを次に示します。

図 8-114　WebSocket のトレース取得ポイント（PING 受信時）



注※1　クライアントアプリケーション情報は新規採番されます。ルートアプリケーション情報は，オープニングハンドシェイク時にクライアントに採番されたルートアプリケーション情報が適用されます。

注※2　クライアントアプリケーション情報は新規採番されます。ルートアプリケーション情報は，項番5と同じルートアプリケーション情報が適用されます。

注※3　クライアントアプリケーション情報は，項番13で採番されたクライアントアプリケーション情報が適用されます。ルートアプリケーション情報は，項番5と同じルートアプリケーション情報が適用されます。

注※4　クライアントアプリケーション情報は，項番5で採番したクライアントアプリケーション情報が適用されます。ルートアプリケーション情報は，項番5と同じルートアプリケーション情報が適用されます。

イベント ID，トレースレベル，トレース取得ポイント，取得できる情報について次の表に示します。

表 8-156　WebSocket のトレース取得ポイントの詳細（PING 受信時）

イベント ID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE102	5	A	受信した WebSocket フレームに対する処理開始直前	READ	－	－
0xE103	12	A	受信した WebSocket フレームに対する処理完了直後	READ	－	• 正常時 なし • 異常時 例外名
0xE112	14	A	WebSocket フレームの送信処理開始直前	WRITE	－	－

イベントID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE113	15	A	WebSocket フレームの送信処理完了直後	WRITE	−	• 正常時 なし • 異常時 例外名
0xE142	17	A	PING フレーム受信後処理開始直前	PING	−	−
0xE143	18	A	PING フレーム受信後処理完了直後	PING	−	• 正常時 なし • 異常時 例外名
0xE130	13	A	javax.websocket.RemoteEndpoint.Async.sendXXX メソッドが呼び出された直後	javax.websocket.RemoteEndpoint.Async	メソッド名（引数の型）	送信先のセッション ID
0xE131	16	A	javax.websocket.RemoteEndpoint.Async.sendXXX メソッドのリターン直前	javax.websocket.RemoteEndpoint.Async	メソッド名（引数の型）	• 正常時 なし • 異常時 例外名
0xE132	13	A	javax.websocket.RemoteEndpoint.Basic.sendXXX メソッドが呼び出された直後	javax.websocket.RemoteEndpoint.Basic	メソッド名（引数の型）	送信先のセッション ID
0xE133	16	A	javax.websocket.RemoteEndpoint.Basic.sendXXX メソッドのリターン直前	javax.websocket.RemoteEndpoint.Basic	メソッド名（引数の型）	• 正常時 なし • 異常時 例外名
0xE134	13	A	javax.websocket.RemoteEndpoint.sendXXX メソッドが呼び出された直後	javax.websocket.RemoteEndpoint	メソッド名（引数の型）	送信先のセッション ID
0xE135	16	A	javax.websocket.RemoteEndpoint.sendXXX メソッドのリターン直前	javax.websocket.RemoteEndpoint	メソッド名（引数の型）	• 正常時 なし • 異常時 例外名

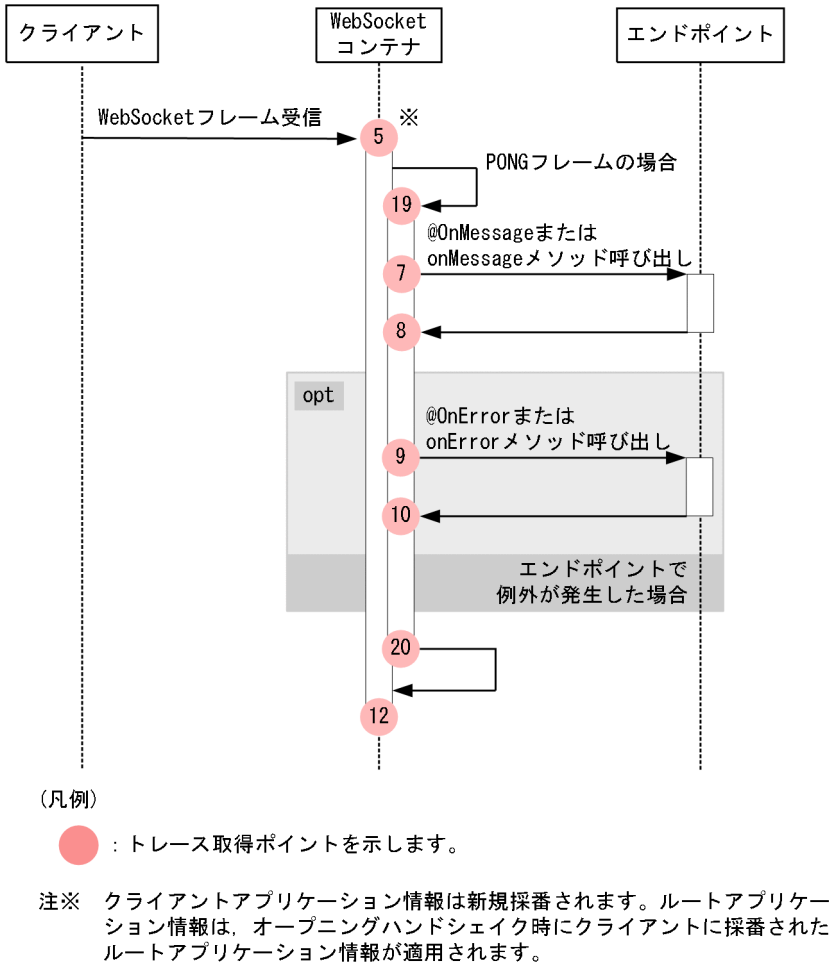
（凡例）A：標準　−：該当なし

注※　図 8-114 中の番号と対応しています。

8.28.5　PONG 受信時

WebSocket でのトレースの取得ポイントを次に示します。

図 8-115　WebSocket のトレース取得ポイント（PONG 受信時）



イベント ID，トレースレベル，トレース取得ポイント，取得できる情報について次の表に示します。

表 8-157　WebSocket のトレース取得ポイントの詳細（PONG 受信時）

イベント ID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE102	5	A	受信した WebSocket フレームに対する処理開始直前	READ	−	−
0xE103	12	A	受信した WebSocket フレームに対する処理完了直後	READ	−	• 正常時 なし • 異常時 例外名

イベントID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE144	19	A	PONG フレーム受信後処理開始直前	PONG	−	−
0xE145	20	A	PONG フレーム受信後処理完了直後	PONG	−	• 正常時 なし • 異常時 例外名
0xE122	7	A	@OnMessage アノテーションが付与されたメソッドの呼び出し直前	エンドポイントのクラス名	メソッド名	−
0xE123	8	A	@OnMessage アノテーションが付与されたメソッド処理完了直後	エンドポイントのクラス名	メソッド名	• 正常時 なし • 異常時 例外名
0xE124	7	A	javax.websocket.MessageHandler.Whole の実装クラスの onMessage メソッド呼び出し直前	javax.websocket.MessageHandler.Whole の実装クラス名	メソッド名	−
0xE125	8	A	javax.websocket.MessageHandler.Whole の実装クラスの onMessage メソッド処理完了直後	javax.websocket.MessageHandler.Whole の実装クラス名	メソッド名	• 正常時 なし • 異常時 例外名
0xE126	7	A	javax.websocket.MessageHandler.Partial の実装クラスの onMessage メソッド呼び出し直前	javax.websocket.MessageHandler.Partial の実装クラス名	メソッド名	−
0xE127	8	A	javax.websocket.MessageHandler.Partial の実装クラスの onMessage メソッド処理完了直後	javax.websocket.MessageHandler.Partial の実装クラス名	メソッド名	• 正常時 なし • 異常時 例外名
0xE128	9	A	@OnError アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onError メソッドの呼び出し直前	エンドポイントのクラス名	メソッド名	−
0xE129	10	A	@OnError アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onError メソッドの処理完了直後	エンドポイントのクラス名	メソッド名	• 正常時 なし • 異常時 例外名

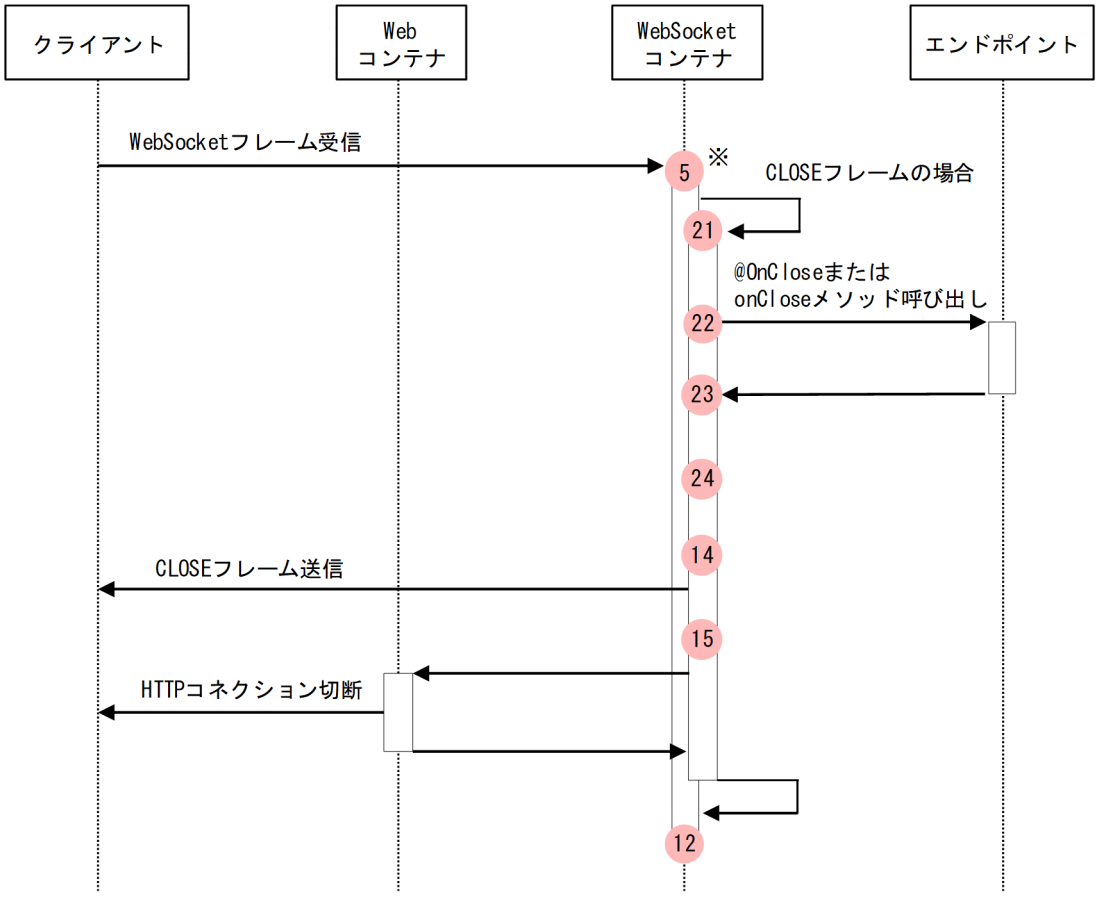
（凡例）A：標準　−：該当なし

注※　図 8-115 中の番号と対応しています。

8.28.6　クロージングハンドシェイク要求受信時

WebSocket でのトレースの取得ポイントを次に示します。

図 8-116　WebSocket のトレース取得ポイント（クロージングハンドシェイク要求受信時（ペイロードデータあり））

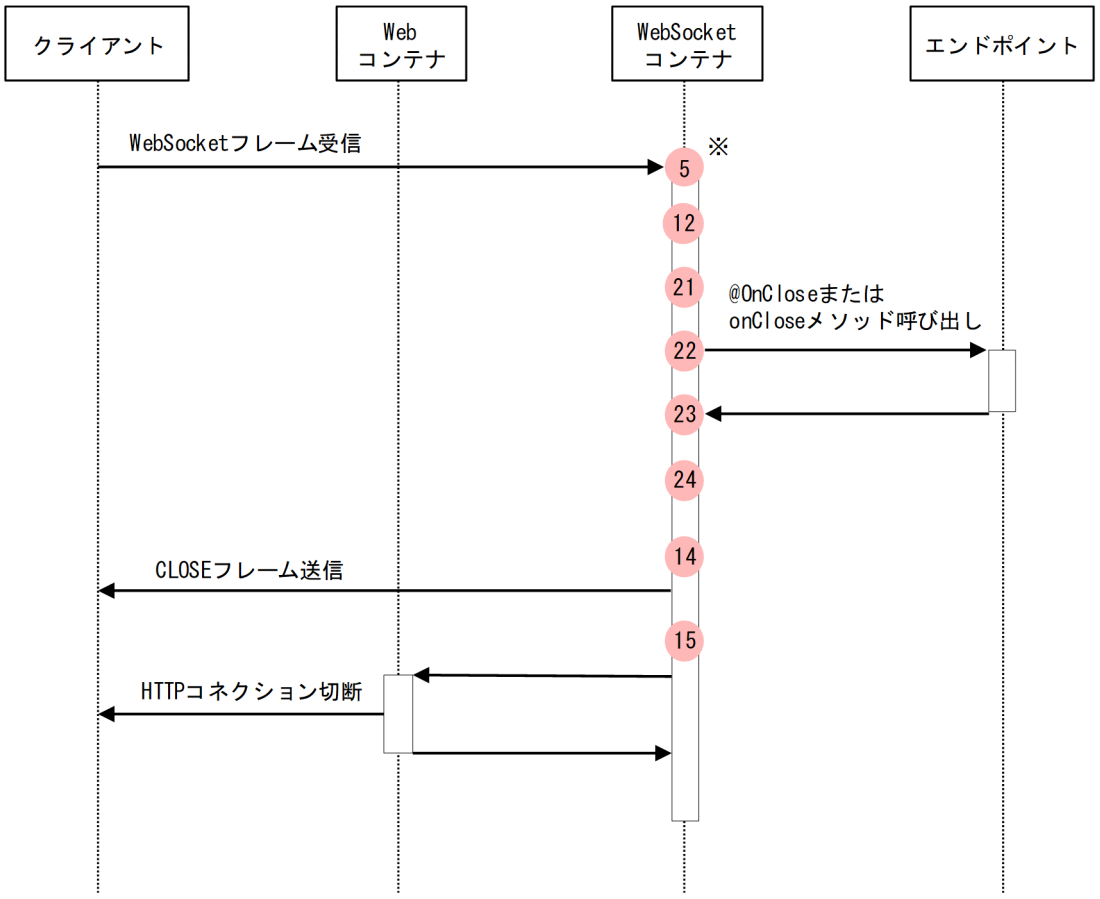


（凡例）

●：トレース取得ポイントを示します。

注※　クライアントアプリケーション情報は新規採番されます。ルートアプリケーション情報は，オープニングハンドシェイク時にクライアントに採番されたルートアプリケーション情報が適用されます。

図 8-117　WebSocket のトレース取得ポイント（クロージングハンドシェイク要求受信時（ペイロードデータなし））



（凡例）

：トレース取得ポイントを示します。

注※　クライアントアプリケーション情報は新規採番されます。ルートアプリケーション情報は，オープニングハンドシェイク時にクライアントに採番されたルートアプリケーション情報が適用されます。

イベント ID，トレースレベル，トレース取得ポイント，取得できる情報について次の表に示します。

表 8-158　WebSocket のトレース取得ポイントの詳細（クロージングハンドシェイク要求受信時）

イベント ID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE102	5	A	受信した WebSocket フレームに対する処理開始直前	READ	−	−
0xE103	12	A	受信した WebSocket フレームに対する処理完了直後	READ	−	• 正常時 なし

イベントID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
						• 異常時 例外名
0xE112	14	A	WebSocket フレームの送信処理開始直前	WRITE	－	－
0xE113	15	A	WebSocket フレームの送信処理完了直後	WRITE	－	• 正常時 なし • 異常時 例外名
0xE146	21	A	クロージングハンドシェイク要求受信後処理または送信後処理開始直前	CLOSE	切断要因	－
0xE147	24	A	クロージングハンドシェイク要求受信後処理または送信後処理完了直後	CLOSE	切断要因	• 正常時 なし • 異常時 例外名
0xE12A	22	A	@OnClose アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onClose メソッドの呼び出し直前	エンドポイントのクラス名	メソッド名	－
0xE12B	23	A	@OnClose アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onClose メソッドの処理完了直後	エンドポイントのクラス名	メソッド名	• 正常時 なし • 異常時 例外名

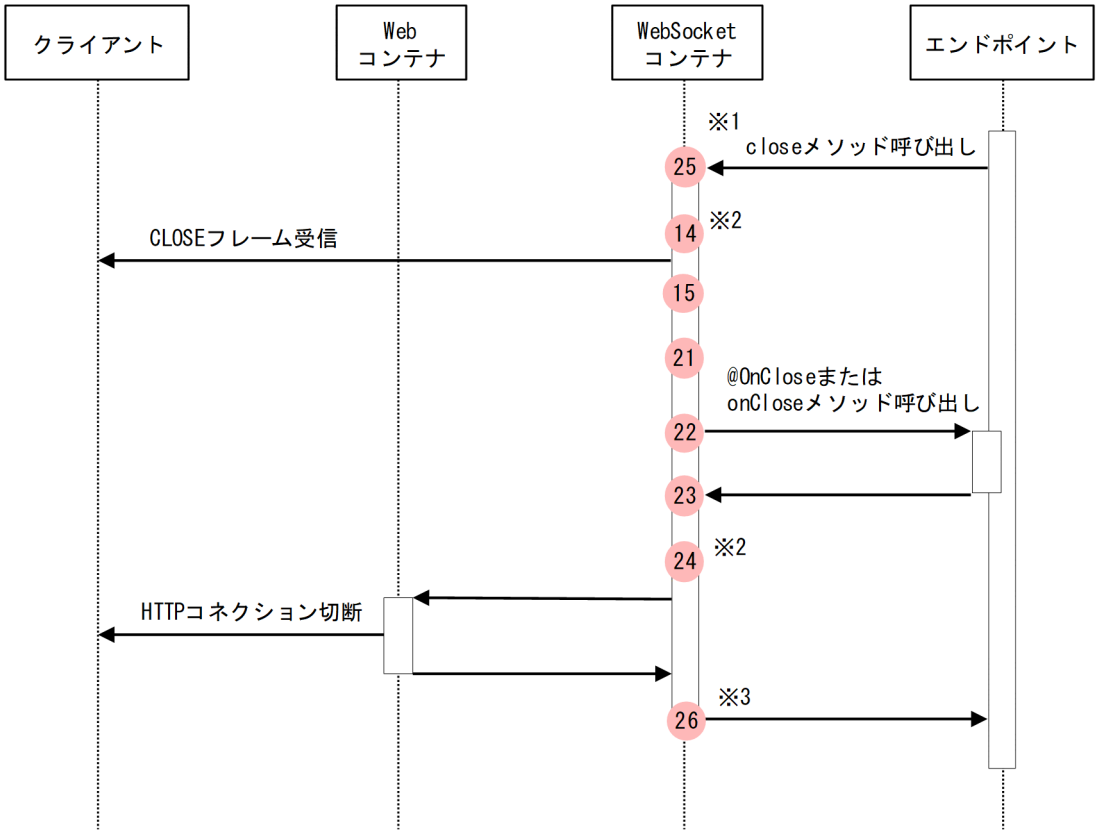
（凡例）A：標準　－：該当なし

注※　図 8-116，図 8-117 中の番号と対応しています。

8.28.7　クロージングハンドシェイク要求送信時

WebSocket でのトレースの取得ポイントを次に示します。

図 8-118　WebSocket のトレース取得ポイント（クロージングハンドシェイク要求送信時）



（凡例）

：トレース取得ポイントを示します。

注※1　クライアントアプリケーション情報は新規採番されます。ルートアプリケーション情報は，sendXXXメソッドの呼び出し元のルートアプリケーション情報が適用されます。

注※2　クライアントアプリケーション情報は，項番25で採番されたクライアントアプリケーション情報が適用されます。ルートアプリケーション情報は，データフレーム送信先のルートアプリケーション情報が適用されます。

注※3　クライアントアプリケーション情報は，項番25で採番されたクライアントアプリケーション情報が適用されます。ルートアプリケーション情報は，sendXXXメソッドの呼び出し元のルートアプリケーション情報が適用されます。

イベント ID，トレースレベル，トレース取得ポイント，取得できる情報について次の表に示します。

表 8-159　WebSocket のトレース取得ポイントの詳細（クロージングハンドシェイク要求送信時

イベント ID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
0xE112	14	A	WebSocket フレームの送信処理開始直前	WRITE	－	－
0xE113	15	A	WebSocket フレームの送信処理完了直後	WRITE	－	• 正常時 なし • 異常時

イベントID	図中の番号※	レベル	トレース取得ポイント	取得できる情報		
				インタフェース名	オペレーション名	オプション
						例外名
0xE146	21	A	クロージングハンドシェイク要求受信後処理または送信後処理開始直前	CLOSE	切断要因	－
0xE147	24	A	クロージングハンドシェイク要求受信後処理または送信後処理完了直後	CLOSE	切断要因	• 正常時 なし • 異常時 例外名
0xE12A	22	A	@OnClose アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onClose メソッドの呼び出し直前	エンドポイントのクラス名	メソッド名	－
0xE12B	23	A	@OnClose アノテーションが付与されたメソッドまたは javax.websocket.Endpoint 実装クラスの onClose メソッドの処理完了直後	エンドポイントのクラス名	メソッド名	• 正常時 なし • 異常時 例外名
0xE136	25	A	javax.websocket.Session.close メソッドが呼び出された直後	javax.websocket.Session	メソッド名（引数の型）	送信先のセッション ID
0xE137	26	A	javax.websocket.Session.close メソッドのリターン直前	javax.websocket.Session	メソッド名（引数の型）	• 正常時 なし • 異常時 例外名

（凡例）A：標準　－：該当なし

注※　図 8-118 中の番号と対応しています。

8.29　Concurrency Utilities のトレース取得ポイント

ここでは，Concurrency Utilities のトレース取得ポイントと，取得できるトレース情報について説明します。

8.29.1　トレース取得ポイントおよび取得できるトレース情報

イベント ID，トレース取得ポイント，および PRF トレース取得レベルについて，次の表に示します。

表 8-160　Concurrency Utilities でのトレース取得ポイントの詳細

イベント ID	図中の番号※	トレース取得ポイント	レベル
0xD050	1	新規タスク登録メソッドの入口	A
0xD051	2	新規タスク登録メソッドの出口	A
0xD052	3	タスクの呼び出し直前	A
0xD053	4	タスクの終了直後	A
0xD054	5	タスク登録時に戻り値として返された Future または ScheduledFuture オブジェクトの get メソッドの入口	A
0xD055	6	タスク登録時に戻り値として返された Future または ScheduledFuture オブジェクトの get メソッドの出口	A
0xD056	7	タスク登録時に戻り値として返された Future または ScheduledFuture オブジェクトの cancel メソッドの入口	A
0xD057	8	タスク登録時に戻り値として返された Future または ScheduledFuture オブジェクトの cancel メソッドの出口	A

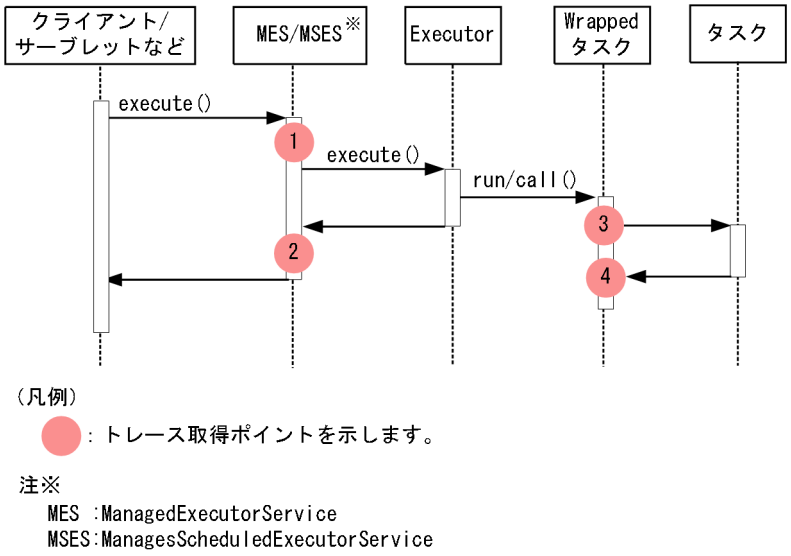
（凡例）A：標準

注※　図 8-119～図 8-126 中の番号と対応しています。

次のメソッドを呼び出した場合のトレース取得ポイントを図 8-119 に示します。

- javax.enterprise.concurrent.ManagedExecutorService#execute(Runnable)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#execute(Runnable)

図 8-119　Concurrency Utilities のトレース取得ポイント（その 1）



次のメソッドを呼び出した場合のトレース取得ポイントを図 8-120～図 8-124 に示します。get メソッド，または cancel メソッドを呼び出すタイミングによって，トレース取得ポイントが異なります。

タスク終了前に get メソッドを呼び出したとき：図 8-120

タスク終了後に get メソッドを呼び出したとき：図 8-121

タスク実行開始前に cancel メソッドを呼び出したとき：図 8-122

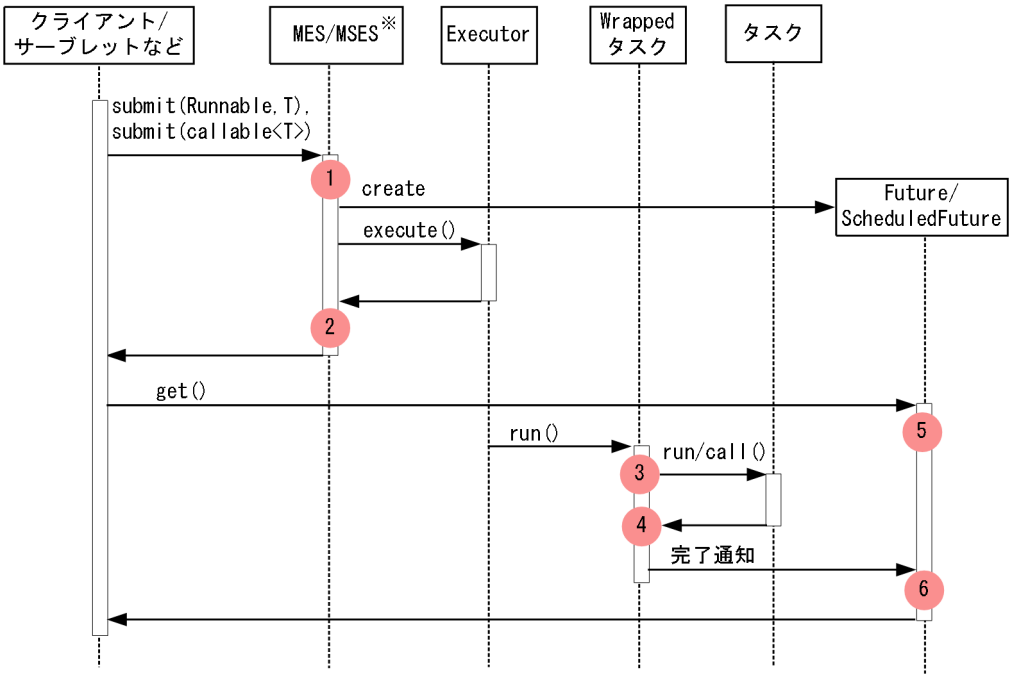
タスク処理中に cancel メソッドを呼び出したとき：図 8-123

タスク終了後に cancel メソッドを呼び出したとき：図 8-124

- javax.enterprise.concurrent.ManagedExecutorService#submit(Runnable)
- javax.enterprise.concurrent.ManagedExecutorService#submit(Runnable, T)
- javax.enterprise.concurrent.ManagedExecutorService#submit(Callable<T>)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#submit(Runnable)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#submit(Runnable, T)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#submit(Callable<T>)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#schedule(Callable<V>, long, TimeUnit)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#schedule(Callable<V>, Trigger)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#schedule(Runnable, long, TimeUnit)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#schedule(Runnable, Trigger)

- javax.enterprise.concurrent.ManagedScheduledExecutorService#scheduleAtFixedRate(Runnable, long, long, TimeUnit)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#scheduleWithFixedDelay(Runnable, long, long, TimeUnit)

図 8-120　Concurrency Utilities のトレース取得ポイント（その 2）（タスク終了前に get メソッドを呼び出したとき）



（凡例）

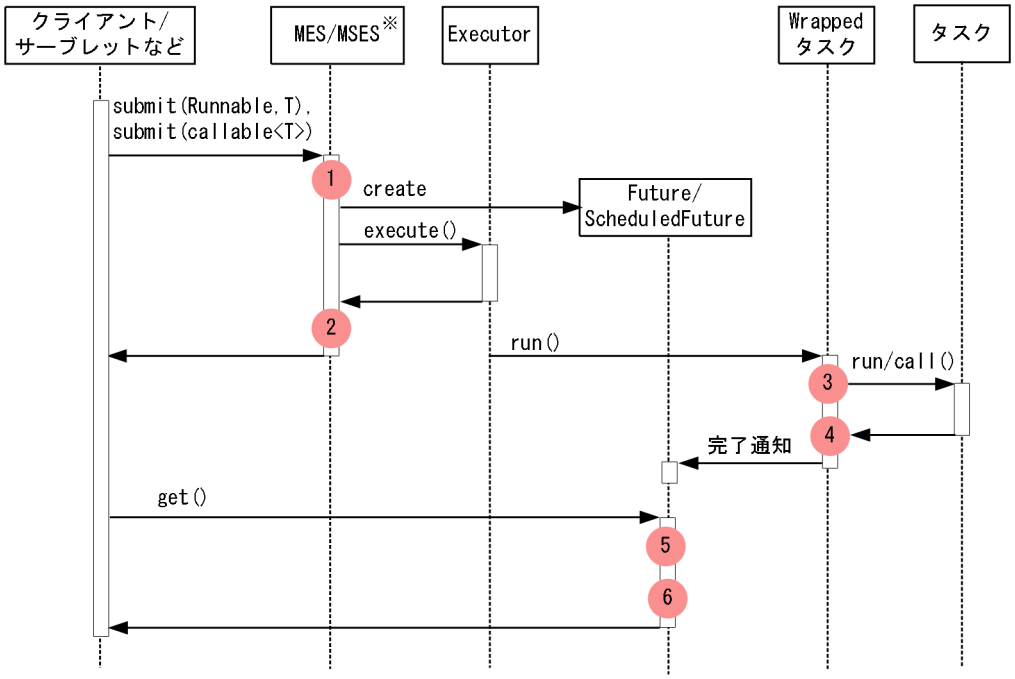
●：トレース取得ポイントを示します。

注※

MES :ManagedExecutorService

MSES:ManagesScheduledExecutorService

図 8-121　Concurrency Utilities のトレース取得ポイント（その 3）（タスク終了後に get メソッドを呼び出したとき）



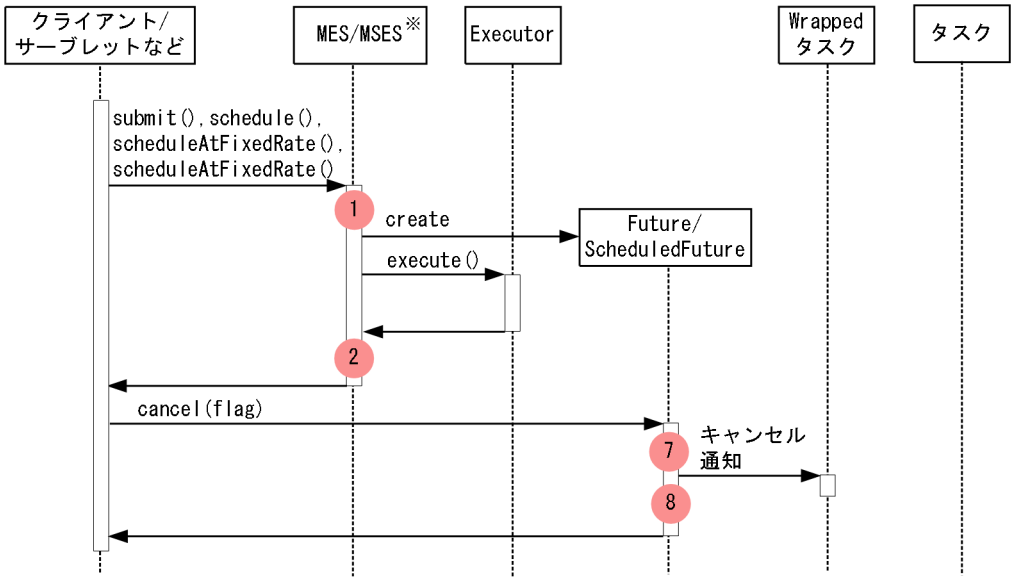
（凡例）

●：トレース取得ポイントを示します。

注※

MES :ManagedExecutorService
MSES:ManagesScheduledExecutorService

図 8-122　Concurrency Utilities のトレース取得ポイント（その 4）（タスク実行開始前に cancel メソッドを呼び出したとき）



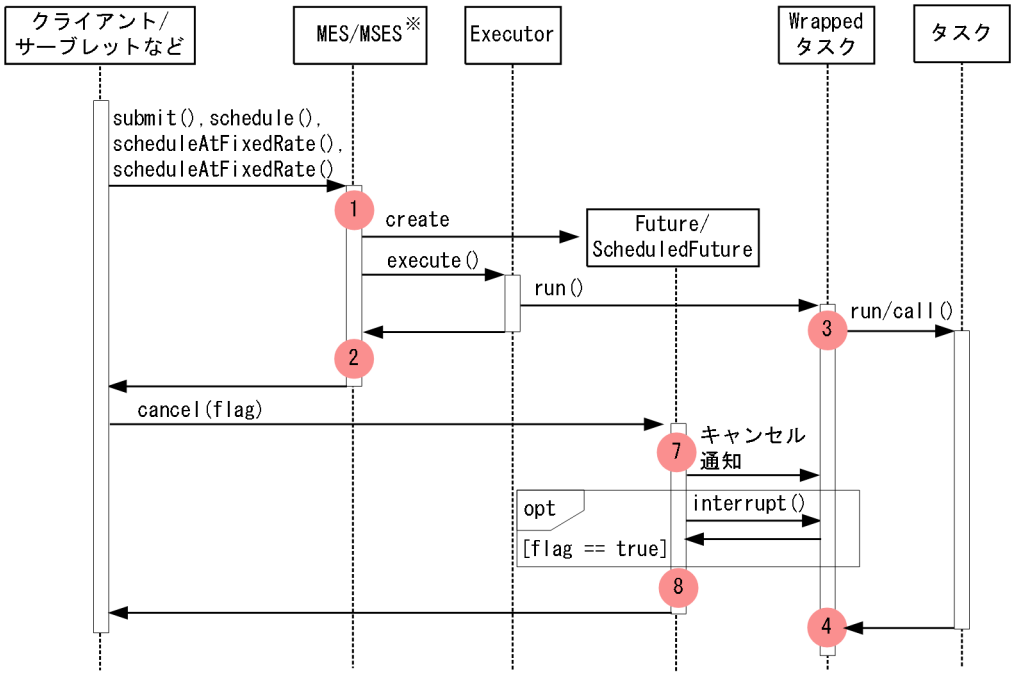
（凡例）

●：トレース取得ポイントを示します。

注※

MES :ManagedExecutorService
MSES:ManagesScheduledExecutorService

図 8-123　Concurrency Utilities のトレース取得ポイント（その 5）（タスク処理中に cancel メソッドを呼び出したとき）



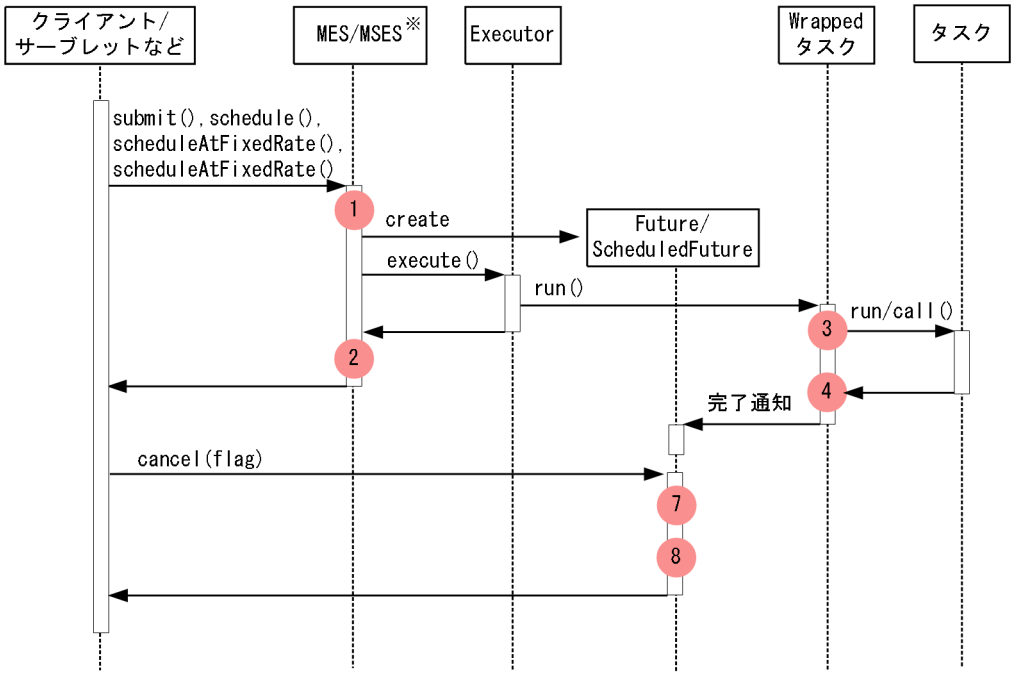
（凡例）

●：トレース取得ポイントを示します。

注※

MES :ManagedExecutorService
MSES:ManagesScheduledExecutorService

図 8-124　Concurrency Utilities のトレース取得ポイント（その 6）（タスク終了後に cancel メソッドを呼び出したとき）



（凡例）

●：トレース取得ポイントを示します。

注※

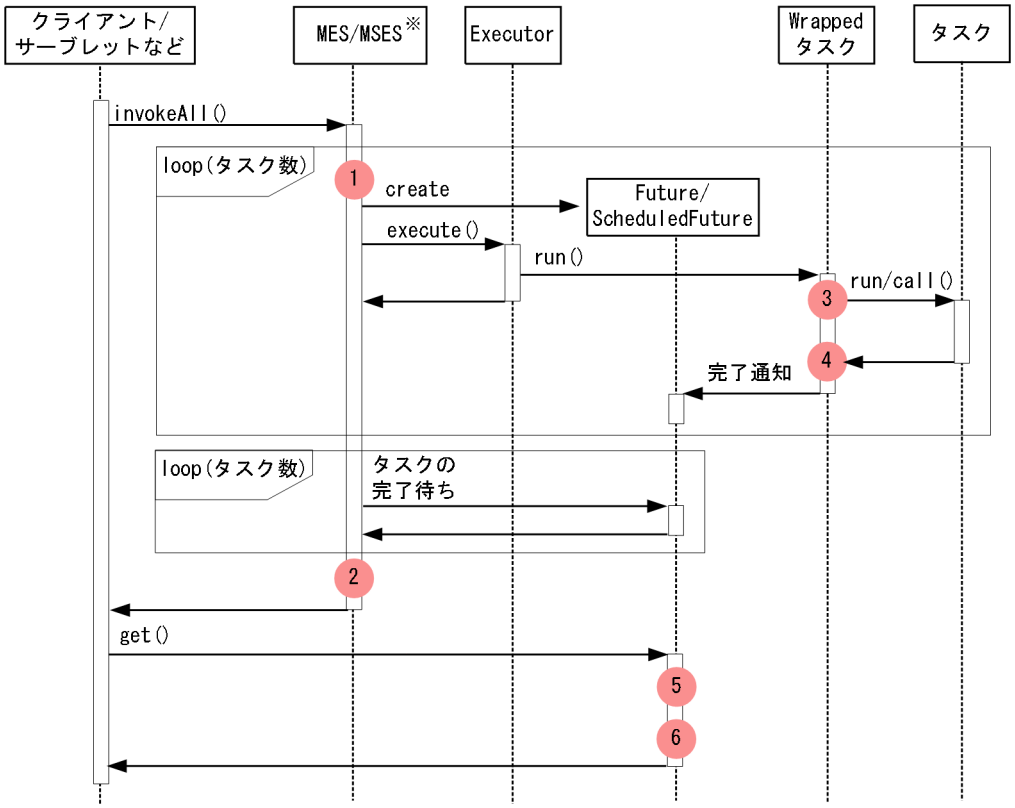
MES :ManagedExecutorService

MSES:ManagesScheduledExecutorService

次のメソッドを呼び出した場合のトレース取得ポイントを図 8-125 に示します。

- javax.enterprise.concurrent.ManagedExecutorService#invokeAll(Collection<? Extends Callable<T>>)
- javax.enterprise.concurrent.ManagedExecutorService#invokeAll(Collection<? Extends Callable<T>>, long, TimeUnit)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#invokeAll(Collection<? Extends Callable<T>>)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#invokeAll(Collection<? Extends Callable<T>>, long, TimeUnit)

図 8-125　Concurrency Utilities のトレース取得ポイント（その 7）



（凡例）

●：トレース取得ポイントを示します。

注※

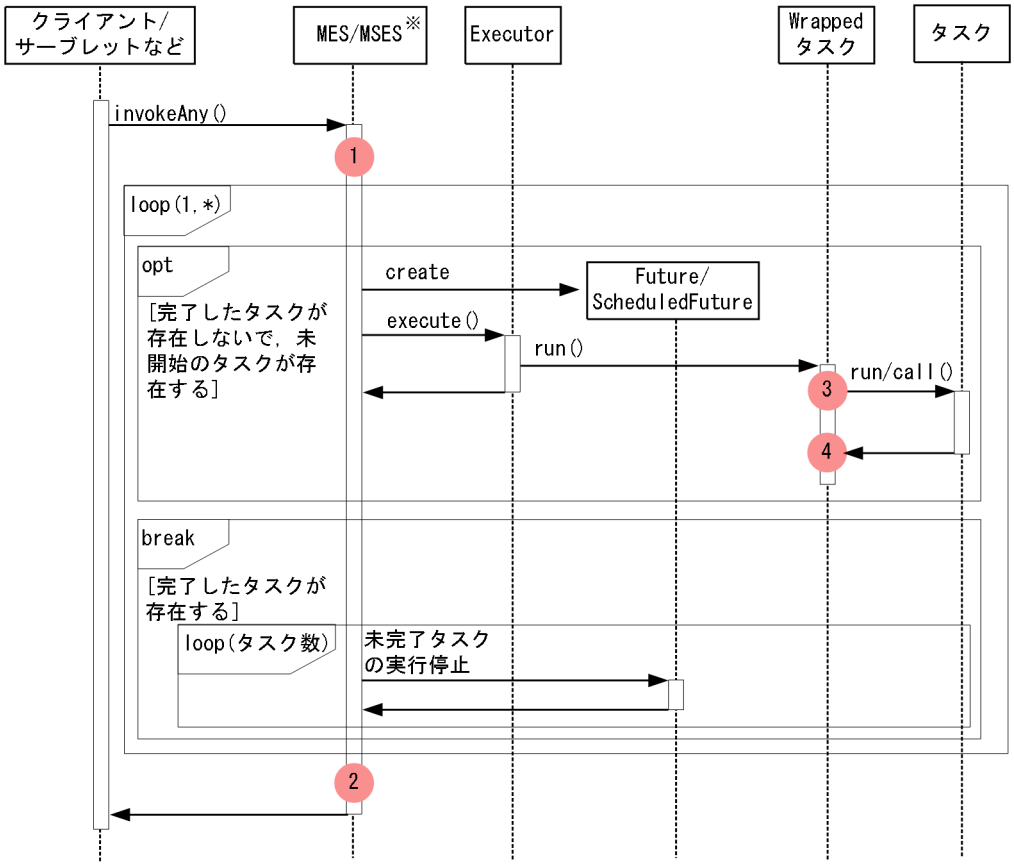
MES :ManagedExecutorService

MSES:ManagesScheduledExecutorService

次のメソッドを呼び出した場合のトレース取得ポイントを図 8-126 に示します。

- javax.enterprise.concurrent.ManagedExecutorService#invokeAny(Collection<? Extends Callable<T>>)
- javax.enterprise.concurrent.ManagedExecutorService#invokeAny(Collection<? Extends Callable<T>>, long, TimeUnit)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#invokeAny(Collection<? Extends Callable<T>>)
- javax.enterprise.concurrent.ManagedScheduledExecutorService#invokeAny(Collection<? Extends Callable<T>>, long, TimeUnit)

図 8-126　Concurrency Utilities のトレース取得ポイント（その 8）



（凡例）

●：トレース取得ポイントを示します。

注※

MES :ManagedExecutorService

MSES:ManagesScheduledExecutorService

8.29.2　取得できるトレース情報

Concurrency Utilities で取得できるトレース情報を次の表に示します。

表 8-161　JMS Connection インタフェースで取得できるトレース情報

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
1	0xD050	A	登録先の ManagedExecutorService または ManagedScheduledExecutorService の JNDI 名	呼び出されたメソッド名（複数のタスクを指定可能なメソッド※2 の場合は，カッコ内にタスク数を出力する）	新規タスクのクラス名（複数の場合はコンマ区切り）
2	0xD051	A	登録先の ManagedExecutorService または	タスク ID	• 正常時 なし

図中の番号※1	イベント ID	レベル	取得できる情報		
			インタフェース名	オペレーション名	オプション
			ManagedScheduledExecutorService の JNDI 名		• 異常時 例外名
3	0xD052	A	呼び出し元の ManagedExecutorService または ManagedScheduledExecutorService の JNDI 名	• 単一タスクの場合 タスク ID • 複数タスクの場合 タスク ID-リスト通番	−
4	0xD053	A	呼び出し元の ManagedExecutorService または ManagedScheduledExecutorService の JNDI 名	• 単一タスクの場合 タスク ID • 複数タスクの場合 タスク ID-リスト通番	• 正常時 なし • 異常時 例外名
5	0xD054	A	登録先の ManagedExecutorService または ManagedScheduledExecutorService の JNDI 名	• 単一タスクの場合 タスク ID • 複数タスクの場合 タスク ID-リスト番号	−
6	0xD055	A	登録先の ManagedExecutorService または ManagedScheduledExecutorService の JNDI 名	• 単一タスクの場合 タスク ID • 複数タスクの場合 タスク ID-リスト番号	• 正常時 なし • 異常時 例外名
7	0xD056	A	登録先の ManagedExecutorService または ManagedScheduledExecutorService の JNDI 名	• 単一タスクの場合 タスク ID • 複数タスクの場合 タスク ID-リスト番号	−
8	0xD057	A	登録先の ManagedExecutorService または ManagedScheduledExecutorService の JNDI 名	• 正常時 cancel メソッドの戻り値 • 異常時 なし	• 正常時 なし • 異常時 例外名

（凡例）A：標準　−：該当なし

注※1　図 8-119～図 8-126 の番号と対応しています。

注※2　複数のタスクを指定可能なメソッドとは，第 1 引数に Collection オブジェクトを入れるものです。実引数の Collection オブジェクトの size をかっこ書きで出力します。

（例）invokeAny メソッドで要素 3 つの Collection を渡した場合

invokeAny(3)

なお，Collection オブジェクトが null だった場合は，カッコ内に「null」を出力します。

（例）Collection オブジェクトが null だった場合

invokeAny(null)

9 製品の JavaVM の機能

製品の JavaVM を使用すると，障害時の資料やチューニングで使用する情報を取得できます。この章では，製品の JavaVM が提供する機能について説明します。なお，製品の JavaVM が提供する機能の一つである明示管理ヒープ機能については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7. 明示管理ヒープ機能を使用した FullGC の抑止」を参照してください。

9.1　この章の構成

製品の JavaVM は，構成ソフトウェアである Developer's Kit for Java によって提供されます。アプリケーションサーバで動作する J2EE サーバまたはバッチサーバのプロセスは，JavaVM 上で実行されます。この節では，製品の JavaVM の機能について説明します。

製品の JavaVM の機能の概要については，「9.2　製品の JavaVM の機能の概要」を参照してください。

アプリケーションサーバで提供する製品の JavaVM の機能と参照先を次の表に示します。

表 9-1　アプリケーションサーバで提供する製品の JavaVM の機能

機能名	参照先
クラス別統計機能	9.3
インスタンス統計機能※	9.4
STATIC メンバ統計機能※	9.5
参照関係情報出力機能※	9.6
統計前の GC 選択機能※	9.7
Tenured 領域内不要オブジェクト統計機能※	9.8
Tenured 増加要因の基点オブジェクトリスト出力機能※	9.9
クラス別統計情報解析機能	9.10
Survivor 領域の年齢分布情報出力機能	9.11
hndlwrap 機能	9.12
JIT コンパイル時の C ヒープ確保量の上限値設定機能	9.13
スレッド数の上限値設定機能	9.14
製品の JavaVM の機能使用時の注意事項	9.15

注※　クラス別統計情報を出力するクラス別統計機能の一つです。

9.2　製品の JavaVM の機能の概要

アプリケーションサーバで動作する J2EE サーバまたはバッチサーバのプロセスは，JavaVM 上で実行されます。製品の JavaVM で提供している機能を次に示します。

- 明示管理ヒープ機能
- クラス別統計機能
  - インスタンス統計機能
  - STATIC メンバ統計機能
  - 参照関係情報出力機能
  - 統計前の GC 選択機能
  - Tenured 領域内不要オブジェクト統計機能
  - Tenured 増加要因の基点オブジェクトリスト出力機能
- クラス別統計情報解析機能※
- Survivor 領域の年齢分布情報出力機能
- hndlwrap 機能
- JIT コンパイル時の C ヒープ確保量の上限値設定機能
- スレッド数の上限値設定機能

注※

クラス別統計情報解析機能を使うと，拡張スレッドダンプファイルに出力されたクラス別統計情報を CSV 形式で出力できます。

ポイント

クラス別統計機能の Tenured 領域内不要オブジェクト統計機能を実行した場合，次に示す機能は無効となります。

- インスタンス統計機能
- STATIC メンバ統計機能
- 統計前の GC 選択機能

製品の JavaVM では，障害発生時の要因分析やシステムの状態確認に利用できるよう，ログの出力内容が拡張されています。このログは，製品の JavaVM ログファイルに出力され，標準の JavaVM よりも，多くのトラブルシュート情報が取得できます。さらに，このログ（拡張 verbosegc 情報）を利用して適切なチューニングを実施することで，システムの可用性向上が図れます。製品の JavaVM ログファイルについては，「5.7　JavaVM ログ（JavaVM ログファイル）」を参照してください。Java VM のチューニングについては，マニュアル「アプリケーションサーバ システム設計ガイド」の「7.　JavaVM のメモリチューニング」を参照してください。

次節以降で，製品の JavaVM の各機能について説明します。なお，明示管理ヒープ機能については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7. 明示管理ヒープ機能を使用した FullGC の抑止」を参照してください。

9.3　クラス別統計機能

この節では，クラス別統計機能について説明します。

クラス別統計機能を使用すると，クラスごとに，参照関係にあるクラスのインスタンスの情報を拡張スレッドダンプに出力できます。

この節の構成を次の表に示します。

表 9-2　この節の構成（クラス別統計機能）

分類	タイトル	参照先
解説	クラス別統計機能の概要	9.3.1
	クラス別統計情報を前提とする機能	9.3.2
運用	クラス別統計情報の出力	9.3.3
注意事項	クラス別統計情報出力時の注意事項	9.3.4

注 「実装」および「設定」について，この機能固有の説明はありません。

9.3.1　クラス別統計機能の概要

各クラスのインスタンスが持つメンバの配下にあるすべてのインスタンスのサイズを，クラスごとに統計情報として拡張スレッドダンプに出力できます。この統計情報を**クラス別統計情報**といいます。クラス別統計情報を複数回出力すると，GC による Java オブジェクトの変化や，寿命が短い Java オブジェクトの状態，クラスごとのサイズの変化，Java オブジェクトの親子関係などを調査できます。これらの情報は，1 トランザクション当たりのメモリ使用量の測定や，メモリリークの検出などに利用できます。

クラス別統計情報は，次に示す機能によって出力されます。

- インスタンス統計機能
- STATIC メンバ統計機能
- 参照関係情報出力機能
- 統計前の GC 選択機能
- Tenured 領域内不要オブジェクト統計機能
- Tenured 増加要因の基点オブジェクトリスト出力機能

各機能については，「9.3.2　クラス別統計機能を前提とする機能」を参照してください。

また，クラス別統計機能では，Java ヒープ（Eden 領域，Survivor 領域および Tenured 領域を合わせたもの）にあるインスタンスを統計対象とします。さらに，明示管理ヒープ機能を使用している場合には，

Explicit ヒープにあるインスタンスも統計対象にできます。クラス別統計情報の出力方法については，「9.3.3　クラス別統計情報の出力」を参照してください。

> **参考**
>
> 拡張スレッドダンプは，デフォルトで出力されるように設定されています。拡張スレッドダンプを取得するための設定については，「3.3.17(1)　JavaVM のスレッドダンプ取得の設定」を参照してください。出力情報については，「5.5　JavaVM のスレッドダンプ」を参照してください。
>
> また，クラス別統計情報は，CSV 形式に出力することもできます。クラス別統計情報を CSV 形式で出力する方法については，「9.10　クラス別統計情報解析機能」を参照してください。

9.3.2　クラス別統計機能を前提とする機能

クラス別統計機能を前提とする機能の種類と概要を次の表に示します。

表 9-3　クラス別統計機能を前提とする機能の種類と概要

種類	概要	参照先
インスタンス統計機能	クラスごとに，クラスのインスタンスがメンバとして持つインスタンスの合計サイズを出力します。	9.4
STATIC メンバ統計機能	各クラスの static メンバが持つインスタンスの合計サイズを出力します。	9.5
参照関係情報出力機能	指定したクラス（インスタンス）をメンバに持つクラスの参照関係を出力します。	9.6
統計前の GC 選択機能	クラス別統計情報を出力する前に，GC を実行するかどうかを選択できます。この機能は，jheapprof コマンド実行時にオプションで指定します。デフォルトの設定では，クラス別統計情報を出力する前に FullGC が実行されます。	9.7
Tenured 領域内不要オブジェクト統計機能	Tenured 領域内で不要となったオブジェクトを特定します。	9.8
Tenured 増加要因の基点オブジェクトリスト出力機能	Tenured 領域内不要オブジェクト統計機能を使って特定した，不要オブジェクトの基点となるオブジェクトの情報を出力します。	9.9

これらの機能のうち，インスタンス統計機能，STATIC メンバ統計機能，参照関係情報出力機能，および Tenured 増加要因の基点オブジェクトリスト出力機能は，クラス別統計機能の実行時に有効になります。

統計前の GC 選択機能は，クラス別統計機能を実行する際に設定できます。クラス別統計情報の調査目的に合わせて選択してください。詳細は，「9.7.2　GC の選択の指針」を参照してください。

9.3.3　クラス別統計情報の出力

ここでは，クラス別統計情報を出力する方法について説明します。

クラス別統計情報を拡張スレッドダンプに出力するには，jheapprof コマンドを利用します。クラス別統計情報を出力したい Java プロセスや，参照関係情報を出力したいクラスを指定して，jheapprof コマンドを実行します。

jheapprof コマンド実行時には，次の指定ができます。

- クラス別統計情報に Explicit ヒープの情報を出力するかどうかの指定
- クラス別統計情報を取得する前に GC を実行するかどうかの指定

次に，jheapprof コマンドの実行形式と実行例，および各指定方法について説明します。

(1)　jheapprof コマンドの実行形式と実行例

jheapprof コマンドの実行形式と実行例を次に示します。jheapprof コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「jheapprof（クラス別統計情報付き拡張スレッドダンプの出力)」を参照してください。

実行形式

- Windows の場合

```
jheapprof [-f|-i] [-explicit|-noexplicit] [-class <クラス名>] [-fullgc|-copygc|-nogc]
[-garbage|-nogarbage] [-rootobjectinfo|-norootobjectinfo] [-rootobjectinfost <サイズ>]
 -p <プロセスID>
```

- UNIX の場合

```
jheapprof [-f|-i] [-explicit|-noexplicit] [-class <クラス名>] [-fullgc|-copygc|-nogc]
[-garbage|-nogarbage] [-rootobjectinfo|-norootobjectinfo] [-rootobjectinfost <サイズ>]
 [-force] -p <プロセスID>
```

実行例

ここでは，プロセス ID が 2463 の Java プロセスのクラス別統計情報を出力します。

1. -p オプションに，クラス別統計情報を出力したい Java プロセスのプロセス ID を指定して，jheapprof コマンドを実行します。

```
% jheapprof -p 2463
```

jheapprof コマンドで-f オプションを省略している場合，次の確認メッセージが表示されます。

Windows の場合

クラス別統計情報付き拡張スレッドダンプを出力するかどうかを確認するメッセージが次の形式で表示されます。

```
Force VM to output HitachiJavaHeapProfile: ? (y/n)
```

UNIX の場合

プロセス ID を確認するメッセージが次の形式で表示されます。

```
send SIGQUIT to 2463: ? (y/n)
```

2. y を入力します。

クラス別統計情報付き拡張スレッドダンプが出力されます。実行中の java プログラムでは次のメッセージが出力されます。

```
Writing Java core to javacore2463.030806215140.txt... OK
```

実行中の java プログラムは，カレントディレクトリにクラス別統計情報付き拡張スレッドダンプ（javacore<プロセス ID>.<日時>.txt）を作成し，プログラムを継続します。

(2) Explicit ヒープの情報をクラス別統計情報に出力する場合

次の条件を満たしている場合，Explicit ヒープの情報をクラス別統計情報に出力できます。

- JavaVM 起動オプションに-XX:+HitachiUseExplicitMemory を指定している。
- アプリケーションの実装，または実行環境（J2EE サーバ）の設定で Explicit ヒープが使用されている。

Explicit ヒープの情報をクラス別統計情報に出力する場合は，jheapprof コマンドに-explicit オプションを指定して実行します。

明示管理ヒープ機能については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7. 明示管理ヒープ機能を使用した FullGC の抑止」を参照してください。

(3) GC の実行有無を指定する場合

クラス別統計情報を出力する前に，GC を実行するかどうかを選択できます。この機能を統計前の GC 選択機能といいます。クラス別統計情報を出力する前に，GC を実行するかどうかは，jheapprof コマンドに次のどれかのオプションを指定して実行します。

- -fullgc
  FullGC を実行してから，クラス別統計情報を出力します。
- -copygc
  CopyGC を実行してから，クラス別統計情報を出力します。
- -nogc
  GC を実行しないで，クラス別統計情報を出力します。

統計前の GC 選択機能については，「9.7 統計前の GC 選択機能」を参照してください。なお，Tenured 領域内不要オブジェクト統計機能を実行する場合，統計前の GC 選択機能は実行できません。

9.3.4　クラス別統計情報出力時の注意事項

ここでは，クラス別統計情報出力時の注意事項について説明します。

- 起動中の Java プロセスに対して，-copygc オプションを設定した jheapprof コマンドを実行すると，統計前に CopyGC を実行しようとします。この場合に，Tenured 領域の空き容量が足りないと，CopyGC が実行できないときがあります。

  CopyGC が実行できない場合，JavaVM 起動オプションに-XX:+HitachiVerboseGC を指定していても，GC が発生した時の拡張 verbosegc 情報は出力されません。なお，クラス別統計情報を含む拡張スレッドダンプは，GC 実行時と同様に出力されます。

- JavaVM 起動時に，-XX:+PrintGCDetails が指定された Java プロセスに対して，-copygc オプションを設定した jheapprof コマンドを実行すると，統計前に CopyGC を実行します。この場合，-XX:+PrintGCDetails の指定によって出力される GC 情報には，「Full GC」と出力されます。

9.4　インスタンス統計機能

この節では，インスタンス統計機能について説明します。

インスタンス統計機能は，クラス別統計情報を出力する機能の一つです。クラスごとに，クラスのインスタンスの数やインスタンスの合計サイズが出力できます。

インスタンス統計機能は jheapprof コマンドを使用して出力します。jheapprof コマンドの実行形式と実行例については，「9.3.3　クラス別統計情報の出力」を参照してください。

この節の構成を次の表に示します。

表 9-4　この節の構成（インスタンス統計機能）

分類	タイトル	参照先
解説	インスタンス統計機能の概要	9.4.1
	インスタンス統計機能で出力するクラス別統計情報	9.4.2

注 「実装」，「設定」，「運用」および「注意事項」について，この機能固有の説明はありません。

9.4.1　インスタンス統計機能の概要

インスタンス統計機能は，アプリケーションのメモリリークの調査で使用します。

インスタンス統計機能では，classA のインスタンス→classA のメンバ変数（クラスは classB）→classB のインスタンス→…とインスタンスの参照関係を調べ，ほかのインスタンスへの参照を持たないインスタンスのサイズをそのインスタンスをメンバに持つクラスに再帰的に加算します。つまり，インスタンス統計機能では，各クラスのインスタンスが参照しているインスタンスの合計サイズが出力されます。

メモリリークの原因を調査する場合は，次のように，メモリリークを調査したいアプリケーションの処理の前後でインスタンス統計機能を実行し，1.と 3.のインスタンス数やインスタンスの合計サイズの差分を取り，その増加量を見てメモリリークの原因となっているクラスを特定します。

1. インスタンス統計機能を実行します。
2. メモリリークを調査したいアプリケーションの処理を実行します。
3. インスタンス統計機能を実行します。

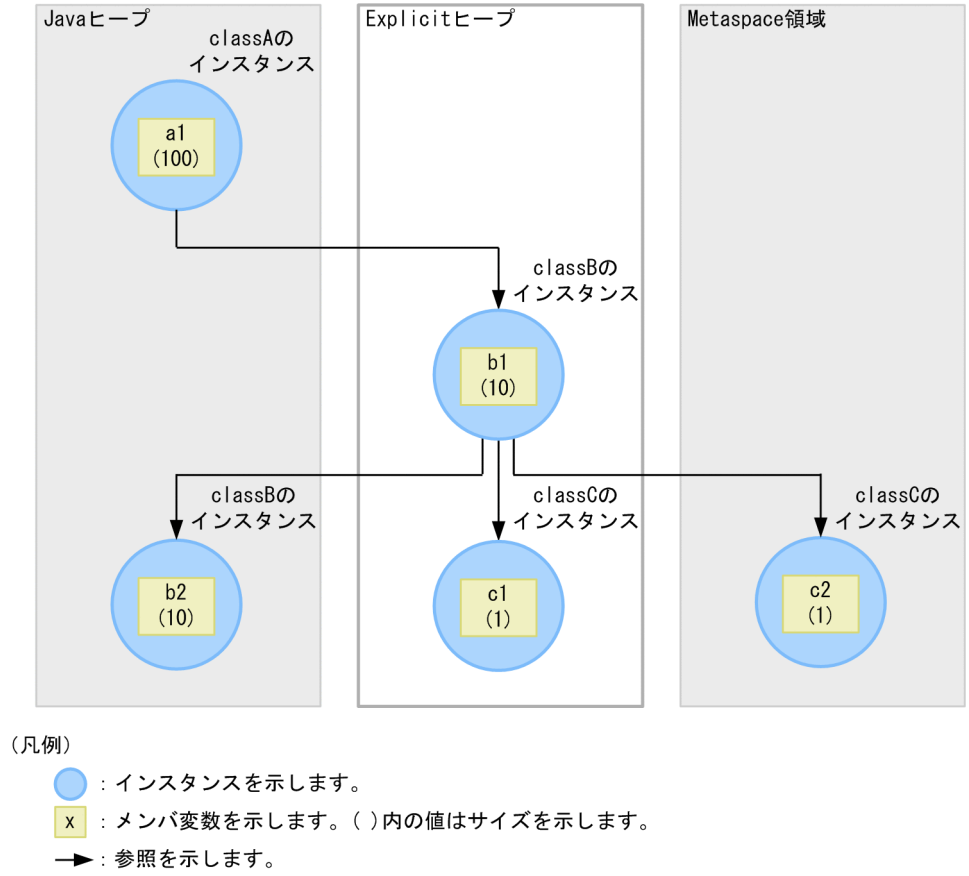
なお，インスタンス統計機能は，各クラスのインスタンスが参照しているすべてのインスタンスのサイズを再帰的に加算するため，各クラスのインスタンスだけのサイズ（参照しているインスタンスのサイズを含まないサイズ）を調べることはできません。

インスタンス統計機能では，明示管理ヒープ機能を使用している場合に，Explicit ヒープのインスタンスを統計対象とするかどうかによって，クラス別統計情報の出力結果が変わります。なお，Explicit ヒープ

のインスタンスを統計対象とする場合には，jheapprof コマンドに-explicit オプションを指定して実行します。

統計対象ごとの出力結果について，次の図に示すインスタンス構造を例に説明します。

図 9-1　インスタンス構造の例



Explicit ヒープを統計対象に含む場合は，インスタンス a1，b1，b2，c1，および c2 がクラス別統計情報の対象となります。Explicit ヒープを統計対象に含まない場合は，インスタンス a1，b2，c2 がクラス別統計情報の対象となります。

統計対象ごとのインスタンス数とインスタンスの合計サイズを次の表に示します。

表 9-5　統計対象ごとのインスタンス数とインスタンスの合計サイズ

jheapprof コマンドの引数	統計対象	クラス A		クラス B		クラス C	
		インスタンス数	合計サイズ※	インスタンス数	合計サイズ※	インスタンス数※	合計サイズ※
-explicit	• Java ヒープ • Explicit ヒープ	1	122	2	22	2	2
-noexplicit	• Java ヒープ	1	111	1	11	1	1

注※　合計サイズは，インスタンスの合計サイズを示します。単位はバイトです。

各クラスのインスタンスの合計サイズの算出式を次に示します。

- Explicit ヒープを統計対象に含む場合
  - クラス A の場合：a1+b1+b2+c1+c2
  - クラス B の場合：b1+b2+c1+c2
  - クラス C の場合：c1+c2
- Explicit ヒープを統計対象に含まない場合
  - クラス A の場合：a1+b2+c2
  - クラス B の場合：b2+c2
    （インスタンス b1 が対象外でも，その配下にあるインスタンス b2，c2 が対象であるため，参照関係にあるそれらのインスタンスのサイズがクラス B に加算される）
  - クラス C の場合：c2

インスタンス統計機能では，基点となるオブジェクトから次の順序に従って参照されるオブジェクトの参照関係を調べます。基点となるオブジェクトは，ほかの参照関係で調べられていないオブジェクトが該当します。1.，2.，3.は調査の優先順を示します。

1. Java ヒープ内のアドレスの低い順
2. 明示管理ヒープ内のアドレスの低い順

参照先のオブジェクトが調査済みの場合は，分岐点まで戻って参照関係を調べます。

また，参照先のオブジェクトがほかの参照関係の基点となるオブジェクトである場合は，参照先オブジェクトとして扱います。すべての基点となるオブジェクトがなくなるまで参照関係を調べます。

インスタンス統計機能の場合，インスタンス数には各クラスのインスタンス数が出力されます。インスタンスの合計サイズには，次の内容が出力されます。

- 基点となるオブジェクトのサイズは，該当するクラスに加算されます。参照先のオブジェクトのサイズは，該当するクラスに加算され，さらに基点となるオブジェクト，および該当するクラスまでの参照関係にあるすべてのオブジェクトの該当するクラスにも加算されます。

9.4.2　インスタンス統計機能で出力するクラス別統計情報

ここでは，インスタンス統計機能で出力するクラス別統計情報の出力形式，出力項目および出力例について説明します。

(1)　出力形式と出力項目

インスタンス統計機能で出力するクラス別統計情報の出力形式を次に示します。

- 出力形式

```
Java Heap Profile
-----------------------
_________________Size__Instances__Class_________________
   <total_size>       <Instance_count>   <class_name>
   <total_size>       <Instance_count>   <class_name>
...
```

- 出力項目

  出力形式で示した各項目について説明します。

  表 9-6　出力項目（インスタンス統計機能）

出力項目	意味
<total_size>	インスタンスの合計サイズがバイト単位で出力されます。
<Instance_count>	インスタンスの数が出力されます。
<class_name>	クラス名が出力されます。

(2)　出力例

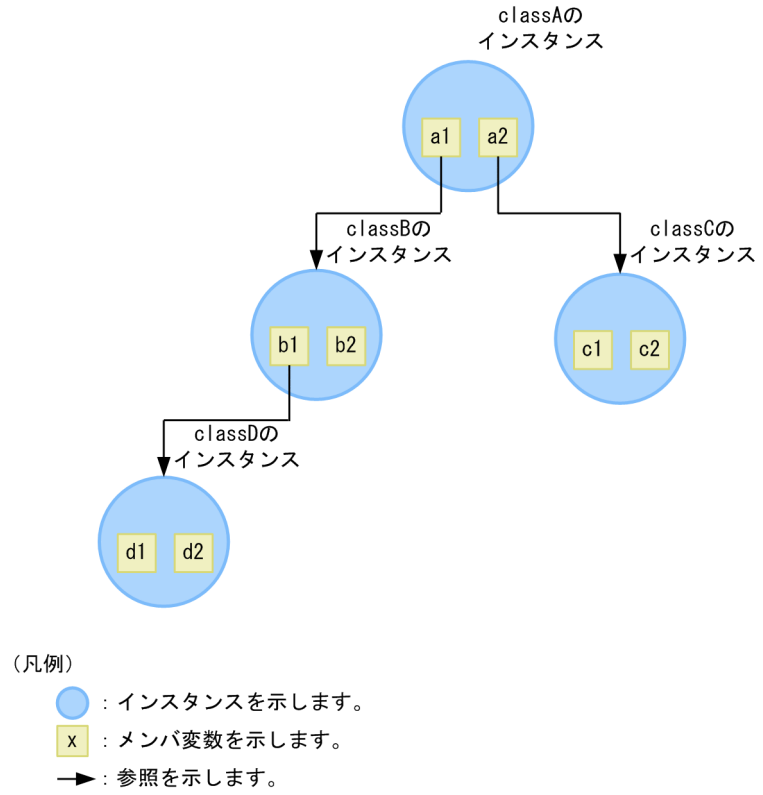
インスタンス統計機能で出力するクラス別統計情報の出力例を，次のソースを例にして説明します。

```
public class instance {
  public static void main(String args[]) {
    classA cls_a = new classA();
    try {
      Thread.sleep(20000);
    } catch (Exception e) {}
  }
}
class classA {
  classB a1;
  classC a2;
  classA() {
    a1 = new classB();
    a2 = new classC();
  }
}
class classB {
  classD b1;
  String b2;
  classB() {
    b1 = new classD();
    b2 = null;
  }
}
class classC {
  String c1, c2;
  classC() {
    c1 = null;
    c2 = null;
  }
```

```
}
class classD {
  String d1, d2;
  classD() {
    d1 = null;
    d2 = null;
  }
}
```

このソースの場合の，インスタンス構造を次の図に示します。

図 9-2　インスタンス構造（インスタンス統計機能）

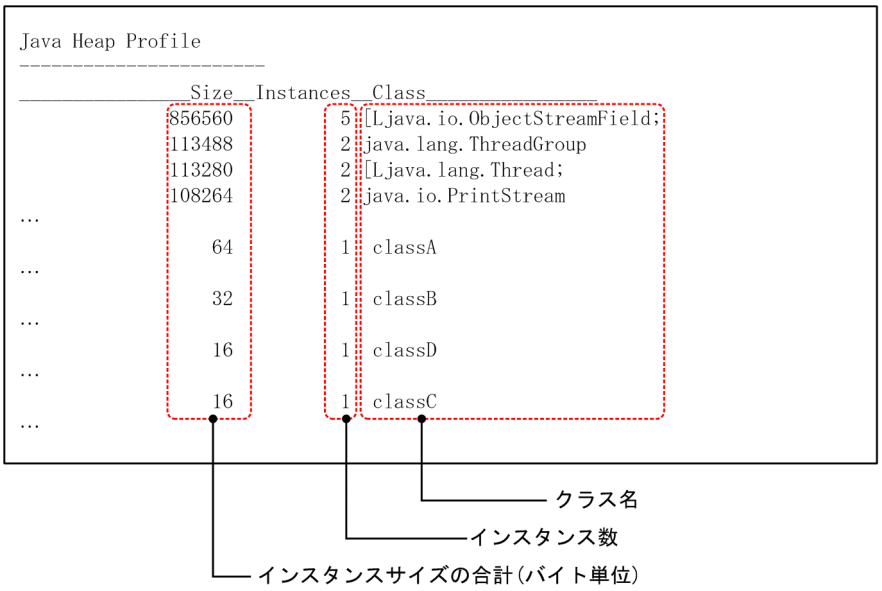


このようなインスタンス構造の場合，インスタンス統計機能では，次のように各クラスのサイズを加算します。

- classA のサイズ：a1+a2+ b1+b2+ c1+c2+ d1+d2
- classB のサイズ：b1+b2+ d1+d2
- classC のサイズ：c1+c2
- classD のサイズ：d1+d2

インスタンス統計機能の出力結果を次の図に示します。

図 9-3　出力結果（インスタンス統計機能）



9.5　STATIC メンバ統計機能

この節では，STATIC メンバ統計機能について説明します。

STATIC メンバ統計機能は，クラス別統計情報を出力する機能の一つです。クラスごとに，static メンバが持つインスタンスの合計サイズが出力できます。

STATIC メンバ統計機能は jheapprof コマンドを使用して出力します。jheapprof コマンドの実行形式と実行例については，「9.3.3　クラス別統計情報の出力」を参照してください。

この節の構成を次の表に示します。

表 9–7　この節の構成（STATIC メンバ統計機能）

分類	タイトル	参照先
解説	STATIC メンバ統計機能の概要	9.5.1
	STATIC メンバ統計機能で出力するクラス別統計情報	9.5.2

注　「実装」，「設定」，「運用」および「注意事項」について，この機能固有の説明はありません。

9.5.1　STATIC メンバ統計機能の概要

STATIC メンバ統計機能は，インスタンス統計機能と同様に，アプリケーションのメモリリークの調査で使用します。

インスタンス統計機能と異なる点は，インスタンス統計機能が最初に取り出したインスタンスの非 static フィールドから参照しているインスタンスのサイズを再帰的に加算するのに対し，STATIC メンバ統計機能は最初に取り出したインスタンスの static フィールド（クラスの static フィールド）から参照しているインスタンスのサイズを再帰的に加算するという点です。これによって，各クラスの static メンバが持つインスタンスの合計サイズを得ることができます。ただし，最初に取り出すインスタンス以外では，インスタンス統計機能および STATIC メンバ統計機能共に，インスタンスの非 static メンバを基に参照関係を調べます。

インスタンス統計機能と STATIC メンバ統計機能の違いについては，「9.5.2(2)　出力例」を参照してください。

メモリリークの原因を調査する場合は，次のように，メモリリークを調査したいアプリケーションの処理の前後で STATIC メンバ統計機能を実行し，1.と 3.のインスタンス数やインスタンスの合計サイズの差分を取り，その増加量を見てメモリリークの原因となっているクラスを特定します。

1. STATIC メンバ統計機能を実行します。
2. メモリリークを調査したいアプリケーションの処理を実行します。
3. STATIC メンバ統計機能を実行します。

なお，STATIC メンバ統計機能は，各クラスの static フィールドが参照しているインスタンスのサイズを再帰的に加算するため，各クラスのインスタンスだけのサイズ（参照しているインスタンスのサイズを含まないサイズ）の合計を調べることはできません。

STATIC メンバ統計機能では，基点となるオブジェクトから参照関係を調べます。基点となるオブジェクトは，JavaVM が持つすべてのクラスの STATIC メンバが参照し，ほかの参照関係で調べられていないオブジェクトが該当します。

また，参照先のオブジェクトが調査済みの場合は，分岐点まで戻って参照関係を調べます。すべての基点となるオブジェクトがなくなるまで参照関係を調べます。

STATIC メンバ統計機能の場合，インスタンス数とインスタンスの合計サイズには，次の内容が出力されます。

- 参照関係にあるすべてのオブジェクトのサイズとオブジェクト数を基点となるオブジェクトに合計します。この値を基点となるオブジェクトが参照している STATIC メンバを持つクラスに合計して統計値とします。

9.5.2 STATIC メンバ統計機能で出力するクラス別統計情報

ここでは，STATIC メンバ統計機能で出力するクラス別統計情報の出力形式，出力項目および出力例について説明します。

(1) 出力形式と出力項目

STATIC メンバ統計機能で出力するクラス別統計情報の出力形式を次に示します。

- 出力形式

```
Java Heap Dump Static Profile
----------------------
_________________Size__Instances__Class_________________
   <total_size>       <Instance_count>   <class_name>
   <total_size>       <Instance_count>   <class_name>
...
```

- 出力項目

  出力形式で示した各項目について説明します。

表 9-8 出力項目（STATIC メンバ統計機能）

出力項目	意味
<total_size>	インスタンスの合計サイズがバイト単位で出力されます。
<Instance_count>	インスタンスの数が出力されます。
<class_name>	クラス名が出力されます。

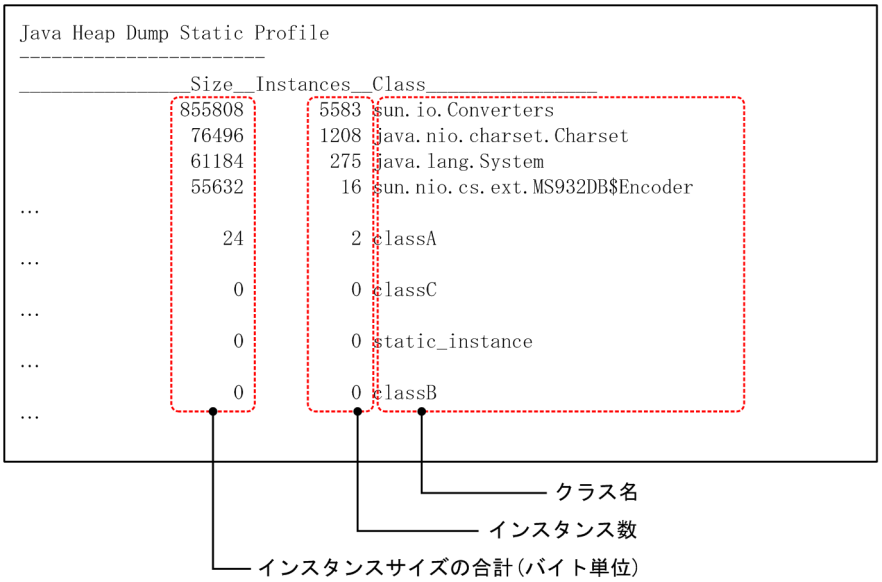
(2) 出力例

STATIC メンバ統計機能で出力するクラス別統計情報の出力例を，次のソースを例にして説明します。

```
public class static_instance {
  public static void main(String args[]) {
    classA cls_a;
    classB cls_b;
    classC cls_c;

    cls_a = new classA();
    cls_b = new classB();
    cls_c = new classC();
    cls_b.cls_c = cls_c;
    cls_a.cls_b = cls_b;

    try {
      Thread.sleep(20000);
    } catch (Exception e) {}
  }
}
class classA {
  static classB cls_b;
}
class classB {
  classC cls_c;
}
class classC {
}
```

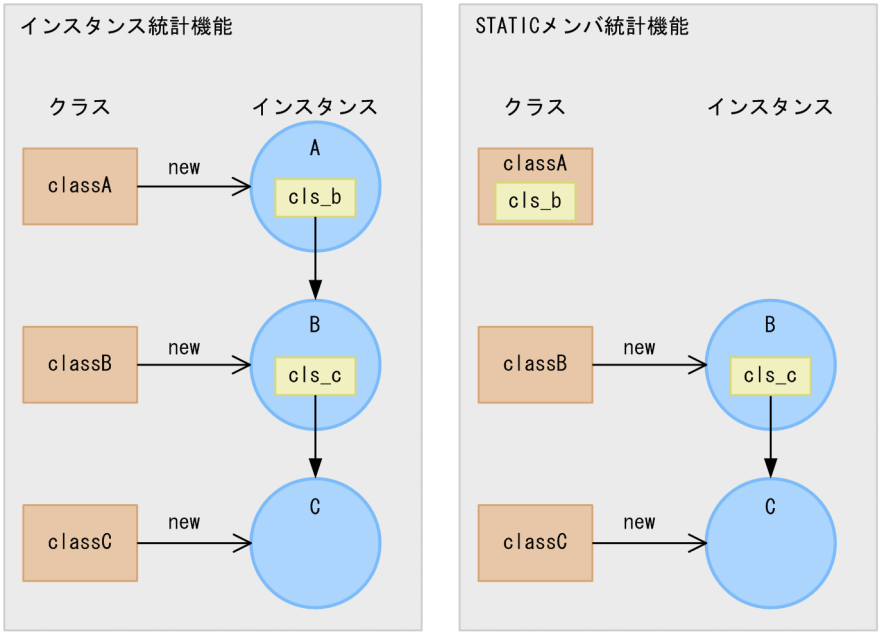
STATIC メンバ統計機能の出力結果を次の図に示します。

図 9-4　出力結果（STATIC メンバ統計機能）



また，上記のソースの場合，インスタンス統計機能と STATIC メンバ統計機能では，参照関係に違いがあります。インスタンス統計機能と STATIC メンバ統計機能の参照関係の相違を次の図に示します。

図 9-5　インスタンス統計機能と STATIC メンバ統計機能の参照関係の相違



（凡例）

x ：クラスを示します。
：インスタンスを示します。
x ：メンバ変数を示します。
：参照を示します。
：インスタンスの生成を示します。

それぞれ機能の参照関係は次のようになっています。

- インスタンス統計機能の参照関係

  インスタンス A のインスタンス変数 cls_b → インスタンス B のインスタンス変数 cls_c → インスタンス C

- STATIC メンバ統計機能の参照関係

  クラス A のクラス変数 cls_b → インスタンス B のインスタンス変数 cls_c → インスタンス C

9.6　参照関係情報出力機能

この節では，参照関係情報出力機能について説明します。

参照関係情報出力機能は，指定したクラスのインスタンスの参照関係が先頭から順番に出力できます。

この節の構成を次の表に示します。

表 9-9　この節の構成（参照関係情報出力機能）

分類	タイトル	参照先
解説	参照関係情報出力機能の概要	9.6.1
	参照関係情報出力機能で出力するクラス別統計情報	9.6.2
	static フィールドを基点とした参照関係情報出力機能で出力するクラス別統計情報	9.6.3
注意事項	static フィールドを基点とした参照関係情報出力時の注意事項	9.6.4

注　「実装」,「設定」，および「運用」について，この機能固有の説明はありません。

9.6.1　参照関係情報出力機能の概要

jheapprof コマンドの-class オプションに指定したクラスのインスタンスがどのクラスから参照されているかを，インスタンスの参照関係の先頭から順番に出力します。

指定したクラスのインスタンスが複数ある場合は，該当するすべてのインスタンスが出力されます。同じ名前のインスタンスが複数ある場合でも，インスタンス名のあとに次の情報が出力されるため，別のインスタンスかどうか識別できます。

- インスタンスのアドレス
- インスタンスが所属している領域名称

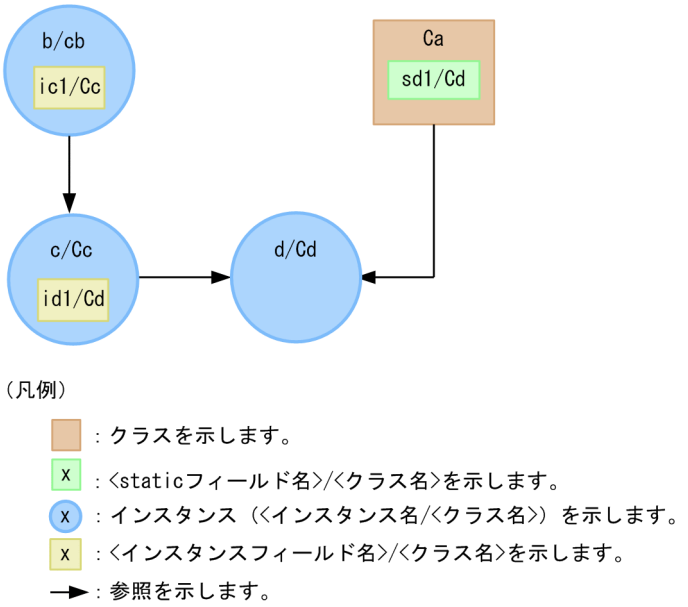
参照関係情報出力機能では，基点となるオブジェクトから次の順序に従って参照されるオブジェクトの参照関係を調べます。基点となるオブジェクトは，ほかの参照関係で調べられていないオブジェクトが該当します。1.，2.，3.は調査の優先順を示します。

1. Java ヒープ内のアドレスの低い順
2. 明示管理ヒープ内のアドレスの低い順

参照先のオブジェクトが-class オプションで指定したクラスの場合，参照関係情報には，基点となるオブジェクトから-class オプションで指定したクラスのオブジェクトまでの参照関係が出力されます。参照先のオブジェクトが調査済みの場合は，分岐点まで戻って参照関係を調べます。参照先のオブジェクトがほかの参照関係の基点となるオブジェクトである場合は，参照先オブジェクトとして扱います。すべての基点となるオブジェクトがなくなるまで参照関係を調べます。

また，jheapprof コマンドで-staticroot オプションを指定すると，static フィールドを基点とした参照関係情報出力機能が有効になります。この機能は，参照関係情報出力機能を前提としています。static フィールドを基点とした参照関係情報出力機能では，参照関係情報出力機能が出力する参照関係情報に，static フィールドを基点とした参照関係情報を追加して出力します。この出力情報は，static フィールドを基点とした参照関係によるメモリリークの原因を究明するために使用します。

メモリリーク状態の参照関係の例を次に示します。



図の参照関係のインスタンス「d/Cd」に static フィールドを基点とした参照関係情報出力機能を実行すると，次の参照関係情報が出力されます。

```
Reference of class Cd
---------------------
Cb(0x088065f0)[Tenured]
  Cc(0x08806668)[Tenured]
    Cd(0x088066e0)[Tenured]
------

Reference of class Cd from static field
---------------------------------------
static field Ca.sd1
  Cd(0x088066e0)[Tenured]
------
```

参照関係情報出力機能で出力する参照関係情報

staticフィールドを基点とした参照関係情報出力機能で出力する参照関係情報

この情報を基に，リーク対策として，次のフィールドの参照をクリアします。このことで，「d/Cd」は GC で回収されるため，メモリリークを解消することができます。

- インスタンス「c/Cc」のインスタンスフィールド「id1」
- クラス「Ca」の static フィールド「sd1」

参照関係情報出力機能の参照関係情報については，「9.6.2　参照関係情報出力機能で出力するクラス別統計情報」を，static フィールドを基点とした参照関係情報については，「9.6.3　static フィールドを基点とした参照関係情報出力機能で出力するクラス別統計情報」を参照してください。

9.6.2　参照関係情報出力機能で出力するクラス別統計情報

ここでは，参照関係情報出力機能で出力するクラス別統計情報の出力形式，出力項目および出力例について説明します。

(1)　出力形式と出力項目

参照関係情報出力機能で出力するクラス別統計情報の出力形式を次に示します。

- 出力形式

```
Reference of class <オプション指定クラス名>
-----------------------------------------※
<クラス名>(<アドレス>)[<領域名称>]
  <クラス名>(<アドレス>)[<領域名称>]
    <オプション指定クラス名>(<アドレス>)[<領域名称>]
------
<クラス名>(<アドレス>)[<領域名称>]
  java.lang.ref.Finalizer<<繰り返し数> times>
    <クラス名>(<アドレス>)[<領域名称>]
      <クラス名>(<アドレス>)[<領域名称>]
        <オプション指定クラス名>(<アドレス>)[<領域名称>]
------
...
```

注※

<オプション指定クラス名>の文字数に 19 を加算した数の「-（ハイフン）」が出力されます。

- 出力項目

出力形式で示した各項目について説明します。

表 9-10　出力項目（参照関係情報出力機能）

出力項目	意味
<クラス名>	jheapprof コマンドの-class オプションに指定したクラスのインスタンスが参照するクラスの名称が出力されます。
<アドレス>	インスタンスのアドレスが出力されます。
<領域名称>	インスタンスが所属する領域が出力されます。 • Eden：Eden 領域を示します。 • Survivor：Survivor 領域を示します。 • Tenured：Tenured 領域を示します。 • EM(eid=<id>)：Explicit メモリブロックを示します。
<オプション指定クラス名>	jheapprof コマンドの-class オプションに指定したクラス名が出力されます。
java.lang.ref.Finalizer	finalize()メソッドを持つクラスで生成する java.lang.ref.Finalizer のオブジェクトを一つにまとめて出力することを示します。
<繰り返し数>	Finalizer インスタンスの参照が連続した回数が出力されます。

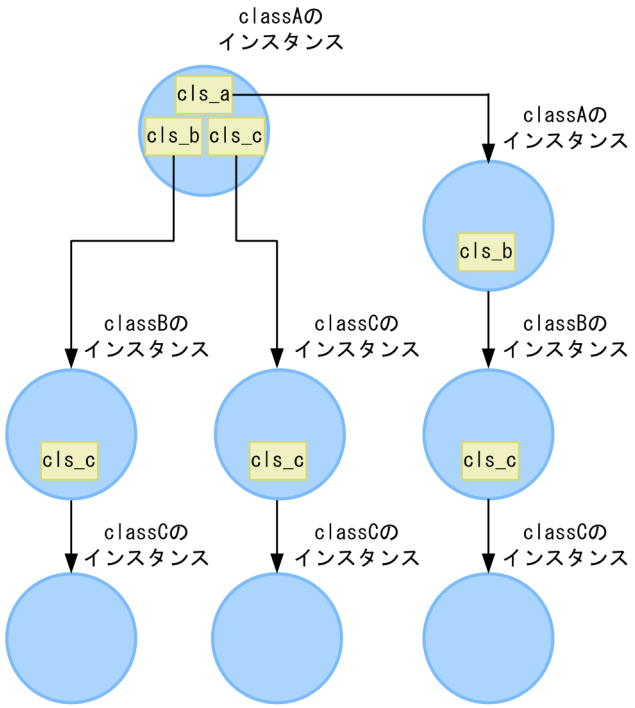
(2) 出力例

参照関係情報出力機能で出力するクラス別統計情報の出力例を，次のソースを例にして説明します。

```
public class instance2 {
  public static void main(String args[]) {
    classA cls_a1 = new classA();    classA cls_a2 = new classA();
    classB cls_b1 = new classB();    classB cls_b2 = new classB();
    classC cls_c1 = new classC();    classC cls_c2 = new classC();
    classC cls_c3 = new classC();    classC cls_c4 = new classC();
    cls_a1.cls_a = cls_a2;    cls_a1.cls_b = cls_b1;
    cls_a1.cls_c = cls_c1;    cls_a2.cls_b = cls_b2;
    cls_b1.cls_c = cls_c2;    cls_b2.cls_c = cls_c3;
    cls_c1.cls_c = cls_c4;
    try {
      Thread.sleep(20000);
    } catch (Exception e) {}
  }
}
class classA {
  classA cls_a;
  classB cls_b;
  classC cls_c;

  classA() {
    classB cls_b;
  }
}
class classB {
  classC cls_c;
}
class classC {
  classC cls_c;
}
```

インスタンス構造を次の図に示します。

図 9-6　インスタンス構造（参照関係情報出力機能）



（凡例）

：インスタンスを示します。

x ：メンバ変数を示します。

→：参照を示します。

参照関係情報出力機能の出力結果を次の図に示します。この場合，jheapprof コマンドに引数「-class <クラス名>」を指定して実行します。

図 9-7　出力結果（参照関係情報出力機能）

```
Reference of class classC
-------------------------
classA(0x10766840)[Eden]
  classB(0x10766998)[Eden]
    classC(0x10766a88)[Survivor]
------                                  } (1)classA→classB→classCの参照を表す
classA(0x10766840)[Eden]
  classC(0x10766920)[Survivor]
------                                  } (2)classA→classC→classCの参照関係のうち
                                            先頭のclassA→classCの部分を表す
classA(0x10766840)[Eden]
  classC(0x10766920)[Survivor]
    classC(0x10766ab8)[EM(eid=1)]
------                                  } (3)classA→classC→classCの参照を表す
classA(0x10766840)[Eden]
  classA(0x10766858)[Tenured]
    classB(0x10766968)[Eden]
      classC(0x10766a28)[Survivor]
------                                  } (4)classA→classA→classB→classCの参照を表す
```

注1　(1)～(4)の出力順はインスタンスの参照関係を調べるときの状態で処理順が変化する場合があります。

注2　(xxxxxxxxxx)のxxxxxxxxxxは，インスタンスのメモリ上のアドレスを示します。
また，[yy...yy]のyy...yyは，インスタンスが所属する領域を示します。

classA のアドレスは，すべて同じアドレス（0x10766840）になっています。したがって，classA はすべて同じインスタンスであることがわかります。一方，(1)と(4)の classB は，アドレスが異なっているので，別のインスタンスであることがわかります。

なお，GC の発生によってインスタンスのメモリ上の配置が変化します。そのため，アドレスおよび領域名称は，出力するたびに変化する場合があります。

9.6.3 static フィールドを基点とした参照関係情報出力機能で出力するクラス別統計情報

ここでは，static フィールドを基点とした参照関係情報出力機能で出力するクラス別統計情報の出力形式，出力項目および出力例について説明します。

(1) 出力形式と出力項目

- 出力形式

  static フィールドを基点とした参照関係情報出力機能で出力するクラス別統計情報の出力形式を次に示します。

```
Reference of class <オプション指定クラス名> from static field
------------------------------------------------------------※1
static field <staticフィールド宣言クラス名>※2.<staticフィールド名>※2
<クラス名>(<アドレス>)[<領域名称>]
  <クラス名>(<アドレス>)[<領域名称>]
    <オプション指定クラス名>(<アドレス>)[<領域名称>]
------
...
```

  注※1

  <オプション指定クラス名>の文字数に 37 を加算した数の「-（ハイフン）」が出力されます。

  注※2

  参照関係の基点を示します。

- 出力項目

  出力形式で示した各項目について説明します。

表 9-11 出力項目（static フィールドを基点とした参照関係情報出力機能）

出力項目	意味
<static フィールド宣言クラス名>	基点となる static フィールドを宣言しているクラス名が出力されます。
<static フィールド名>	基点となる static フィールドの名称が出力されます。
<クラス名>	jheapprof コマンドの-class オプションに指定したクラスのインスタンスを参照する，インスタンスのクラスの名称が出力されます。
<アドレス>	インスタンスのアドレスが出力されます。

出力項目	意味
<領域名称>	インスタンスが所属する領域が出力されます。 • Eden：Eden 領域を示します。 • Survivor：Survivor 領域を示します。 • Tenured：Tenured 領域を示します。 • EM(eid=<id>)：Explicit メモリブロックを示します。
<オプション指定クラス名>	jheapprof コマンドの-class オプションに指定したクラス名が出力されます。

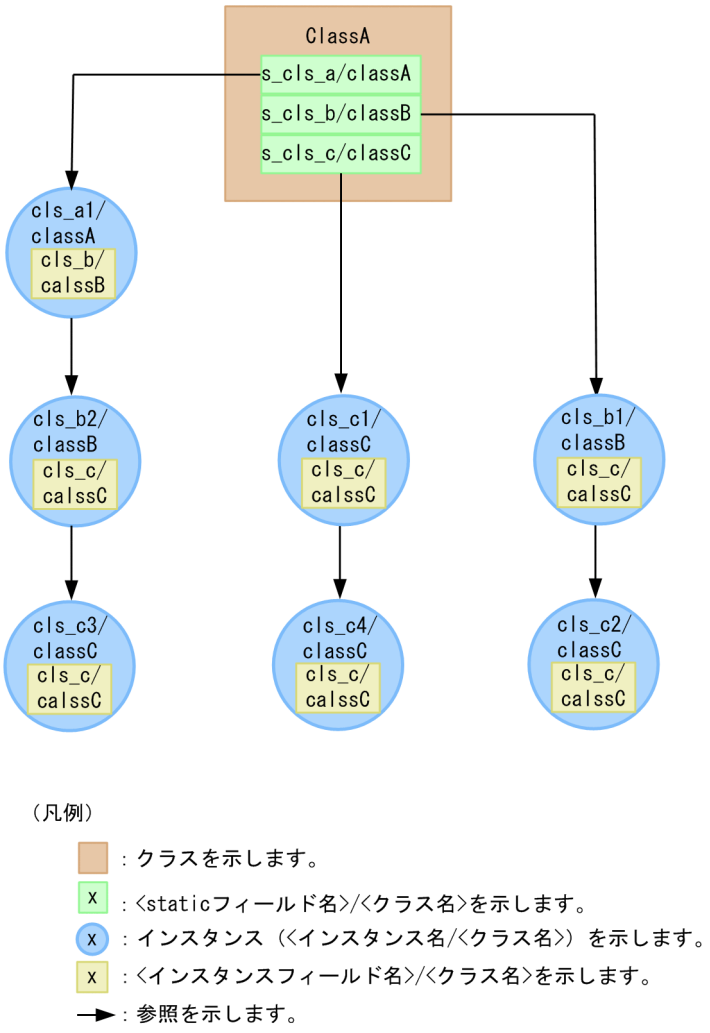
(2) 出力例

static フィールドを基点とした参照関係情報出力機能で出力するクラス別統計情報の出力例を，次のソースを例にして説明します。

```
import JP.co.Hitachi.soft.jvm.MemoryArea.*;
public class static_reference {
  public static void main(String args[]) {
   try {
      classA cls_a1 = new classA();
      classB cls_b1 = new classB();
      classB cls_b2 = new classB();
      classC cls_c1 = new classC();
      classC cls_c2 = new classC();
      classC cls_c3 = new classC();
      BasicExplicitMemory emem = new BasicExplicitMemory();
      classC cls_c4 = (classC)emem.newInstance(classC.class);
      cls_a1.s_cls_a = cls_a1;
      cls_a1.s_cls_b = cls_b1;
      cls_a1.s_cls_c = cls_c1;
      cls_a1.cls_b = cls_b2;
      cls_b1.cls_c = cls_c2;
      cls_b2.cls_c = cls_c3;
      cls_c1.cls_c = cls_c4;
      Thread.sleep(20000);
    } catch (Exception e) {e.printStackTrace();}
  }
}
class classA {
  static classA s_cls_a;
  static classB s_cls_b;
  static classC s_cls_c;
  classB cls_b;
}
class classB {
  classC cls_c;
}
class classC {
  classC cls_c;
  public classC(){
  }
}
```

インスタンス構造を次の図に示します。

図 9-8　インスタンス構造（static フィールドを基点とした参照関係情報出力機能）



static フィールドを基点とした参照関係情報出力機能の出力結果を次の図に示します。この場合，jheapprof コマンドに引数「-class <クラス名> -staticroot」を指定して実行します。

図 9-9　出力結果（static フィールドを基点とした参照関係情報出力機能）

```
Reference of class classC from static field
-------------------------------------------
static field classA.s_cls_a                  ┐
  classA(0x10511d08)[Tenured]                │
    classB(0x10511d78)[Eden]                 ├(1)classAのstaticフィールドs_cls_a
      classC(0x10511df0)[Survivor]           │   →classA→classB→classCの参照を表す
------                                       ┘
static field classA.s_cls_b                  ┐
  classB(0x10511d68)[Tenured]                ├(2)classAのstaticフィールドs_cls_b
    classC(0x10511de0)[Tenured]              │   →classB→classCの参照を表す
------                                       ┘
static field classA.s_cls_c                  ┐
  classC(0x10511dd0)[Tenured]                ├(3)classAのstaticフィールドs_cls_c
    classC(0x03bc0000)[EM(eid=1)]            │   →classC→classCの参照を表す
------                                       ┘
static field classA.s_cls_c                  ┐
  classC(0x10511dd0)[Tenured]                ├(4)classAのstaticフィールドs_cls_c
------                                       │   →classC→classCの参照関係のうち，先頭のclassAの
                                             │   staticフィールドs_cls_c→classCの部分を表す
                                             ┘
```

注1　(1)～(4)の出力順はインスタンスの参照関係を調べるときの状態で処理順が変化する場合があります。
注2　(xxxxxxxxxx)のxxxxxxxxxxは，インスタンスのメモリ上のアドレスを示します。
　　また，[yy...yy]のyy...yyは，インスタンスが所属する領域を示します。

9.6.4　static フィールドを基点とした参照関係情報出力時の注意事項

jheapprof コマンドの実行時間は，static フィールドを基点とした参照関係情報出力機能を有効にすると，static フィールドを基点とした参照関係情報出力機能が無効な場合に比べて，参照関係情報出力機能の実行時間分長くなります。

9.7　統計前の GC 選択機能

この節では，統計前の GC 選択機能について説明します。

統計前の GC 選択機能を使用すると，クラス別統計情報の出力前に実行する処理を選択できます。

この節の構成を次の表に示します。

表 9-12　この節の構成（統計前の GC 選択機能）

分類	タイトル	参照先
解説	統計前の GC 選択機能の概要	9.7.1
	GC の選択の指針	9.7.2

注　「実装」,「設定」,「運用」，および「注意事項」について，この機能固有の説明はありません。

9.7.1　統計前の GC 選択機能の概要

クラス別統計機能を実行すると，拡張スレッドダンプへクラス別統計情報が出力できます。統計前の GC 選択機能では，クラス別統計情報を出力する前に実施する処理を選択できます。調査目的に合わせて，実施する処理を選択することで，Java オブジェクトのさまざまな変化の様子をクラス別統計情報に取得できます。

統計前の GC 選択機能を使用する場合，jheapprof コマンドの引数で実行する処理を指定します。クラス別統計機能実行前に実施できる処理と jheapprof コマンドの引数を次の表に示します。

表 9-13　クラス別統計機能実行前に実施できる処理と jheapprof コマンドの引数

処理の種類	処理内容	jheapprof コマンドの引数
FullGC の実行	Tenured 領域も含む，JavaVM 固有領域全体を対象に，使用済みのオブジェクトを回収します。	-fullgc
CopyGC の実行	Eden 領域および Survivor 領域だけを対象に，使用済みのオブジェクトを回収します。	-copygc
GC を実行しない	使用済みのオブジェクトがあっても回収しません。	-nogc

なお，JavaVM 起動オプションで-XX:+HitachiVerboseGC および-XX:+HitachiVerboseGCPrintCause を指定している場合にクラス別統計機能を実行すると，拡張 verbosegc 情報に次の情報が出力されます。

- GC 種別
- 拡張スレッドダンプに GC が発生した要因

これらの情報は，jheapprof コマンドに指定する引数によって，出力される情報が異なります。jheapprof コマンドの引数と出力情報の関係を次の表に示します。

表 9–14　jheapprof コマンドの引数と出力情報の関係

jheapprof コマンドの引数	GC 種別	GC が発生した要因
-fullgc	FullGC	JHeapProf Command
-copygc	GC	JHeapProf Command

9.7.2　GC の選択の指針

統計前の GC 選択機能で，どの処理を指定するかは，出力されるクラス別統計情報の調査目的によって異なります。

ここでは，調査目的に合わせて，処理を選択する指針について説明します。

統計前の GC 選択機能で選択できる処理は，jheapprof コマンドの引数で指定します。処理は，どのオブジェクトを調査対象とするか，クラス別統計情報をどのような調査で使用するかによって選択します。

処理の選択の指針を次の表に示します。

表 9–15　処理の選択の指針

処理 （jheapprof コマンドの引数）	調査対象	クラス別統計情報の調査方法の例
FullGC の実行 （-fullgc）	FullGC によるオブジェクトの変化	FullGC によって，使用済みのオブジェクトが回収されるため，Java ヒープのメモリリークの原因となるオブジェクトを特定できます。
CopyGC の実行 （-copygc）	CopyGC によるオブジェクトの変化	CopyGC によって，Tenured 領域へ移動する寿命の長いオブジェクトを特定できます。この情報から，FullGC の発生頻度が増大する原因となるオブジェクトを特定できます。
GC は実行しない （-nogc）※	GC の実行で回収されてしまう短寿命オブジェクト	使用済みのオブジェクトを含むすべてのオブジェクトの情報が出力されます。CopyGC が多発している場合などに，その原因となるメモリ占有率の高いオブジェクト（超大オブジェクト）を特定できます。

注※

jheapprof コマンドの引数に-nogc を指定した場合，すべての寿命の短いオブジェクトの情報が出力されるため，ログの出力量が増加します。

9.8 Tenured 領域内不要オブジェクト統計機能

この節では，Tenured 領域内不要オブジェクト統計機能について説明します。

Tenured 領域内不要オブジェクト統計機能は，クラス別統計情報を出力する機能の一つです。Tenured 領域内不要オブジェクト統計機能を使用すると，Tenured 領域内で不要となったオブジェクトを特定できます。

Tenured 領域内不要オブジェクト統計機能を使用する場合は，jheapprof コマンドの引数に-garbage を指定します。jheapprof コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「jheapprof（クラス別統計情報付き拡張スレッドダンプの出力)」を参照してください。

この節の構成を次の表に示します。

表 9-16　この節の構成（Tenured 領域内不要オブジェクト統計機能）

分類	タイトル	参照先
解説	Tenured 領域内不要オブジェクト統計機能の概要	9.8.1
	Tenured 領域内不要オブジェクト統計機能で出力するクラス別統計情報	9.8.2
注意事項	Tenured 領域内不要オブジェクト統計機能の実行に関する注意事項	9.8.3

注　「実装」,「設定」および「運用」について，この機能固有の説明はありません。

9.8.1 Tenured 領域内不要オブジェクト統計機能の概要

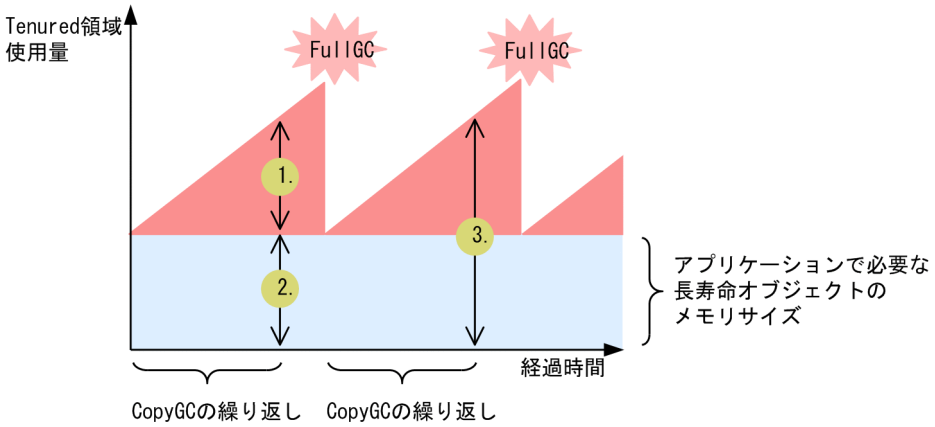
Tenured 領域内不要オブジェクト統計機能では，Tenured 領域内に蓄積された不要となったオブジェクトだけを特定して，スレッドダンプファイルに出力します。Tenured 領域内不要オブジェクト統計機能の仕組みについて説明します。

(1) 不要なオブジェクトのサイズの出力

CopyGC の繰り返しによって，長寿命オブジェクトが Tenured 領域に蓄積します。蓄積した長寿命オブジェクトのうち，時間が経過して用途を失ったオブジェクトは，不要なオブジェクトとなって Tenured 領域内に残ります。その後，メモリがいっぱいになったタイミングで FullGC が発生します。CopyGC の発生から FullGC の発生までの Tenured 領域の使用量は，Tenured 領域内不要オブジェクト統計機能，およびインスタンス統計機能で確認できます。

Tenured 領域内不要オブジェクト統計機能，およびインスタンス統計機能を使って特定できる内容を次の図に示します。

図 9-10　Tenured 領域内不要オブジェクト統計機能，およびインスタンス統計機能を使って特定できる内容



統計前 GC を実施しないでインスタンス統計機能を実行した場合，図 9-10 の 3.のサイズが出力されます。このサイズは，図 9-10 の 1.に該当する Tenured 領域内で不要となったオブジェクトのサイズ，および図 9-10 の 2.に該当する Tenured 領域内で使用中のオブジェクトを含んだ Tenured 領域内のメモリ使用状況になります。

一方，Tenured 領域内不要オブジェクト統計機能を実行した場合は，図 9-10 の 2.の使用中のオブジェクトを除いた Tenured 領域内のメモリ使用状況（図 9-10 の 1.に該当）を出力できます。Tenured 領域内不要オブジェクト統計機能を使うことで，Tenured 領域の増加要因となる不要となったオブジェクトを特定できるため，FullGC を抑止できます。

(2)　不要なオブジェクトの参照関係の確認

Tenured 領域内不要オブジェクト統計機能では，基点となるオブジェクトは Tenured 領域内のアドレスの低い順に検索されます。検索されたオブジェクトの中でも，ほかの参照関係で調べられていないオブジェクトが基点となるオブジェクトになります。

参照先のオブジェクトが調査済みの場合は，分岐点まで戻って参照関係を調べます。また，参照先のオブジェクトがほかの参照関係の基点となるオブジェクトである場合は，参照先オブジェクトとして扱います。すべての基点となるオブジェクトがなくなるまで参照関係を調べます。

Tenured 領域内不要オブジェクト統計機能の場合，インスタンス数，およびインスタンスサイズの合計が出力されます。インスタンス数は該当するクラスを加算します。インスタンスの合計サイズには次の内容が出力されます。

- 基点となるオブジェクトのサイズは，該当するクラスに加算されます。参照先のオブジェクトのサイズは，該当するクラスに加算され，さらに基点となるオブジェクト，および該当するクラスまでの参照関係にあるすべてのオブジェクトの該当するクラスにも加算されます。

なお，Tenured 領域内不要オブジェクト統計機能を実行した場合，インスタンス統計機能，STATIC メンバ統計機能，および統計前の GC 選択機能は無効となります。

Tenured 領域内の不要オブジェクトによる参照関係の例を示します。

図 9-11　Tenured 領域内の不要なオブジェクトによる参照関係の例

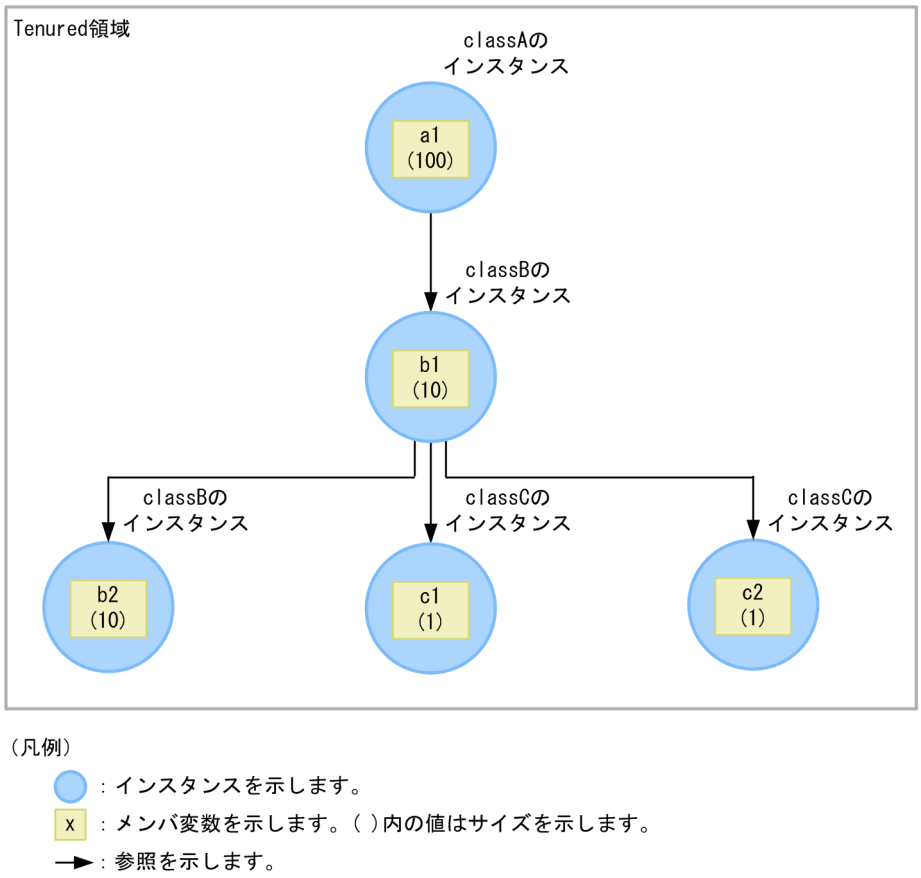


図 9-11 の参照関係について説明します。

インスタンス数

- classA：a1 が存在するため 1
- classB：b1，b2 の二つが存在するため 2
- classC：c1，c2 の二つが存在するため 2

インスタンスサイズの合計

- classA：classA のインスタンス（a1）と，その参照先のインスタンス（b1，b2，c1，c2）のサイズを合計した 122
- classB：classB のインスタンス（b1，b2）と，その参照先のインスタンス（c1，c2）のサイズを合計した 22
- classC：classC のインスタンス（c1，c2）を合計した 2

Tenured 領域内不要オブジェクト統計機能を実行して，図 9-11 の参照関係の情報を出力した場合の出力例を次に示します。

```
Garbage Profile
---------------
_________________Size__Instances__Class_________________
                  122           1 A
```

```
                22          2 B
                 2          2 C
```

9.8.2 Tenured 領域内不要オブジェクト統計機能で出力するクラス別統計情報

ここでは，Tenured 領域内不要オブジェクト統計機能で出力するクラス別統計情報の出力形式，出力項目および出力例について説明します。

(1) 出力形式と出力項目

Tenured 領域内不要オブジェクト統計機能で出力するクラス別統計情報の出力形式を次に示します。

- 出力形式

```
Garbage Profile
-----------------------
_________________Size__Instances__Class_________________
             〈サイズ〉       〈個数〉          〈クラス名〉
             〈サイズ〉       〈個数〉          〈クラス名〉...
```

- 出力項目

  出力形式で示した各項目について説明します。

  表 9-17 出力項目（Tenured 領域内不要オブジェクト統計機能）

出力項目	意味
<サイズ>	インスタンスの合計サイズがバイト単位で出力されます。
<個数>	インスタンスの数が出力されます。
<クラス名>	クラス名が出力されます。

(2) 出力例

Tenured 領域内不要オブジェクト統計機能で出力するクラス別統計情報の出力例を示します。

```
Garbage Profile
---------------
_________________Size__Instances__Class_________________
             35234568      10648 java.util.HashMap
              5678900      10668 [Ljava.util.HashMap$Entry;
              4456788       7436 java.util.HashMap$Entry
              4321000        200 java.util.WeakHashMap
              1234568        190 [Ljava.util.WeakHashMap$Entry
               454400          4 java.util.WeakHashMap$Entry
                    0          0 java.lang.Class
...
```

9.8.3　Tenured 領域内不要オブジェクト統計機能の実行に関する注意事項

Tenured 領域内不要オブジェクト統計機能の注意事項を次に示します。

(1)　Tenured 領域内不要オブジェクト統計機能を FullGC 直後に実行した場合の注意事項

Tenured 領域内不要オブジェクト統計機能を FullGC 直後に実行すると，統計対象の Tenured 領域内の不要なオブジェクトが回収されている状態で統計処理を実行します。そのため，クラス別統計情報中のインスタンスサイズの合計，およびインスタンス数が小さくなり，不要なオブジェクトが効果的に特定されません。不要なオブジェクトを効果的に特定するためには Tenured 領域内不要オブジェクト統計機能を実行してください。なお，Tenured 領域内不要オブジェクト統計機能の実行について，FullGC の発生のタイミングが判明している場合と，判明していない場合に分けて説明します。

(a)　FullGC が発生するタイミングが判明している場合

Tenured 領域内不要オブジェクト統計機能を FullGC の直前に実行すると，統計対象である Tenured 領域内の不要オブジェクトの数が多い状態で統計処理が実行されます。そのため，クラス別統計情報中のインスタンスサイズの合計，およびインスタンス数が大きくなり，不要なオブジェクトを効果的に特定できます。

(b)　FullGC が発生するタイミングが判明していない場合

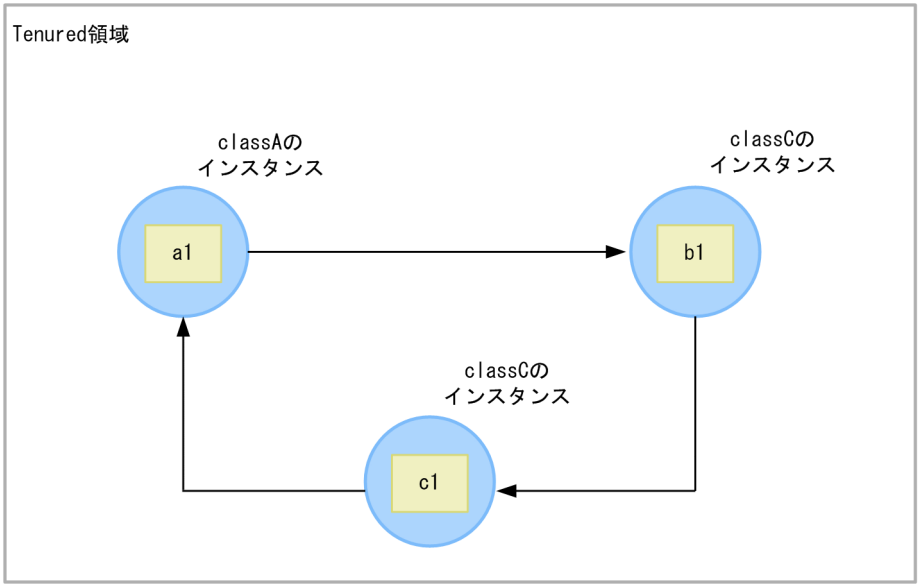
クラス別統計情報中のインスタンスサイズの合計，およびインスタンス数をできるだけ大きくし，不要なオブジェクトを効果的に特定する方法を示します。

1. FullGC の発生のタイミングを知るために JavaVM 起動時に拡張 verbosegc 情報を出力するオプション-XX:+HitachiVerboseGC を設定します。オプションを指定することで GC の情報を取得できます。
2. JavaVM に対して Tenured 領域内不要オブジェクト統計機能を一定の間隔で実行します。それによって，GC の情報と，複数のクラス別統計情報を取得できます。
3. 拡張 verbosegc 情報から FullGC の日時を取得できるため，FullGC により近いクラス別統計情報を選択します。FullGC に近いクラス別統計情報は，統計対象である Tenured 領域内の不要なオブジェクトの数が多い状態で統計処理された情報です。

(2)　統計結果に関する注意事項

図 9-12 のような参照関係の場合を例に統計結果に関する注意事項を説明します。

図 9-12　参照関係の例（統計結果に関する注意事項）



（凡例）

：インスタンスを示します。

x ：メンバ変数を示します。（ ）内の値はサイズを示します。

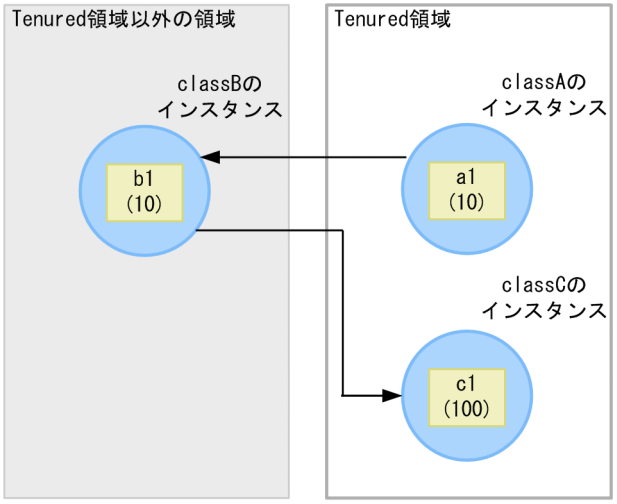
→：参照を示します。

この図で，a1 が最も下位のアドレスとした場合，a1 を基点となるオブジェクトとして，a1→b1→c1 という参照関係で統計処理を実行します。このとき，b1 または c1 を基点となるオブジェクトとしている場合，Tenured 領域内不要オブジェクト統計機能，および Tenured 増加要因の基点オブジェクトリスト機能の統計結果には，意図したとおりの結果が出力できません。

(3)　統計対象とならないオブジェクトに関する注意事項

図 9-13 のように Tenured 領域のオブジェクト（a1 および c1）が，Tenured 領域以外のオブジェクト（b1）と参照関係がある場合を例に，統計対象とならないオブジェクトに関する注意事項を説明します。

図 9-13　参照関係の例（統計対象とならないオブジェクト）



（凡例）

：インスタンスを示します。

x　：メンバ変数を示します。（ ）内の値はサイズを示します。

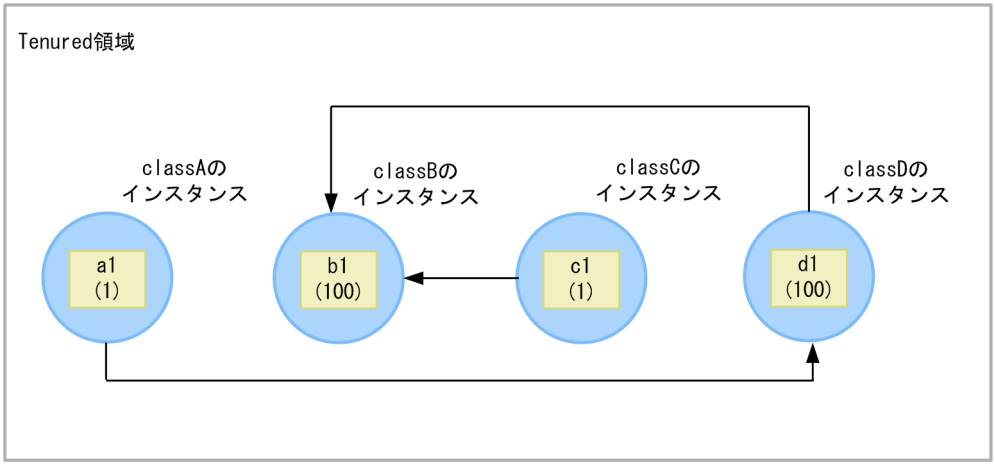
→：参照を示します。

この図の場合，Tenured 領域以外のオブジェクト（b1）は統計対象とはなりません。しかし，クラス B のインスタンスサイズの合計には，参照先の Tenured 領域のオブジェクト（c1）のサイズが加算されます。

(4)　複数のオブジェクトから参照関係がある場合の注意事項

図 9-14 のように複数の参照元（c1 および d1）から一つのオブジェクト（b1）を参照している参照関係がある場合を例に，複数のオブジェクトから参照関係がある場合の注意事項を説明します。

図 9-14　参照関係の例（複数のオブジェクトから参照関係がある場合）



（凡例）

：インスタンスを示します。

x　：メンバ変数を示します。（ ）内の値はサイズを示します。

→：参照を示します。

この図の場合，Tenured 領域内不要オブジェクト統計機能を実行すると，その参照元が属している参照関係の基点となるオブジェクト（c1 および a1）のうち，アドレスが最も低位である基点となるオブジェクトから統計処理が実行されます。そのため，アドレスが最も低位のオブジェクトが a1 で，基点となるオブジェクトが c1 である場合，Tenured 領域内不要オブジェクト統計機能，および Tenured 増加要因の基点オブジェクトリスト機能の統計結果には，意図したとおりの結果が出力できません。

(5) 統計値の増加に関する注意事項

オプション-XX:+HitachiAutoExplicitMemory を指定した Java プロセスに対して，Tenured 領域内不要オブジェクト統計機能，および Tenured 増加要因の基点オブジェクトリスト出力機能を実行すると，次に示すような現象が発生する場合があります。

- Tenured 領域内不要オブジェクト統計機能で出力されるクラス別統計情報のうち，float 型の配列型（クラス名に[F と出力される情報）のインスタンスサイズの合計，およびインスタンス数の統計値が，本来の統計値よりも大きくなります。クラス別統計情報の出力例を次に示します。

```
Garbage Profile
---------------
_________________Size__Instances__Class_________________
             43861400     473859 [F
                    0          0 java.util.Collections$EmptyMap
                    0          0 sun.security.util.Debug
                    0          0 java.nio.ByteOrder
```

- Tenured 増加要因の基点オブジェクトリスト出力機能で出力される Tenured 増加要因の基点オブジェクトリストに，float 型の配列型（クラス名に[F と出力される情報）が出力されることでインスタンスサイズの合計の統計値が，本来の統計値よりも大きくなります。Tenured 増加要因の基点オブジェクトリストの出力例を次に示します。

```
Garbage Profile Root Object Information
---------------------------------------
*, [F # 43861400
```

9.9　Tenured 増加要因の基点オブジェクトリスト出力機能

この節では，Tenured 増加要因の基点オブジェクトリスト出力機能について説明します。

Tenured 増加要因の基点オブジェクトリスト出力機能は，クラス別統計情報を出力する機能の一つです。Tenured 増加要因の基点オブジェクトリスト出力機能は，Tenured 領域内不要オブジェクト統計機能を使って特定した，不要なオブジェクトの基点となるオブジェクトの情報を出力できます。

Tenured 増加要因の基点オブジェクトリスト出力機能を使用する場合は，jheapprof コマンドの引数に-rootobjectinfo を指定します。jheapprof コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「jheapprof（クラス別統計情報付き拡張スレッドダンプの出力）」を参照してください。

この節の構成を次の表に示します。

表 9-18　この節の構成（Tenured 増加要因の基点オブジェクトリスト出力機能）

分類	タイトル	参照先
解説	Tenured 増加要因の基点オブジェクトリスト出力機能の概要	9.9.1
	Tenured 増加要因の基点オブジェクトリスト出力機能で出力するクラス別統計情報	9.9.2

注　「実装」，「設定」，「運用」および「注意事項」について，この機能固有の説明はありません。

9.9.1　Tenured 増加要因の基点オブジェクトリスト出力機能の概要

Tenured 増加要因の基点オブジェクトリスト出力機能では，Tenured 領域内不要オブジェクト統計機能を使って特定した，不要なオブジェクトの基点となるオブジェクトの情報をリストにしてスレッドダンプファイルに出力します。

また，Tenured 増加要因の基点オブジェクトリスト出力機能は，Tenured 領域内不要オブジェクト統計機能が前提になります。

ポイント

Tenured 増加要因の基点オブジェクトリスト出力機能を使って取得した情報は，明示管理ヒープ機能を使用する場合に指定する，自動配置設定ファイルに指定できます。自動配置設定ファイルを使った明示管理ヒープの使用については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7.13.2　自動配置設定ファイルを使った明示管理ヒープ機能の使用」を参照してください。

Tenured 増加要因の基点オブジェクトリスト出力機能の仕組みについて説明します。

(1) 基点となるオブジェクトのリスト出力

Tenured 増加要因の基点オブジェクトリスト出力機能は，Tenured 領域内不要オブジェクト統計機能の処理のあとに続けて実行されます。Tenured 領域内不要オブジェクト統計機能で実行する処理については，「9.8.1　Tenured 領域内不要オブジェクト統計機能の概要」を参照してください。

Tenured 増加要因の基点オブジェクトリスト出力機能は，Tenured 領域内のアドレスの低い順にオブジェクトを検索します。検索したオブジェクトの中で，すでに参照関係で調べられているオブジェクト，および不要なオブジェクトのクラス情報を取得して，出力候補リストに保存します。

出力候補リストに保存した情報をインスタンスサイズの合計でソートし，合計サイズが jheapprof コマンドの-rootobjectinfost オプションで指定した値以上のクラス情報だけが出力されます。

(2) 不要なオブジェクトの参照関係の確認

Tenured 増加要因の基点オブジェクトリスト出力機能を実行して基点となるオブジェクトを取得する際の，Tenured 領域内の不要オブジェクトによる参照関係の例を示します。

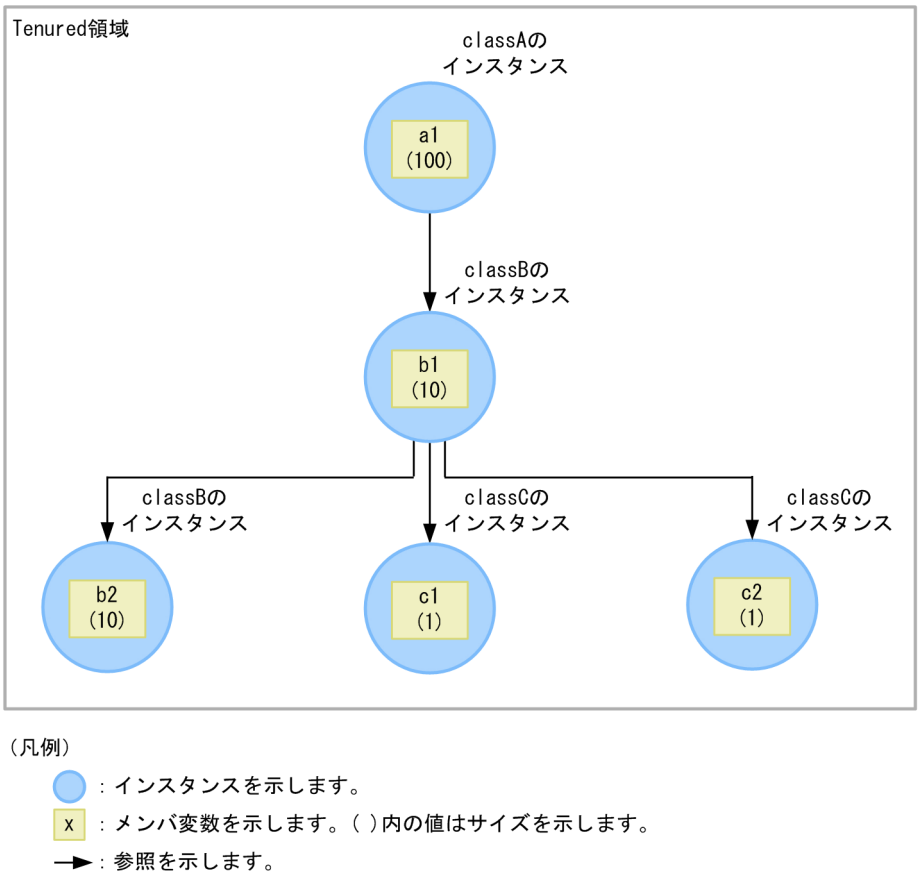
インスタンス数

- classA：a1 が存在するため 1
- classB：b1，b2 の二つが存在するため 2
- classC：c1，c2 の二つが存在するため 2

インスタンスサイズの合計

- classA：classA のインスタンス（a1）と，その参照先のインスタンス（b1，b2，c1，c2）のサイズを合計した 122
- classB：classB のインスタンス（b1，b2）と，その参照先のインスタンス（c1，c2）のサイズを合計した 22
- classC：classC のインスタンス（c1，c2）を合計した 2

図 9-15　Tenured 領域内の不要なオブジェクトによる参照関係の例



Tenured 領域内不要オブジェクト統計機能を実行して図 9-15 のような参照関係の情報を取得した場合に，Tenured 増加要因の基点オブジェクトリスト出力機能で取得する情報を次に示します。

- 基点オブジェクト：a1
- 基点オブジェクトのクラス：A
- 基点オブジェクトのクラス A のインスタンスサイズの合計：122

図 9-15 の参照関係の情報を Tenured 増加要因の基点オブジェクトリスト出力機能を実行して出力した場合の出力例を次に示します。

```
Garbage Profile Root Object Information
---------------------------------------
*, A # 122
```

9.9.2　Tenured 増加要因の基点オブジェクトリスト出力機能で出力するクラス別統計情報

ここでは，Tenured 増加要因の基点オブジェクトリスト出力機能で出力するクラス別統計情報の出力形式，および出力項目について説明します。

- 出力形式

```
Garbage Profile Root Object Information
---------------------------------------
*,〈クラス名〉#〈サイズ〉
…
```

- 出力項目

  出力形式で示した各項目について説明します。

表 9-19　出力項目（Tenured 領域内不要オブジェクト統計機能）

出力項目	意味
<クラス名>	Tenured 増加要因となった基点となるオブジェクトのクラス名が出力されます。
<サイズ>	Tenured 領域内不要オブジェクト統計機能のクラス別統計情報から取得したインスタンスサイズの合計がバイト単位で出力されます。

9.10　クラス別統計情報解析機能

この節では，クラス別統計情報解析機能について説明します。

クラス別統計情報解析機能を使用すると，クラス別統計情報で取得した情報を CSV 形式で出力できます。

この節の構成を次の表に示します。

表 9-20　この節の構成（クラス別統計情報解析機能）

分類	タイトル	参照先
解説	クラス別統計情報解析機能の概要	9.10.1
	クラス別統計情報解析機能の出力例	9.10.2
注意事項	クラス別統計情報解析機能の注意事項	9.10.3

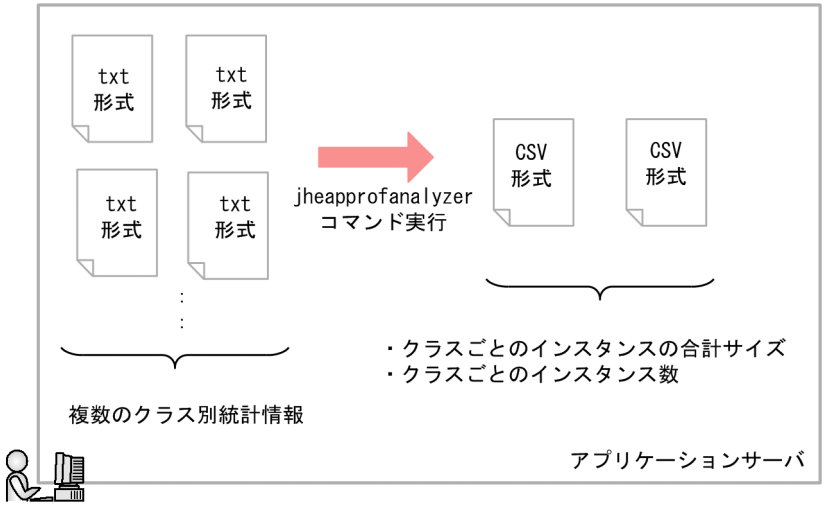
注　「実装」,「設定」および「運用」について，この機能固有の説明はありません。

9.10.1　クラス別統計情報解析機能の概要

jheapprofanalyzer コマンド（クラス別統計情報解析機能）を実行することで，クラス別統計情報の付いた複数の拡張スレッドダンプファイルを入力ファイルとして，クラスごとのインスタンスの合計サイズ，およびクラスごとのインスタンス数を時系列に出力します。出力されるファイルは，CSV 形式で出力されます。

クラス別統計情報として取得した情報を基に，クラス別統計情報解析機能を実行して CSV 形式に出力する場合の流れを次に示します。

図 9-16　クラス別統計情報解析機能を実行して CSV 形式に出力する場合の流れ



クラス別統計情報解析機能では，インスタンスの合計サイズが大きいインスタンスの情報を出力して，そのインスタンスのメモリ使用量だけを確認することもできます。インスタンスの合計サイズが大きいもの

だけを出力する場合には-DJP.co.Hitachi.soft.jvm.tools.jheapprofanalyzer.threshold にしきい値を指定し，jheapprofanalyzer コマンドに指定して実行します。jheapprofanalyzer コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「jheapprofanalyzer（クラス別統計情報解析ファイルの CSV 出力)」を参照してください。

9.10.2　クラス別統計情報解析機能の出力例

ここでは，クラス別統計情報解析機能の入力ファイル，出力ファイル，および出力形式について説明します。

(1)　入力ファイル

クラス別統計情報解析機能で使用する入力ファイルは，クラス別統計情報を出力した拡張スレッドダンプファイルです。

(2)　出力ファイル

クラス別統計情報解析機能で出力するファイルは，クラスごとのインスタンスの合計サイズを出力したファイル，およびクラスごとのインスタンス数を出力したファイルの 2 種類です。出力ファイルはカレントディレクトリに次のファイル名で作成されます。

表 9–21　出力ファイルのファイル名

出力ファイルの種類	出力ファイルのファイル名の例
インスタンス合計サイズファイル	JheapprofAnalyzer_size_*nnn*.csv
インスタンス数ファイル	JheapprofAnalyzer_num_*nnn*.csv

（凡例）

nnn：ファイルの分割番号が出力されます。分割番号の範囲は 001～999 です。

列が 201 列を超える場合は出力ファイルが分割されます。また，999 ファイルを超えた場合は，001 に戻りファイルは書き換えられます。

分割する列数は，201 列（クラス名 1 列＋値 200 列）を超えた場合とし，出力形式は分割したファイルも同じになります。

(3)　出力形式

クラス別統計情報解析機能で出力されるファイルの出力形式を次の図に示します。なお，インスタンス合計サイズ，およびインスタンス数が出力された CSV ファイルの出力形式は同じです。

図 9-17　クラス別統計情報解析機能で出力されるファイルの出力形式

1列目はクラス名　　最大200個（列）

class name,	入力ファイル名,	入力ファイル名,	・・・	入力ファイル名
クラス名,	値-1-1,	値-1-2,	・・・	値-1-xxx
:	:	:	・・・	:
クラス名,	値-y-1,	値-y-2,	・・・	値-y-xxx

（凡例）
入力ファイル名：処理対象に指定したクラス別統計情報
クラス名：入力ファイルに出力されていたクラス名
値：インスタンスの合計サイズ，またはインスタンス数

クラス名と値，および値と値の間はコンマで区切ります。また，行の最後は値（空白も含む）で終了します。

クラス名の出力順はランダムです。値は入力ファイルの先頭行にある日付を基に，日付の古いものから横に並びます。同じ日付の入力ファイルがある場合はランダムに連続して横に並びます。

参考

クラス別統計情報解析機能を複数回実行すると，処理途中のクラスが消滅したり追加されたりする場合があります。また，該当するクラスがない場合の値には 0 が出力されます。次に示す図のクラス情報の例を使用して説明します。

図 9-18　クラス情報の例

1回目のクラス別統計情報（A.txt）

```
ClassA      100
ClassB      100
```

2回目のクラス別統計情報（B.txt）

```
ClassA      100
ClassB       30
ClassC       50
ClassB        0
```

3回目のクラス別統計情報（C.txt）

```
ClassA      100

ClassC       50
```

上記のようなクラス情報の場合で，-DJP.co.Hitachi.soft.jvm.tools.jheapprofanalyzer.threshold のしきい値を 0 にしたときの出力結果は次に示す図のようになります。

図 9-19　クラス別統計情報解析機能の出力例

```
class name,A.txt,B.txt,C.txt
ClassA,100,100,100
ClassB,100,30,0
ClassC,0,50,50
ClassD,0,0,0
```

インスタンス合計サイズの最大値は 0～2^{63}-1，インスタンス数の最大値は 0～2^{31}-1 です。一つの入力ファイルに同じクラス名がある場合は，インスタンスサイズの合計が加算されます。また，インスタンス数も加算されます。加算されたことによって，それぞれの最大値を超えた場合は，指定

した最大値が出力されます。なお，一つのクラスについて，すべての入力ファイルで該当するクラスの情報がない，またはしきい値未満の場合は，そのクラスの情報は出力されません。

9.10.3　クラス別統計情報解析機能の注意事項

クラス別統計情報解析機能を使用する場合，jheapprofanalyzer コマンド実行中に入力ファイルの更新，および削除の操作はしないでください。クラス別統計情報解析機能は，日付を取得するとき，およびデータを読み込むときの 2 回ファイルを開きます。そのため，コマンド実行中に入力ファイルの更新，および削除の操作をした場合の結果は保証されません。

9.11　Survivor 領域の年齢分布情報出力機能

この節では，Survivor 領域の年齢分布情報出力機能について説明します。

Survivor 領域の年齢分布情報出力機能を使用すると，CopyGC 実行時に，Survivor 領域の使用状況が調査できます。この情報は，メモリサイズのチューニングに使用できます。

この節の構成を次の表に示します。

表 9-22　この節の構成（Survivor 領域の年齢分布情報出力機能）

分類	タイトル	参照先
解説	Survivor 領域の年齢分布情報出力機能の概要	9.11.1
	Survivor 領域の年齢分布情報の出力形式と出力例	9.11.2
設定	実行環境での設定	9.11.3
注意事項	Survivor 領域の年齢分布情報出力機能使用時の注意事項	9.11.4

注 「実装」および「運用」について，この機能固有の説明はありません。

9.11.1　Survivor 領域の年齢分布情報出力機能の概要

製品の JavaVM では，多くのトラブルシュート情報が取得できるように，ログの出力内容が標準の JavaVM よりも拡張されています。製品の JavaVM ログは，製品の JavaVM ログファイルに出力されます。Survivor 領域の年齢分布情報出力機能は，CopyGC 実行時，製品の JavaVM ログファイルに，Survivor 領域の Java オブジェクトの年齢分布情報を出力できる機能です。この情報を使用すると，Survivor 領域のオブジェクトの使用状況が調査でき，Survivor 領域のメモリサイズのチューニングができます。Survivor 領域のメモリサイズのチューニングについては，マニュアル「アプリケーションサーバ システム設計ガイド」の「7.6.1　Java ヒープ内の Survivor 領域のメモリサイズの見積もり」を参照してください。

Survivor 領域の年齢分布情報出力機能は，Survivor 領域の年齢分布情報に加えて日時も出力できます。また，出力先が製品の JavaVM ログファイルなので，ほかのログとの同期が取れます。

製品の JavaVM ログファイルについては，「4.10　JavaVM ログ（JavaVM ログファイル）」を参照してください。また，Java VM のチューニングについては，マニュアル「アプリケーションサーバ システム設計ガイド」の「7.　JavaVM のメモリチューニング」を参照してください。

9.11.2　Survivor 領域の年齢分布情報の出力形式と出力例

Survivor 領域の年齢分布情報は，CopyGC 発生時に，CopyGC のログのあとに続けて出力されます。Survivor 領域の年齢分布情報の出力形式と出力例を次に示します。

出力形式

```
[PTD]<date>[Desired survivor:size bytes][New threshold:value][MaxTenuringThreshold: max_value][age1:total_age1][age2:total_age2]...[agen:total_agen]
```

説明

- PTD：Survivor 領域の年齢情報であることを示す識別子
- date：GC が発生した日時（-XX:+HitachiVerboseGCPrintDate（拡張 verbosegc 情報日付出力オプション）を指定した場合だけ出力）※1
- size：Survivor 領域のサイズ（単位：バイト）
- value：次回の GC 発生時に Tenured 領域へ移動するオブジェクトの年齢のしきい値
- max_value：-XX:MaxTenuringThreshold の指定値※2
- total_age1：1 歳のオブジェクトが使用しているメモリサイズの合計（単位：バイト）
- total_age2：1 歳から 2 歳までのオブジェクトが使用しているメモリサイズの合計（単位：バイト）
- total_agen：1 歳から n 歳までのオブジェクトが使用しているメモリサイズの合計（単位：バイト）※3

注

-XX:+HitachiCommaVerboseGC を指定している場合は，次の形式で出力されます。

```
PTD,date,size,value,max_value,total_age1,total_age2,...,total_agen
```

注※1

対応する CopyGC のログと同じ時刻が表示されます。

注※2

-XX:MaxTenuringThreshold の指定値には，CopyGC 実行時に，From 空間と To 空間で Java オブジェクトを入れ替える回数のしきい値を設定します。

注※3

存在するオブジェクトを最小年齢から最大年齢まで順番に表示します。表示されるオブジェクトの最大年齢が max_value の値に近いと，寿命の長いオブジェクトが存在することを示します。

出力例

```
[VGC]<Wed May 28 11:45:23 2008>[GC 648K->136K(1984K), 0.0013020 secs][DefNew::Eden: 512K->0K(512K)][DefNew::Survivor: 0K->0K(64K)][Tenured: 136K->136K(1408K)][Metaspace: 3634K(4492K, 4492K)->3634K(4492K, 4492K)][class space: 356K(388K, 388K)->356K(388K, 388K)][cause: ObjAllocFail][User: 0.0000000 secs][Sys: 0.0000000 secs]
[PTD]<Wed May 28 11:45:23 2008>[Desired survivor:5467547 bytes][New  threshold:30][MaxTenuringThreshold:31][age1:1357527][age2:1539661]
```

この出力例では，次の内容が確認できます。

- 出力契機は，2008 年 5 月 28 日(水)11 時 45 分 23 秒に発生した CopyGC です。

- Survivor 領域のメモリサイズは 5,467,547 バイトです。Survivor 領域のオブジェクトは 2 歳までです。1 歳のオブジェクトが使用しているメモリサイズは 1,357,527 バイト，1 歳から 2 歳までのオブジェクトが使用しているメモリサイズは 1,539,661 バイトです。

9.11.3　実行環境での設定

Survivor 領域の年齢分布情報出力機能を使用する場合，次の設定が必要です。

- J2EE サーバ
- バッチサーバ
- Java アプリケーション

(1)　J2EE サーバの設定

J2EE サーバの設定は，簡易構築定義ファイルで実施します。Survivor 領域の年齢分布情報出力機能の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に JavaVM 起動パラメタ（add.jvm.arg）で指定します。指定するパラメタ値を次に示します。

- -XX:+HitachiVerboseGCPrintTenuringDistribution
  Survivor 領域の年齢分布情報を製品の JavaVM ログファイルへ出力します。デフォルト値は，-XX:-HitachiVerboseGCPrintTenuringDistribution です。

簡易構築定義ファイルおよびパラメタについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

(2)　バッチサーバの設定

バッチサーバの設定は，簡易構築定義ファイルで実施します。Survivor 領域の年齢分布情報出力機能の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に JavaVM 起動パラメタ（add.jvm.arg）で指定します。指定するパラメタ値を次に示します。

- -XX:+HitachiVerboseGCPrintTenuringDistribution
  Survivor 領域の年齢分布情報を製品の JavaVM ログファイルへ出力します。デフォルト値は，-XX:-HitachiVerboseGCPrintTenuringDistribution です。

簡易構築定義ファイルおよびパラメタについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

(3)　Java アプリケーションの設定

Java アプリケーションの設定は，usrconf.cfg（Java アプリケーション用オプション定義ファイル）で実施します。Survivor 領域の年齢分布情報出力機能の定義は，usrconf.cfg の add.jvm.arg キーで指定します。指定するパラメタ値を次に示します。

- -XX:+HitachiVerboseGCPrintTenuringDistribution
  Survivor 領域の年齢分布情報を製品の JavaVM ログファイルへ出力します。デフォルト値は，-XX:-HitachiVerboseGCPrintTenuringDistribution です。

usrconf.cfg（Java アプリケーション用オプション定義ファイル）については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「12.2.1　usrconf.cfg（Java アプリケーション用オプション定義ファイル)」を参照してください。

9.11.4　Survivor 領域の年齢分布情報出力機能使用時の注意事項

Survivor 領域の年齢分布情報出力機能を使用すると，CopyGC 実行時のログが使用しない場合に比べて倍以上に増えます。この機能は，Survivor 領域のチューニング時だけに使用することをお勧めします。

9.12　hndlwrap 機能

この節では，hndlwrap 機能について説明します。

hndlwrap 機能は，ログオフ時の JavaVM のログオフイベントの発生を抑止できます。なお，この機能は Windows の場合だけ使用できます。

この節の構成を次の表に示します。

表 9-23　この節の構成（hndlwrap 機能）

分類	タイトル	参照先
解説	hndlwrap 機能の概要	9.12.1
注意事項	hndlwrap 機能使用時の注意事項	9.12.2

注　「実装」,「設定」および「運用」について，この機能固有の説明はありません。

9.12.1　hndlwrap 機能の概要

hndlwrap 機能は，ログオフした場合にログオフイベント，およびウィンドウクローズが発生しないようにする機能です。

この機能は，Java Virtual Machine Tool Interface（JVMTI）を使用します。「hndlwrap2.dll」を JVMTI インタフェースでローディングすると，クラスの準備イベントを検出してログオフとウィンドウクローズのイベントを無視するイベントハンドラがインストールされます。これによって，-agentlib:hndlwrap2 オプションを指定して hndlwrap 機能を実行した場合には，ログオフ後もコマンドを動作させることができます。指定するオプションの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.5　Application Server で指定できる Java HotSpot VM のオプション」を参照してください。

9.12.2　hndlwrap 機能使用時の注意事項

hndlwrap 機能使用時の注意事項を次に示します。

- -XX:+EagerXrunInit オプションを指定したときは，-Xrunhndlwrap オプションは無効になります。
- コマンドプロンプト上で hndlwrap 機能を使用した Java アプリケーションを実行している場合にログオフすると，エラー用のポップアップが表示され，ログオフできません。
- -agentlib:hndlwrap2 オプションはほかの JVMTI プログラムと同時に実行できません。実行した場合の動作は保証されません。
- -Xrunhndlwrap オプション，および-agentlib:hndlwrap2 オプションは同時に指定できません。

9.13　JIT コンパイル時の C ヒープ確保量の上限値設定機能

JavaVM がサポートしている Just In Time 方式でのコンパイル（**JIT コンパイル**）では，コンパイル実行中に C ヒープを使用します。処理数が多いメソッドや大量のメソッドに対して JIT コンパイルを実行した場合，コンパイル処理のために確保される C ヒープのサイズも多大になり，C ヒープ不足が発生することがあります。この場合，JavaVM の強制終了や J2EE サーバの異常終了などのトラブルが発生して，システムが全面停止してしまうおそれがあります。

このような問題の発生を防止するために，JIT コンパイルで使用する C ヒープのサイズに上限値を設定できます。上限値を超えた場合は，JIT コンパイルが中止され，以降のコンパイルはインタプリタ方式で実行されます。これによって，JavaVM の強制終了を防ぎ，システムの停止を抑止できます。

JIT コンパイル時の C ヒープ確保量の上限は，-XX:HitachiJITCompileMaxMemorySize オプションで指定します。-XX:HitachiJITCompileMaxMemorySize オプションの詳細は，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:HitachiJITCompileMaxMemorySize（JIT コンパイル時の確保メモリ上限値指定オプション）」を参照してください。

9.14　スレッド数の上限値設定機能

アプリケーションで使用するスレッド数が多くなると，C ヒープで使用するメモリ使用量が増加します。メモリ使用量の増加に伴い C ヒープ不足が発生した場合，JavaVM の強制終了，J2EE サーバの異常終了などのトラブルが発生して，システムが全面停止してしまうおそれがあります。

このような問題の発生を防止するために，使用できるスレッド数に上限値を設定できます。あらかじめ使用するスレッド数の上限を把握し，その数を基に C ヒープに割り当てるメモリサイズを決定することで，C ヒープ不足の発生を防止します。なお，設定した上限値を超えたスレッドが生成された場合は，例外がスローされます。アプリケーションでこの例外をキャッチして，適切な対処をすることで，システムの停止を抑止できます。

スレッド数の上限は，-XX:HitachiThreadLimit オプションで指定します。-XX:HitachiThreadLimit オプションの詳細は，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「-XX:HitachiThreadLimit（スレッド数の上限値指定オプション)」を参照してください。

9.15　製品の JavaVM の機能使用時の注意事項（UNIX の場合）

製品の JavaVM の機能を使用する際の注意事項について説明します。

9.15.1　UNIX 共通の場合

UNIX で共通の注意事項について説明します。

- SIGXFSZ シグナル受信時の動作について

  JavaVM が SIGXFSZ シグナルを受信した場合，次のメッセージを標準出力に出力して処理を継続します。

  ```
  Java HotSpot(TM) 64-Bit Server VM warning: File size limit exceeded.
  ```

  ただし，ファイルサイズの上限を超える場合は，ファイルへ書き込みません。

- SIGXCPU シグナル受信時の動作について

  JavaVM が SIGXCPU シグナルを受信した場合，次のメッセージを標準出力に出力して処理を継続します。

  ```
  Java HotSpot(TM) 64-Bit Server VM warning: CPU time limit exceeded.
  ```

- AWT について（AIX の場合）

  AIX の場合，アプリケーションサーバの AWT は，XToolkit（sun.awt.X11.XToolkit）を使用しています。Motif ベースの MToolkit はサポートしていません。

9.15.2　AIX の場合

AIX での注意事項について説明します。

- ページングスペース見積もりについて

  J2EE サーバおよび Web コンテナサーバを起動するときは，プログラム稼動中にページング不足でプログラムを停止（SIGKILL）させられないようにするため，起動シェルで AIX の早期ページングスペース割り当てを指定する環境変数 PSALLOC=early を指定してください。ただし，/etc/environment ファイルに PSALLOC=early を指定すると，すべてのプロセスが早期ページングスペース割り当て指定となり，ページングスペース不足に陥ります。このため，該当するサーバを起動するシェルだけで，環境変数を指定するようにしてください。また，環境変数 NODISCLAIM=true も同時に指定してください。早期ページングスペース割り当てのページングスペースを見積もる上で考慮する点については，AIX のマニュアルを参照してください。

- ネイティブライブラリ検索パスについて

java コマンドを使用する場合

Developer's Kit for Java に含まれる java コマンドを使って Java プログラムを実行する場合，System.loadLibrary()でローディングされるシステムライブラリのディレクトリ検索パスは，環境変数 LIBPATH，および環境変数 LD_LIBRARY_PATH で指定できます。検索順序は，LD_LIBRARY_PATH，LIBPATH の順に優先されます。

java コマンドを使用しない場合

JNI を使って JavaVM を起動し，Java プログラムを実行する場合，System.loadLibrary()でローディングされるシステムライブラリのディレクトリ検索パスは，環境変数 LIBPATH で指定されたパスだけになります。

- JNI を使用した JavaVM の起動について

  JNI を使って JavaVM を起動する場合は，次の点に注意してください。

  - 次の AIX 固有環境変数を必ず設定してください。

    csh（C シェル）の場合

    ```
    setenv AIXTHREAD_MUTEX_DEBUG OFF
    setenv AIXTHREAD_RWLOCK_DEBUG OFF
    setenv AIXTHREAD_COND_DEBUG OFF
    ```

    sh（標準シェル）および ksh の場合

    ```
    export AIXTHREAD_MUTEX_DEBUG=OFF
    export AIXTHREAD_RWLOCK_DEBUG=OFF
    export AIXTHREAD_COND_DEBUG=OFF
    ```

    これらの環境変数については，該当ページ（https://www.ibm.com/support/knowledgecenter/ja/ssw_aix_72/performance/thread_env_vars.html）を参照してください。

  - 環境変数 LIBPATH に次のパスを追加してから起動プログラム実行してください。

    /opt/Cosminexus/jdk/lib

    /opt/Cosminexus/jdk/lib/server

    なお，起動プログラム生成時のリンケージオプションとして，これらのパスを設定している場合は，環境変数 LIBPATH は設定不要です。

- フォントパスについて

  GUI で表示されるフォントは，システムのフォントパスから検索されます。Developer's Kit for Java の起動前にシステムのフォントパスを変更するプログラムを実行すると，GUI が正しく表示されないことがあります。また，Developer's Kit for Java によってもシステムのフォントパスが変更されます。このため，Developer's Kit for Java の起動後にシステムのフォントパスを検索するようなプログラムでは，GUI が正しく表示されないことがあります。

  Developer's Kit for Java の起動前や起動後に次の設定をすると，システムのフォントパスをデフォルトの状態に戻すことができます。

  ```
  % xset fp default
  ```

- プログラムの起動時間について

Developer's Kit for Java の Java 仮想マシンの実装は，サーバ VM（Java HotSpot Server VM）です。サーバ VM は，旧バージョンのクライアント VM（Java HotSpot Client VM）に比べて，一般に，プログラムの実行速度は高速ですが，J2EE サーバなどのプログラムの起動が遅くなることがあります。プログラムの起動を速くしたい場合には，次のオプションを指定してください。ただし，このオプションを指定した場合，プログラムの実行速度は通常よりも遅くなることがあります。

```
-XX:CompileThreshold=3000
```

- 32 ビット版の JavaVM（アプリケーションサーバ Version 8）から 64 ビット版の JavaVM（アプリケーションサーバ Version 9）に移行する際の注意事項

  一般的に，64 ビット版の JavaVM では，32 ビット版の JavaVM と比べて，大きなアドレス空間を利用できるという利点があります。また，オブジェクトのポインタ値は，32 ビット版の JavaVM では 32 ビット，64 ビット版の JavaVM では 64 ビットとなります。このため，32 ビット版の JavaVM から 64 ビット版の JavaVM へ移行する際には，アプリケーションの見直し，メモリチューニング，環境の見直しなどが必要になることがあります。次の点に注意して移行してください。

  - オブジェクトへのアクセスコストの増加

    オブジェクトのポインタ値が 32 ビットから 64 ビットになるため，オブジェクトへのアクセスが最大で 2 倍となります。オブジェクトへ頻繁にアクセスするようなアプリケーションの場合，64 ビット版の JavaVM では実行性能に影響が出ることがあります。

  - Java ヒープのメモリサイズの増加

    オブジェクトのポインタ値が 32 ビットから 64 ビットになるため，オブジェクト 1 個あたりのサイズが最大で 2 倍に増加します。これに伴って，アプリケーションが使用する Java ヒープのメモリサイズも増加します。

  - プロセスが使用するメモリサイズの増加

    64 ビット版の JavaVM では，Java ヒープのメモリサイズだけでなく，プロセスが使用するメモリサイズも増加します。プロセスが使用するメモリサイズの見積もり方法は，core ファイルサイズの見積もり方法と同じです。見積もり方法については，「3.3.16(1)　core ファイルのサイズの上限値の設定」を参照してください。

  - JNI の互換性について

    32 ビットのバイナリとして作成された，JNI で呼び出されるネイティブプログラムは，64 ビット版の JavaVM 上では動作できません。また，一つのプロセス内で，32 ビットと 64 ビットのネイティブプログラムを同時に動作させることもできません。そのため，JNI で呼び出されるネイティブプログラムが使用されている場合は，64 ビット版の JavaVM 用にそのプログラムをコンパイルし直す必要があります。

9.15.3　Linux の場合

Linux での注意事項について説明します。

- フォントに関するメッセージについて

X サーバのようにコンソール以外のリモート端末上で，GUI のアプリケーションを起動する際，次のようなメッセージが出力される場合があります。

```
Warning: Cannot convert string
”-monotype-arial-regular-r-normal--*-140-*-*-p-*-iso8859-1” to type FontStruct
```

この場合，xfs コマンドでフォントサーバを起動し，X サーバ側で xset コマンドを使用してフォントパスにフォントサーバを指定すると回避できます。その際，フォントサーバの設定ファイルに不足しているフォントパスが指定されていることを確認してください。

9.16　ファイナライズ滞留解消機能

この節では，ファイナライズ滞留解消機能について説明します。ファイナライズ滞留解消機能を使用すると，ファイナライズ処理の滞留を解消でき，OS 資源の解放遅れなどの発生を抑止できます。

9.16.1　概要

finalize()メソッドで OS 資源の解放処理を定義しているアプリケーションでは，ファイナライズ処理が滞留して，OS 資源の解放遅れが発生することがあります。ファイナライズ滞留解消機能を使用すると，JavaVM 内でファイナライズ処理が滞留している状態を検知して，滞留しているファイナライズ処理を解消します。

ファイナライズ滞留解消機能では，ファイナライズ滞留を検知するために，Java 起動時に FinalizerThread を監視するファイナライズ処理監視スレッドを生成します。FinalizerThread は，Java 実行時に常に 1 つあり，オブジェクトの finalize()メソッドを 1 つずつ処理するスレッドです。

ファイナライズ処理監視スレッドでは，FinalizerThread で処理しているオブジェクトを定期的に監視します。次の条件を満たす場合，ファイナライズ処理が滞留していると見なして，新たにファイナライズスレッドを生成します。

- FinalizerThread でオブジェクトのファイナライズ処理が進んでいない。
- ファイナライズキューにオブジェクトがある。

FinalizerThread とファイナライズスレッドの両方で，ファイナライズ処理を実行して，滞留しているファイナライズ処理の解消を促します。なお，ファイナライズスレッドは，ファイナライズキューが空になると終了します。

9.16.2　出力情報

ファイナライズ滞留解消機能では，次に示すタイミングでメッセージを標準出力に出力します。

- ファイナライズ処理の滞留を検知して，ファイナライズスレッドを新たに生成する場合
- 生成したファイナライズスレッドが終了した場合

それぞれのタイミングで出力するメッセージの形式と出力例を次に示します。

- ファイナライズ処理の滞留を検知して，ファイナライズスレッドを新たに生成する場合
  ファイナライズ処理の滞留を検知して，ファイナライズスレッドを新たに生成する場合に出力されるメッセージの形式を次に示します。

```
# FinalizerWatcherThread: Create: create secondary finalizer thread. [queue length = queue] <date>
```

説明

queue：ファイナライズキューの要素数

date：ファイナライズスレッドを新たに生成した日時

出力例を次に示します。

```
# FinalizerWatcherThread: Create: create secondary finalizer thread. [queue length = 128] <M
on May 26 18:00:36 JST 2008>
```

- 生成したファイナライズスレッドが終了した場合

  生成したファイナライズスレッドが終了した場合に出力されるメッセージの形式を次に示します。

  ```
  # FinalizerWatcherThread: Finish: secondary finalizer thread is finished. <date>
  ```

  説明

  date：生成したファイナライズスレッドが終了した日時

出力例を次に示します。

```
# FinalizerWatcherThread: Finish: secondary finalizer thread is finished.  <Mon May 26 20:12
:26 JST 2008>
```

9.16.3　実行環境での設定

ファイナライズ滞留解消機能を使用する場合，次の設定が必要です。

- J2EE サーバ
- バッチサーバ
- Java アプリケーション

(1)　J2EE サーバの設定

J2EE サーバの設定は，簡易構築定義ファイルで実施します。ファイナライズ滞留解消機能の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に JavaVM 起動パラメタ（add.jvm.arg）で指定します。指定するパラメタ値を次に示します。

- -DJP.co.Hitachi.soft.jvm.autofinalizer=true

  ファイナライズ滞留解消機能を有効にします。この機能を有効にすると，Java 起動時に監視スレッドを生成します。デフォルト値は true のため，無効にする場合は false を指定してください。

簡易構築定義ファイルおよびパラメタについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

(2) バッチサーバの設定

バッチサーバの設定は，簡易構築定義ファイルで実施します。ファイナライズ滞留解消機能の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に JavaVM 起動パラメタ（add.jvm.arg）で指定します。指定するパラメタ値を次に示します。

- -DJP.co.Hitachi.soft.jvm.autofinalizer=true
  ファイナライズ滞留解消機能を有効にします。この機能を有効にすると，Java 起動時に監視スレッドを生成します。デフォルト値は true のため，無効にする場合は false を指定してください。

簡易構築定義ファイルおよびパラメタについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

(3) Java アプリケーションの設定

Java アプリケーションの設定は，usrconf.cfg（Java アプリケーション用オプション定義ファイル）で実施します。ファイナライズ滞留解消機能の定義は，usrconf.cfg の add.jvm.arg キーで指定します。指定するパラメタ値を次に示します。

- -DJP.co.Hitachi.soft.jvm.autofinalizer=true
  ファイナライズ滞留解消機能を有効にします。この機能を有効にすると，Java 起動時に監視スレッドを生成します。デフォルト値は true のため，無効にする場合は false を指定してください。

usrconf.cfg（Java アプリケーション用オプション定義ファイル）については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「12.2.1　usrconf.cfg（Java アプリケーション用オプション定義ファイル)」を参照してください。

9.16.4 注意事項

ファイナライズ滞留解消機能を使用する場合の注意事項を次に示します。

- JavaVM 起動パラメタの値は，Java 起動時に参照されます。このため，JavaVM 実行中に java.lang.System.setProperty()メソッドなどで，JavaVM 起動パラメタの値を変更しても，ファイナライズ滞留解消機能は有効になりません。
- JavaVM 起動パラメタが複数指定された場合，最後に指定されたパラメタの値が有効になります。
- ファイナライズ滞留解消機能では，ファイナライズの滞留を監視するための常駐スレッドを 1 つ生成します。また，ファイナライズの滞留を検知すると，滞留を解消するためのスレッドを 1 つ生成します。ファイナライズの滞留が発生してもシステムに影響しない場合は，add.jvm.arg パラメタまたは add.jvm.arg キーに「-DJP.co.Hitachi.soft.jvm.autofinalizer=false」を指定して，ファイナライズ滞留解消機能を無効にすることで，ファイナライズの滞留を監視するスレッドとファイナライズの滞留を解消するスレッドの生成を抑止できます。

9.17　ログファイルの非同期出力機能

ここではログファイルの非同期出力機能について説明します。

9.17.1　概要

この機能は，ログのファイル出力を非同期でできるように変更する機能です。ここに記載している「非同期」とは，ログのファイル出力処理を専用でするスレッドを用意することで，ログのファイル出力以外の処理を実行するスレッドとは処理のタイミングを合わせることなく，ログをファイルに出力していくことを指します。ここでは例として，GC 発生時に出力されるログファイルについて記載します。

GC の実行中は，プログラムの処理が停止します。日立 JavaVM ログファイル機能では，GC を実行しているスレッドが，GC の情報のログファイル出力処理もしているため，GC の情報をログファイルに出力する処理が完了するまで次の処理へ進むことができません。

そこで，この機能を有効にすることによって，GC 情報のログファイル出力処理は専用のスレッドが実行するようになります。そのため，それ以外の処理を実行するスレッドが GC 情報のログファイル出力処理の完了を待つことによる処理の遅延を削減することができます。

しかし，この機能を使用することでログの一部が欠けるおそれがあります。この機能は，一時的にログをバッファにためておき，その後ログファイルへ出力するという処理をしています。ログの出力量が非常に多い場合や，ファイル入出力処理が極端に遅い場合に，一部のログが出力されないおそれがあります。そのため，GC によるプログラムの処理の停止時間として 100msec 以下を達成するようなチューニングをしている場合は，この機能の使用を推奨します。一方で，100msec 以下を達成するようなチューニングをしていない場合には，この機能の使用は推奨しません。

9.17.2　対象ログファイル

この機能の対象とするファイルは次の 2 つです。

- 日立 JavaVM ログファイル
- 明示管理ヒープ機能のイベントログ

9.17.3　エラーケース

ファイルへの I/O 処理速度より，バッファへの書き込み処理速度が速い場合，バッファがラウンドしてログが出力されない場合があります。

バッファがラウンドしてログが出力されなかった場合の出力様式

```
[ASY]<skip_count> log is skipped.

skip_count：バッファがラウンドして出力できなかった場合のスキップした行数
```

この機能で使用するログに出力する専用のスレッドの生成に失敗した場合は，次のメッセージを標準出力に出力し，JavaVM を起動しません。

日立 JavaVM ログファイル用のスレッドの生成に失敗

```
Error occurred during initialization of VM
Could not create thread for VM: Asynchronous javalog thread creation failed. thread_count th
reads exist.

thread_count: 存在するスレッド数
```

明示管理ヒープログファイル用のスレッドの生成に失敗

```
Error occurred during initialization of VM
Could not create thread for VM: Asynchronous ehjavalog thread creation failed. thread_count
threads exist.

thread_count: 存在するスレッド数
```

9.17.4　注意事項

ログの非同期出力機能使用時，-XX:HitachiOutOfMemoryStackTraceLineSize または-XX:HitachiJavaClassLibTraceLineSize オプションに 4096 より大きな値を指定した場合は，オプションの値として 4096 が指定されたものとして動作します。その場合の動作の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.2　JavaVM 拡張オプションの詳細」を参照してください。

9.17.5　メモリ所要量

この機能は，新たにバッファとスレッドを生成します。この機能が有効の場合，このバッファとスレッドは JavaVM 終了時まで生存します。そのため，この機能を使用することでバッファとスレッド用にメモリを消費します。そのメモリ消費量を次に記載します。

まず，バッファのメモリ消費量を次に示します。

```
（ログ1行分のサイズ = 4096byte）×（バッファにためこめる行数 = 1024行）= 4Mbyte
```

さらに明示管理ヒープ機能を使用している場合は，明示管理ヒープ機能のイベントログ用のバッファも生成するため，2 倍のメモリ消費量となり 8Mbyte となります。

また，スレッドのメモリ消費量は-Xss オプションに指定された値です。こちらも，明示管理ヒープ機能を使用している場合は，明示管理ヒープ機能のイベントログ用のスレッドも生成するため，メモリ消費量は2倍となります。

表 9–24　プラットフォームごとの-Xss のデフォルト値

プラットフォーム	-Xss のデフォルト値
Linux(EM64T)	1Mbyte
Windows	1Mbyte
AIX	1Mbyte

9.18　圧縮オブジェクトポインタ機能

ここでは圧縮オブジェクトポインタ機能について説明します。

9.18.1　概要

圧縮オブジェクトポインタ機能は，Java アプリケーションによって作成される Java オブジェクトのサイズを圧縮することで，Java アプリケーションのメモリ使用効率を向上させる機能です。この機能を有効にすることによって，Application Server 上で動作している Java アプリケーション実行中での，Java ヒープ領域と Explicit ヒープ領域※の使用量を少なくできます。

圧縮オブジェクトポインタ機能は，-XX:+UseCompressedOops を指定することで有効にできます。UseCompressedOops オプションの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「14.2　JavaVM 拡張オプションの詳細」を参照してください。

この機能を有効にすると，実行時のメモリ使用量と JavaVM の実行性能が変化する場合があります。また，JavaVM 実行時の OS 環境の違いによっても，この機能有効時の JavaVM の実行性能が変動します。

注※　Explicit ヒープ領域は，明示管理ヒープ機能が有効な場合にだけ該当します。

9.18.2　前提条件

各ヒープ領域（Java ヒープ領域，および Explicit ヒープ領域※）の設定サイズの総和が 32GB 未満のときに有効です。設定サイズの総和が 32GB 以上の場合は，-XX:+UseCompressedOops を指定しても，この機能は無効となります。

```
圧縮オブジェクトポインタ機能が有効となるヒープ領域サイズ指定値

(1)明示管理ヒープ機能を使用する場合
  （-Xmxオプションの指定値）+（-XX:HitachiExplicitHeapMaxSizeオプションの指定値）< 32GB

(2)明示管理ヒープ機能を使用しない場合
  （-Xmxオプションの指定値）< 32GB
```

注※

Explicit ヒープ領域は，明示管理ヒープ機能が有効な場合にだけ該当します。

9.18.3 注意事項

1. この機能を有効にすることによって，Java アプリケーション実行時の Java ヒープ領域と Explicit ヒープ領域の使用量が減少し，その変化量は Java アプリケーションに依存します。もし，この機能を有効にしたことによるメモリ使用量の変化を問題視する場合は，この機能を OFF にしてください。
2. この機能を有効にすることによって，Java アプリケーション実行時のスループットなどの実行性能に影響を与える場合があるのでご注意ください。この機能が実行性能に影響する点は，次の 2 点です。

   (a)JavaVM が扱うデータ(Java オブジェクト)サイズが減少し，より効率的なメモリアクセスができる(性能向上要因)

   (b)JavaVM が Java オブジェクトのサイズを圧縮するための計算が必要となる(性能低下要因)

   (a)(b)の 2 点の要因から，この機能有効時 JavaVM 実行時の性能が変化することになります。また(b)については，JavaVM 起動時の OS 環境の違いによって，計算方法(詳細は次の(補足))も変化し実行性能に影響します。もし，この機能を有効にしたことによる性能の変化を問題視する場合は，この機能を OFF にしてください。

   (補足) Java オブジェクトのサイズの圧縮について

   圧縮オブジェクトポインタ機能有効時，JavaVM は Java オブジェクトのサイズを圧縮して管理しますが，その圧縮方法(以降，モードと呼びます)は複数存在します。JavaVM は起動時に，その実行環境状況によってどのモードを選択するかを決定します。以降，JavaVM は起動時に選択したモードを適用して，Java オブジェクトのサイズを圧縮します。

   どのモードを選択するかの条件は，次の 2 つとなります。

   1.Java ヒープ領域と Explicit ヒープ領域サイズの指定合計値

   2. 1.のヒープ領域が仮想アドレス空間のどこに割り当てられるか

   そのため，JavaVM 起動時のヒープ領域サイズの指定値を変更した場合，または，同一 OS 上で動作しているほかのプロセスやライブラリのロード状況によっては，JavaVM 起動ごとに異なるモードが適用されます。

   どのモードが選択されるかは，実際にヒープ領域が仮想メモリ空間上のどこに割り当てられるかに依存するため，特定の圧縮モードが常に適用されるようにこの機能を有効とすることは困難です。この機能を無効・有効とした場合だけ性能の差異がでるだけでなく，この機能を有効とした場合も，JavaVM の起動ごとに性能の差異が発生する場合もあることにご注意ください。

   参考として圧縮オブジェクトポインタ機能有効時に JavaVM が選択できる各モードについて，選択条件と圧縮方法の概要を次の表※1※2 に示します。

表 9-25　圧縮オブジェクトポインタ機能のモード一覧

	選択条件	概要説明
モード 1	各ヒープ領域（Java ヒープ領域と Explicit ヒープ領域※3）の総サイズが 4GB 未満であり，仮想アドレス空間上，全ヒープ領域が 4GB よりも低位側に存在する場合	最も簡単な計算方法で，Java オブジェクトのサイズを圧縮します。
モード 2	モード 1 の適用ができないで，仮想アドレス空間上，その全ヒープ領域が 32GB よりも低位側に存在する場合	モード 1 の次に簡単な計算方法により Java オブジェクトのサイズを圧縮します。

	選択条件	概要説明
モード 3	モード 1 とモード 2 が適用できなかった場合	これらモードのうち，最も複雑な方法で Java オブジェクトの圧縮をします。

注※1

表に示した内容は JavaVM の内部処理を参考として示したものです。以降の JDK のアップデートや修正によって予告なく変更される場合があります。

注※2

AIX では，プラットフォーム固有の制約によって，表 9-25 中のモード 1 およびモード 2 が選択できるメモリが確保できないため，必ずモード 3 が選択されます。

注※3

Explicit ヒープ領域は，明示管理ヒープ機能を利用している場合だけ該当します。

9.19 Oracle 社の JDK とアプリケーションサーバが提供する JDK との非互換性

ここでは Oracle 社の JDK とアプリケーションサーバが提供する JDK との非互換性について説明します。

9.19.1 デフォルトで選択されるメモリ管理方式

Oracle 社の JDK11 以降では，デフォルトで選択されるメモリ管理方式は，ユーザ環境に依存します。基本的にはガベージファースト・コレクタ（G1GC）が選択されます。論理プロセッサが 2 未満または物理メモリが 1792 メガバイト未満の場合はシリアルガベージコレクタ（SerialGC）が選択されます。アプリケーションサーバが提供する JDK11 ベースの，デフォルトで選択されるメモリ管理方式は，シリアルガベージコレクタ（SerialGC）です。

アプリケーションサーバが提供する JDK17 ベースでは，Oracle 社の JDK11 以降と同様です。

9.19.2 ランタイムイメージ

アプリケーションサーバではランタイムイメージの作成および再構成はサポートしていません。また，ランタイムイメージを作成する jlink コマンドは同梱していません。

9.19.3 モジュール関連オプション

アプリケーションサーバでは，モジュールに関連したオプションで，サポートしていないオプションまたは制限事項のあるオプションがあります。

(1) サポートしていないオプション

次に示すオプションは，アプリケーションサーバではサポートしていません。

- --limit-modules
- --module（または-m）
- --patch-module
- --upgrade-module-path

各オプションの詳細は，次の Web ページを参照してください。

https://docs.oracle.com/en/java/javase/11/tools/java.html

https://docs.oracle.com/en/java/javase/17/docs/specs/man/java.html

(2) 制限事項のあるオプション

次に示すオプションには制限事項があります。これらのオプションには，更新対象モジュールに JDK 内モジュール※を指定しないでください。

- --add-exports
- --add-opens
- --add-reads

注※

JDK 内モジュールとは，次の Web ページに記載されているすべてのモジュールです。
https://docs.oracle.com/javase/jp/11/docs/api/index.html
https://docs.oracle.com/javase/jp/17/docs/api/index.html

各オプションの詳細は，次の Web ページを参照してください。

https://docs.oracle.com/en/java/javase/11/tools/java.html

https://docs.oracle.com/en/java/javase/17/docs/specs/man/java.html

制限事項の対象となる使用例を次に示します。

（使用例 1）

--add-exports=java.base/jdk.internal.jimage=my.module
更新対象に java.base モジュールを指定しないでください。

（使用例 2）

--add-opens=java.base/jdk.internal.jimage=my.module
更新対象に java.base モジュールを指定しないでください。

（使用例 3）

--add-reads=java.logging=my.module
更新対象に java.logging モジュールを指定しないでください。

10 旧バージョンのアプリケーションサーバからの移行（J2EE サーバモードの場合）

この章では，旧バージョンのアプリケーションサーバから移行する場合の方法について説明します。なお，ここでは，J2EE サーバモードを使ったシステムを移行する場合に必要な移行情報を記載しています。

10.1　アプリケーションサーバの移行の概要

旧バージョンからアプリケーションサーバ 11-40 への移行方法について説明します。移行の対象となる旧バージョンを次に示します。

- アプリケーションサーバ 08-00
- アプリケーションサーバ 08-50
- アプリケーションサーバ 08-53
- アプリケーションサーバ 08-70
- アプリケーションサーバ 09-00
- アプリケーションサーバ 09-50
- アプリケーションサーバ 09-60
- アプリケーションサーバ 09-70
- アプリケーションサーバ 09-80
- アプリケーションサーバ 09-87
- アプリケーションサーバ 11-00
- アプリケーションサーバ 11-10
- アプリケーションサーバ 11-20
- アプリケーションサーバ 11-30

なお，このマニュアルでは，次に示す構成ソフトウェアの移行を中心に説明します。

- Performance Tracer
- Component Container
- Component Transaction Monitor
- Component Container - Client
- Component Container - Redirector
- Developer's Kit for Java

旧バージョンからの移行を行う場合，使用しているアプリケーションサーバの全構成ソフトウェアを同時に移行する必要があります。上記で示していないほかの構成ソフトウェア（以降，他構成ソフトウェアと呼びます）の移行については，次に示すマニュアルを参照してください。

表 10–1　構成ソフトウェアごとの参照先マニュアル

構成ソフトウェア	参照先マニュアル
HTTP Server	HTTP Server

構成ソフトウェア	参照先マニュアル
Reliable Messaging	Reliable Messaging
Web Services - Security	アプリケーションサーバ Web サービスセキュリティ構築ガイド
XML Processor	XML Processor ユーザーズガイド

注意事項

構成ソフトウェアは，バージョンごとに体系が異なる場合があります。

バージョンごとの構成ソフトウェアの体系について次に示します。

表 10-2　バージョンごとの構成ソフトウェアの体系

構成ソフトウェア	アプリケーションサーバのバージョン		備考
	V8	V9	
Component Container	○	○	なし
Component Transaction Monitor	○	○	V8 までは Application Server Enterprise だけに同梱。
DABroker Library	○	○	互換用としてサポート。
Developer's Kit for Java	○	○	なし
Performance Tracer	○	○	なし
Reliable Messaging	○	○	なし
Server Plug-in	○	－	UNIX で動作する製品には含まれない。
TPBroker	○	○	なし
Web Services - Security	○	○	なし
XML Processor	○	○	なし
HTTP Server	○	○	V9 から名称を Hitachi Web Server から HTTP Server へ変更。

（凡例）○：対応するバージョンで構成ソフトウェアあり
－：対応するバージョンで構成ソフトウェアなし

参考

TPBroker については，明示的な移行作業はありません。このほか，アプリケーションサーバ上で動作する SOAP アプリケーションや，Web サービスなどの移行については，次のマニュアルを参照してください。

- SOAP アプリケーション
  マニュアル「アプリケーションサーバ SOAP アプリケーション開発の手引」
- JAX-WS 仕様に対応した SOAP Web サービス／JAX-RS 仕様に対応した RESTful Web サービス
  マニュアル「アプリケーションサーバ Web サービス開発ガイド」

アプリケーションサーバの移行の概要について説明します。

10.1.1　移行のための新規インストール，または更新インストール

旧バージョンからアプリケーションサーバ 11-40 へ移行するためには，新規インストール，または更新インストールを行います。新規インストールとは，アプリケーションサーバがインストールされていない新規のマシンに，アプリケーションサーバをインストールすることです。更新インストールとは，インストール先のマシンに旧バージョンのアプリケーションサーバがインストール済みの場合のインストールです。

なお，更新インストールの場合は，アプリケーションサーバのインストールディレクトリを旧バージョンのインストールディレクトリから変更できません。旧バージョンがインストールされているディレクトリに上書きインストールされます。

新規インストールの場合の移行手順は，「10.2.1　新規インストールの場合」を参照してください。

更新インストールの場合の移行手順は，「10.2.2　更新インストールの場合」を参照してください。

10.1.2　移行時に引き継がれる情報

旧バージョンで使用されていた，<製品のインストールディレクトリ>以下に配置される定義情報，DD（Deployment Descriptor）情報，構成情報などのユーザ資産は，移行コマンドなどで必要に応じて自動変換され，11-40 で引き続き使用されます（一部，手動での定義修正が必要な場合もあります）。また，移行コマンドを実行すると，移行元バージョンから移行先バージョン間のすべてのバージョンごとに移行処理が実施されます。例えば，09-00 から 11-20 へ移行する場合，09-00 から 09-50 への移行，09-50 から 09-70 への移行，09-70 から 09-80 への移行，09-80 から 09-87 への移行，09-87 から 11-00 への移行，11-00 から 11-10 への移行，11-10 から 11-20 への移行，11-20 から 11-30 への移行，11-30 から 11-40 への移行というように処理が実施されます。

Component Container で引き継がれる定義情報，DD 情報，構成情報などを次の表に示します。他構成ソフトウェアについては，それぞれのドキュメントを参照してください。

表 10–3　Component Container で移行時に引き継がれる情報

機能	引き継がれる情報	ファイル名称，または格納場所
J2EE サーバ	オプション定義ファイル	usrconf.cfg

機能	引き継がれる情報	ファイル名称，または格納場所
	ユーザプロパティファイル	usrconf.properties
	セキュリティポリシーファイル	server.policy
	Web アプリケーションプロパティファイル	hitachi_web.properties
	作業ディレクトリ • 定義情報 • アプリケーション • DD 情報 • ユーザ・パスワード情報	オプション定義ファイルで指定されている ejb.public.directory の値。デフォルトは，次のとおり。 • Windows の場合 <製品のインストールディレクトリ>¥CC¥server¥public 以下 • UNIX の場合 /opt/Cosminexus/CC/server/public 以下
バッチサーバ	オプション定義ファイル	usrconf.cfg
	ユーザプロパティファイル	usrconf.properties
	セキュリティポリシーファイル	server.policy
	作業ディレクトリ • 定義情報 • DD 情報 • ユーザ・パスワード情報	オプション定義ファイルで指定されている ejb.public.directory の値。デフォルトは，次のとおり。 • Windows の場合 <製品のインストールディレクトリ>¥CC¥server¥public 以下 • UNIX の場合 /opt/Cosminexus/CC/server/public 以下
サーバ管理コマンド	オプション定義ファイル	• Windows の場合 usrconf.bat • UNIX の場合 usrconf
	ユーザプロパティファイル	usrconf.properties
Web サーバ連携	Web Server 用リダイレクタ動作定義ファイル	mod_jk.conf
	Microsoft IIS 用リダイレクタ動作定義ファイル（Windows だけのファイル）	isapi_redirect.conf
	Microsoft IIS 用マッピング定義ファイル（Windows だけのファイル）	uriworkermap.properties
	ワーカ定義ファイル	workers.properties
Management Server	運用管理エージェントプロパティファイル	adminagent.properties
	運用管理エージェント設定ファイル	adminagent.xml 別のファイル名称でバックアップされたあと，新規のファイルで上書きされます。

機能	引き継がれる情報	ファイル名称，または格納場所
		Management Server の「論理サーバの起動／停止」で使用する環境変数を指定する機能を利用している場合，手動による定義修正が必要となります。
	運用監視エージェント設定ファイル	mngagent.<実サーバ名>.properties
	Management Server 環境設定ファイル	mserver.properties
	mngsvrutil コマンドのサーバ側定義ファイル	mngsvrutil.properties
	構成情報定義	• Windows の場合 <製品のインストールディレクトリ>¥manager¥config 以下 • UNIX の場合 /opt/Cosminexus/manager/config 以下
	Management アクション実行用プロパティファイル	maction.properties
	Management Server 用メッセージマッピングファイル	mserver.jp1event.system.mapping.properties
	J2EE サーバ共通用メッセージマッピングファイル	manager.jp1event.system.mapping.properties
	J2EE サーバ個別用メッセージマッピングファイル	manager.<論理サーバ名>.jp1event.system.mapping.properties
	J2EE アプリケーションの登録ディレクトリのデフォルトパス	• Windows の場合 <製品のインストールディレクトリ>¥manager¥apps 以下 • UNIX の場合 /opt/Cosminexus/manager/apps 以下
統合ユーザ管理	JAAS のコンフィグレーションファイル	jaas.conf
	統合ユーザ管理のコンフィグレーションファイル	ua.conf
	カスタムログインモジュール格納用ディレクトリ	• Windows の場合 <製品のインストールディレクトリ>¥manager¥modules • UNIX の場合 /opt/Cosminexus/manager/modules
Smart Composer 機能	サーバ設定プロパティファイル	cmxserver.properties
	負荷分散機定義プロパティファイル	lb.properties

10.1.3　移行時の J2EE アプリケーションの動作

アプリケーションサーバを更新インストールする場合，または旧バージョンの J2EE アプリケーションをインポートする場合には，次に示す処理で，自動的に J2EE アプリケーションに含まれる EJB の再デプロイが実行されるため，一時的に性能が悪くなります。

- 開始状態の J2EE アプリケーションが存在する状態でアプリケーションサーバを更新インストールしたあとの初回 J2EE サーバ起動時
- 停止状態の J2EE アプリケーションが存在する状態でアプリケーションサーバを更新インストールしたあとの初回 J2EE アプリケーション開始時
- 旧バージョンのアプリケーションサーバでエクスポートした実行時情報を含む，開始状態の J2EE アプリケーションのインポート時
- 旧バージョンのアプリケーションサーバでエクスポートした実行時情報を含む，停止状態の J2EE アプリケーションをインポートしたあとの初回 J2EE アプリケーション開始時

10.2　アプリケーションサーバの移行手順の詳細

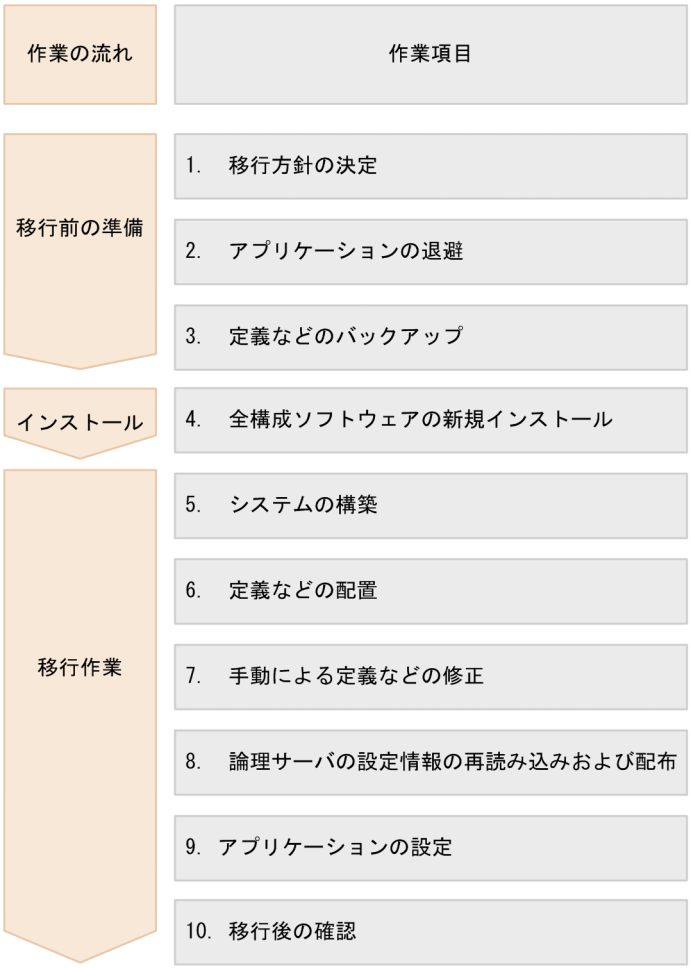
10.2.1　新規インストールの場合

ここでは，旧バージョンのアプリケーションサーバを移行する場合に，新しいアプリケーションサーバを，旧バージョンのアプリケーションサーバを構築している環境とは別の環境へ構築（新規インストール）する際の手順について示します。

新規インストールについては，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「2.2.1　Application Server のインストールについて」を参照してください。

次の手順に従って，旧バージョンからアプリケーションサーバ 11-40 へ移行してください。

図 10–1　旧バージョンからアプリケーションサーバ 11-40 への移行作業（新規インストールの場合）



(1)　移行方針の決定

移行後の環境で使用する機能や，使用する Web サーバの種類などを決めます。

また，旧バージョンのアプリケーションサーバの環境での設定を引き継ぐのか（互換性を重視したシステムへ移行），または移行後のアプリケーションサーバが推奨する機能を使用する設定にするのか（推奨機能を使用したシステムへ移行）についても検討します。互換重視のシステム，および推奨機能を使用したシステムの特徴を次に示します。

表 10–4　互換重視のシステムおよび推奨機能を使用したシステムの特徴

移行方針	特徴
互換性を重視したシステムへ移行	• 推奨されていない機能を，移行後の環境でも使用します。 推奨されていない機能については，マニュアル「アプリケーションサーバ 機能解説 互換編」を参照してください。 • 機能が廃止されたり，互換性がない仕様変更があったりするため，旧システムと完全に同じ環境は構築できません（J2SE や J2EE のバージョン変更，サーバマネージャやユーザマネージャの廃止など）。 • 最新のバージョンのアプリケーションサーバで追加された機能には，互換性を重視したシステムの設定では使用できない場合もあります（コネクション自動クローズ機能など）。
推奨機能を使用したシステムへ移行	最新のバージョンのアプリケーションサーバで使用できる機能でも，推奨される機能がほかにある場合は，推奨機能が選択されます。

これらの移行方針を決定する上で，旧バージョンと移行するバージョンとの仕様差を確認する必要があります。機能の変更や追加に伴い，旧バージョンで使用している機能，使用条件，使用方法によっては，次の作業が必要となります。

- 操作手順の変更
- 運用手順の変更
- アプリケーション，および DD の変更

詳細は，「10.3　旧バージョンから 11-40 までの仕様変更の確認」を参照してください。

(2)　アプリケーションの退避

旧バージョンのアプリケーションサーバで使用していたアプリケーションを退避します。退避の手順を次に示します。

1. J2EE アプリケーション，および J2EE アプリケーション内の EJB-JAR，WAR 属性ファイルを取得します。
2. J2EE リソースアダプタを，実行時情報を含む形式でエクスポートします。
   開始状態の場合は停止してからエクスポートしてください。
3. Connector 属性ファイルを取得します。
   Connector 属性ファイルは，移行後に属性変更が必要になった場合に使用します。
4. データソース属性ファイルを取得します。

5. メール属性ファイルを取得します。

6. サーバ管理コマンドで追加したユーザおよびロールの情報を記録しておきます。

なお，Reliable Messaging を使用している場合は，上記の J2EE リソースアダプタと同様に，Reliable Messaging のリソースアダプタおよび Connector 属性ファイルを退避してください。

(3)　定義などのバックアップ

旧バージョンのアプリケーションサーバで使用していた定義ファイルを退避します。新規インストールの場合，ここで退避したバックアップ情報を基に，新規インストールする環境で使用する定義ファイルなどを作成します。詳細は，「10.6　定義などのバックアップ」を参照してください。なお，adminagent.xml，snapshotlog.conf および snapshotlog.2.conf については，カスタマイズした情報を控えてください。

(4)　全構成ソフトウェアの新規インストール

使用しているアプリケーションサーバの全構成ソフトウェアを新規インストールします。

なお，インストール方法，および注意事項については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「2.2.1　Application Server のインストールについて」を参照してください。

(5)　システムの構築

新規インストールを実施した環境のシステムを構築します。「(3)　定義などのバックアップ」で退避した，運用管理ポータルの［運用管理ドメインの構成定義］画面の情報，または簡易構築定義ファイルの情報を基に，システムを構築します。システムの構築とは，セットアップウィザードの実行，運用管理ポータルの［運用管理ドメインの構成定義］画面での操作，または Smart Composer 機能の cmx_build_system コマンド実行を指します。

システムの構築については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

(6)　定義などの配置

adminagent.xml，snapshotlog.conf および snapshotlog.2.conf 以外の定義ファイルは，旧バージョンのアプリケーションサーバから退避した定義ファイルを，新規インストールする環境に配置します。詳細は，「10.6　定義などのバックアップ」のバックアップを取った情報の設定先に関する説明を参照してください。

これらの定義ファイルの場合は旧バージョンのアプリケーションサーバから退避した定義ファイルに，最新のバージョンの情報を追加します。情報の追加は，「(8)　手動による定義などの修正」で記載している作業になります。

adminagent.xml，snapshotlog.conf および snapshotlog.2.conf の場合は，「(3)　定義などのバックアップ」で控えた情報を基に新規インストールされた定義ファイルに，旧バージョンのアプリケーションサーバでカスタマイズした情報を追加します。

参考

adminagent.xml，snapshotlog.conf および snapshotlog.2.conf は，旧バージョンのアプリケーションサーバで使用していた定義ファイルを上書きして使用しないでください。

(7)　手動による定義などの修正

「(3)　定義などのバックアップ」で退避した情報を基に，手動で定義などを修正します。機能の変更，追加に伴い，互換性を保つためのオプションが提供されています。旧バージョンで使用している機能，使用方法，使用条件によっては，次の作業が必要となります。

- ユーザプロパティファイルなどの定義ファイルの変更
- Java のユーザクラスパスの変更
- 環境変数の変更

旧バージョンで使用している機能，使用方法，使用条件については，「10.3　旧バージョンから 11-40 までの仕様変更の確認」を参照してください。

なお，このマニュアルで説明していない構成ソフトウェアの旧バージョンに関する説明については，次に示すマニュアルを参照してください。

HTTP Server
: マニュアル「HTTP Server」の旧バージョンからの移行に関する注意事項を参照してください。

Web Services - Security
: マニュアル「アプリケーションサーバ Web サービスセキュリティ構築ガイド」の下位バージョンからの移行情報に関する説明を参照してください。

XML Processor
: マニュアル「XML Processor ユーザーズガイド」のバージョン間の差異に関する説明を参照してください。

(8)　論理サーバの設定情報の再読み込みおよび配布

設定した定義情報をシステムへ反映させます。詳細は，「10.10　論理サーバの設定情報の再読み込みおよび配布」を参照してください。

配布したあと，運用管理ポータルから，「(3)　定義などのバックアップ」で控えた論理 J2EE サーバの［JP1 連携の設定］画面で，JP1 連携の再設定をしてください。再設定後は，再度配布してください。

(9)　アプリケーションの設定

アプリケーションを設定します。詳細は，「10.11　アプリケーションの設定（新規インストールの場合）」を参照してください。

なお，Reliable Messaging を使用している場合は，J2EE リソースアダプタの手順と同様に，退避していた Reliable Messaging のリソースアダプタを更新してください。この際，Reliable Messaging を使用するために必要なシステム構築を実施してください。

Reliable Messaging のシステム構築の詳細については，マニュアル「Reliable Messaging」を参照してください。

(10)　移行後の確認

移行後にアプリケーションサーバの動作確認を実施します。

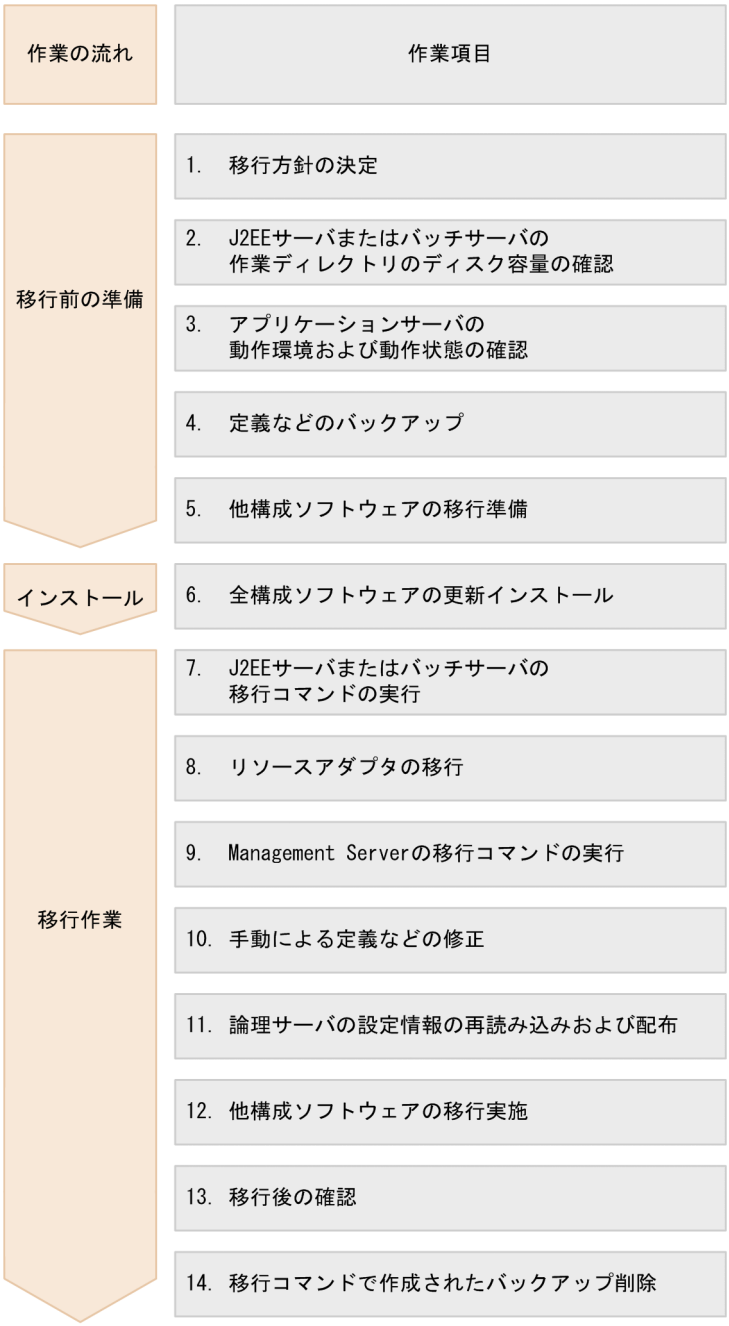
10.2.2　更新インストールの場合

ここでは，旧バージョンのアプリケーションサーバを移行する場合に，旧バージョンのアプリケーションサーバを構築している環境の上に，新規のアプリケーションサーバの環境を構築（更新インストール）する際の手順について示します。

更新インストールについては，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「2.2.1　Application Server のインストールについて」を参照してください。

次の手順に従って，旧バージョンからアプリケーションサーバ 11-40 へ移行してください。

図 10–2　旧バージョンからアプリケーションサーバ 11-40 への移行作業（更新インストールの場合）



(1)　移行方針の決定

詳細は「10.2.1(1)　移行方針の決定」を参照してください。

(2)　J2EE サーバまたはバッチサーバの作業ディレクトリのディスク容量の確認

移行のために十分なディスク容量があるかを確認します。詳細については，「10.4　J2EE サーバまたはバッチサーバの作業ディレクトリのディスク容量の確認」を参照してください。

(3)　アプリケーションサーバの動作環境および動作状態の確認

移行前にアプリケーションサーバの動作環境，動作状態などを確認します。詳細については，「10.5　アプリケーションサーバの動作環境および動作状態の確認」を参照してください。

(4)　定義などのバックアップ

旧バージョンのアプリケーションサーバで使用していた定義ファイルを退避します。詳細については，「10.6　定義などのバックアップ」を参照してください。

(5)　他構成ソフトウェアの移行準備

使用しているアプリケーションサーバの全構成ソフトウェアについて移行の準備をします。

構成ソフトウェアごとの参照先マニュアルについては，「10.1　アプリケーションサーバの移行の概要」の構成ソフトウェアごとの参照先マニュアルに関する説明を参照してください。

なお，ここで準備した情報を基に，「(12)　他構成ソフトウェアの移行実施」で他構成ソフトウェアを移行してください。

(6)　全構成ソフトウェアの更新インストール

使用しているアプリケーションサーバの全構成ソフトウェアを更新インストールします。

なお，インストール方法，および注意事項については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「2.2.1　Application Server のインストールについて」を参照してください。

(7)　J2EE サーバまたはバッチサーバの移行コマンドの実行

定義情報，DD 情報などを継続して利用できるようにするため，J2EE サーバまたはバッチサーバの移行コマンド（cjenvupdate）を実行します。詳細については，「10.7　J2EE サーバまたはバッチサーバの移行コマンドの実行」を参照してください。

(8)　リソースアダプタの移行

旧バージョンのアプリケーションサーバで使用していたリソースアダプタを使用する場合，リソースアダプタの移行処理が必要です。

DB Connector やほかのリソースアダプタの場合は，リソースアダプタの移行コマンド（cjrarupdate コマンド）を実行します。ただし，CJMSP リソースアダプタの場合，リソースアダプタの移行コマンド（cjrarupdate コマンド）は使用できません。サーバ管理コマンドを使用して CJMSP リソースアダプタを入れ替えてください。

(a)　リソースアダプタの移行コマンドの実行

旧バージョンで DB Connector を使用していた場合，またはほかのリソースアダプタを更新インストールする場合，DB Connector やほかのリソースアダプタを入れ替えるためには，リソースアダプタの移行コマンド（cjrarupdate）を実行します。詳細については，「10.8.1　リソースアダプタの移行コマンドの実行」を参照してください。

(b)　CJMSP リソースアダプタの入れ替え

CJMS プロバイダのリソースアダプタ（CJMSP リソースアダプタ）をバージョンアップする場合，リソースアダプタの移行コマンド（cjrarupdate）は使用できません。この場合，サーバ管理コマンドを使用して CJMSP リソースアダプタを入れ替えてください。詳細は，「10.8.2　CJMSP リソースアダプタの入れ替え」を参照してください。

(9)　Management Server の移行コマンドの実行

アプリケーションサーバ Version 8 から移行する場合，Management Server の移行コマンド（mngenvupdate）を実行します。詳細については，「10.9　Management Server の移行コマンドの実行」を参照してください。

(10)　手動による定義などの修正

手動で定義などを修正します。機能の変更，追加に伴い，互換性を保つためのオプションが提供されています。旧バージョンで使用している機能，使用方法，使用条件によっては，次の作業が必要となります。

- ユーザプロパティファイルなどの定義ファイルの変更
- Java のユーザクラスパスの変更
- 環境変数の変更

詳細については，「10.3　旧バージョンから 11-40 までの仕様変更の確認」を参照してください。

(11)　論理サーバの設定情報の再読み込みおよび配布

設定した定義情報をシステムへ反映させます。詳細は，「10.10　論理サーバの設定情報の再読み込みおよび配布」を参照してください。

(12)　他構成ソフトウェアの移行実施

他構成ソフトウェアを移行します。移行の詳細はそれぞれの他構成ソフトウェアのドキュメントを参照してください。なお，「(5)　他構成ソフトウェアの移行準備」で準備した情報を基に移行を実施してください。

(13)　移行後の確認

移行後にアプリケーションサーバの動作確認を実施します。

(14) 移行コマンドで作成されたバックアップ削除

移行コマンドでは，各種バックアップを作成します。移行後の確認が完了したあと，バックアップを削除してください。詳細については，「10.12 移行コマンドで作成されたバックアップの削除」を参照してください。

10.3　旧バージョンから 11-40 までの仕様変更の確認

旧バージョンから 11-40 までの仕様変更について説明します。

10.3.1　アプリケーションサーバ Version 8 から 11-40 までの仕様変更

この項にある表で，アプリケーションサーバ Version 8 から 11-40 までの仕様変更の項目を示します。

アプリケーションサーバ Version 8 から 11-40 に移行する場合，この項にある表の「移行前のアプリケーションサーバのバージョン」列で該当する列を確認してください。「○」のときは仕様変更があるため，「仕様変更の項目」列に記載の参照先を確認してください。「－」のときは仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要です。なお，「仕様変更の項目」列の項番は，「10.3.5　旧バージョンから Version 9 までの仕様変更の一覧」の説明の項番と対応しています。

- アプリケーションに関する仕様変更

表 10–5　アプリケーションサーバ Version 8 から 11-40 までの仕様変更（アプリケーションに関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		08-70	08-53	08-50	08-00
全体	10.3.4　11-40 への移行と V9 互換モードについて	○	○	○	○
	10.3.5(1)　Java SE のバージョン	○	○	○	○
J2EE サーバ	10.3.5(2)　JSP から生成されたサーブレット用コンパイラのバージョン	○	○	○	○
	10.3.5(3)　POST リクエストのフォームデータの最大サイズの制限	－	－	－	○
	10.3.5(23)　javax.servlet.http.HttpServletRequest インタフェースの getSession メソッドでスローされる例外の変更	－	○	○	○
	10.3.5(24)　アプリケーションサーバで実行できるアプリケーションの構成の変更	○	○	○	○
	10.3.5(25)　アプリケーションサーバで実行できる Web アプリケーションの構成の変更	○	○	○	○
	10.3.5(4)　同一 J2EE サーバ内のほかのアプリケーションで動作する Enterprise Bean の呼び出し手順の改善	○	○	○	○
	10.3.5(26)　Portable Global JNDI 対応	○	○	○	○
	10.3.5(43)　MimeUtility API の仕様変更	○	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

－：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

- 構築環境に関する仕様変更

表 10–6　アプリケーションサーバ Version 8 から 11-40 までの仕様変更（構築環境に関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		08-70	08-53	08-50	08-00
全体	10.3.4　11-40 への移行と V9 互換モードについて	○	○	○	○
Management Server	10.3.5(6)　論理サーバの環境変数定義の再設定	○	○	○	○
	10.3.5(7)　JP1 イベントメッセージマッピングファイルの再設定	○	○	○	○
	10.3.5(8)　snapshot ログ収集先対象リストファイルの再設定	○	○	○	○
	10.3.5(9)　OutOfMemoryError 発生時の運用管理エージェントの動作変更	○	○	○	○
	10.3.5(10)　負荷分散機の Cookie スイッチング機能のモード変更	－	－	○	○
	10.3.5(11)　運用管理ポータルで操作対象とするリソースアダプタの変更	－	○	○	○
	10.3.5(12)　Smart Composer 機能で cmx_build_system（-s オプション指定）コマンド実行時の設定情報の配布に関する動作変更	○	○	○	○
	10.3.5(13)　運用管理エージェントおよび Management Server の自動起動の設定方法変更（UNIX の場合）	○	○	○	○
	10.3.5(14)　Management Server のプロセス構成の変更	○	○	○	○
	10.3.5(15)　cmx_list_status コマンドによるサービスユニットの稼働状況のステータス変更	○	○	○	○
	10.3.5(16)　Management Server の機能を使用した明示管理ヒープ機能の自動配置設定ファイルの設定方法の追加	○	○	○	－
	10.3.5(17)　Management Server の移行	○	○	○	○
	10.3.5(18)　WebPasswordJDBCLoginModule 使用時の DB 接続設定の見直し	－	○	○	○
	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	○	○	○	○
J2EE サーバ	10.3.5(27)　TP1 インバウンド連携機能でのコネクション保持のサポート	－	－	○	－
	10.3.5(28)　TP1 インバウンド連携機能での最大同時接続数のデフォルト値の変更	－	－	○	－

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		08-70	08-53	08-50	08-00
	10.3.5(29)　TP1 インバウンド連携機能で使用するスレッド数，ファイルディスクリプタ数，ポート番号の変更	−	−	○	−
	10.3.5(30)　データベースセッションフェイルオーバ機能での完全性保障モードの変更	−	○	○	○
	10.3.5(31)　TP1 インバウンド連携機能の同期点待ち受けポートの追加	−	−	○	○
	10.3.5(32)　データベースセッションフェイルオーバ機能のデータベーステーブルの再作成	○	○	○	○
	10.3.5(33)　データベースセッションフェイルオーバ機能での DB Connector に設定するステートメントプールの数の変更	○	○	○	○
	10.3.5(34)　@EJB アノテーションの mappedName 属性の仕様変更	○	○	○	○
	10.3.5(19)　証明書のインポート	−	−	−	○
	10.3.5(20)　DBConnector_SQLServer_CP.rar の非サポート	○	○	○	○
	10.3.5(21)　DBConnector_SQLServer2005_CP.rar の名称変更	○	○	○	○
	10.3.5(38)　サーバ ID のデフォルト値の変更	○	○	○	○
	10.3.5(41)　server.policy（J2EE サーバ用セキュリティポリシーファイル）への定義の追加	○	○	○	○
	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	○	○	○	○
サーバ管理コマンド	10.3.5(39)　cjimportapp コマンドの-a または-d オプション実行時の重複ディレクトリチェックの変更	○	○	○	○
Web サーバ連携	10.3.5(35)　Microsoft IIS 7.0，7.5 の設定変更	○	○	−	−
パフォーマンストレーサ	10.3.5(22)　PRF トレースバックアップの設定変更	○	○	○	○
JavaVM	10.3.5(36)　明示管理ヒープ機能の自動配置機能で使用する設定ファイルの内容変更	○	○	○	−
	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	○	○	○	○
EJB クライアント	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	○	○	○	○
Web コンテナサーバ	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	○	○	○	○

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		08-70	08-53	08-50	08-00
バッチアプリケーション実行基盤	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	○	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

－：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

- 運用環境に関する仕様変更

表 10–7　アプリケーションサーバ Version 8 から 11-40 までの仕様変更（運用環境に関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		08-70	08-53	08-50	08-00
全体	10.3.4　11-40 への移行と V9 互換モードについて	○	○	○	○
Management Server	10.3.5(42)　Management イベント発行用メッセージ ID の追加	○	○	○	○
J2EE サーバ	10.3.5(40)　J2EE アプリケーション開始時の Java のコンパイルの仕様	○	○	○	○
	10.3.5(5)　J2EE アプリケーション起動時および JSP コンパイル時の Java プログラムのコンパイル処理	○	○	○	○
	10.3.5(38)　サーバ ID のデフォルト値の変更	○	○	○	○
Web サーバ連携	10.3.3(4)　リダイレクタ機能	○	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

－：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

10.3.2　アプリケーションサーバ Version 9 から 11-40 までの仕様変更

この項にある表で，アプリケーションサーバ Version 9 から 11-40 までの仕様変更の項目を示します。

アプリケーションサーバ Version 9 から 11-40 に移行する場合，この項にある表の「移行前のアプリケーションサーバのバージョン」列で該当する列を確認してください。「○」のときは仕様変更があるため，「仕様変更の項目」列に記載の参照先を確認してください。「－」のときは仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要です。なお，「仕様変更の項目」列の項番は，「10.3.5　旧バージョンから Version 9 までの仕様変更の一覧」の説明の項番と対応しています。

- アプリケーションに関する仕様変更

表 10–8　アプリケーションサーバ Version 9 から 11-40 までの仕様変更（アプリケーションに関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン					
		09-87	09-80	09-70	09-60	09-50	09-00
全体	10.3.4　11-40 への移行と V9 互換モードについて	○	○	○	○	○	○
	10.3.5(1)　Java SE のバージョン	−	○	○	○	○	○
J2EE サーバ	10.3.5(2)　JSP から生成されたサーブレット用コンパイラのバージョン	−	−	−	○	○	○
	10.3.5(43)　MimeUtility API の仕様変更	−	−	−	−	−	○
JavaVM	10.3.5(46)　JDK 内部のファイル構成変更	−	−	○	○	○	○
	10.3.5(47)　削除された API／機能／オプション	−	−	○	○	○	○
	10.3.5(48)　バージョン文字列形式の変更	−	−	○	○	○	○
	10.3.5(49)　内部 API のカプセル化	−	−	○	○	○	○
	10.3.5(53)　システムクラスローダの継承関係の変更	−	−	○	○	○	○
	10.3.5(56)　java.io.FileInputStream と java.io.FileOuputStream クラスの finalize メソッドの実装変更	−	○	○	○	○	○
	10.3.5(58)　Java EE 向けモジュールと CORBA モジュールの削除	−	○	○	○	○	○
	10.3.5(59)　クラスファイルバージョンの変更	−	○	○	○	○	○
	10.3.5(60)　javac コマンドの source/target/release オプションに指定できる値	−	○	○	○	○	○
	10.3.5(61)　Java プログラム実行中のシステムプロパティの変更	−	○	○	○	○	○
	10.3.5(62)　3DES 暗号スイートの無効化	−	○	○	○	○	○
	10.3.5(63)　runFinalizersOnExit メソッドの削除	−	○	○	○	○	○
	10.3.5(64)　ThreadPoolExecutor と ScheduledThreadPoolExecutor クラスの finalize メソッドの実装変更	−	○	○	○	○	○
	10.3.5(65)　getAnnotation メソッド呼び出しで発生する例外の変更	−	○	○	○	○	○
	10.3.5(72)　クラスライブラリのスタックトレース出力オプションの対象となる API の変更	−	○	○	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

－：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

- 構築環境に関する仕様変更

表 10–9　アプリケーションサーバ Version 9 から 11-40 までの仕様変更（構築環境に関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン					
		09-87	09-80	09-70	09-60	09-50	09-00
全体	10.3.4　11-40 への移行と V9 互換モードについて	○	○	○	○	○	○
Management Server	10.3.5(6)　論理サーバの環境変数定義の再設定	－	－	－	－	－	○
	10.3.5(7)　JP1 イベントメッセージマッピングファイルの再設定	－	－	－	－	－	○
	10.3.5(8)　snapshot ログ収集先対象リストファイルの再設定	－	－	－	－	－	○
	10.3.5(9)　OutOfMemoryError 発生時の運用管理エージェントの動作変更	－	－	－	－	－	○
	10.3.5(14)　Management Server のプロセス構成の変更	－	－	－	－	－	○
	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	－	－	－	○	○	○
J2EE サーバ	10.3.5(21)　DBConnector_SQLServer2005_CP.rar の名称変更	－	－	－	－	－	○
	10.3.5(38)　サーバ ID のデフォルト値の変更	－	－	－	－	－	○
	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	－	－	－	○	○	○
サーバ管理コマンド	10.3.5(39)　cjimportapp コマンドの-a または-d オプション実行時の重複ディレクトリチェックの変更	－	－	－	－	－	○
JavaVM	10.3.5(36)　明示管理ヒープ機能の自動配置機能で使用する設定ファイルの内容変更	－	－	－	－	－	○
	10.3.5(44)　RSA BSAFE SSL-J の削除	－	－	－	○	－	－
	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	－	－	－	○	○	○
	10.3.5(50)　-Xbootclasspath オプションと-Xbootclasspath/p オプションの削除	－	－	○	○	○	○

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン					
		09-87	09-80	09-70	09-60	09-50	09-00
	10.3.5(51)　推奨標準機構（Endorsed Standards Override Mechanism）の削除	－	－	○	○	○	○
	10.3.5(52)　拡張機能機構（Extensions Mechanism）の削除	－	－	○	○	○	○
	10.3.5(54)　javah コマンドの削除	－	○	○	○	○	○
	10.3.5(55)　-Xprof オプションの削除	－	○	○	○	○	○
	10.3.5(57)　JMX エージェントのパスワード暗号化	－	○	○	○	○	○
	10.3.5(66)　日立拡張機能が提供するファイルの配置場所の変更	－	－	○	○	○	○
	10.3.5(67)　拡張 verbosegc 出力機能のログフォーマット変更（G1GC を使用する場合）	－	－	○	○	○	○
	10.3.5(70)　インストーラ	－	－	○	○	○	○
	10.3.5(71)　更新インストール時に退避するファイルと退避先	－	－	○	○	○	○
EJB クライアント	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	－	－	－	○	○	○
Web コンテナサーバ	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	－	－	－	○	○	○
バッチアプリケーション実行基盤	10.3.5(45)　Permanent 領域から Metaspace 領域への移行	－	－	－	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

－：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

- 運用環境に関する仕様変更

表 10–10　アプリケーションサーバ Version 9 から 11-40 までの仕様変更（運用環境に関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン					
		09-87	09-80	09-70	09-60	09-50	09-00
全体	10.3.4　11-40 への移行と V9 互換モードについて	○	○	○	○	○	○
J2EE サーバ	10.3.5(37)　Servlet 3.0 を使用したアプリケーションのリロードに関する仕様の変更	－	－	－	－	－	○
	10.3.5(38)　サーバ ID のデフォルト値の変更	－	－	－	－	－	○

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン					
		09-87	09-80	09-70	09-60	09-50	09-00
Web サーバ連携	10.3.3(4)　リダイレクタ機能	○	○	○	○	○	○
JavaVM	10.3.5(68)　スレッドダンプ中の JavaVM 内部メモリマップ情報の出力フォーマット変更	−	−	○	○	○	○
	10.3.5(69)　エラーリポートファイル中のメモリ情報の出力フォーマット変更	−	−	○	○	○	○
	10.3.5(73)　G1GC の処理変更に伴う JavaVM ログファイルの出力形式変更	−	−	○	○	○	○
	10.3.5(74)　StackOverflowError	−	−	○	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

−：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

10.3.3　アプリケーションサーバ 11-00，11-10，11-20，または 11-30 から，11-40 までの仕様変更

この項にある表で，アプリケーションサーバ 11-00，11-10，11-20，または 11-30 から，11-40 までの仕様変更の項目を示します。

アプリケーションサーバ 11-00，11-10，11-20，または 11-30 から，11-40 に移行する場合，この項にある表の「移行前のアプリケーションサーバのバージョン」列で該当する列を確認してください。「○」のときは仕様変更があるため，「仕様変更の項目」列に記載の参照先を確認してください。「−」のときは仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要です。

- アプリケーションに関する仕様変更

表 10–11　アプリケーションサーバ 11-00，11-10，11-20，または 11-30 から，11-40 までの仕様変更（アプリケーションに関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		11-30	11-20	11-10	11-00
J2EE サーバ	(2)　アプリケーションサーバの JPA 機能の CDI Managed Bean 対応	−	−	○	○
	(3)　Web コンテナの welcome ファイル転送方式	−	−	○	○
JavaVM※	(6)　sun.nio.cs.map システムプロパティの削除	○	○	○	○
	(7)　javac コマンドの source/target/release オプションに指定できる値	○	○	○	○

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		11-30	11-20	11-10	11-00
	(8)　Java SE のバージョン	○	○	○	○
	(9)　クラスファイルバージョンの変更	○	○	○	○
	(10)　JDK のバージョンによる明示管理ヒープ機能の使用可否	○	○	○	○
	(11)　JSP から生成されたサーブレット用コンパイラのバージョン	○	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

－：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

注※

JDK17 以降を使用する場合に参照してください。

- 構築環境に関する仕様変更

表 10–12　アプリケーションサーバ 11-00，11-10，11-20，または 11-30 から，11-40 までの仕様変更（構築環境に関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		11-30	11-20	11-10	11-00
J2EE サーバ	(1)　アプリケーションサーバの JPA 機能	－	－	－	○
J2EE サーバおよびバッチサーバ	(5)　リソース枯渇監視機能	○	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

－：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

- 運用環境に関する仕様変更

表 10–13　アプリケーションサーバ 11-00，11-10，11-20，または 11-30 から，11-40 までの仕様変更（運用環境に関する仕様変更）

分類	仕様変更の項目	移行前のアプリケーションサーバのバージョン			
		11-30	11-20	11-10	11-00
J2EE サーバ	(1)　アプリケーションサーバの JPA 機能	－	－	－	○
Web サーバ連携	(4)　リダイレクタ機能	－	－	○	○
JavaVM	(12)　GC のデフォルト	○	○	○	○

（凡例）

○：仕様変更があるため，参照先の確認が必要

－：仕様変更がない，または移行前のバージョンにその仕様がないため，参照先の確認は不要

(1) アプリケーションサーバの JPA 機能

アプリケーションサーバ 11-10 以降，アプリケーションサーバの JPA 機能はデフォルトで有効になります。11-00 から移行した場合でも同様です。

11-00 から移行した場合に，11-00 の J2EE サーバで JPA2.1 を利用していたアプリケーションが，メッセージ KDJE56515-E を出力して開始に失敗するときは，アプリケーションサーバの JPA 機能を無効にしてください。アプリケーションサーバの JPA 機能を無効にするには，ユーザプロパティファイルに次のプロパティを設定してください。

```
ejbserver.jpa.disabled=true
```

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(2) アプリケーションサーバの JPA 機能の CDI Managed Bean 対応

11-20 以降では，CDI Managed Bean（それに関連づくインターセプタ含む）からアプリケーションサーバの JPA 機能の使用をサポートします。そのため，CDI Managed Bean で@PersistenceContext および@PersistenceUnit が有効になり，アノテーションによってコンテナ管理の EntityManager と EntityManagerFactory がインジェクトされます。11-10 以前から移行した場合でも同様です。

CDI Managed Bean で@PersistenceContext および@PersistenceUnit を無効にしたい場合は，ユーザプロパティファイルに次のプロパティを設定してください。

```
ejbserver.jpa.cdiEnabled=false
```

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(3) Web コンテナの welcome ファイル転送方式

11-20 以降では，Web コンテナの welcome ファイル転送方式がリダイレクト方式からフォワード方式に変更になります。このため，welcome ファイルをリダイレクトで処理することを期待していたアプリケーションを 11-10 以前から移行する場合，次の動作が異なります。

- welcome ファイル内で指定した相対パスは，welcome ファイルを配置した位置からの相対パスではなく，リクエスト URL からの相対パスとして処理します。
- welcome ファイルを配置した位置からの相対パスとリクエスト URL からの相対パスとが異なる場合，welcome ファイルからの遷移先が 11-10 以前と異なることがあります。

互換性を重視したシステムへ移行する場合に必要な作業

V9 互換モードでは，リダイレクト方式から変更がないため作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

welcome ファイル内で相対パスを指定していて，リクエスト URL からの相対パスとは異なる位置を期待している場合，指定したパスの変更が必要です。

影響を受けるもの

welcome ファイルから他のページへ遷移する場合に，期待したページへ遷移できない場合があります。

(4) リダイレクタ機能

11-20 以降では，リダイレクタ機能を使用するために HTTP Server を起動する際は，環境変数を設定する必要があります。

互換性を重視したシステムへ移行する場合に必要な作業

リダイレクタ機能を使用するために HTTP Server を起動する際は，次の環境変数を設定する必要があります。

Linux の場合

LD_LIBRARY_PATH=/opt/Cosminexus/common/lib

Windows の場合

PATH=<Application Server のインストールディレクトリ>¥common¥lib

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(5) リソース枯渇監視機能

11-40 以降，リソース枯渇監視機能のメモリ枯渇監視情報の Rate2 のアラート出力は，デフォルトで無効になります。11-30 以前から移行した場合でも同様です。

リソース枯渇監視機能のメモリ枯渇監視情報の Rate2 のアラート出力を有効にしたい場合は，ユーザプロパティファイルに次のプロパティを設定してください。

```
ejbserver.watch.memory.rate2alert.enabled=true
```

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(6) sun.nio.cs.map システムプロパティの削除

JDK17 以降の場合，sun.nio.cs.map システムプロパティを使用できません。そのため，エンコーディング名 shift_jis，csshiftjis，ms_kanji，x-sjis を MS932 の別名として実行させることができません。この変更によって文字化けが発生する場合は，アプリケーションを改修してください。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(7) javac コマンドの source/target/release オプションに指定できる値

JDK17 以降の場合，javac コマンドの source オプション，target オプションおよび release オプションに「6」を指定することはできません。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(8) Java SE のバージョン

アプリケーションサーバ 11-40 以降で使用されている Java SE 17 は，Java SE の旧バージョンと，一部の機能に互換性がありません。詳細は，Java SE のドキュメントを参照してください。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(9) クラスファイルバージョンの変更

JDK17 のクラスファイルバージョンは「61.0」です。

また，このクラスファイルバージョンの変更に伴い，クラスファイルフォーマットも変更されています。

互換性を重視したシステムへ移行する場合に必要な作業

クラスファイルの解析などを実施するプログラムやライブラリを使用している場合は，実装の変更やライブラリの更新などが必要となる場合があります。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

ありません。

(10) JDK のバージョンによる明示管理ヒープ機能の使用可否

アプリケーションサーバ 11-40 に同梱されている JDK17 では，明示管理ヒープ機能は非サポートです。明示管理ヒープ機能は，SerialGC における GC 停止時間の長期化を防ぐために使用します。11-40 ではアプリケーションサーバでサポートしている G1GC，または ZGC を使用することで GC 停止時間の長期化を防ぐことができます。継続して明示管理ヒープ機能をご使用になる場合は，JDK11 をご使用ください。

JDK17 の使用時に GC 停止時間の長期化を防ぐときは，G1GC または ZGC を使用してください。どの GC を選択するかによって，スループットや停止時間など，アプリケーションに与える影響が異なります。それぞれの GC の特徴を考慮して，アプリケーションの要件に合う GC を選択してください。それぞれの GC の特徴については，マニュアル「アプリケーションサーバ システム設計ガイド」の「7.1　GC と JavaVM のメモリ管理の概要」を参照してください。

明示管理ヒープ機能の使用状況については，-XX:[+|-]HitachiUseExplicitMemory オプションの指定有無をご確認ください。詳細は，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7.13.1　明示管理ヒープ機能を利用するための共通の設定(JavaVM オプションの設定)」を参照してください。

これまでに明示管理ヒープ機能を使用していて，これから JDK17 を使用する場合，次の対応をしてください。

- 明示管理ヒープ機能の JavaVM オプションを指定している場合，プロセスの起動に失敗します。そのため，明示管理ヒープ機能の JavaVM オプションはすべて削除してください。明示管理ヒープ機能の

JavaVM オプションについては，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7.13.1 明示管理ヒープ機能を利用するための共通の設定(JavaVM オプションの設定)」を参照してください。

- 明示管理ヒープ機能の API を使用している場合，アプリケーションの改修は必要ありません。明示管理ヒープ機能が無効の場合と同じ挙動となります。明示管理ヒープ機能の API については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7.12　明示管理ヒープ機能 API を使った Java プログラムの実装」を参照してください。
- 明示管理ヒープ機能は，Java ヒープとは別の Explicit ヒープという独自の領域を使用しています。これまで明示管理ヒープ機能を使用していた場合，Explicit ヒープに配置されていたオブジェクトは，すべて Java ヒープに配置されることになります。そのため，これまで Explicit ヒープに設定していたヒープサイズ（-XX:HitachiExplicitHeapMaxSize に指定した値）を Java ヒープに加算してください。G1GC，ZGC を使用する場合は，さらにそれぞれの GC で必要な Java ヒープサイズを加算して使用してください。G1GC のチューニングについては，マニュアル「アプリケーションサーバ システム設計ガイド」の「7.16　G1GC のチューニング」，ZGC のチューニングについては，マニュアル「アプリケーションサーバ システム設計ガイド」の「7.18　ZGC のチューニング（JDK17 以降の場合）」を参照してください。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(11)　JSP から生成されたサーブレット用コンパイラのバージョン

JSP ファイルをコンパイルする過程で，中間ファイルとしてサーブレットの Java ソースを生成します。生成されたサーブレットの Java ソースは Java コンパイラによって，最終的に class ファイルを生成します。このとき使用される Java コンパイラの Java 言語仕様のバージョンは，製品のバージョンによって異なります。J2EE サーバでは，インストールされている javac コンパイラのデフォルトバージョンの Java 言語仕様に従って JSP から生成されたサーブレットのソースファイルをコンパイルします。

コンパイルする Java 言語仕様のバージョンを次の表に示します。

表 10–14　コンパイルする Java 言語仕様のバージョン

アプリケーションサーバのバージョン	JDK のバージョン	コンパイル Java 言語仕様バージョン
09-87 から 11-40 より前	JDK11	Java SE 11
11-40	JDK11	Java SE 11
	JDK17	Java SE 17

JSP ファイルのスクリプティングに移行後の Java 言語仕様（JDK11 を使用している場合は Java SE 11，JDK17 を使用している場合は Java SE 17）に反したコーディングをしていて，JSP から生成されたサーブレットのソースファイルがコンパイルできない場合，J2EE サーバのユーザプロパティファイルの webserver.jsp.compile.backcompat キーに，移行前と同じコンパイラの Java 言語仕様のバージョンを指定します。

cjjspc コマンドを使用して JSP 事前コンパイルを実行する場合は，cjjspc コマンドの-source オプションに移行前と同じ値を指定します。

JSP ファイル内にスクリプティングを記述していない場合や，JSP ファイルのスクリプティングが移行後の Java 言語仕様に反さず JSP から生成したサーブレットのソースファイルのコンパイルが正常にできる場合，上記の設定は不要です。

互換性を重視したシステムへ移行する場合に必要な作業

移行前のアプリケーションサーバのコンパイル Java 言語仕様バージョンに合わせて，次の値を設定してください。

J2EE サーバ用のユーザプロパティファイル（usrconf.properties）の webserver.jsp.compile.backcompat キー

- JDK11 を使用している場合：「1.6」「6」「1.7」「7」「1.8」または「8」
- JDK17 を使用している場合：「1.7」「7」「1.8」「8」または「11」

cjjspc コマンドの-source オプション（cjjspc コマンドを使用して JSP 事前コンパイルを実行する場合）

- JDK11 を使用している場合：「1.6」「6」「1.7」「7」「1.8」「8」または「9」
- JDK17 を使用している場合：「1.7」「7」「1.8」「8」「9」または「11」

推奨機能を使用したシステムへ移行する場合に必要な作業

JSP ファイルのスクリプティングに移行後の Java 言語仕様に反したコーディングをしている場合，移行後の Java 言語仕様に従うように JSP ファイルを修正してください。

影響を受けるもの

ありません。

(12) GC のデフォルト

GC のデフォルトは，11-30 以前は SerialGC ですが，11-40 以降は G1GC になります。ただし，論理プロセッサが 2 未満または物理メモリが 1792 メガバイト未満の場合は，自動で SerialGC がデフォルトとして選択されます。

互換性を重視したシステムへ移行する場合に必要な作業

移行前に JDK で GC のオプションを指定していない，かつ 11-40 以降も 11-30 以前と同様に SerialGC を使用する場合は，-XX:+UseSerialGC オプションを指定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

10.3.4　11-40 への移行と V9 互換モードについて

11-40 には「推奨モード」と「V9 互換モード」の 2 つのモードがあり，モードによって使用できる機能が異なります。そのため，11-40 に移行する場合は，使用したい機能を基に，どちらのモードを使用するかを検討してください。

推奨モード

推奨モードでは，11-00 以降の新機能を使用できます。11-00 以降の新機能については，この項の「・11-00 以降の主な機能」を参照してください。

また，09-87 から 11-40 に移行し，「推奨モード」を使用するユーザ向けに，09-87 から 11-40 の変更点を説明する移行ガイドを提供しています。移行ガイドについては，「12.　アプリケーションサーバ 09-87 から 11-40 への移行」を参照してください。

V9 互換モード

V9 互換モードでは，アプリケーションサーバが 09-87 相当の動作になります。11-00 では Java EE 7 の新機能に対応するために大幅な機能の変更をしています。11-40 の新機能を使用しないで，09-87 以前との互換性を重視するユーザは「V9 互換モード」を使用してください。

V9 互換モードでは，09-87 以前との互換性がある機能を使用できます。互換性がある機能の詳細は，この項の「・旧バージョンとの互換のための機能」を参照してください。

なお，「V9 互換モード」の詳細は，マニュアル「アプリケーションサーバ 機能解説 互換編」の「第 2 編　V9 互換モード」を参照してください。

なお，11-00 以降で非サポートとなった機能は，推奨モードと V9 互換モードのどちらでも使用できません。詳細は，この項の「・11-00 以降で非サポートとなった機能」を参照してください。

11-00 以降の主な機能，旧バージョンとの互換のための機能，および 11-00 以降で非サポートとなった機能について説明します。

- 11-00 以降の主な機能

  11-00 以降の主な機能を次の表に示します。この表に記載している機能は，推奨モードでは使用できますが，V9 互換モードでは使用できません。

表 10–15　11-00 以降の主な機能

分類	機能名	マニュアル参照先
Web コンテナ	JSF 2.3 対応	アプリケーションサーバ 機能解説　基本・開発編(Web コンテナ) 「3.　JSF および JSTL の利用」
	JAX-RS 2.1 対応	アプリケーションサーバ 機能解説　基本・開発編(Web コンテナ)

分類	機能名	マニュアル参照先
		「4. JAX-RS 2.1 の利用」
	WebSocket 1.1 対応	アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ) 「5. WebSocket」
	NIO HTTP サーバ機能	アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ) 「7. NIO HTTP サーバ」
	非同期サーブレット対応	アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ) 「8. サーブレットおよび JSP の実装」
	Servlet 4.0 対応	
	JSP 2.3 対応	
コンテナ共通機能	JPA 2.2 対応	アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能) 「6. JPA 2.2 の利用」
	CDI 2.0 対応	アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能) 「9. アプリケーションサーバでの CDI の利用」
	Bean Validation 2.0 対応	アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能) 「10. アプリケーションサーバでの Bean Validation の利用」
	Java Batch 1.0 対応	アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能) 「11. Java Batch」
	JSON-P 1.1 対応	アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能) 「12. JSON-P」
	JSON-B 1.0 対応	アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能) 「13. JSON-B」
	Concurrency Utilities for Java EE 1.0 対応	アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能) 「14. Concurrency Utilities」
	Interceptors 1.2 対応	アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能) 「15. Interceptors」

- 旧バージョンとの互換のための機能

旧バージョンとの互換のための機能を次の表に示します。この表に記載している機能は，推奨モードとV9 互換モードで使用可否が異なります。

表 10-16　旧バージョンとの互換のための機能

分類	機能名	V9 互換モードでの使用可否	推奨モードでの使用可否	推奨モードでの代替機能
Web コンテナ	HTTP レスポンス圧縮機能	○	×	Web アプリケーション側で Filter として実装してください。
	Web サーバ連携機能（リダイレクタ機能）	○	×	HTTP Server のリバースプロキシ機能と，J2EE サーバの NIO HTTP サーバ機能を組み合わせることで代替できます。 ただし，次の機能は代替できません。 • ラウンドロビン方式によるリクエストの振り分け • POST データサイズでのリクエストの振り分け • Web コンテナへのゲートウェイ情報の通知 詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発機能編(Web コンテナ)」 の「7.　NIO HTTP サーバ」，およびマニュアル「HTTP Server」を参照してください。
	インプロセス HTTP サーバ機能	○	×	NIO HTTP サーバ機能で代替できます。 ただし，次の機能は代替できません。 • リダイレクトによるリクエストの振り分け • 有効な HTTP メソッドの制限によるアクセス制御 • エラーページのカスタマイズ • Web コンテナへのゲートウェイ情報の通知 詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発機能編(Web コンテナ)」の「7.　NIO HTTP サーバ」を参照してください。
コンテナ共通機能	JPA コンテナ機能（11-00 の場合）	○	×	次の API を使用してアプリケーション管理の永続化ユニットを取得してください。 • javax.persistence.Persistence クラスの createEntityManagerFactory メソッド 詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発機能編(コンテナ共通機能)」の「6.　JPA 2.2 の利用」を参照してください。
	JPA コンテナ機能（11-10 以降の場合）	○	○	－
	CJPA プロバイダ機能	○	×	JPA 2.2 対応の JPA プロバイダで代替できます。 詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発機能編(コンテナ共通機能)」の「6.　JPA 2.2 の利用」を参照してください。

分類	機能名	V9 互換モードでの使用可否	推奨モードでの使用可否	推奨モードでの代替機能
	クラスタコネクションプール機能	△	△	Oracle RAC の機能で代替できます。
Web サービス機能	JAX-RS 機能（JAX-RS 1.1）	○	×	JAX-RS 2.1 で代替できます。 詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発機能編(Web コンテナ)」の「4. JAX-RS 2.1 の利用」を参照してください。

（凡例）

○：使用できます。

△：使用できますが，互換機能のため推奨しません。

×：使用できません。

- 11-00 以降で非サポートとなった機能

Version 11-00 以降で非サポートとなった機能を次の表に示します。この表に記載している機能は，推奨モードと V9 互換モードのどちらでも使用できません。

表 10–17　11-00 以降で非サポートとなった機能

分類	機能名	代替機能
拡張機能	EADs セッションフェイルオーバー機能	データベースセッションフェイルオーバ機能を使用してください。 詳細は，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「6. データベースセッションフェイルオーバ機能」を参照してください。
互換機能	ベーシックモード	J2EE サーバを使用してください。
	サーブレットエンジンモード (Web コンテナサーバ)	J2EE サーバを使用してください。
	簡易 Web サーバ機能	NIO HTTP サーバ機能を使用してください。 詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発機能編(Web コンテナ)」の「7. NIO HTTP サーバ」を参照してください。
	EJB クライアントアプリケーションのログのサブディレクトリ専有モード	デフォルトのサブディレクトリ共有モードを使用してください。
	メモリセッションフェイルオーバ機能	データベースセッションフェイルオーバ機能を使用してください。 詳細は，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「6. データベースセッションフェイルオーバ機能」を参照してください。
	複数の構築済み実行環境の切り替え	なし

分類	機能名	代替機能
	J2EE アプリケーションのテスト機能	なし
	JSP1.1 仕様および JSP1.2 仕様に準拠した JSP ソースのチェック（cjjsp2java）	なし
	論理サーバのアプリケーション管理の V7 互換モード	なし
セキュリティ管理機能	セッションフェイルオーバ機能を使用したログイン状態の引き継ぎ	データベースセッションフェイルオーバ機能を使用してください。 詳細は，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「6. データベースセッションフェイルオーバ機能」を参照してください。
コマンド	Management Server で使用するコマンド（旧バージョンとの互換用のコマンド） • cmx_define_application コマンド • cmx_deploy_application コマンド • cmx_register_application コマンド • cmx_start_application コマンド • cmx_stop_application コマンド • cmx_undefine_application コマンド • cmx_undeploy_application コマンド • cmx_unregister_application コマンド • cmx_define_resource コマンド • cmx_deploy_resource コマンド • cmx_register_resource コマンド • cmx_start_resource コマンド • cmx_stop_resource コマンド • cmx_undefine_resource コマンド • cmx_undeploy_resource コマンド • cmx_unregister_resource コマンド • cmx_add_serverref コマンド • cmx_delete_serverref コマンド	J2EE サーバのコマンドを使用してください。 詳細は，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「2. J2EE サーバで使用するコマンド」を参照してください。

10.3.5 旧バージョンから Version 9 までの仕様変更の一覧

旧バージョンから Version 9 までの仕様変更を示します。

(1) Java SE のバージョン

09-87 以降で使用されている Java SE 11 は，以前の J2SE 1.4.2，J2SE 5.0，JavaSE 6，Java SE7，Java SE 8，Java SE 9 と，一部の機能に互換性がありません。詳細については，JavaSE のドキュメントを参照してください。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

ありません。

(2) JSP から生成されたサーブレット用コンパイラのバージョン

変更内容

JSP ファイルをコンパイルする過程で，中間ファイルとしてサーブレットの Java ソースを生成します。生成されたサーブレットの Java ソースは Java コンパイラによって，最終的に class ファイルを生成します。このとき使用される Java コンパイラの Java 言語仕様のバージョンは，製品のバージョンによって異なります。J2EE サーバでは，インストールされている javac コンパイラのデフォルトバージョンの Java 言語仕様に従って JSP から生成されたサーブレットのソースファイルをコンパイルします。

コンパイルする Java 言語仕様のバージョンを次の表に示します。

表 10–18 コンパイルする Java 言語仕様のバージョン

アプリケーションサーバのバージョン	JDK のバージョン	コンパイル Java 言語仕様バージョン
Version 8	JDK 5.0	J2SE 5.0
	JDK 6	Java SE 6
09-00 から 09-60 より前	JDK 6	Java SE 6
09-60 から 09-70 より前	JDK 7	Java SE 7
09-70	JDK 8	Java SE 8
09-80	JDK 9	Java SE 9
09-87	JDK 11	Java SE 11

JSP ファイルのスクリプティングに Java SE 8 の言語仕様に反したコーディングをしていて，JSP から生成されたサーブレットのソースファイルがコンパイルできない場合，コンパイラの Java 言語仕様のバージョンを J2EE サーバ，または Web コンテナサーバのユーザプロパティファイルの webserver.jsp.compile.backcompat キーに移行前と同じ値になるように設定します。

cjjspc コマンドを使用して JSP 事前コンパイルを実行する場合は，cjjspc コマンドの-source オプションに移行前と同じ値を指定します。

JSP ファイル内にスクリプティングを記述していない場合や JSP ファイルのスクリプティングが Java SE 8 の言語仕様に反していなくて JSP から生成したサーブレットのソースファイルのコンパイルが正常にできる場合，上記の設定は不要です。

互換性を重視したシステムへ移行する場合に必要な作業

移行前のアプリケーションサーバのコンパイル Java 言語仕様バージョンに合わせて，J2EE サーバ用のユーザプロパティファイル（usrconf.properties）の webserver.jsp.compile.backcompat キーに「1.6」「6」「1.7」「7」「1.8」または「8」を設定してください。

cjjspc コマンドを使用して JSP 事前コンパイルを実行する場合は，同様に移行前のアプリケーションサーバでのコンパイラが受け付けるソースコードのバージョンに合わせて，cjjspc コマンドの-source オプションに 1.6,6,1.7,7,1.8 または 8 を設定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

JSP ファイルのスクリプティングに Java SE 11 の言語仕様に反したコーディングをしている場合，Java SE 11 の言語仕様に従うよう，JSP ファイルを修正してください。

影響を受けるもの

「10.3.5(1)　Java SE のバージョン」

(3)　POST リクエストのフォームデータの最大サイズの制限

変更内容

POST リクエストのフォームデータ（Content-Type ヘッダの値が application/x-www-form-urlencoded の場合の POST データ）の最大サイズがデフォルトで 2MB に制限されます。更新インストールをした場合，すでにセットアップされている J2EE サーバには，POST リクエストのフォームデータの最大サイズを無制限にする設定（webserver.connector.limit.max_post_form_data=-1）が自動的に実施されます。

互換性を重視したシステムへ移行する場合に必要な作業

J2EE サーバ用 usrconf.properties の webserver.connector.limit.max_post_form_data プロパティに-1（無制限）を指定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

J2EE サーバ用 usrconf.properties の webserver.connector.limit.max_post_form_data プロパティに適切な上限値を指定してください。

影響を受けるもの

アプリケーションが POST リクエストを行う際，2MB 以上のフォームデータを送信している場合は実行時エラーになります。

(4)　同一 J2EE サーバ内のほかのアプリケーションで動作する Enterprise Bean の呼び出し手順の改善

変更内容

09-00 以降では，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に次のプロパティを設定した場合，またはこのプロパティの指定を省略した場合，同一 J2EE サーバ内のほかのアプリケーションで動作する Enterprise Bean の呼び出しができます。

パラメタ名

ejbserver.rmi.localinvocation.scope

値

app

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(5) J2EE アプリケーション起動時および JSP コンパイル時の Java プログラムのコンパイル処理

変更内容

バージョンアップに伴い，J2EE サーバの設定ファイルの変更が必要です。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

Java ヒープ使用量が以前のバージョンに比べて約 40MB 増加します。

(6) 論理サーバの環境変数定義の再設定

変更内容

バージョンアップに伴い，adminagent.xml ファイルの変更が必要です。

互換性を重視したシステムへ移行する場合に必要な作業

Management Server を使用している場合，運用管理ポータルの「論理サーバの起動／停止」で使用する環境変数を設定していたときは，新たにインストールされた adminagent.xml ファイルに環境変数を再設定する必要があります。adminagent.xml は次のディレクトリに格納されています。

- Windows の場合
  <製品のインストールディレクトリ>¥manager¥config
- UNIX の場合
  /opt/Cosminexus/manager/config

adminagent.xml での環境変数の設定については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.4　adminagent.xml（運用管理エージェント設定ファイル)」を参照してください。

注意事項

- adminagent.xml はインストール時に上書きされ，既存のファイルは adminagent.xml.bak に退避されます。
- adminagent.xml はバージョンアップによって変更されている可能性があります。adminagent.xml を編集して運用している場合は，新たにインストールされたファイルに，以前に編集した項目を再編集してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

移行で実施する作業を行わないと，「論理サーバの起動／停止」で使用する環境変数の設定が引き継がれません。

(7)　JP1 イベントメッセージマッピングファイルの再設定

変更内容

バージョンアップに伴い，JP1 イベントメッセージマッピングファイルの変更が必要です。

互換性を重視したシステムへ移行する場合に必要な作業

次に示す JP1 イベントメッセージマッピングファイルは，更新インストール時に上書きされないで旧バージョンのファイルが引き継がれます。

- Management Server 用メッセージマッピングファイル
  mserver.jp1event.system.mapping.properties
- J2EE サーバ共通用メッセージマッピングファイル
  manager.jp1event.system.mapping.properties
- J2EE サーバ個別用メッセージマッピングファイル
  manager.<論理サーバ名>.jp1event.system.mapping.properties

JP1 イベントメッセージマッピングファイルをカスタマイズしないで使用していた場合

次の場所に格納されている Management Server 用メッセージマッピングファイルと J2EE サーバ共通用メッセージマッピングファイルを修正しないで，旧バージョンのファイルに上書きしてください。

- Windows の場合
  <製品のインストールディレクトリ>¥manager¥config¥templates
- UNIX の場合
  /opt/Cosminexus/manager/config/templates

JP1 イベントメッセージマッピングファイルをカスタマイズして使用していた場合

次の場所に格納されている Management Server 用メッセージマッピングファイルと J2EE サーバ共通用メッセージマッピングファイルをコピーして，旧バージョンのときと同様にカスタマイズした上で，旧バージョンのファイルに上書きしてください。

- Windows の場合
  <製品のインストールディレクトリ>¥manager¥config¥templates
- UNIX の場合
  /opt/Cosminexus/manager/config/templates

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

移行で実施する作業を行わないと，新しいバージョンで追加されたメッセージに対する JP1 イベントが発行されません。

(8) snapshot ログ収集先対象リストファイルの再設定

変更内容

バージョンアップに伴い，snapshot ログ収集先対象リストの変更が必要です。

互換性を重視したシステムへ移行する場合に必要な作業

snapshot ログを収集する場合に，snapshot ログ収集先対象リストファイル（snapshotlog.conf，snapshotlog.2.conf，および snapshotlog.param.conf）を編集して使用していたときは，新たにインストールされた snapshot ログ収集先対象リストファイルで snapshot ログの収集先を再設定する必要があります。

snapshot ログ収集先対象リストファイルは，次のディレクトリに格納されています。

- Windows の場合
  <製品のインストールディレクトリ>¥manager¥config
- UNIX の場合
  /opt/Cosminexus/manager/config

snapshot ログ収集先対象リストファイルの設定については，「3.3.3(3) snapshot ログの収集先のカスタマイズ」を参照してください。

注意事項

- 更新インストールの場合，snapshot ログ収集先対象リストファイルはバージョンアップ時に上書きされ，既存のファイルは snapshotlog.conf.bak，snapshotlog.2.conf.bak，および snapshotlog.param.conf.bak として退避されます。

- バージョンアップによって snapshot ログの収集先が変更されることがあります。snapshot ログの収集対象を編集して運用している場合は，新たにインストールされたファイルに，以前に編集した項目を再編集してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

移行で実施する作業を行わないと，バージョンアップ前の snapshot ログ収集先対象リストファイルに対するユーザ変更が失われます。

(9) OutOfMemoryError 発生時の運用管理エージェントの動作変更

変更内容

OutOfMemoryError 発生時に保守調査用ログを保護したり，不安定な状態で運用管理エージェントが動作することを防いだりするため，08-50 から運用管理エージェントの動作が変更になります。

- 08-50 より前の動作
  OutOfMemoryError が発生しても処理を続行する。
- 08-50 以降の動作
  OutOfMemoryError が発生したら強制終了する。

旧バージョンと同等の動作にするには，adminagentuser.cfg に次のオプションを追加してください。

add.jvm.arg=-XX:-HitachiOutOfMemoryAbort

互換性を重視したシステムへ移行する場合に必要な作業

運用管理エージェント用オプション定義ファイル（adminagentuser.cfg）で add.jvm.arg=-XX:-HitachiOutOfMemoryAbort を指定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

デフォルト値（OutOfMemoryError 発生時に強制終了）を使用してください。

影響を受けるもの

旧バージョンと同等の動作にする作業を行わないと，OutOfMemoryError が発生したら運用管理エージェントが強制終了されます。

(10) 負荷分散機の Cookie スイッチング機能のモード変更

変更内容

08-53 から，負荷分散機の Cookie スイッチング機能で使用しているスイッチングモードが，ハッシュモードからインサートモードへ変更されます。

互換性を重視したシステムへ移行する場合に必要な作業

08-53 より前に，Smart Composer 機能で BIG-IP v9 の Cookie スイッチング機能を使用してシステムを構築している場合は，cmx_delete_system コマンドを実行してシステムを削除してから，再度，cmx_build_system コマンドを実行してシステムを構築してください。

また，08-53 以降では，簡易構築定義ファイルの<server-id-rule>タグが非推奨となります。<server-id-rule>タグに値を設定しても無視されます。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

移行で実施する作業を行わないと，HTTP リクエストが特定の Web サーバに割り振られなくなる場合があります。

(11) 運用管理ポータルで操作対象とするリソースアダプタの変更

変更内容

08-70 から，運用管理ポータルで操作対象とするリソースアダプタが，デプロイしたリソースアダプタに変更されます。運用管理ポータルでリソースアダプタを削除しても，そのデプロイ元のリソースは削除されません。

互換性を重視したシステムへ移行する場合に必要な作業

更新インストールした環境ではデプロイ元のリソースが残ります。この場合，サーバ管理コマンド（cjdeleteres コマンド）でデプロイ元のリソースを削除してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

移行で実施する作業を行わないと，デプロイ前リソースと同名のリソースアダプタのデプロイができません。

(12) Smart Composer 機能で cmx_build_system（-s オプション指定）コマンド実行時の設定情報の配布に関する動作変更

変更内容

09-00 以降，Smart Composer 機能で-s オプションを指定して cmx_build_system コマンドを実行した場合，各ホストのサーバに設定情報が配布されているかどうかに関係なく，常に設定情報を配布するように動作が変更になります。

- 09-00 より前の動作

  各ホストのサーバに設定情報が配布されていない論理サーバの設定情報を配布する。

- 09-00 以降の動作

各ホストのサーバに設定情報が配布されているかどうかに関係なく，常に論理サーバの設定情報を配布する。

互換性を重視したシステムへ移行する場合に必要な作業

旧バージョンと同等の動作にするには，-s オプションを指定して cmx_build_system コマンドを実行する際には，-sd オプションを指定して実行してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

移行で実施する作業を行わないと各ホストのサーバに設定情報が配布されているかどうかに関係なく，常に論理サーバの設定情報が配布されるようになります。

(13) 運用管理エージェントおよび Management Server の自動起動の設定方法変更（UNIX の場合）

変更内容

UNIX の場合，09-00 以降では，運用管理エージェントおよび Management Server の自動起動の設定方法が変更になります。

09-00 より前に運用管理エージェントまたは Management Server で自動起動を設定している場合に，09-00 以降の設定方法を使用するときは，次の手順で自動起動の設定方法を変更してください。なお，09-00 より前の設定のまま自動起動を運用しても問題はありません。09-00 より前の設定のまま自動起動を運用する場合は，09-00 以降の自動起動の設定は有効にしないでください。

- **運用管理エージェントの自動起動の設定変更手順**

1. 09-00 より前の自動起動の設定を解除します。

   OS ごとに説明します。

   AIX の場合

   rmitab コマンドを使用して，起動スクリプト（/opt/Cosminexus/manager/bin/adminagentctl start）を/etc/inittab ファイルから削除します。

   （指定例）

   ```
   # rmitab adminagentctl
   ```

   AIX 以外の場合

   rm コマンドを使用して，自動起動の設定時に作成したシンボリックリンクを削除します。

   （指定例）

   ```
   # rm /sbin/rc2.d/S800AdminAgent
   ```

2. mngautorun コマンドを使用して，運用管理エージェントの自動起動を設定します。

   （指定例）

   ```
   mngautorun once agent
   ```

- Management Server の自動起動の設定変更手順

1. 09-00 より前の自動起動の設定を解除します。

   OS ごとに説明します。

   AIX の場合

   rmitab コマンドを使用して，起動スクリプト（/opt/Cosminexus/manager/bin/mngsvrctl start）を，/etc/inittab ファイルから削除します。

   （指定例）

   ```
   # rmitab mngsvrctl
   ```

   AIX 以外の場合

   rm コマンドを使用して，自動起動の設定時に作成したシンボリックリンクを削除します。

   （指定例）

   ```
   # rm sbin/rc2.d/S900MngSvr
   ```

2. mngautorun コマンドを使用して，Management Server の自動起動を設定します。

   （指定例）

   ```
   mngautorun once server
   ```

互換性を重視したシステムへ移行する場合に必要な作業

mngautorun コマンドを使用して自動起動設定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

mngautorun コマンドによる運用管理エージェントおよび Management Server の自動起動を有効した場合，09-00 より前の自動起動の設定と mngautorun コマンドによる設定が有効となってしまうため，マシン起動時にどちらかの自動起動処理がエラーとなります。

(14) Management Server のプロセス構成の変更

変更内容

09-00 から Management Server のプロセス構成が変更になります。

- 09-00 より前のプロセス構成

  mngsvrctl プロセスおよび cjstartweb プロセス

- 09-00 以降のプロセス構成

  cjstartsv プロセス

Management Server のプロセスや，Management Server の起動時に出力されるメッセージを監視する運用をしている場合は，監視対象を変更する必要があります。

互換性を重視したシステムへ移行する場合に必要な作業

- スクリプトファイルなどで mngsvrctl プロセスを監視している場合は，cjstartsv プロセスを監視するように変更してください。
- Management Server の起動失敗時にコンソールに出力されていた KEOS10112-E メッセージが出力されないことがあります。mngsvrctl コマンドの戻り値が 0 以外の場合は，「<Manager のログ出力ディレクトリ>¥cjmessage[n].log」，または「<Manager のログ出力ディレクトリ>/cjmessage[n].log」を参照してエラーの内容を確認してください。

  また，Management Server の起動成功時にコンソールに出力されていた KEOS10110-I メッセージが出力されません。Management Server が起動されたかどうかは，次のどちらかの方法で確認してください。

1. KEOS10019-I メッセージが，次に示す Management Server のログに出力されているかどうかを確認する。

   Windows の場合：<Manager のログ出力ディレクトリ>¥mngsvr[n].log または<Manager のログ出力ディレクトリ>¥message¥mngmessage[n].log

   UNIX の場合：<Manager のログ出力ディレクトリ>/mngsvr[n].log または<Manager のログ出力ディレクトリ>/message/mngmessage[n].log

2. mngsvrutil コマンドのサブコマンド check を使用して，Management Server が稼働しているかどうかを確認する。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

監視システムなどで Management Server の稼働監視を行っている場合，不当に異常を検知したり，異常を検知できなかったりする場合があります。

(15) cmx_list_status コマンドによるサービスユニットの稼働状況のステータス変更

変更内容

09-00 から，Smart Composer 機能で cmx_list_status コマンド実行時に取得されるサービスユニットの稼働状況のステータスが，「unknown」から「no working」と「no ready」に変更となります。

互換性を重視したシステムへ移行する場合に必要な作業

監視システムなどで「unknown」を参照している場合は，「no working」または「no ready」のどちらかを参照するように変更してください。「no working」および「no ready」については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cmx_list_status（サービスユニット状況の表示)」を参照してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

監視システムなどでサービスユニットの稼働状況を監視している場合，不当に異常を検知したり，異常を検知できなかったりする場合があります。

(16) Management Server の機能を使用した明示管理ヒープ機能の自動配置設定ファイルの設定方法の追加

変更内容

09-00 から，明示管理ヒープ機能の自動配置設定ファイルを，Management Server の機能を使用して設定できます。

互換性を重視したシステムへ移行する場合に必要な作業

09-00 より前に設定した明示管理ヒープ機能の自動配置設定ファイルを使用する場合は次の作業を実施してください。

1. 明示管理ヒープ機能の自動配置設定ファイルのリネームなどを実施し，<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<実サーバ名>/auto_explicit_memory.cfg に配置します。
2. 明示管理ヒープ機能の自動配置設定ファイルを使用している論理 J2EE サーバの usrconf.cfg 内の-XX:HitachiAutoExplicitMemoryFile オプションに，<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<実サーバ名>/auto_explicit_memory.cfg を指定します。
3. 運用管理ポータルにログインして，［運用管理ポータル］画面で「論理サーバの環境設定」をクリックします。
4. 運用管理ドメインに構成定義された既存の論理 J2EE サーバに対して，［サーバの設定読み込み］画面で「接続先ホストから設定を読み込みます」をチェックして［読み込み］ボタンをクリックします。
5. 論理 J2EE サーバの定義する画面で［適用］ボタンをクリックしたあと，［設定情報の配布］画面で再配布します。

> **参考**
>
> mngsvrutil コマンドでも接続先ホストからの読み込みを実行できます。mngsvrutil コマンドで実行する場合は，サブコマンド「reload」で引数に「env」を指定します。mngsvrutil コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「7.3 mngsvrutil コマンドのサブコマンドの詳細」を参照してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

移行で実施する作業を行わないと明示管理ヒープ機能の自動配置設定ファイルの設定は従来どおり手動での運用となります。

(17) Management Server の移行

変更内容

09-00 以降，Management Server が使用する Web コンテナが変更されます。

互換性を重視したシステムへ移行する場合に必要な作業

アプリケーションサーバ Version 8 から移行する場合，Management Server の移行コマンド（mngenvupdate）を実行します。移行方法については，「10.9 Management Server の移行コマンドの実行」を参照してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

移行で実施する作業を行わないと，バージョンアップ前の Management Server 用オプション定義ファイル（mserver.cfg）への設定が引き継がれない場合があります。アプリケーション管理機能・統合ユーザ管理機能を使用していた場合は正常に動作しなくなります。

また，新規インストールの場合，Management Server 用オプション定義ファイル（mserver.cfg）に指定できるキーが Web コンテナサーバ用オプション定義ファイル（usrconf.cfg）から J2EE サーバ用オプション定義ファイル（usrconf.cfg）へ変更されたことを考慮して，mserver.cfg を設定する必要があります。

(18) WebPasswordJDBCLoginModule 使用時の DB 接続設定の見直し

変更内容

DABroker を使用した接続が，サポート外になりました。08-70 以降，データベース製品が提供する JDBC ドライバを使用してください。なお，HiRDB の場合は Type4 ドライバを，Oracle の場合は Thin ドライバを使用してください。

互換性を重視したシステムへ移行する場合に必要な作業

J2EE サーバ用 usrconf.cfg の add.class.path キーに，JDBC ドライバの絶対パスを追加してください。また，JDBC ドライバのマニュアルに従い，ua.conf の com.cosminexus.admin.auth.jdbc.driver パラメタにドライバのクラス名，com.cosminexus.admin.auth.jdbc.conn.url パラメタに接続 URL を指定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

移行で実施する作業を行わないと WebPasswordJDBCLoginModule が使用できません。

(19)　証明書のインポート

変更内容

08-50 以降では，cacerts 証明書ファイルを同梱していません。証明書が必要な場合は，個別に入手してインポートする必要があります。

インポート手順の詳細は，次に示す Java SE のドキュメントを参照してください。

Windows の場合

http://docs.oracle.com/javase/jp/6/technotes/tools/windows/keytool.html

UNIX（Linux，AIX）の場合

http://docs.oracle.com/javase/jp/6/technotes/tools/solaris/keytool.html

互換性を重視したシステムへ移行する場合に必要な作業

必要な証明書を入手してインポートしてください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

「10.3.5(1)　Java SE のバージョン」

(20)　DBConnector_SQLServer_CP.rar の非サポート

変更内容

09-00 以降では 08-70 以前の「DBConnector_SQLServer_CP.rar」が非サポートになります。

「DBConnector_SQLServer2005_CP.rar」は 08-70 以前の「DBConnector_SQLServer_CP.rar」と互換性がないため，cjrarupdate コマンドによる自動での移行処理はできません。このため，手動で 08-70 以前の「DBConnector_SQLServer_CP.rar」の削除，
「DBConnector_SQLServer2005_CP.rar」のインポートおよびプロパティ設定を実施する必要があります。

互換性を重視したシステムへ移行する場合に必要な作業

08-70 以前の「DBConnector_SQLServer_CP.rar」の削除，
「DBConnector_SQLServer2005_CP.rar」のインポートおよびプロパティ設定※を実施してください。

注※　08-70 以前と 09-50 以降の「DBConnector_SQLServer_CP.rar」では一部設定できるプロパティが異なります。09-50 以降で設定できるプロパティについては，マニュアル「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)」の「4.　リソースの設定で使用する属性ファイル」を参照してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(21) DBConnector_SQLServer2005_CP.rar の名称変更

変更内容

09-00 以前の「DBConnector_SQLServer2005_CP.rar」は，09-50 以降では「DBConnector_SQLServer_CP.rar」に名称が変更になります。09-50 以降の「DBConnector_SQLServer_CP.rar」は 08-70 以前の「DBConnector_SQLServer_CP.rar」と互換性がないため，cjrarupdate コマンドによる自動での移行処理はできません。このため，手動で 08-70 以前の「DBConnector_SQLServer_CP.rar」の削除，09-50 以降の「DBConnector_SQLServer_CP.rar」のインポートおよびプロパティ設定を実施する必要があります。

互換性を重視したシステムへ移行する場合に必要な作業

08-70 以前の「DBConnector_SQLServer_CP.rar」の削除，09-50 以降の「DBConnector_SQLServer_CP.rar」のインポートおよびプロパティ設定※を実施してください。

注※　08-70 以前と 09-50 以降の「DBConnector_SQLServer_CP.rar」では一部設定できるプロパティが異なります。09-50 以降で設定できるプロパティについては，マニュアル「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)」の「4. リソースの設定で使用する属性ファイル」を参照してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(22) PRF トレースバックアップの設定変更

変更内容

09-00 以降は，PRF デーモンの起動時，および停止時に PRF トレースをバックアップしなくなります。

互換性を重視したシステムへ移行する場合に必要な作業

09-00 より前のバージョンのように PRF デーモンをバックアップするには，cprfstart コマンドに-PrfNoBackUp 0 オプションを指定してください。詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cprfstart（PRF デーモンの開始)」を参照してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(23) javax.servlet.http.HttpServletRequest インタフェースの getSession メソッドでスローされる例外の変更

変更内容

web.xml に指定したエラーページ内で，データベースセッションフェイルオーバ機能の抑止によってデータベースセッションフェイルオーバ機能が無効となったリクエストに対して，javax.servlet.http.HttpServletRequest インタフェースの getSession()メソッド，または getSession(boolean create)メソッドを呼び出した時にスローされる例外は，バージョンごとに異なります。バージョンごとにスローされる例外を次に示します。

- 08-53 以前

  com.hitachi.software.web.dbsfo.DatabaseAccessException
- 08-70 以降

  com.hitachi.software.web.dbsfo.SessionOperationException

互換性を重視したシステムへ移行する場合に必要な作業

08-70 以降で 08-53 以前と同じ例外をスローさせる場合は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に次のパラメタを設定する必要があります。

パラメタ名

webserver.dbsfo.exception_type_backcompat

値

true

推奨機能を使用したシステムへ移行する場合に必要な作業

業務アプリケーションでキャッチする例外を com.hitachi.software.web.dbsfo.DatabaseAccessException から com.hitachi.software.web.dbsfo.SessionOperationException へ変更してください。

影響を受けるもの

なし。

(24) アプリケーションサーバで実行できるアプリケーションの構成の変更

変更内容

09-00 以降では，J2EE アプリケーションの構成として，JavaEE 6.0 の application.xml と，EJB 3.1 の ejb-jar.xml をサポートしました。

そのため，DD を省略した場合に適用されるバージョンが，次のとおり変更されます。

- application.xml を省略した J2EE アプリケーションは，JavaEE 6 と認識されます。
- application.xml を省略した J2EE アプリケーションに含まれる，ejb-jar.xml を省略した EJB コンポーネントは EJB 3.1 と認識されます。

互換性を重視したシステムへ移行する場合に必要な作業

ejb-jar.xml を省略した EJB コンポーネントを EJB 3.0 として扱いたい場合は，JavaEE 5 仕様に準じた application.xml を J2EE アプリケーションに含めるか，08-70 以前でエクスポートした実行時情報付きの J2EE アプリケーションをインポートしてください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(25) アプリケーションサーバで実行できる Web アプリケーションの構成の変更

変更内容

09-00 以降では，J2EE アプリケーションの構成として，Servlet 3.0 の web.xml をサポートしました。

そのため，application.xml を省略した J2EE アプリケーションに含まれる，web.xml を省略した Web アプリケーションは Servlet 3.0 と認識されます。

互換性を重視したシステムへ移行する場合に必要な作業

web.xml を省略した Web アプリケーションを Servlet 2.5 として扱いたい場合は，JavaEE 5 仕様に準じた application.xml を J2EE アプリケーションに含めるか，08-70 以前でエクスポートした実行時情報付きの J2EE アプリケーションをインポートしてください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(26) Portable Global JNDI 対応

変更内容

09-00 以降では，Java EE 6 の新機能である Portable Global JNDI をサポートします。そのため，アプリケーションの開始時に各アプリケーションやアプリケーション内の各モジュールに対して，Portable Global JNDI 名として使用する標準アプリケーション名（application-name）と標準モジュール名（module-name）が割り当てられ，ネーミングサービスに登録されます。また，EJB のリファレンスなど標準仕様で Portable Global JNDI 名が決まっているものに対しても，自動的に Portable Global JNDI 名でリソースが登録されます。

09-00 以降で Portable Global JNDI 名の登録機能が有効になっている場合，08-70 以前の既存のアプリケーションで，標準アプリケーション名，標準モジュール名，Enterprise Bean 名が重複する場合や，Portable Global JNDI 名として使用できない文字が含まれている場合は，KDJE477xx-W が出力される場合があります。KDJE477xx-W が出力されていてもアプリケーションの開始は続行されますので，Portable Global JNDI 名を使用していないアプリケーションの場合は問題ありません。

J2EE サーバ全体で Portable Global JNDI 名の登録機能を無効にしたい場合は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に次のプロパティを設定する必要があります。

パラメタ名

ejbserver.jndi.global.enabled

値

false

更新インストール時，すでにセットアップされている J2EE サーバに対しては，このプロパティが自動的に設定されます。

互換性を重視したシステムへ移行する場合に必要な作業

J2EE サーバ用 usrconf.properties の ejbserver.jndi.global.enabled プロパティに false を指定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

Portable Global JNDI 機能を使ってアプリケーションを開発する場合は，J2EE サーバ用 usrconf.properties の ejbserver.jndi.global.enabled プロパティをデフォルト値（有効）で使用してください。アプリケーション開始時に警告メッセージが出力される場合は，メッセージに従い名前の重複を解消してください。

使わない場合は false を指定してください。

影響を受けるもの

互換機能を利用した場合は業務アプリケーション，およびユーザ運用への影響はありませんが，Portable Global JNDI 機能を使ったアプリケーションをエンハンスできません。推奨機能を利用した場合，既存の業務アプリケーションが正常に動作しなくなることがあります。

(27) TP1 インバウンド連携機能でのコネクション保持のサポート

変更内容

08-53 以降の TP1 インバウンド連携機能では，OpenTP1 とのコネクションを保持できるようになりました。

互換性を重視したシステムへ移行する場合に必要な作業

TP1 インバウンドアダプタで使用する Connector 属性ファイルの rpc_close_after_send プロパティに"true"を指定して，OpenTP1 の rpc_close_after_send 句に"Y"を指定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

TP1 インバウンドアダプタで使用する Connector 属性ファイルの rpc_close_after_send プロパティに"false"を指定して，OpenTP1 の rpc_close_after_send 句を削除してください。

影響を受けるもの

推奨機能を使用することで，コネクションの接続と切断に伴うボトルネックを削減でき，通信効率を向上できます。

(28) TP1 インバウンド連携機能での最大同時接続数のデフォルト値の変更

変更内容

08-53 以降の TP1 インバウンド連携機能では，RPC 受信コネクションの最大同時接続数（Connector 属性ファイルの max_connections プロパティ）のデフォルト値が 10 から 64 に変更になりました。

互換性を重視したシステムへ移行する場合に必要な作業

デフォルト値（10）を使用している場合，移行後，設定値として 10 を設定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

最大同時接続数が 64 になります。関連機能の最大同時接続数を見直してください。

影響を受けるもの

なし

(29) TP1 インバウンド連携機能で使用するスレッド数，ファイルディスクリプタ数，ポート番号の変更

変更内容

08-53 以降の TP1 インバウンド連携アダプタでは，使用するスレッド数，ファイルディスクリプタ数，およびポート番号が 08-50 と異なります。08-50 から 08-53 以降のバージョンに移行する場合は，再度リソースを見積もり直してください。

互換性を重視したシステムへ移行する場合に必要な作業

再度，リソース見積もりを行ってください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし

(30) データベースセッションフェイルオーバ機能での完全性保障モードの変更

変更内容

08-70 以降のデータベースセッションフェイルオーバ機能では，完全性保障モードを無効にする機能をサポートしました。完全性保障モードを無効にすると，同一セッション ID を持つリクエスト処理を同時に実行できるようになります。

08-53 以前は，完全性保障モードは常に有効でしたが，08-70 以降ではデフォルトで無効になります。

互換性を重視したシステムへ移行する場合に必要な作業

簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に次のプロパティを設定する必要があります。

パラメタ名

webserver.dbsfo.integrity_mode.enabled

値

true

推奨機能を使用したシステムへ移行する場合に必要な作業

現状のシステムが，完全性保証モード「無効」が適用可能か確認してください。「無効」は，冗長化された複数の J2EE サーバから同じセッション ID のグローバルセッション情報を同時に更新するような処理が発生しないことを前提としています。

「無効」への移行が可能な場合，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に次のプロパティを設定してください。

パラメタ名

webserver.dbsfo.integrity_mode.enabled

値

false

影響を受けるもの

なし。

(31) TP1 インバウンド連携機能の同期点待ち受けポートの追加

変更内容

08-53 以降の TP1 インバウンド連携機能では，同期点待ち受けポートが追加になりました。デフォルトは 23900 です。

互換性を重視したシステムへ移行する場合に必要な作業

別の機能で 23900 を使用している場合，同期点待ち受けポートのポート番号を変更してください。また，複数の TP1 インバウンドアダプタを使用している場合は，cjrarupdate コマンド実行後に，同期点待ち受けポートにそれぞれ別のポート番号を設定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(32) データベースセッションフェイルオーバ機能のデータベーステーブルの再作成

変更内容

08-70 以降のデータベースセッションフェイルオーバ機能では，データベースのアプリケーション情報テーブルとセッション情報格納テーブルのテーブル構成が 08-53 以前のテーブル構成と異なります。

互換性を重視したシステムへ移行する場合に必要な作業

08-70 以降のバージョンに移行する場合は，08-70 以降のデータベースセッションフェイルオーバ機能が提供するテーブル削除用 SQL を使用してテーブルを削除してから，テーブル作成用 SQL を使用してテーブルを作成してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(33) データベースセッションフェイルオーバ機能での DB Connector に設定するステートメントプールの数の変更

変更内容

08-70 以降のデータベースセッションフェイルオーバ機能では，DB Connector に設定するステートメントプールの数が変更となります。Connector 属性ファイルに次のプロパティを設定してください。

プロパティ名

PreparedStatementPoolSize

値

30×J2EE サーバ内のデータベースセッションフェイルオーバ機能を使用する Web アプリケーション数

互換性を重視したシステムへ移行する場合に必要な作業

Connector 属性ファイルに次のプロパティを設定してください。

プロパティ名

PreparedStatementPoolSize

値

30×J2EE サーバ内のデータベースセッションフェイルオーバ機能を使用する Web アプリケーション数

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(34) @EJB アノテーションの mappedName 属性の仕様変更

変更内容

08-70 以前では，@EJB アノテーションの mappedName 属性は無視されていましたが，09-00 以降では，mappedName 属性への GlobalJNDI 名（Portable Global JNDI 名，EJB の別名，物理名）の記述をサポートします。

互換性を重視したシステムへ移行する場合に必要な作業

08-70 以前で動作していたアプリケーションで，mappedName 属性の指定がある場合，mappedName 属性を削除してください。mappedName 属性が残っていると，09-00 以降に移行したことで，従来無視されていた mappedName 属性の値が有効になり，アプリケーションの動作が変わってしまうおそれがあります。

推奨機能を使用したシステムへ移行する場合に必要な作業

09-00 以降の@EJB アノテーションの mappedName 属性には，DI 対象の EJB の Portable Global JNDI 名，または EJB の別名を指定できます。

影響を受けるもの

ユーザアプリケーション（@EJB の mappedName 属性を指定していた場合だけ影響します）。

(35) Microsoft IIS 7.0，7.5 の設定変更

変更内容

09-00 以降のリダイレクタは，64bit アプリケーションになりました。

互換性を重視したシステムへ移行する場合に必要な作業

Microsoft IIS 7.0 または Microsoft IIS 7.5 を使用して Web サーバ連携する場合，Microsoft IIS のアプリケーションプールの詳細設定で[32bit アプリケーションの有効化]に false を指定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(36) 明示管理ヒープ機能の自動配置機能で使用する設定ファイルの内容変更

変更内容

09-50 以降では，明示管理ヒープ機能の自動配置機能で使用する設定ファイル（自動配置設定ファイル）で，Hibernate に関する設定項目が削除され，初期設定時の内容が変更になりました。

互換性を重視したシステムへ移行する場合に必要な作業

自動配置設定ファイルに次の内容を記述してください。記述方法については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7.13.2　自動配置設定ファイルを使った明示管理ヒープ機能の使用」を参照してください。

```
org.hibernate.cfg.Configuration.buildSessionFactory(), org.hibernate.impl.SessionFactoryI
mpl
```

推奨機能を使用したシステムへ移行する場合に必要な作業

Java ヒープ領域および Explicit ヒープ領域のメモリサイズを再チューニングしてください。チューニング方法については，マニュアル「アプリケーションサーバ システム設計ガイド」の「7. JavaVM のメモリチューニング」を参照してください。

影響を受けるもの

アプリケーションサーバと Hibernate の両方を動作させていた場合に，09-50 より前から移行すると，Java ヒープ領域および Explicit ヒープ領域の使用サイズが変化することがあります。

(37) Servlet 3.0 を使用したアプリケーションのリロードに関する仕様の変更

変更内容

09-50 より前では，Servlet 3.0 を使用したアプリケーションで，次に示すメソッドを削除することで設定を解除した場合，リロードの際にその設定が反映されません。

- SessionCookieConfig.setName(String name)
- SessionCookieConfig.setDomain(String domain)
- SessionCookieConfig.setPath(String path)
- SessionCookieConfig.setComment(String comment)
- SessionCookieConfig.setHttpOnly(boolean httpOnly)
- SessionCookieConfig.setSecure(boolean secure)
- SessionCookieConfig.setMaxAge(int maxAge)
- ServletContext.setSessionTrackingModes(Set<SessionTrackingMode> sessionTrackingModes)
- ServletContext.setInitParameter(String name, String value)

09-50 以降では，リロードの際に設定が反映されるようになりました。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(38) サーバ ID のデフォルト値の変更

変更内容

09-50 以降，サーバ ID のデフォルト値は 64 バイト固定です。サーバ ID を明示的に指定するには，usrconf.properties に次のプロパティを指定してください。

- webserver.session.server_id.value
- webserver.container.server_id.value

usrconf.properties については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.3　usrconf.properties（J2EE サーバ用ユーザプロパティファイル)」を参照してください。

互換性を重視したシステムへ移行する場合に必要な作業

サーバ ID がデフォルト値の場合，サーバ ID を明示的に指定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

サーバ ID がデフォルト値の場合，セッション ID の文字列長が増加します。

セッション ID の長さによって問題が起こらないかどうかを確認してください。

(39) cjimportapp コマンドの-a または-d オプション実行時の重複ディレクトリチェックの変更

変更内容

cjimportapp コマンドで-a または-d オプションを指定して実行した際のディレクトリ重複チェックの仕様が，次のとおり変更となります。

09-00 以前の場合

cjimportapp コマンドで-a または-d オプションを指定したディレクトリと同じディレクトリを持つ J2EE アプリケーションが J2EE サーバ内に存在する場合，KDJE42325-E を出力してインポートに失敗します。

09-50 以降の場合

- cjimportapp コマンドで-a または-d オプションを指定したディレクトリと同じディレクトリを持つ J2EE アプリケーションが J2EE サーバ内に存在する場合，KDJE42325-E を出力してインポートに失敗します。
- cjimportapp コマンドで-a または-d オプションを指定したディレクトリの上位ディレクトリを持つ J2EE アプリケーションが J2EE サーバ内に存在する場合，KDJE42406-E を出力してインポートに失敗します。
- cjimportapp コマンドで-a または-d オプションを指定したディレクトリの下位ディレクトリを持つ J2EE アプリケーションが J2EE サーバ内に存在する場合，KDJE42410-E を出力してインポートに失敗します。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(40) J2EE アプリケーション開始時の Java のコンパイルの仕様

変更内容

J2EE アプリケーション開始時の Java のコンパイルは，javac コマンドではなく J2EE サーバプロセス内で行われるようになります。

互換性を重視したシステムへ移行する場合に必要な作業

J2EE サーバのヒープサイズの見積りを実施してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(41) server.policy（J2EE サーバ用セキュリティポリシーファイル）への定義の追加

変更内容

新規インストールした環境へ移行する場合，server.policy に定義を追加する必要があります。

更新インストールの場合は，移行コマンドを実行すると自動で定義が追加されます。

互換性を重視したシステムへ移行する場合に必要な作業

バージョンごとに，追加が必要な定義を次に示します。なお，説明中の括弧内の数字は，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.4　server.policy（J2EE サーバ用セキュリティポリシーファイル)」の記述例に関する説明を参照してください。

- 08-50 で追加された定義

```
// (3)
// Grant all permissions to anything loaded from the
// EJB server itself
grant codeBase "file:${cosminexus.home}/jaxws/lib/*" {
permission java.security.AllPermission;
};
```

- 08-53 で追加された定義

```
// (6)
// Grant permissions to resource adapters
grant codeBase "file:${ejbserver.http.root}/ejb/${ejbserver.serverName}/rarjars/-" {
// For Cosminexus SOA FTP Inbound Adapter
permission java.lang.RuntimePermission "getClassLoader";
permission java.lang.RuntimePermission "setContextClassLoader";
permission java.lang.RuntimePermission "accessDeclaredMembers";
};
```

- 08-70 で追加された定義

```
// (7)
// Grant permissions to JSP/Servlet
grant codeBase "file:${ejbserver.http.root}/web/${ejbserver.serverName}/-" {
permission javax.security.auth.AuthPermission "createLoginContext.{name}";
permission javax.security.auth.AuthPermission "getSubject";
// (9)
// Grant permissions to custom login modules
//
grant codeBase "file:${cosminexus.home}/manager/modules/-" {
permission java.io.FilePermission "<<ALL FILES>>", "read";
permission javax.security.auth.AuthPermission "modifyPrincipals";
permission javax.security.auth.AuthPermission "modifyPublicCredentials";
};
```

- 09-00 で追加された定義

```
// (3)
// Grant all permissions to anything loaded from the
// EJB server itself
grant codeBase "file:${cosminexus.home}/jaxrs/lib/*" {
permission java.security.AllPermission;
};
grant codeBase "file:${ejbserver.install.root}/weld/lib/*" {
permission java.security.AllPermission;
};
grant codeBase "file:${cosminexus.home}/common/lib/*" {
permission java.security.AllPermission;
```

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(42) Management イベント発行用メッセージ ID の追加

変更内容

Management イベント発行用メッセージ ID を追加しました。

互換性を重視したシステムへ移行する場合に必要な作業

追加されたメッセージ ID でイベントが発行されないよう，追加されたメッセージ ID の前に「-」（ハイフン）を付けたメッセージ ID を Management イベント発行用メッセージ ID リストファイルに追加してください。

追加された Management イベント発行用メッセージ ID をバージョンごとに示します。

表 10–19　追加された Management イベント発行用メッセージ ID

追加されたアプリケーションサーバのバージョン	機能	メッセージ ID
09-00	Web コンテナ単位の全体実行待ちリクエスト数の監視	KDJE53862-W KDJE53863-I KDJE53864-W KDJE53865-I KDJE53866-W KDJE53867-I KDJE53868-W KDJE53869-I

Management イベント発行用メッセージ ID リストファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.12　Management イベント発行用メッセージ ID リストファイル」を参照してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(43) MimeUtility API の仕様変更

変更内容

javax.mail.internet.MimeUtility クラスの encode メソッドでエンコード方式に"base64"を用いた場合，Component Container のバージョンおよび戻り値として得た OutputStream に対する close メソッドの実行有無によって，末尾への改行（"¥r¥n"）付与の有無が次のように異なります。

Component Container のバージョン	OutputStream.close メソッドの実行の有無	末尾への改行付与
09-00 以前	有	付与しない
	無	付与しない
09-50 以降	有	付与する
	無	付与しない

互換性を重視したシステムへ移行する場合に必要な作業

MimeUtility.encode メソッドの戻り値として得た OutputStream の末尾に不要な改行が存在する場合があります。不要な改行が存在する場合は，改行文字を削除してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

次の条件がすべて重なるアプリケーションに影響がありますので，上記の必要な作業を実施してください。

- MimeUtility.encode メソッドを使用して Base64 エンコードを実行している。
- 戻り値として得られる OutputStream で close メソッドを実行している。

(44) RSA BSAFE SSL-J の削除

変更内容

JDK 6 をベースとしているアプリケーションサーバ Version 9 と 9.5 では，TLSv1.2 による SSL/TLS 通信を目的として，RSA BSAFE SSL-J(以降，SSL-J)を Cosminexus Developer's Kit for Java(TM) に同梱していました。JDK 7 以降では TLSv1.2 による SSL/TLS 通信を標準で提供していて，JDK 7 以降をベースとしているこの製品の Cosminexus Developer's Kit for Java(TM)では，SSL-J を同梱していません。

互換性を重視したシステムへ移行する場合に必要な作業

アプリケーションサーバ Version 9 と 9.5 で，同梱していた SSL-J を使用していた場合，この製品にバージョンアップをする際には，SSL-J を利用するために行っていた手順や設定を取り消してください。具体的には次の手順(a)～(c)が必要となります。

(a) コピーしていた SSL-J プロバイダのファイルの削除

次のファイルを拡張機能ディレクトリから削除します。

- sslj.jar
- cryptoj.jar
- certj.jar

拡張機能ディレクトリ

```
<JDKインストールパス>/jre/lib/ext
```

(b) セキュリティプロパティファイルの変更取り消し

SSL-J が提供しているセキュリティプロパイダを削除してください。その際，削除した行以降については，プロバイダの優先順位を表す番号を繰り上げるようにしてください。次に例を示します。

＜取り消し前のセキュリティプロパティファイルの内容＞

```
security.provider.1=com.rsa.jsafe.provider.JsafeJCE
security.provider.2=com.rsa.jsse.JsseProvider
security.provider.3=example.provider.
security.provider.4=…..
…..
```

＜取り消し後のセキュリティプロパティファイルの内容＞

```
security.provider.1=example.provider
security.provider.2=…..
…..
```

(c) ユーザプロパティファイル usrconf.properties の編集

SSL-J による SSL/TLS 通信機能を利用する場合に行った設定変更を元に戻します。この手順では，ユーザプロパティファイル usrconf.properties の編集を行います。使用している usrconf.properties によって，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の次の項目を参照してください。

「usrconf.properties（J2EE サーバ用ユーザプロパティファイル)」

「usrconf.properties（バッチサーバ用ユーザプロパティファイル)」

「usrconf.properties（Java アプリケーション用ユーザプロパティファイル)」

「usrconf.properties（バッチアプリケーション用ユーザプロパティファイル)」

SSL-J が同梱されていた Cosminexus 製品の usrconf.properties には，次のように，SSL-J が提供している SSL/TLS 通信が有効となるような設定記載が行われています。

```
…..
# JDK SSL-J Settings
# If you want to enable SSL-J, please uncomment.
https.protocols=SSLv3,TLSv1,TLSv1.1,TLSv1.2
https.cipherSuites=TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384…..
…..
```

SSL-J を同梱していないこの製品では，これらの設定は不要ですので，usrconf.properties からこれらの行を削除してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

アプリケーションサーバ Version 9 と 9.5 で，同梱していた SSL-J を使用していた場合に影響があります。

(45) Permanent 領域から Metaspace 領域への移行

変更内容

09-70 では，ロードされたクラスなどの情報が格納される領域が Metaspace 領域になりました。09-60 まではロードされたクラスなどの情報が格納される領域は Permanent 領域でしたが，09-70 では廃止されました。

互換性を重視したシステムへ移行する場合に必要な作業

Permanent 領域を 09-70 で使用することはできません。

それに伴い，-XX:PermSize，-XX:MaxPermSize は 09-70 以降指定できません。

代わりに-XX:MetaspaceSize，-XX:MaxMetaspaceSize を使用してください。

表 10–20　移行時の各オプションの設定

オプション名	設定値
-XX:MetaspaceSize	アプリケーションで必要なクラス情報の見積もり値（09-60 までの-XX:PermSize と同じ値）
-XX:MaxMetaspaceSize	アプリケーションで必要なクラス情報の見積もり値（09-60 までの-XX:MaxPermSize と同じ値）
-XX:CompressedClassSpaceSize	アプリケーションで必要なクラス情報の見積もり値（09-60 までの-XX:MaxPermSize と同じ値）

なお，Compressed Class Space は，圧縮オブジェクトポインタ機能が有効な場合，Metaspace 領域内に作成される領域です。この領域に，Java ヒープ内のオブジェクトから参照されるクラス情報が配置されます。そして，それ以外のメソッド情報などが Compressed Class Space 以外の Metaspace 領域に配置されます。

推奨機能を使用したシステムへ移行する場合に必要な作業

Metaspace 領域はロードされたクラス情報が格納される領域です。そのため，Metaspace 領域のサイズの設定は，アプリケーションで必要なクラス情報のサイズを見積もり，その値を-XX:MetaspaceSize や-XX:MaxMetaspaceSize に設定することを推奨します。これは，09-60 までの Permanent 領域の設定の考え方と同じです。

アプリケーションで必要なクラス情報のサイズの見積もり値を正しく設定することで，Metaspace 領域の OutOfMemoryError の発生を防ぎ，また-XX:MetaspaceSize と-XX:MaxMetaspaceSize を同じ値にすることで Metaspace 領域に起因する FullGC の発生を抑止することができます。

影響を受けるもの

Permanent 領域に関するオプションを使用している場合に影響を受けます。

(46) JDK 内部のファイル構成変更

変更内容

JDK 9 では，JDK 内部のディレクトリとファイル構成が一部変更されています。主要な変更点を次に示します。

- <Application Server のインストールディレクトリ>¥jdk¥下のディレクトリ構成の変更
  jre ディレクトリの削除など，いくつかの変更点があります。詳細は次の Web ページを参照してください。
  https://docs.oracle.com/javase/9/migrate/toc.htm#JSMIG-GUID-BC5E3EDE-C1FB-4B60-AF7D-8D4D7FE79C84
- rt.jar ファイルと tools.jar ファイルの削除
  rt.jar ファイルと tools.jar ファイルは削除されました。詳細は次の Web ページを参照してください。
  https://docs.oracle.com/javase/9/migrate/toc.htm#JSMIG-GUID-A78CC891-701D-4549-AA4E-B8DD90228B4B

互換性を重視したシステムへ移行する場合に必要な作業

変更内容に記載されているディレクトリまたはファイルに依存している設定や実装方式のアプリケーションの場合，設定または実装方式の変更が必要です。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(47) 削除された API／機能／オプション

変更内容

JDK 9 で削除または変更された API，機能，オプションについては次に示す Web ページを参照し，JDK 9 移行時に使用している項目がないかどうかを確認してください。

http://www.oracle.com/technetwork/java/javase/9-removed-features-3745614.html

https://docs.oracle.com/javase/9/migrate/toc.htm#JSMIG-GUID-F7696E02-A1FB-4D5A-B1F2-89E7007D4096

互換性を重視したシステムへ移行する場合に必要な作業

変更内容に記載されている API，機能，オプションを使用している場合，設定または実装方式の変更が必要です。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(48) バージョン文字列形式の変更

変更内容

JDK 9 ではバージョン文字列のフォーマットが変更され，従来の”1.8.0”のようなフォーマットではなくなりました。JDK 9 のバージョン文字列は”9”や”9.0.1”といったフォーマットになります。それに関連し，バージョン文字列に依存した，次のシステムプロパティのデフォルト値も変更されています。

- java.version
- java.specification.version
- java.vm.specification.version

互換性を重視したシステムへ移行する場合に必要な作業

バージョン文字列または変更内容で示した 3 つのシステムプロパティに依存する設定や実装方式のアプリケーションの場合，設定または実装方式の変更が必要です。JDK 9 のバージョン文字列フォーマットやシステムプロパティの詳細は，次の Web ページを参照してください。

http://www.oracle.com/technetwork/java/javase/9-relnote-issues-3704069.html#JDK-8085822

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(49) 内部 API のカプセル化

変更内容

JSR 376「Java Platform Module System」の導入に伴い，一部の内部 API がカプセル化され，利用できなくなりました（このような内部 API は，JDK 8 において非サポートとされていたものです）。そのため，そのような内部 API を使用したソースコードを JDK 9 でコンパイルした場合，エラーや警告が発生するおそれがあります。

また，JDK 8 以前の JDK を利用して作成したアプリケーションを JDK 9 で実行させた場合でも同様に，カプセル化した内部 API を使用していると，エラーや警告が発生するおそれがあります。

詳細は次の Web ページを参照してください。

http://www.oracle.com/technetwork/java/javase/9-relnote-issues-3704069.html#JDK-8142968

互換性を重視したシステムへ移行する場合に必要な作業

内部 API を使用している場合，設定または実装方式の変更が必要です。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(50) -Xbootclasspath オプションと-Xbootclasspath/p オプションの削除

変更内容

JDK 9 では，-Xbootclasspath オプションと-Xbootclasspath/p オプションは削除されました。一方，-Xbootclasspath/a オプションは削除されていません。ただし，これらのオプションは，旧バージョンを含め，削除されているかどうかに関係なく，アプリケーションサーバではサポートしていません。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(51) 推奨標準機構（Endorsed Standards Override Mechanism）の削除

変更内容

JDK 9 では推奨標準機構（Endorsed Standards Override Mechanism）が削除され，java.endorsed.dirs システムプロパティと<Application Server のインストールディレクトリ>¥jdk¥lib¥endorsed ディレクトリは使用できません。JavaVM の起動時，このシステムプロパティ指定またはディレクトリが存在する場合は，JDK の起動に失敗します。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(52) 拡張機能機構（Extensions Mechanism）の削除

変更内容

JDK 9 では拡張標準機構（Extensions Mechanism）が削除され，java.ext.dirs システムプロパティと<Application Server のインストールディレクトリ>¥jdk¥lib¥ext ディレクトリは使用できません。

JDK の起動時，このシステムプロパティ指定またはディレクトリが存在する場合は，JDK の起動に失敗します。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(53) システムクラスローダの継承関係の変更

変更内容

JDK 8 以前のシステムクラスローダは，JDK 内部実装として java.net.URLClassLoader を継承していましたが，JDK 9 のシステムクラスローダは java.net.URLClassLoader を継承していません。そのため，システムクラスローダが java.net.URLClassLoader を継承していることを前提としていないことを確認してください。

詳細は次の Web ページを参照してください。

http://www.oracle.com/technetwork/java/javase/9-relnote-issues-3704069.html

互換性を重視したシステムへ移行する場合に必要な作業

システムクラスローダが java.net.URLClassLoader を継承していることを前提としている場合，実装方式の変更が必要です。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(54) javah コマンドの削除

変更内容

javah コマンドは JDK 10 で削除されました。javah コマンドの機能は，javac コマンドの-h オプションに移管されています。詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/10-relnote-issues-4108729.html#JDK-8182758

互換性を重視したシステムへ移行する場合に必要な作業

javac コマンドの-h オプションを使用してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(55) -Xprof オプションの削除

変更内容

JavaVM オプションである，-Xprof オプションは JDK 10 で削除されました。詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/10-relnote-issues-4108729.html#JDK-8173715

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(56) java.io.FileInputStream と java.io.FileOuputStream クラスの finalize メソッドの実装変更

変更内容

FileInputStream クラスと FileOutputStream クラスの finalize メソッドは，以前は close メソッドを呼び出していましたが，JDK 10 以降では close メソッドを呼び出しません。もし，これらクラスの finalize メソッドが close メソッドを直接呼び出すことを前提とした処理がある場合には，非互換性が発生するおそれがあります。その場合は，close メソッドを明示的に呼び出すか，try-with-resouce 構文を使用してリソースの解放をしてください。ただし，Application Server を使用した場合は，互換性を考慮し，旧バージョンと同様に FileInputStream クラスと FileOutputStream クラスの finalize メソッドは close メソッドを呼び出す方式になります。詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/10-relnote-issues-4108729.html#JDK-8080225

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(57) JMX エージェントのパスワード暗号化

変更内容

JMX エージェントを使用したリモート接続時，設定ファイル（jmxremote.password）に記載されたパスワードはハッシュ化された値で上書きされるようになりました。詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/10-relnote-issues-4108729.html#JDK-5016517

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(58) Java EE 向けモジュールと CORBA モジュールの削除

変更内容

次に示す Java EE 向けモジュールと CORBA モジュールは JDK 11 で削除されました。

- java.xml.ws
- java.xml.bind
- java.activation
- java.xml.ws.annotation
- java.corba
- java.transaction
- java.se.ee
- jdk.xml.ws
- jdk.xml.bind

モジュールの削除に伴い，これらのモジュールに関連したコマンド群やシステムプロパティも削除されています。詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/11-relnote-issues-5012449.html#JDK-8190378

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(59) クラスファイルバージョンの変更

変更内容

JDK11 のクラスファイルバージョンは”55.0”です。クラスファイルバージョン”54.0”や”55.0”のクラスファイルを JDK9 で動作させることはできません。

また，このクラスファイルバージョンの変更に伴い，クラスファイルフォーマットも変更されています。

互換性を重視したシステムへ移行する場合に必要な作業

クラスファイルの解析などをするプログラムやライブラリを使用している場合は，実装の変更やライブラリの更新などが必要となる場合があります。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(60) javac コマンドの source/target/release オプションに指定できる値

変更内容

09-87 の javac コマンドでは，source オプション，target オプションそれぞれに指定できる値を次に示します。

- 「6」または「1.6」(非推奨。次バージョンで指定できなくなるおそれがあります。)
- 「7」または「1.7」
- 「8」または「1.8」
- 「9」または「1.9」

09-80 以降では「10」または「1.10」は指定できません。「10」または「1.10」を指定した場合の動作は保証しません。また，--release オプションも同様に，「10」を指定した場合の動作は保証しません。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(61) Java プログラム実行中のシステムプロパティの変更

変更内容

Java プログラム実行中，次に示すシステムプロパティ値を変更しても，その変更が java.base モジュール内の API の動作に反映されない場合があります。JavaVM 初期化時の値が参照される方式になりました。

- java.home
- user.home
- user.dir
- user.name

詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/11-relnote-issues-5012449.html#JDK-8066709

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(62) 3DES 暗号スイートの無効化

変更内容

SSL/TLS 通信で使用する 3DES 暗号スイートは，使用できなくなりました。

詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/11-relnote-issues-5012449.html#JDK-8175075

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(63) runFinalizersOnExit メソッドの削除

変更内容

java.lang.System クラスと java.lang.Runtime クラスの runFinalizersOnExit メソッドは，JDK 11 で削除されました。runFinalizersOnExit メソッドを使用すると java.lang.NoSuchMethodError がスローされます。

互換性を重視したシステムへ移行する場合に必要な作業

runFinalizersOnExit メソッドを使用している場合は，使用しない方式へプログラムを変更してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

(64) ThreadPoolExecutor と ScheduledThreadPoolExecutor クラスの finalize メソッドの実装変更

変更内容

java.util.concurrent.ThreadPoolExecutor クラスと，そのサブクラスである java.util.concurrent.ScheduledThreadPoolExecutor クラスの finalize メソッドは，以前は shutdown メソッドを呼び出していましたが，JDK11 以降では shutdown メソッドを呼び出しません。もし，これらのクラスの finalize メソッドが shutdown メソッドを呼び出すことを前提とした処理がある場合には，非互換性が発生するおそれがあります。その場合は，shutdown メソッドを明示的に呼び出すようにしてください。

ただし，Application Server を使用した場合は，互換性を考慮し，旧バージョンと同様に ThreadPoolExecutor クラスと ScheduledThreadPoolExecutor クラスの finalize メソッドは shutdown メソッドを呼び出す方式になります。これらのクラスの shutdown メソッドは複数回呼ばれても問題ないメソッドです。

また，Application Server を使用する際，ThreadPoolExecutor クラスまたは ScheduledThreadPoolExecutor クラスのサブクラスを作成していて，かつ shutdown メソッドをオーバーライドしている場合，オーバーライドした shutdown メソッドが複数回呼ばれるおそれがあります。オーバーライドする shutdown メソッドが複数回呼ばれても問題ないように実装してください。

詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/11-relnote-issues-5012449.html#JDK-8190324

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(65) getAnnotation メソッド呼び出しで発生する例外の変更

変更内容

アノテーションに指定したクラスがクラスパスに存在しない場合に，そのアノテーションを java.lang.Class クラスの getAnnotation メソッドで取得しようとすると，JDK10 以前は java.lang.ArrayStoreException をスローしていました。JDK11 以降から，同様の処理を実施しても java.lang.ArrayStoreException をスローしないで，getAnnotation メソッドで取得したアノテーションの値を読み込もうとしたタイミングで，java.lang.TypeNotPresentException をスローするようになりました。もし，java.lang.ArrayStoreException を前提とした処理がある場合には，非互換性が発生するおそれがあります。その場合は，java.lang.TypeNotPresentException を明示的にキャッチするようにしてください。

ただし，Application Server を使用した場合だけ，JDK10 以前と同様に java.lang.ArrayStoreException をスローする方式になります。従って，java.lang.ArrayStoreException を明示的にキャッチしていたプログラムで，java.lang.TypeNotPresentException を処理するように変更する必要はありません。

詳細は次の Web ページを参照してください。

https://www.oracle.com/technetwork/java/javase/11-relnote-issues-5012449.html#JDK-7183985

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(66) 日立拡張機能が提供するファイルの配置場所の変更

変更内容

jre ディレクトリの削除に伴い，旧バージョンの jre ディレクトリ内の，日立拡張機能が提供するコマンドまたはファイルの配置場所が変更されました。jre ディレクトリの削除については「10.3.5(46) JDK 内部のファイル構成変更」を参照してください。

配置場所が変更されたコマンドまたはファイルについて，旧バージョンである 09-70 の配置場所と 09-80 以降での配置場所を次の表に示します。

表 10–21　配置場所が変更されたコマンドまたはファイル一覧

項番	09-70	09-80 以降
1	jdk¥jre¥bin¥eheapprof.exe	jdk¥bin¥eheapprof.exe
2	jdk¥jre¥bin¥javacore.exe	jdk¥bin¥javacore.exe
3	jdk¥jre¥bin¥javagc.exe	jdk¥bin¥javagc.exe
4	jdk¥jre¥bin¥jheapprof.exe	jdk¥bin¥jheapprof.exe
5	jdk¥jre¥bin¥jheapprofanalyzer.exe	jdk¥bin¥jheapprofanalyzer.exe
6	jdk¥jre¥bin¥hndlwrap2.dll	jdk¥bin¥hndlwrap2.dll
7	jdk¥jre¥lib¥explicitmemory¥sysexmemexcludeclass.cfg	jdk¥lib¥explicitmemory¥sysexmemexcludeclass.cfg
8	jdk¥jre¥bin¥car_tar_gz	jdk¥bin¥car_tar_gz
9	jdk¥jre¥bin¥car_tar_Z	jdk¥bin¥car_tar_Z
10	jdk¥jre¥bin¥javatrace	jdk¥bin¥javatrace

注 1　jdk ディレクトリは，<Application Server のインストールディレクトリ>¥jdk を指します。

注 2　項番 1～5 のファイル名文字列は，Linux 環境では拡張子.exe が存在しない文字列です。

注 3　項番 6 は Windows 固有のファイル，項番 8～10 は Linux 環境固有のファイルです。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(67)　拡張 verbosegc 出力機能のログフォーマット変更（G1GC を使用する場合）

変更内容

ガベージファースト・コレクタ（G1GC）使用時，拡張 verbosegc 出力機能のログファイルフォーマットに GCID という項目が追加されます。GCID とは，GC ごとに付与される一意の値です。G1GC 以外のガーベージコレクション方式使用時には，拡張 verbosegc 出力機能のログファイルフォーマットに変更はありません。

GCID は，次に示す GC を開始するタイミングでインクリメントされます。

- Young GC
- Concurrent Marking(CM)
- Mixed GC

- Full GC

G1GC 使用時の GCID の出力イメージを次に示します。各行末尾の太字の項目が GCID の出力項目です。

G1GC 使用時の GCID 出力イメージ

```
[VG1]<Thu Sep 15 11:01:45 2016>[Young GC(initial-mark) 974K/1024K(131072K)->1044K/2048K(1
31072K), 0.0017569 secs]…[GCID: 1]
[VCM]<Thu Sep 15 11:01:45 2016>[Concurrent Cycle Start][User: 0.0000000 secs][Sys: 0.0000
000 secs][GCID: 2]
[VCM]<Thu Sep 15 11:01:45 2016>[Concurrent Root Region Scan Start][User: 0.0000000 secs][
Sys: 0.0000000 secs][GCID: 2]
[VCM]<Thu Sep 15 11:01:45 2016>[Concurrent Root Region Scan End][User: 0.0000000 secs][Sy
s: 0.0000000 secs][GCID: 2]
[VCM]<Thu Sep 15 11:01:45 2016>[Concurrent Mark Start][User: 0.0000000 secs][Sys: 0.00000
00 secs][GCID: 2]
[VG1]<Thu Sep 15 11:01:45 2016>[Young GC 1022K/1024K(131072K)->1024K/1024K(131072K), 0.00
15565 secs]…[GCID: 3]
[VCM]<Thu Sep 15 11:01:45 2016>[Concurrent Mark End][User: 0.0000000 secs][Sys: 0.0000000
 secs][GCID: 2]
[VG1]<Thu Sep 15 11:01:45 2016>[Young GC 1024K/1024K(131072K)->1019K/1024K(157696K), 0.00
21122 secs]…[GCID: 4]
[VG1]<Thu Sep 15 11:01:45 2016>[CM Remark 1024K/1024K(131072K)->1019K/1024K(157696K), 0.0
007291 secs]…[GCID: 2]
[VG1]<Thu Sep 15 11:01:45 2016>[CM Cleanup 1024K/1024K(131072K)->1019K/2048K(157696K), 0.
0002869 secs]…[GCID: 2]
[VCM]<Thu Sep 15 11:01:45 2016>[Concurrent Cycle End][User: 0.0000000 secs][Sys: 0.000000
0 secs][GCID: 2]
```

注

…は省略を示します。

GIGC 使用時の拡張 verbosegc 出力機能のログフォーマット（非 CSV 形式）を次に示します。

gc_id が GC ごとに付与される一意の値を示します。フォーマット中の *gc_id* 以外の出力項目（*id*, *data* などの太字・斜字部分）の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください

G1GC 使用時の拡張 verbosegc 出力機能のログフォーマット（非 CSV 形式）

```
 [id]<date>[gc_kind gc_info, gc_time secs][Status:gc_status][G1GC::Eden: eden_info][G1GC:
:Survivor: survivor_info][G1GC::Tenured: tenured_info][G1GC::Humongous: humongous_info][G
1GC::Free: free_info][G1GC::Perm: perm_info][Metaspace: metaspace_info][class space: clas
s_space_info] [cause:cause_info][RegionSize: region_sizeK][Target: target_time secs][Pred
icted: predicted_time secs][TargetTenured: target_sizeK][Reclaimable: reclaimable_info][U
ser: user_cpu secs][Sys: system_cpu secs][IM: jvm_alloc_sizeK, mmap_total_sizeK, malloc_t
otal_sizeK][TC: thread_count][DOE: doe_alloc_sizeK, called_count][CCI: cc_used_sizeK, cc_
max_sizeK, cc_infoK][GCID: gc_id]
```

注

上記のログフォーマットは 1 文です。

G1GC 使用時の拡張 verbosegc 出力機能のログ出力例（非 CSV 形式）を次に示します。

G1GC 使用時の拡張 verbosegc 出力機能のログ出力例（非 CSV 形式）

```
[VG1]<Wed Sep 14 13:48:48.563 2016>[Full GC 841216K/841728K(1344512K)->632147K/841728K(13
44512K), 0.2313101 secs][Status:-][G1GC::Eden: 6144K(57344K)->0K(66560K)][G1GC::Survivor:
 9216K->0K][G1GC::Tenured: 826368K->841728K][G1GC::Humongous: 0K->0K][G1GC::Free: 502784K
->502784K][Metaspace: 6562K(6900K, 7040K)->6562K(6900K, 7040K)][class space: 662K(774K, 8
96K)->662K(774K, 896K)][cause:System.gc][RegionSize: 1024K][Target: 0.2000000 secs][Predi
cted: 0.0000000 secs][TargetTenured: 0K][Reclaimable: 0K(0.00%)][User: 0.2343750 secs][Sy
s: 0.0000000 secs][IM: 20459K, 21920K, 0K][TC: 34][DOE: 0K, 0][CCI: 2126K, 245760K, 7488K
][GCID: 22]
```

GIGC 使用時の拡張 verbosegc 出力機能のログフォーマット（CSV 形式）を次に示します。

gc_id が GC ごとに付与される一意の値を示します。フォーマット中の*gc_id* 以外の出力項目（*id*, *data* などの太字・斜字部分）の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください

G1GC 使用時の拡張 verbosegc 出力機能のログフォーマット（非 CSV 形式）

```
id,date,gc_kind,gc_info,gc_time,gc_status,eden_info,survivor_info,tenured_info,humongous_
info,free_info,perm_info,metaspace_info,classspace_info,cause_info,region_size,target_tim
e,predicted_time,target_size,reclaimable_info,user_cpu,system_cpu,jvm_alloc_size,mmap_tot
al_size,malloc_total_size,thread_count,doe_alloc_size,called_count,cc_used_size,cc_max_si
ze,cc_info,gc_id
```

注

上記 format 文は 1 文です。

G1GC 使用時の拡張 verbosegc 出力機能のログ出力例（CSV 形式）を次に示します。

G1GC 使用時の拡張 verbosegc 出力機能のログ出力例（CSV 形式）

```
VG1,Thu Oct 02 10:40:54.920 2016,Full GC,753,2048,8192,678,1024,8192,0.0064767,-,1024,204
8,0,2048,0,0,1024,1024,0,0,6144,7168,3634,3634,4492,3634,3634,4492,356,356,388,356,356,38
8,1,1024,0.2000000,0.0000000,0,0,0.00,0.0000000,0.0000000,20459,21920,0,35,0,0,1171,24576
0,2496,3
```

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(68) スレッドダンプ中の JavaVM 内部メモリマップ情報の出力フォーマット変更

変更内容

スレッドダンプには，JavaVM 自身が確保しているメモリ領域情報として，JavaVM 内部メモリマップ情報が出力されます。この JavaVM 内部メモリマップ情報の出力フォーマットが変更されました。スレッドダンプや JavaVM 内部メモリマップ情報の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

旧バージョンでは，JavaVM 内部メモリマップ情報は JVM Internal Memory Map の 1 項目としてすべて出力していましたが，09-80 以降では，JavaVM 内部メモリマップ情報を JVM Internal Memory Map と JVM Internal Memory Map (Large Block)の 2 項目に分けて出力します。JavaVM 自身が確保しているメモリ領域のうち，巨大なメモリを割り当てるために JavaVM が確保したメモリ情報を JVM Internal Memory Map（Large Block）として出力し，それ以外の，通常の大きさのメモリを割り当てるために JavaVM が確保したメモリ情報を JVM Internal Memory Map として出力します。

JavaVM 内部メモリマップ情報の出力フォーマットと出力例を次に示します。

JavaVM 内部メモリマップ情報の出力フォーマット

```
JVM Internal Memory Map
-----------------------
        <メモリ確保関数>  : address = <開始アドレス> - <終了アドレス> (size:<サイズ>)
...

JVM Internal Memory Map (Large Block)
-------------------------------------
        <メモリ確保関数>  : address = <開始アドレス> - <終了アドレス> (size:<サイズ>)
...
```

注

<メモリ確保関数>：mmap()か malloc()のどちらかです。

<開始アドレス>：メモリ領域の開始アドレスが 16 進数で表示されます。

<終了アドレス>：メモリ領域の終了アドレスが 16 進数で表示されます。

<サイズ>：確保しているメモリ領域のサイズが出力されます（単位：バイト）。

JavaVM 内部メモリマップ情報の出力例

```
JVM Internal Memory Map
-----------------------
        mmap()   : address = 0x000000ee20010000 - 0x000000ee20050000 (size:262144)
        mmap()   : address = 0x000000ee2d590000 - 0x000000ee2d5d0000 (size:262144)
        mmap()   : address = 0x000000ee33280000 - 0x000000ee332c0000 (size:262144)

JVM Internal Memory Map (Large Block)
-------------------------------------
        mmap()   : address = 0x000000ee2d500000 - 0x000000ee2d528000 (size:163840)
        mmap()   : address = 0x000000ee2d530000 - 0x000000ee2d58b000 (size:372736)
        mmap()   : address = 0x000000ee33200000 - 0x000000ee33276000 (size:483328)
```

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(69)　エラーリポートファイル中のメモリ情報の出力フォーマット変更

変更内容

エラーリポートファイルには，JavaVM 自身が確保しているメモリ領域情報として，メモリ情報が出力されます。このメモリ情報の出力フォーマットが変更されました。エラーリポートファイルやメモリ情報の詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

09-80 以降では，「10.3.5(68)　スレッドダンプ中の JavaVM 内部メモリマップ情報の出力フォーマット変更」の JVM Internal Memory Map に対応するメモリ情報のあとに空行 1 行が挿入され，JVM Internal Memory Map （Large Block）に対応するメモリ情報を出力します。

メモリ情報の出力フォーマットと出力例を次に示します。

メモリ情報の出力フォーマット

```
JVM Internal Memory Map
-----------------------
        <メモリ確保関数>  : address = <開始アドレス> - <終了アドレス> (size:<サイズ>)
...

JVM Internal Memory Map (Large Block)
-------------------------------------
        <メモリ確保関数>  : address = <開始アドレス> - <終了アドレス> (size:<サイズ>)
...
        Heap Size  : <確保しているメモリサイズ>
        Alloc Size : <使用中のメモリサイズ>
        Free Size  : <未使用のメモリサイズ>
```

注

<メモリ確保関数>：mmap()か malloc()のどちらかです。

<開始アドレス>：メモリ領域の開始アドレスが 16 進数で表示されます。

<終了アドレス>：メモリ領域の終了アドレスが 16 進数で表示されます。

各種メモリサイズの単位はバイトです。

メモリ情報の出力例

```
Memory:
        mmap()   : address = 0x000000ca0d380000 - 0x000000ca0d400000 (size:524288)
        mmap()   : address = 0x000000ca136d0000 - 0x000000ca13710000 (size:262144)
        mmap()   : address = 0x000000ca13d10000 - 0x000000ca13dd0000 (size:786432)

        mmap()   : address = 0x000000ca0d270000 - 0x000000ca0d298000 (size:163840)
        mmap()   : address = 0x000000ca0d2a0000 - 0x000000ca0d2fb000 (size:372736)
        mmap()   : address = 0x000000ca0d300000 - 0x000000ca0d376000 (size:483328)

        Heap Size  : 3919872
        Alloc Size : 3302992
        Free Size  : 616880
```

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(70)　インストーラ

変更内容

更新インストールする前のバージョンが 09-70 と 09-80 以降では，更新インストール時に退避するファイルの退避場所と退避後のファイル名が異なります。

「10.3.5(71)　更新インストール時に退避するファイルと退避先」で，09-70 と 09-80 以降のインストール時の動作の違いについて説明します。

(71)　更新インストール時に退避するファイルと退避先

変更内容

更新インストールする前のバージョンが 09-70 と 09-80 以降では，更新インストール時に退避するファイルの退避場所と退避後のファイル名が異なります。

更新インストールする前のバージョンが 09-70 以前の場合

09-80 以降のインストーラでは，<Application Server のインストールディレクトリ>¥jdk¥jre_バージョン番号_日付時刻¥下にファイルを退避します。また，ファイル名はインストール前のファイル名のままとなります。

2017 年 7 月 10 日 18:02:02 に 09-60 から 09-80 に更新インストールした場合の例を次の表に示します。

表 10–22　09-60 から 09-80 に更新インストールした場合の例

退避対象ファイル	退避後のファイルパス
jdk¥jre¥lib¥security¥cacerts	jdk¥jre_0960_20170710180202¥lib¥security¥cacerts
jdk¥jre¥lib¥logging.properties	jdk¥jre_0960_20170710180202¥lib¥logging.properties
jdk¥jre¥lib¥net.properties	jdk¥jre_0960_20170710180202¥lib¥net.properties
jdk¥jre¥lib¥sound.properties	jdk¥jre_0960_20170710180202¥lib¥sound.properties
jdk¥jre¥lib¥management¥jmxremote.access	jdk¥jre_0960_20170710180202¥lib¥management¥jmxremote.access
jdk¥jre¥lib¥management¥management.properties	jdk¥jre_0960_20170710180202¥lib¥management¥management.properties
jdk¥jre¥lib¥security¥java.policy	jdk¥jre_0960_20170710180202¥lib¥security¥java.policy
jdk¥jre¥lib¥security¥java.security	jdk¥jre_0960_20170710180202¥lib¥security¥java.security

注

jdk ディレクトリは，<Application Server のインストールディレクトリ>¥jdk を指します。

更新インストールする前のバージョンが 09-80 以降の場合

退避したファイルの別名（ファイル名_バージョン番号_日付時刻）の命名規則に変更はなく，同じディレクトリ内に別名ファイルとして退避します。

09-80 以降のインストーラが退避する対象ファイルを次に示します。

- jdk¥lib¥security¥cacerts
- jdk¥conf¥logging.properties
- jdk¥conf¥net.properties
- jdk¥conf¥sound.properties
- jdk¥conf¥management¥jmxremote.access
- jdk¥conf¥management¥management.properties
- jdk¥conf¥security¥java.policy
- jdk¥conf¥security¥java.security

注

jdk ディレクトリは，<Application Server のインストールディレクトリ>¥jdk を指します。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(72) クラスライブラリのスタックトレース出力オプションの対象となる API の変更

変更内容

次に示す API は Oracle 社製 JDK 11 で削除されたため，クラスライブラリのスタックトレース出力オプション(-XX:[+¦-]HitachiJavaClassLibTrace)を有効にした際，トレース対象となる API ではなくなりました。

- java.lang.System.runFinalizersOnExit
- java.lang.Runtime.runFinalizersOnExit

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(73) G1GC の処理変更に伴う JavaVM ログファイルの出力形式変更

変更内容

JDK9（09-80）以前では，拡張 verbosegc 機能と G1GC の両方を有効にした場合，G1GC の Concurrent Marking 処理中に Concurrent Cleanup が実施されると，JavaVM ログファイルには VG1 ログの cm_event 欄に Concurrent Cleanup Start（または End）が出力されました。
JDK11(09-87)では，Concurrent Cleanup が削除されたため，VG1 ログの cm_event 欄には [Concurrent Cleanup Start（または End）が出力されなくなります。

拡張 verbosegc 情報出力オプション，JavaVM ログファイルの出力形式については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

互換性を重視したシステムへ移行する場合に必要な作業

作業はありません。

推奨機能を使用したシステムへ移行する場合に必要な作業

作業はありません。

影響を受けるもの

なし。

(74) StackOverflowError

変更内容

AIX 環境では，次に示す処理でスタック領域が枯渇すると，java.lang.StackOverflowError 例外がスローされないで Java プロセスが異常終了する場合があります。

- cjstartsv コマンド実行もしくは Management Server による，J2EE サーバの起動処理
- cjstartweb コマンド実行もしくは Management Server による， Web コンテナサーバの起動処理
- cjclstartap コマンド実行時，Java アプリケーションのメインスレッド 上で行われる処理

この時，次のメッセージを出力して Java プロセスが終了します。

- java.lang.StackOverflowError occurred in primordial thread.

互換性を重視したシステムへ移行する場合に必要な作業

OS のスタック領域の最大サイズの設定値と-Xss オプション（デフォルト:1MB）に設定する値を大きくして，スタック領域が枯渇しないようにしてください。この際，OS のスタック領域の最大サイズには，-Xss オプション値以上の値を設定してください。

推奨機能を使用したシステムへ移行する場合に必要な作業

互換性を重視したシステムへ移行する場合に必要な作業と同じです。

影響を受けるもの

なし。

10.4　J2EE サーバまたはバッチサーバの作業ディレクトリのディスク容量の確認

J2EE サーバまたはバッチサーバの作業ディレクトリ用に十分なディスク容量があるかを確認します。バッチサーバの場合は，「J2EE サーバ」を「バッチサーバ」と読み替えてください。

旧バージョンからの移行で幾つかの移行コマンドを実行する必要がありますが，移行コマンドを実行すると，J2EE サーバの作業ディレクトリ内のファイルを変換する前にバックアップを作成します。バックアップは J2EE サーバの作業ディレクトリ以下に作成されます。また，移行コマンドを実行すると，バックアップ分の容量に加えて，J2EE サーバの作業ディレクトリ自体のサイズも大きくなります。

これらの理由から，移行の前に作業ディレクトリに割り当てられているディスク領域の容量が十分かどうかを確認してください。バックアップと移行後の作業ディレクトリのサイズは次のとおりです。

なお，ここでのバックアップとは，バックアップ作業ディレクトリとバックアップ RAR ディレクトリを示します。詳細については，「10.7　J2EE サーバまたはバッチサーバの移行コマンドの実行」，および「10.8.1　リソースアダプタの移行コマンドの実行」を参照してください。

旧バージョンでの J2EE サーバの作業ディレクトリのサイズが N バイトの場合，バックアップ領域のサイズは，次の計算式で算出してください。

N×（cjenvupdate コマンドによって作業ディレクトリが移行される場合は 1，移行されない場合は 0）＋ N×（DB Connector を使用している場合は 1，使用していない場合は 0）+N×（リソースアダプタを使用している場合は，バージョンアップするリソースアダプタの数）

また，旧バージョンでの J2EE サーバの作業ディレクトリのサイズが N バイトの場合，作業ディレクトリ自体の増加分のサイズは次の計算式で算出してください。

N×（cjenvupdate コマンドによって作業ディレクトリが移行される場合は 1，移行されない場合は 0）＋ N×（DB Connector を使用している場合は 1，使用していない場合は 0）+N×（リソースアダプタを使用している場合は，バージョンアップするリソースアダプタの数）

注意事項

移行するためには，J2EE サーバの作業ディレクトリが割り当てられているディスクの空き容量が，バックアップ領域分と作業ディレクトリ自体の増加分を足した値よりも大きい必要があります。なお，移行が完了したあと，バックアップ領域は削除できます。

10.5　アプリケーションサーバの動作環境および動作状態の確認

移行前に，アプリケーションサーバの動作環境，および動作状態を確認します。次に確認項目を示します。

- 運用管理サーバ（Management Server だけ起動するマシン）を利用するシステム構成の場合，運用管理サーバにインストールしているアプリケーションサーバは 09-70 に移行する必要があります。
- 旧バージョンのアプリケーションサーバが終了していることを確認してください。アプリケーションサーバの全構成ソフトウェアを終了させます。
- Microsoft IIS を使用している場合は，Microsoft IIS を終了させます。
- JavaVM のプロセスは，プロセス起動時に取得した CPU 情報を使って動作します。このため，プロセス起動時と異なる CPU 情報（プロセッサ数など）の環境へ移行した場合の動作は保証できません。
- AIX の場合，旧バージョンで環境変数 LDR_CNTRL に MAXDATA を指定して運用していたときは，移行後に旧バージョンの指定値のまま運用すると，C ヒープ領域のサイズが不足するおそれがあります。移行後も環境変数 LDR_CNTRL に MAXDATA を指定して運用するときは，MAXDATA の指定値を見直してください。

10.6　定義などのバックアップ

定義情報のバックアップについて説明します。なお，ここで説明している定義情報は，次に示す構成ソフトウェアに関するものです。

- Performance Tracer
- Component Container
- Component Transaction Monitor
- Component Container - Client
- Component Container - Redirector
- Developer's Kit for Java
- Application Development Plug-in

このほかの構成ソフトウェアについては，次に示すマニュアルを参照してください。また，必要な場合は，保守員に連絡してください。

表 10–23　構成ソフトウェアごとの参照先マニュアル

構成ソフトウェア	参照先マニュアル
HTTP Server	HTTP Server
Reliable Messaging	Reliable Messaging
Web Services - Security	アプリケーションサーバ Web サービスセキュリティ構築ガイド
XML Processor	XML Processor ユーザーズガイド

10.6.1　新規インストールの場合の定義情報のバックアップ

新規インストールの場合，バックアップを取った定義情報を基に，新しいアプリケーションサーバの定義情報を作成します。

バックアップを取る情報を次の表に示します。

表 10–24　バックアップを取る情報（新規インストールの場合）

機能	バックアップを取る情報	バックアップを取った情報の設定先
Management Server／統合ユーザ管理	次に示すファイルをバックアップします。 • keyfile.key（ua.conf の com.cosminexus.admin.auth.sso.keyfile キーに指定した暗号鍵ファイル）※	• Windows の場合 <製品のインストールディレクトリ>¥manager¥config • UNIX の場合 /opt/Cosminexus/manager/config

機能	バックアップを取る情報	バックアップを取った情報の設定先
	次に示すファイルをバックアップします。 • Manager で使用するファイル • Smart Composer 機能で使用するファイル • 統合ユーザ管理で使用するファイル • ログの運用で使用するファイル	設定先については，次のマニュアルを参照してください。 Manager で使用するファイル マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8. Manager で使用するファイル」 Smart Composer 機能で使用するファイル マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4. Smart Composer 機能で使用するファイル」 統合ユーザ管理で使用するファイル マニュアル「アプリケーションサーバ 機能解説 セキュリティ管理機能編」の「14. 統合ユーザ管理で使用するファイル」 ログの運用で使用するファイル マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「10. ログの運用で使用するファイル」
	次に示すディレクトリをバックアップします。 • Windows の場合 <製品のインストールディレクトリ>¥manager¥modules • UNIX の場合 /opt/Cosminexus/manager/modules	• Windows の場合 <製品のインストールディレクトリ>¥manager • UNIX の場合 /opt/Cosminexus/manager
	次に示すディレクトリをバックアップします。 • Windows の場合 <製品のインストールディレクトリ>¥manager¥apps • UNIX の場合 /opt/Cosminexus/manager/apps	• Windows の場合 <製品のインストールディレクトリ>¥manager • UNIX の場合 /opt/Cosminexus/manager
	運用管理ポータルで，次に示す情報を控えます。 • ［運用管理ドメインの構成定義］画面で設定する情報 • ［論理サーバの環境設定］画面で設定する情報。論理 J2EE サーバの設定項目に関しては，［JP1 連携の設定］画面，［オプションの設定］画面，［環境変数の設定］画面，および［J2EE サーバの基本設定］画面の「セキュリティマネージャの使用」の設定項目だけ控えます。 • ［論理サーバの起動／停止］画面で設定する情報 • ［論理サーバのアプリケーション管理］画面で設定する情報。ただし，旧バージョンのアプリケーションサーバが 08-00 未満の場合，設定項目を控えます。 • 統合ユーザ管理の［リポジトリ管理］画面で設定する情報	設定項目を控えたときと同じ運用管理ポータルの画面
J2EE サーバ	次に示すディレクトリをバックアップします。 • Windows の場合	• Windows の場合

機能	バックアップを取る情報	バックアップを取った情報の設定先
	<製品のインストールディレクトリ>¥CC¥server¥usrconf¥ejb • UNIX の場合 /opt/Cosminexus/CC/server/usrconf/ejb	<製品のインストールディレクトリ>¥CC¥server¥usrconf • UNIX の場合 /opt/Cosminexus/CC/server/usrconf
	Management Server を使用していない場合は，J2EE サーバ起動時に独自に定義していた環境変数を退避してください。	運用管理ポータルの［環境変数の設定］画面，または簡易構築定義ファイルの user.env.variable パラメタに設定します。 環境変数の設定については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「4.1　Web サーバを別ホストに配置してマシン性能を向上するシステムの構築」の環境変数に関する説明を参照してください。
バッチサーバ	次に示すディレクトリをバックアップします。 • Windows の場合 <製品のインストールディレクトリ>¥CC¥server¥usrconf¥ejb • UNIX の場合 /opt/Cosminexus/CC/server/usrconf/ejb	• Windows の場合 <製品のインストールディレクトリ>¥CC¥server¥usrconf • UNIX の場合 /opt/Cosminexus/CC/server/usrconf
サーバ管理コマンド	次に示すディレクトリをバックアップします。 • Windows の場合 <製品のインストールディレクトリ>¥CC¥admin¥usrconf • UNIX の場合 /opt/Cosminexus/CC/admin/usrconf	• Windows の場合 <製品のインストールディレクトリ>¥CC¥admin • UNIX の場合 /opt/Cosminexus/CC/admin
Web サーバ連携	次に示すファイルをバックアップします。 Microsoft IIS の場合 • isapi_redirect.conf（Microsoft IIS 用リダイレクタ動作定義ファイル） • uriworkermap.properties（Microsoft IIS 用マッピング定義ファイル） • workers.properties（ワーカ定義ファイル） HTTP Server の場合 • mod_jk.conf（HTTP Server 用リダイレクタ動作定義ファイル） • workers.properties（ワーカ定義ファイル）	• Windows の場合 <製品のインストールディレクトリ>¥CC¥web¥redirector • UNIX の場合 /opt/Cosminexus/CC/web/redirector
HTTP Server	次に示すファイルをバックアップします。 • httpsd.conf（HTTP Server 定義ファイル） • mime.types（HTTP Server 定義ファイル）	• Windows の場合 <製品のインストールディレクトリ>¥httpsd¥conf • UNIX の場合 /opt/hitachi/httpsd/conf Smart Composer 機能を使って Web システムを構築した場合，または運用管理ポータル画面

機能	バックアップを取る情報	バックアップを取った情報の設定先
		で論理サーバを構築した場合の httpsd.conf ファイルの設定先を次に示します。 • Windows の場合 <製品のインストールディレクトリ>¥httpsd¥servers¥HWS_<論理 Web サーバ名>¥conf • UNIX の場合 /opt/hitachi/httpsd/servers/HWS_<論理 Web サーバ名>/conf
	Management Server を使用していない場合は，HTTP Server 起動時に独自に定義していた環境変数を退避してください。	adminagent.xml（運用管理エージェント設定ファイル）へ設定してください。環境変数の設定については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「4.1.12　論理サーバの環境変数を設定する」の環境変数に関する説明を参照してください。
	次のディレクティブに設定されたディレクトリ，およびファイルは，HTTP Server の設定，または HTTP Server の出力するコンテンツに関連するものです。各ディレクティブに設定されたディレクトリ，およびファイルの内容を確認し，移行する必要があるディレクトリ，およびファイルを選択し，バックアップを取ってください。 <Directory>，<DirectoryMatch>，AccessFileName，Alias，AliasMatch，AuthGroupFile，AuthUserFile，CoreDumpDirectory（UNIX の場合），DocumentRoot，HWSTraceIdFile，HWSTraceLogFile，Include，LoadFile，PidFile，QOSResponse（UNIX の場合），ScriptAlias，ScriptAliasMatch，ScriptLog，ServerRoot，SSLCACertificateFile，SSLCACertificatePath（UNIX の場合），SSLCacheServerPath（UNIX の場合），SSLCacheServerPort（UNIX の場合），SSLCacheServerRunDir（UNIX の場合），SSLCertificateFile，SSLCertificateKeyFile，SSLCertificateKeyPassword，SSLCRLDERPath，SSLCRLPEMPath，TypesConfig，UserDir	次のディレクティブの設定に合致するように，退避したディレクトリおよびファイルを設定してください。 <Directory>，<DirectoryMatch>，AccessFileName，Alias，AliasMatch，AuthGroupFile，AuthUserFile，CoreDumpDirectory（UNIX の場合），CustomLog，DocumentRoot，ErrorLog，HWSRequestLog，HWSTraceIdFile，HWSTraceLogFile，Include，LoadFile，PidFile，QOSResponse（UNIX の場合），ScriptAlias，ScriptAliasMatch，ScriptLog，ServerRoot，SSLCACertificateFile，SSLCACertificatePath（UNIX の場合），SSLCacheServerPath（UNIX の場合），SSLCacheServerPort（UNIX の場合），SSLCacheServerRunDir（UNIX の場合），SSLCertificateFile，SSLCertificateKeyFile，SSLCertificateKeyPassword，SSLCRLDERPath，SSLCRLPEMPath，TransferLog，TypesConfig，UserDir なお，移行先でディレクトリおよびファイルの構成が変更される場合は，これらのディレクティブの設定値を見直してください。
	LoadModule ディレクティブに設定されたモジュールは，09-00 より前と 09-00 以降のバージョン間で互換性がありません。09-00 以降が提供するモジュールを使用してください。	移行前の環境で LoadModule ディレクティブに設定していたモジュールについて，09-00 以降が提供するモジュールを使用し，LoadModule ディレクティブの指定先に格納してください。 モジュールの仕様差を確認し，必要であればモジュールに関する設定値を見直してください。

機能	バックアップを取る情報	バックアップを取った情報の設定先
	次のディレクティブに設定されたログファイルのバックアップを取ってください。rotatelogs または rotatelogs2 を使用している場合，ユティリティの退避は不要です。 CustomLog，ErrorLog，HWSRequestLog，TransferLog	設定先なし

注※

存在する場合だけバックアップを取ります。

注

ここで説明していない定義情報については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」およびマニュアル「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)」を参照してください。

10.6.2 更新インストールの場合の定義情報のバックアップ

更新インストールに失敗するといった問題が発生したときに対応できるよう，バックアップを取ります。

バックアップを取るディレクトリを次の表に示します。

表 10–25 バックアップを取るディレクトリ

機能	バックアップを取るディレクトリ
Management Server／統合ユーザ管理	• Windows の場合 <製品のインストールディレクトリ>¥manager • UNIX の場合 /opt/Cosminexus/manager
J2EE サーバ	• Windows の場合 <製品のインストールディレクトリ>¥CC¥server¥usrconf J2EE サーバ用作業ディレクトリ • UNIX の場合 /opt/Cosminexus/CC/server/usrconf J2EE サーバ用作業ディレクトリ
バッチサーバ	• Windows の場合 <製品のインストールディレクトリ>¥CC¥server¥usrconf バッチサーバ用作業ディレクトリ • UNIX の場合 /opt/Cosminexus/CC/server/usrconf バッチサーバ用作業ディレクトリ
サーバ管理コマンド	• Windows の場合 <製品のインストールディレクトリ>¥CC¥admin¥usrconf

機能	バックアップを取るディレクトリ
	• UNIX の場合 /opt/Cosminexus/CC/admin/usrconf
Web サーバ連携	• Windows の場合 <製品のインストールディレクトリ>¥CC¥web¥redirector • UNIX の場合 /opt/Cosminexus/CC/web/redirector

10.7　J2EE サーバまたはバッチサーバの移行コマンドの実行

ここでは，J2EE サーバおよびバッチサーバの移行コマンドである cjenvupdate コマンドの実行について説明します。バッチサーバの場合は，「J2EE サーバ」を「バッチサーバ」と読み替えてください。

10.7.1　バックアップ作業ディレクトリ

J2EE サーバの移行コマンドを実行すると，J2EE サーバの作業ディレクトリ内のファイルを変換する前にバックアップを作成します。このマニュアルでは，バックアップされた作業ディレクトリのことを，バックアップ作業ディレクトリと呼びます。バックアップ作業ディレクトリには，移行前の情報（ファイル）が残ります。移行コマンドの実行が終了したとき，バックアップ作業ディレクトリは自動的に削除されません。

注意事項

- Windows および UNIX 共に，バックアップ作業ディレクトリは作業ディレクトリの直下に作成されます。その際，バックアップ作業ディレクトリ名は，作業ディレクトリに「_old」が付いた名称になります。

  バックアップ作業ディレクトリの例を次に示します。

  Windows の場合

  作業ディレクトリ

  <製品のインストールディレクトリ>¥CC¥server¥public

  バックアップ作業ディレクトリ

  <製品のインストールディレクトリ>¥CC¥server¥public¥public_old

  UNIX の場合

  作業ディレクトリ

  /opt/Cosminexus/CC/server/public

  バックアップ作業ディレクトリ

  /opt/Cosminexus/CC/server/public/public_old

- 作業ディレクトリの移行が必要ないバージョンからのバージョンアップの場合，cjenvupdate コマンドは作業ディレクトリの移行は行いません。この場合，バックアップ作業ディレクトリは作成されません。

10.7.2　J2EE サーバの移行方法

作業ディレクトリおよびユーザ定義ファイルを移行するため，cjenvupdate コマンドを実行します。cjenvupdate コマンドの実行形式を次に示します。

- Windows の場合
  <製品のインストールディレクトリ>¥CC¥server¥bin¥cjenvupdate
- UNIX の場合
  /opt/Cosminexus/CC/server/bin/cjenvupdate

cjenvupdate コマンドを実行すると，作業ディレクトリおよびユーザ定義ファイルが移行されます。
cjenvupdate コマンドのログは次の場所に出力されます。

- Windows の場合
  <製品のインストールディレクトリ>¥CC¥logs
- UNIX の場合
  /opt/Cosminexus/CC/logs

cjenvupdate コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjenvupdate（作業ディレクトリおよびユーザ定義ファイルの移行)」を参照してください。

10.8　リソースアダプタの移行

旧バージョンのアプリケーションサーバで使用していたリソースアダプタを使用する場合，リソースアダプタの移行処理が必要です。

DB Connector やほかのリソースアダプタの場合は，リソースアダプタの移行コマンド（cjrarupdate コマンド）を実行します。ただし，CJMSP リソースアダプタの場合，リソースアダプタの移行コマンド（cjrarupdate コマンド）は使用できません。サーバ管理コマンドを使用して CJMSP リソースアダプタを入れ替えてください。

注意事項

DB Connector を使用しリソース接続を行う場合，Cosminexus でサポートしている JDBC ドライバはリリースノートでご確認ください。

10.8.1　リソースアダプタの移行コマンドの実行

ここでは，リソースアダプタの移行コマンドである cjrarupdate コマンドの実行について説明します。バッチサーバの場合は，「J2EE サーバ」を「バッチサーバ」と読み替えてください。

(1)　バックアップ RAR ディレクトリ

Windows および UNIX 共に，リソースアダプタの移行コマンドを実行すると，J2EE サーバの作業ディレクトリ内のファイルを変換する前にバックアップを作成します。このマニュアルでは，バックアップされた作業ディレクトリのことを，バックアップ RAR ディレクトリと呼びます。バックアップ RAR ディレクトリには，移行前の情報（ファイル）が残ります。リソースアダプタの移行コマンドの実行が終了したとき，バックアップ RAR ディレクトリは自動的に削除されません。

また，リソースアダプタの移行コマンドを複数回実行する場合（例えば，DB Connector 用に 1 回と TP1 Connector 用に 1 回の合計 2 回実行する），バックアップ RAR ディレクトリはコマンド実行ごとに生成され，最終的には複数残ります。

バックアップ RAR ディレクトリの例，および移行処理に失敗した場合に自動で回復する方法について説明します。

- バックアップ RAR ディレクトリの例

  バックアップ RAR ディレクトリは，作業ディレクトリの直下に作成されます。バックアップ RAR ディレクトリ名は，「rarupdate_bk」です。バックアップ RAR ディレクトリの例を次に示します。

- Windows の場合

  作業ディレクトリ

  　<製品のインストールディレクトリ>¥CC¥server¥public

バックアップ RAR ディレクトリ

<製品のインストールディレクトリ>¥CC¥server¥public¥rarupdate_bk

- UNIX の場合

作業ディレクトリ

/opt/Cosminexus/CC/server/public

バックアップ RAR ディレクトリ

/opt/Cosminexus/CC/server/public/rarupdate_bk

すでにバックアップ RAR ディレクトリ（rarupdate_bk）が作成されている場合，既存のバックアップディレクトリは「rarupdate_bk_0000000[n]」（n は 17 けたの番号で年月日時間の情報）にリネームされたあと，新たにバックアップ RAR ディレクトリが作成されます。

- 移行処理に失敗した場合に自動で回復する方法

移行処理に失敗した場合に自動で回復するには，cjrarupdate コマンドに引数-recoverfrom を指定して実行してください。引数-recoverfrom には，バックアップ格納先ディレクトリを指定します。バックアップ処理の開始時，および完了時には，メッセージが出力されます。また，引数-recoverfrom に指定したバックアップ格納先ディレクトリが不正な場合，および回復処理に失敗した場合，エラーメッセージが出力されます。

(2) 移行コマンドの実行

cjrarupdate コマンドを実行します。DB Connector，およびバージョンアップする各リソースアダプタに対して，一回ずつ cjrarupdate コマンドを実行します。

cjrarupdate コマンドの実行形式を次に示します。

- DB Connector を使用している場合
  - Windows の場合
    <製品のインストールディレクトリ>¥CC¥server¥bin¥cjrarupdate -type dbconnector
  - UNIX の場合
    /opt/Cosminexus/CC/server/bin/cjrarupdate -type dbconnector
- DB Connector 以外のリソースアダプタを使用している場合
  - Windows の場合
    <製品のインストールディレクトリ>¥CC¥server¥bin¥cjrarupdate -type rar -f <リソースアダプタのパス>
  - UNIX の場合
    /opt/Cosminexus/CC/server/bin/cjrarupdate -type rar -f <リソースアダプタのパス>

なお，cjrarupdate コマンドのログは，次の場所に出力されます。

- Windows の場合

<製品のインストールディレクトリ>¥CC¥logs

- UNIX の場合

  /opt/Cosminexus/CC/logs

cjrarupdate コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjrarupdate（リソースアダプタのバージョンアップ）」を参照してください。

注意事項

- cjrarupdate コマンドを同時に実行すると正しく環境が移行されないことがあります。コマンドを同時に実行しないでください。
- 複数のリソースアダプタを使用している場合，cjrarupdate コマンドを 1 回実行して，すべてのリソースアダプタを一括で移行できます。また，バックアップ RAR ディレクトリのディレクトリパスを明示的に指定できます。
- TP1/Message Queue-Access を移行する場合に，新しい RAR ファイルが同一バージョンと認識されることがあります。その場合は-force オプションを指定して強制的に移行処理を実行してください。
- cjrarupdate コマンドに引数-recoverfrom を指定して，J2EE アプリケーションに含めて使用するリソースアダプタを自動で回復する場合，移行処理に失敗したときに，回復が完了するまで J2EE サーバに対する操作を実行しないでください。回復が完了する前に J2EE サーバに対する操作を実行すると，回復処理が正しく実行されないことがあります。

10.8.2 CJMSP リソースアダプタの入れ替え

CJMSP プロバイダのリソースアダプタ（CJMSP リソースアダプタ）をバージョンアップする場合は，サーバ管理コマンドを使用して CJMSP リソースアダプタを入れ替えて，設定値を引き継いでください。入れ替え手順を次に示します。

1. cjgetrarprop コマンドを使用して，入れ替え前の CJMSP リソースアダプタの Connector 属性ファイルを取得します。
2. CJMSP リソースアダプタを使用中のアプリケーションがある場合は，アプリケーションを停止します。
3. CJMSP リソースアダプタを削除します。
4. 入れ替え前の CJMSP リソースアダプタを削除します。
5. cjimportres コマンドを使用して，次に示す CJMSP リソースアダプタの RAR ファイルをインポートします。

   Windows の場合

   <製品のインストールディレクトリ>¥CC¥cjmsp¥lib¥cjmsra.rar

UNIX の場合

/opt/Cosminexus/CC/cjmsp/lib/cjmsra.rar

6. cjdeployrar コマンドを使用して，CJMSP リソースアダプタをデプロイします。

7. cjsetrarprop コマンドを使用して，CJMSP リソースアダプタの Connector 属性ファイルを入れ替え前に指定されていた値へ変更します。

コマンドの使用方法については，マニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

10.9 Management Server の移行コマンドの実行

ここでは，Management Server の移行コマンドの実行について説明します。

09-00 より前の製品から移行する場合は，09-00 以降にバージョンアップしたあと，Management Server が動作するホストで mngenvupdate コマンドを実行します。mngenvupdate コマンドの実行形式を次に示します。

- Windows の場合
  <製品のインストールディレクトリ>¥manager¥bin¥mngenvupdate
- UNIX の場合
  /opt/Cosminexus/manager/bin/mngenvupdate

注意事項

- mngenvupdate コマンドは，Administrator 権限（Windows の場合），または root 権限（UNIX の場合）のあるユーザが実行してください。
- mngenvupdate コマンドの実行に失敗した場合，<のインストールディレクトリ>/manager/config ディレクトリの直下に一時ファイル（mserver.cfgxxxxxxxxxxxxxxxxxxxx.tmp）が残る場合がありますが，この一時ファイルを残しても削除しても，そのあとの操作への影響はありません。

10.10　論理サーバの設定情報の再読み込みおよび配布

バージョンアップ前の環境で使用していた設定を引き継ぐため，論理サーバの設定情報の再読み込みが必要です。この作業を行わないと，バージョンアップ前の論理 J2EE サーバの設定が失われることがあります。

Management Server を使用している場合，移行後に Management Server を最初に起動したときに，運用管理ドメインに構成定義された既存の論理 J2EE サーバに対して，接続先ホストから設定情報の読み込みを実施する必要があります。移行の手順中で，既存の論理 J2EE サーバの設定ファイルの内容が自動的に変更されるためです。

読み込みを実施したあと，設定情報の配布も実施してください。読み込みを実施したときにワーニングメッセージが出力され，配布ステータスが空白になった場合には，ワーニングメッセージに従って設定情報を見直してから配布してください。また，配布ステータスが空白以外の場合には，そのまま配布を実施してください。

設定情報の読み込みは，Management Server の運用管理ポータルの論理サーバごとの［サーバの設定読み込み］画面で実施できます。「論理サーバの環境設定」については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.8.28　論理 J2EE サーバの設定読み込み」を参照してください。

注意事項

- 設定情報の読み込みを実施したあと，Management Server の運用管理ポータルの［設定情報の配布］画面，または Smart Composer 機能の cmx_build_system コマンドで，設定情報の配布を実施してください。［設定情報の配布］画面で設定情報の読み込みを実施した際にワーニングメッセージが出力されて，ステータスに「空白」が表示された場合は，ワーニングメッセージに従って設定情報を見直してから配布してください。Smart Composer 機能で設定情報の配布を実施すると，論理 J2EE サーバだけでなく論理 Web サーバの設定情報も配布されます。設定情報が未配布の論理サーバだけを配布したい場合は，-sd オプションを指定して cmx_build_system コマンドを実行してください。［設定情報の配布］画面については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.10.1　設定情報の配布」を参照してください。また，cmx_build_system コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cmx_build_system（Web システムの構築）」を参照してください。

10.11　アプリケーションの設定（新規インストールの場合）

新しい環境で使用するアプリケーションを設定します。アプリケーションの設定手順を次に示します。

1. J2EE アプリケーションが利用するリソースを J2EE サーバに設定します。
   サーバ管理コマンドと属性ファイルを使用して設定します。属性ファイルについては，マニュアル「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)」を参照してください。
   - 記録していたセキュリティグループ（ユーザおよびロール）の情報を基に，セキュリティグループを追加してください。
   - 退避していた属性ファイルを使用してメールコンフィグレーションを作成してください。
   - 退避していた実行時情報付き J2EE リソースアダプタをインポートしてください。
   - データソースを使用していた場合は，退避していた属性ファイルの情報を基に DB Connector を登録し，開始してください。
2. 退避していた実行時情報付き J2EE リソースアダプタをインポートします。
3. J2EE サーバを停止します。
4. cjrarupdate コマンドを実行します。
   コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「2.2 J2EE サーバを操作するコマンド」を参照してください。
5. J2EE サーバを開始します。
6. 「10.3　旧バージョンから 11-40 までの仕様変更の確認」に従い，J2EE アプリケーションを修正します。
   統合ユーザ管理が提供する uastartup.war または uastartup.ear をデプロイしていた場合は，移行先のアプリケーションサーバが提供するものを使用してください。09-00 以降では，次の場所にインストールされています。

   Windows の場合

   <製品のインストールディレクトリ>¥manager¥config

   UNIX の場合

   /opt/Cosminexus/manager/config
7. J2EE アプリケーションをインポートします。
8. cjgetappprop コマンドでアプリケーション統合属性ファイルを取得します。
9. 記録しておいた J2EE アプリケーション内の EJB-JAR，WAR のプロパティ情報，J2EE アプリケーションのカスタマイズ情報を基に，アプリケーション統合属性ファイルを編集します。
10. cjsetappprop コマンドでアプリケーション統合属性ファイルを設定します。

11. J2EE アプリケーションを開始します。

10.12　移行コマンドで作成されたバックアップの削除

アプリケーションの動作確認後，手動でバックアップ作業ディレクトリ，およびバックアップ RAR ディレクトリを削除する必要があります。ただし，保守のためにバックアップ作業ディレクトリ，およびバックアップ RAR ディレクトリを別の場所に保存しておくことをお勧めします。

移行前の作業ディレクトリ，移行後の作業ディレクトリ，バックアップ作業ディレクトリ，およびバックアップ RAR ディレクトリの例を次の表に示します。

表 10–26　ディレクトリの移行の例（Windows の場合）

J2EE サーバのサーバ名称	移行前の作業ディレクトリ	移行後の作業ディレクトリ，およびバックアップ作業ディレクトリ
サーバ A	<製品のインストールディレクトリ>¥CC¥server¥public （デフォルト）	変換後の作業ディレクトリ 　<製品のインストールディレクトリ>¥CC¥server¥public バックアップ作業ディレクトリ 　<製品のインストールディレクトリ>¥CC¥server¥public¥public_old バックアップ RAR ディレクトリ 　<製品のインストールディレクトリ>¥CC¥server¥public¥rarupdate_bk
サーバ B	C:¥work_dir	変換後の作業ディレクトリ 　C:¥work_dir バックアップ作業ディレクトリ 　C:¥work_dir¥work_dir_old バックアップ RAR ディレクトリ 　C:¥work_dir¥rarupdate_bk
サーバ C		
－	D:¥replication_dir	D:¥replication_dir 　未使用のディレクトリは移行されません。

（凡例）

－：作業ディレクトリとして使用しているサーバがないことを示す

表 10–27　ディレクトリの移行の例（UNIX の場合）

J2EE サーバのサーバ名称	移行前の作業ディレクトリ	移行後の作業ディレクトリ，およびバックアップ作業ディレクトリ
サーバ A	/opt/Cosminexus/CC/server/public （デフォルト）	変換後の作業ディレクトリ 　/opt/Cosminexus/CC/server/public

J2EE サーバのサーバ名称	移行前の作業ディレクトリ	移行後の作業ディレクトリ，およびバックアップ作業ディレクトリ
		バックアップ作業ディレクトリ /opt/Cosminexus/CC/server/public/public_old バックアップ RAR ディレクトリ /opt/Cosminexus/CC/server/public/rarupdate_bk
サーバ B サーバ C	/<任意のディレクトリ>/work_dir	変換後の作業ディレクトリ /<任意のディレクトリ>/work_dir バックアップ作業ディレクトリ /<任意のディレクトリ>/work_dir/work_dir_old バックアップ RAR ディレクトリ /<任意のディレクトリ>/work_dir/rarupdate_bk
－	/<任意のディレクトリ>/replication_dir	/<任意のディレクトリ>/replication_dir 未使用のディレクトリは移行されません。

（凡例）

－：作業ディレクトリとして使用しているサーバがないことを示す

上記の表の例では，更新インストール前に，「サーバ A」，「サーバ B」，および「サーバ C」の J2EE サーバがセットアップされています。「サーバ A」，および「サーバ B」では DB Connector を使用しています。これらの J2EE サーバが利用している作業ディレクトリが，Windows の場合は「<製品のインストールディレクトリ>¥CC¥server¥public」と「C:¥work_dir」，UNIX の場合は「/opt/Cosminexus/CC/server/public」と「<任意のディレクトリ>/work_dir」です。Windows の場合の「D:¥replication_dir」，UNIX の場合の「<任意のディレクトリ>/replication_dir」はユーザが複製した作業ディレクトリで，J2EE サーバからは利用されていません。

このような状況で cjenvupdate コマンド，および cjrarupdate コマンドを実行した場合，それぞれ使用中の作業ディレクトリに対して，バックアップが作成されます。アプリケーションの動作確認後，手動でバックアップを削除してください。なお，未使用の作業ディレクトリ「D:¥replication_dir」または「<任意のディレクトリ>/replication_dir」は，移行されません。

注意事項

作業ディレクトリの移行が必要ないバージョンからのバージョンアップの場合，cjenvupdate コマンドは作業ディレクトリを移行しません。この場合，バックアップ作業ディレクトリは作成されません。

10.13　J2EE サーバまたはバッチサーバの移行コマンドで自動的に追加／変更／削除される定義

J2EE サーバまたはバッチサーバのユーザプロパティファイルや作業ディレクトリ内の DD 情報は，旧バージョンのものが 09-70 の環境でもそのまま継続使用されます。ただし，新バージョンで追加された機能，およびデフォルト値が変更された機能があり，互換性を維持するために，J2EE サーバまたはバッチサーバの移行コマンドがユーザプロパティファイルのプロパティや作業ディレクトリ内の DD 情報を部分的に変換します。移行コマンド実行時に追加／変更されるキーと値，およびコマンドで取得した属性ファイルに追加されるタグについて説明します。

また，移行コマンドを実行すると，移行元バージョンから移行先バージョン間のすべてのバージョンごとに移行処理が実施されます。例えば，08-00 から 09-70 へ移行する場合，08-00 から 08-50 への移行，08-50 から 08-53 への移行，08-53 から 08-70 への移行，08-70 から 09-00 への移行，09-00 から 09-50 への移行，09-50 から 09-70 への移行というように処理が実施されます。

10.13.1　移行コマンドで自動的に追加／変更／削除される定義（アプリケーションサーバ Version 8 からの移行）

移行コマンド実行時に追加／変更／削除されるキーと値を次の表に示します。

表 10–28　J2EE サーバの usrconf.properties に自動的に追加される定義（アプリケーションサーバ Version 8 からの移行）

追加されるキーと値	旧バージョンでの環境条件	08-70	08-53	08-50	08-00
webserver.connector.limit.max_post_form_data=-1	－	－	－	－	○
webserver.dbsfo.integrity_mode.enabled=true	－	－	○	○	○
ejbserver.jndi.global.enabled=false	－	○	○	○	○

（凡例）　○：定義が自動的に追加されることを示す　－：該当しない

表 10–29　J2EE サーバの usrconf.cfg に自動的に追加／変更／削除される定義（アプリケーションサーバ Version 8 からの移行）

追加／変更／削除されるキーと値	旧バージョンでの環境条件	08-70	08-53	08-50	08-00
add.jvm.arg=-XX:+HitachiExplicitMemoryCompatibleToV8	－	－	－	－	○
add.jvm.arg=-XX:-HitachiOutOfMemoryHandling	－	－	－	○	○

追加／変更／削除されるキーと値	旧バージョンでの環境条件	08-70	08-53	08-50	08-00
add.jvm.arg=-XX:MetaspaceSize=<size> <size>は-XX:PermSize の指定値	-XX:PermSize を指定している場合	○	○	○	○
add.jvm.arg=-XX:MaxMetaspaceSize=<size> <size>は-XX:MaxPermSize の指定値	-XX:MaxPermSize を指定している場合	○	○	○	○
add.jvm.arg=-XX:+UseParNewGC	−	×	×	×	×

（凡例）　○：定義が自動的に追加／変更されることを示す　×：定義が自動的に削除されることを示す　−：該当しない

表 10-30　J2EE サーバの server.policy に自動的に追加される定義（アプリケーションサーバ Version 8 からの移行）

追加される定義	旧バージョンでの環境条件	08-70	08-53	08-50	08-00

grant codeBase "file:${cosminexus.home}/jaxws/lib/*" {
permission java.security.AllPermission;
};
	−	−	−	−	○

grant codeBase "file:${ejbserver.http.root}/ejb/${ejbserver.serverName}/rarjars/-" {
permission java.lang.RuntimePermission "getClassLoader";
permission java.lang.RuntimePermission "setContextClassLoader";
permission java.lang.RuntimePermission "accessDeclaredMembers";
};
	−	−	−	○	○

grant codeBase "file:${ejbserver.http.root}/web/${ejbserver.serverName}/-" {
permission javax.security.auth.AuthPermission "getSubject";
permission javax.security.auth.AuthPermission "createLoginContext.*";
};
	−	−	○	○	○

grant codeBase "file:${cosminexus.home}/manager/modules/-" { permission java.io.FilePermission "<<ALL FILES>>", "read";	−	−	○	○	○

追加される定義	旧バージョンでの環境条件	08-70	08-53	08-50	08-00
permission javax.security.auth.AuthPermission "modifyPrincipals"; permission javax.security.auth.AuthPermission "modifyPublicCredentials"; };					

grant codeBase "file:${cosminexus.home}/common/lib/*" {
permission java.security.AllPermission;
};
	−	○	○	○	○

grant codeBase "file:${ejbserver.install.root}/weld/lib/*" {
permission java.security.AllPermission;
};
	−	○	○	○	○

grant codeBase "file:${cosminexus.home}/jaxrs/lib/*" {
permission java.security.AllPermission;
};
	−	○	○	○	○

（凡例）　○：定義が自動的に追加されることを示す　−：該当しない

表 10–31　サーバ管理コマンドの usrconf.bat，または usrconf に自動的に追加／変更／削除される定義（アプリケーションサーバ Version 8 からの移行）

追加／変更／削除されるキーと値	旧バージョンでの環境条件	08-70	08-53	08-50	08-00
USRCONF_JVM_ARGS 内の-XX:MetaspaceSize=<size> <size>は-XX:PermSize の指定値	-XX:PermSize を指定している場合	○	○	○	○
USRCONF_JVM_ARGS 内の-XX:MaxMetaspaceSize=<size> <size>は-XX:MaxPermSize の指定値	-XX:MaxPermSize を指定している場合	○	○	○	○

（凡例）　○：定義が自動的に追加／変更されることを示す

10.13.2　移行コマンドで自動的に追加／変更／削除される定義（アプリケーションサーバ Version 9 からの移行）

移行コマンド実行時に変更／削除されるキーと値を次の表に示します。

表 10–32　J2EE サーバの usrconf.cfg に自動的に追加／変更／削除される定義（アプリケーションサーバ Version 9 からの移行）

追加／変更／削除されるキーと値	旧バージョンでの環境条件	09-60	09-50	09-00
add.jvm.arg=-XX:MetaspaceSize=<size> <size>は-XX:PermSize の指定値	-XX:PermSize を指定している場合	○	○	○
add.jvm.arg=-XX:MaxMetaspaceSize=<size> <size>は-XX:MaxPermSize の指定値	-XX:MaxPermSize を指定している場合	○	○	○
add.jvm.arg=-XX:+UseParNewGC	－	×	×	×

（凡例）　○：定義が自動的に追加／変更されることを示す　×：定義が自動的に削除されることを示す　－：該当しない

表 10–33　サーバ管理コマンドの usrconf.bat，または usrconf に自動的に追加／変更／削除される定義（アプリケーションサーバ Version 9 からの移行）

追加／変更／削除されるキーと値	旧バージョンでの環境条件	09-60	09-50	09-00
USRCONF_JVM_ARGS 内の-XX:MetaspaceSize=<size> <size>は-XX:PermSize の指定値	-XX:PermSize を指定している場合	○	○	○
USRCONF_JVM_ARGS 内の-XX:MaxMetaspaceSize=<size> <size>は-XX:MaxPermSize の指定値	-XX:MaxPermSize を指定している場合	○	○	○

（凡例）　○：定義が自動的に追加／変更されることを示す

表 10–34　J2EE サーバの usrconf.properties に自動的に追加される定義（アプリケーションサーバ Version 9 からの移行）

追加／変更／削除されるキーと値	旧バージョンでの環境条件	09-87	09-70
ejbserver.rmi.remote.listener.port=0	ejbserver.rmi.remote.listener.port プロパティが存在しない場合	－	○
ejbserver.http.port=8080	ejbserver.http.port プロパティが存在しない場合	○	○

（凡例）　○：定義が自動的に追加／変更されることを示す　－：該当しない

10.13.3　移行コマンドで自動的に追加／変更／削除される定義（アプリケーションサーバ Version 11 からの移行）

移行コマンド実行時に追加されるキーと値を次の表に示します。

表 10–35　J2EE サーバの usrconf.properties に自動的に追加される定義（アプリケーションサーバ Version 11 からの移行）

追加されるキーと値	旧バージョンでの環境条件	11-10	11-00
ejbserver.javaee.cdi.webInfLibJarsEnabled=false	ejbserver.javaee.cdi.webInfLibJarsEnabled が存在しない場合	○	○

（凡例）　○：定義が自動的に追加／変更されることを示す

11 推奨機能への移行

この章では，アプリケーションサーバで従来サポートしている機能から推奨機能への移行について説明します。

11.1 HiRDB Type4 JDBC Driver を使用したデータベース接続への移行時の注意事項

DABroker Library を使用したデータベース接続から HiRDB Type4 JDBC Driver を使用したデータベース接続へ移行する場合，マニュアル「HiRDB UAP 開発ガイド」の DABroker for Java からの移行の説明を参照してください。

11.2 DABroker Library から Oracle JDBC Thin Driver を使用したデータベース接続への移行

DABroker Library と Oracle JDBC Thin Driver の JDBC ドライバには仕様差があります。そのため，Oracle への接続方法を DABroker Library から Oracle JDBC Thin Driver へ移行する場合，互換性を保つために Connector 属性ファイルの config-property タグに次のプロパティを設定してください。

- appendZero

  Connector 属性ファイルで appendZero プロパティを true に設定してください。

  appendZero プロパティの詳細については，マニュアル「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)」の「4.1.10　DB Connector に設定する<config-property>タグに指定できるプロパティ」を参照してください。

- forceFixedString

  Connector 属性ファイルで forceFixedString プロパティを true に設定してください。

  forceFixedString プロパティの詳細については，マニュアル「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)」の「4.1.10　DB Connector に設定する<config-property>タグに指定できるプロパティ」を参照してください。

12 アプリケーションサーバ 09-87 から 11-40 への移行

アプリケーションサーバ 09-87 から 11-40 への移行について説明します。

12.1 概要

アプリケーションサーバ 11-00 では，Servlet 3.1 や WebSocket 1.0 などの Java EE 7 の新仕様に対応しています。これによって，Java EE 7 の仕様を前提とする Hitachi Application Framework や他社製フレームワークに対応しています。

11-40 では Java EE 7/8 の仕様に対応するため，09-87 以前との互換性が保てない機能変更があります。この章では 09-87 から 11-40 へ移行する場合の，機能変更と移行方法について説明します。

12.2　09-87 から 11-40 までの変更点

09-87 から 11-40 までの変更点について説明します。

12.2.1　NIO HTTP サーバ機能

アプリケーションサーバ 11-40 では，Servlet 3.0 の非同期サーブレット，Servlet 3.1 の非同期 I/O API，WebSocket 1.1 など，主に Web アプリケーションを非同期で処理する機能をサポートしています。これに伴い，従来の同期処理を前提としていた「リダイレクタによる Web サーバ連携機能」および「インプロセス HTTP サーバ機能」を刷新し，Java NIO の技術を用いて実装したインプロセスの HTTP サーバ機能である「NIO HTTP サーバ機能」を搭載しています。

アプリケーションサーバ 11-40 では，NIO HTTP サーバ機能によるノンブロッキング I/O を前提としているため，従来のリダイレクタ機能やインプロセス HTTP サーバ機能は使用できません。J2EE サーバ内の Web アプリケーションへのリクエストは，Web サーバを介する場合と直接 J2EE サーバにアクセスする場合，NIO HTTP サーバ機能による HTTP 通信に一本化されます。

そのため，リダイレクタ機能やインプロセス HTTP サーバ機能に対するシステム設計を，NIO HTTP サーバ向けの設計に見直す必要があります。

詳細は，「12.4　システム設計の移行ガイド」を参照してください。

図 12–1　リダイレクタによる Web サーバ連携構成をアプリケーションサーバ 11-40 に移行する場合

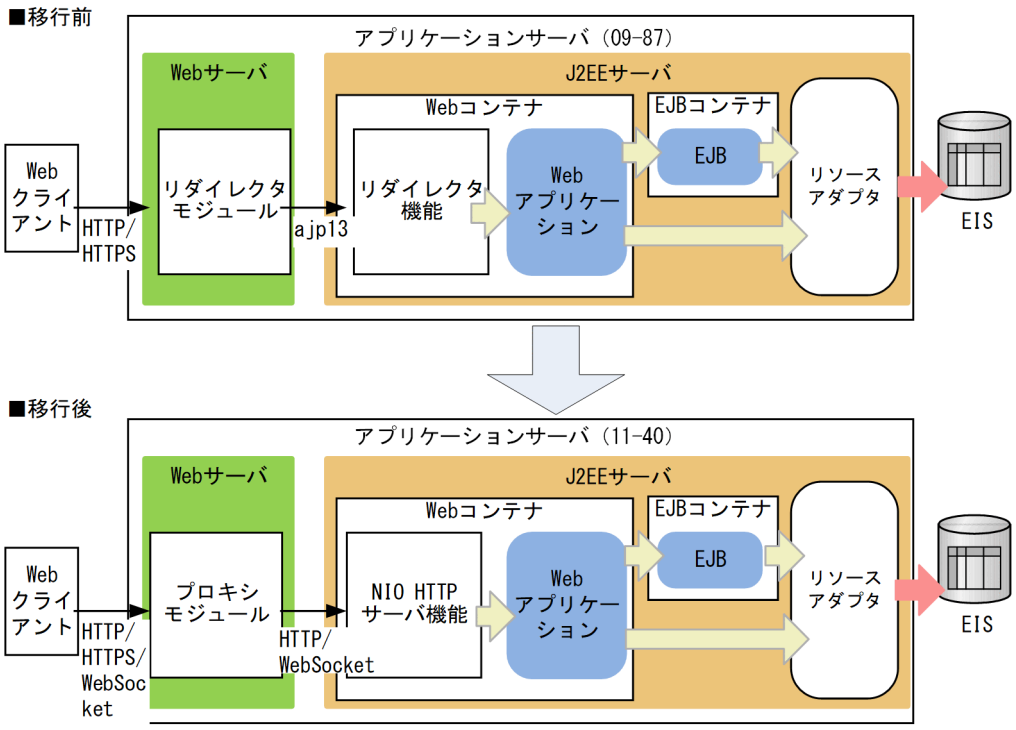
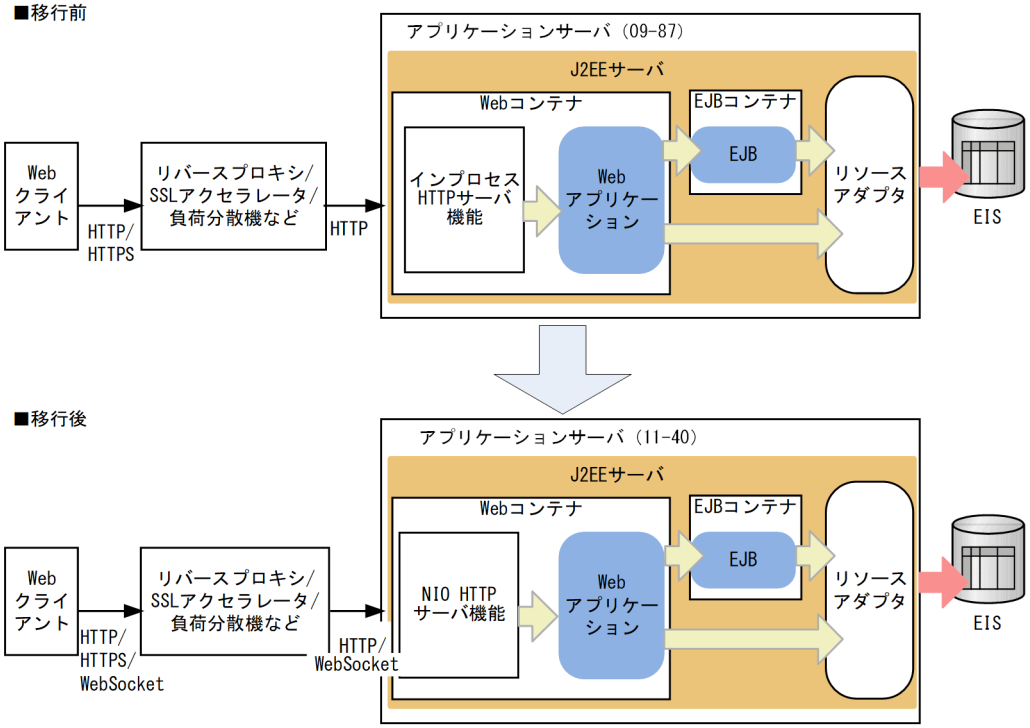


図 12–2　インプロセス HTTP サーバ構成をアプリケーションサーバ 11-40 に移行する場合



12.2.2　Java EE 7/8 新仕様対応

アプリケーションサーバ 11-40 では，次の Java EE コンポーネントについてサポートバージョンやサポート範囲を変更しています。一部の仕様については，アプリケーション側で移行作業が必要です。該当する仕様をアプリケーション内で利用している場合は，各仕様の参照先を確認してください。

表 12–1　Java EE 7/8　標準仕様

仕様名	アプリケーションサーバでのサポートバージョン		参照先
	09-87	11-40※3	
Java Servlet (Servlet)	3.0	4.0	12.3.2
Contexts and Dependency Injection for Java (CDI)	1.0	2.0	12.3.3
Java API for RESTful Web Services (JAX-RS)	1.1	2.1	12.3.4
Java Persistence API (JPA)	1.0	2.2	12.3.5
Java Server Faces (JSF)	2.1	2.3	12.3.6
JavaServer Pages (JSP)	2.1	2.3	移行不要
Expression Language (EL)	2.2	3.0	
Bean Validation (BV)	1.0	2.0	

仕様名	アプリケーションサーバでのサポートバージョン		参照先
	09-87	11-40※3	
Common Annotations for the Java Platform (Common Annotations)	1.1	1.2	
Java Transaction API (JTA)	1.1	1.2※1	
Batch Applications for the Java Platform (JavaBatch)	－	1.0	
Java API for JSON Processing (JSON-P)	－	1.1	
Java API for JSON Binding (JSON-B)	－	1.0	
Java API for WebSocket (WebSocket)	－	1.1	
Concurrency Utilities for Java EE	－	1.0※2	
Interceptors	1.1	1.2	

注

アプリケーションサーバ 09-87 から変更のない仕様については記載を省略しています。

注※1

09-87 以前までの機能に加えて，@javax.transaction.Transactional アノテーションをサポートします。

注※2

ContextService 機能は使用できません。

注※3

パッケージ名変換機能でサポートする Jakarta EE アプリケーションのバージョンは，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「20.5.1　Jakarta EE アプリケーションのサポート範囲」を参照してください。

12.2.3　V9 互換モード

アプリケーションサーバ 11-40 では，新機能は使用しないで 09-87 以前からの互換性を重視するシステム向けや，uCosminexus Service Platform のユーザ向けに，アプリケーションサーバの動作を 09-87 相当の仕様に戻す「V9 互換モード」を提供しています。

V9 互換モードについては，マニュアル「アプリケーションサーバ 機能解説 互換編」を参照してください。

なお，この章では V9 互換モードを使用しないユーザを前提に説明しています。

12.2.4　11-40 で非サポートの機能

アプリケーションサーバ 11-40 には，09-87 からの移行をサポートしていない機能があります。なお，一部の機能を除き，V9 互換モードでは使用できます。

(1)　非サポートの Web コンテナの機能

(a)　ラウンドロビン方式によるリクエストの振り分け

アプリケーションサーバ 11-40 の Web サーバに内蔵されたプロキシモジュールでは，09-87 のリダイレクタによる Web サーバ連携でサポートしていたラウンドロビン方式によるリクエストの振り分け（ロードバランサ）をサポートしていません。

ラウンドロビン方式によるリクエストの振り分けを実現するには，負荷分散機やロードバランサ機能を持つ Web サーバが別途必要です。

(b)　POST データサイズでのリクエストの振り分け

アプリケーションサーバ 11-40 の Web サーバに内蔵されたプロキシモジュールでは，09-87 のリダイレクタによる Web サーバ連携でサポートしていた，POST データサイズでのリクエストの振り分けをサポートしていません。

POST データサイズによるリクエストの振り分けを実現するには，その機能を持つ負荷分散機やロードバランサ機能を持つ Web サーバが別途必要です。

(c)　J2EE サーバによるゲートウェイ指定機能

アプリケーションサーバ 11-40 の NIO HTTP サーバには，09-87 のリダイレクタやインプロセス HTTP サーバで提供していた「ゲートウェイ指定機能」がありません。

URL 転送の際に Location ヘッダに付与する URL として，負荷分散機や SSL アクセラレータなど，ゲートウェイの URL をクライアントに返却する必要がある場合，負荷分散機や SSL アクセラレータの機能で Location ヘッダの書き換えなどの適切なゲートウェイの URL を返す設定をしてください。

Web コンテナが生成したセッション ID を Cookie によってクライアントに返すとき，Cookie に secure 属性を付与するには，Servlet 標準仕様に従って web.xml（/web-app/session-config/cookie-config/secure 要素）で設定してください。または，Cosminexus HTTP Server でのレスポンスヘッダ書き換えをご検討ください。詳細はマニュアル「HTTP Server」の「4.10.2(3)　バックエンドのレスポンスヘッダ値を書き換える場合」を参照してください。

「ゲートウェイ指定機能」使用時に動作が変わっていた次の API は，Servlet 標準仕様およびマニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「8.2　サーブレットおよび JSP の実装時の注意事項」に従って動作します。

- javax.servlet.http.HttpServletResponse クラスの sendRedirect メソッド
- javax.servlet.ServletRequest インタフェースの getRequestURL メソッド

- javax.servlet.ServletRequest インタフェースの getServerName メソッド
- javax.servlet.ServletRequest インタフェースの getServerPort メソッド
- javax.servlet.ServletRequest インタフェースの getScheme メソッド
- javax.servlet.ServletRequest インタフェースの isSecure メソッド

また，「ゲートウェイ指定機能」使用時に javax.servlet.ServletRequest インタフェースの getAttribute メソッドで取得できない次の属性は，NIO HTTP サーバでも取得できません。

- javax.servlet.request.cipher_suite
- javax.servlet.request.key_size
- javax.servlet.request.X509Certificate

(d) Web コンテナ単位のエラーページのカスタマイズ機能

アプリケーションサーバ 11-40 の NIO HTTP サーバには，09-87 のリダイレクタやインプロセス HTTP サーバで提供していた「Web コンテナ単位のエラーページのカスタマイズ機能」がありません。

エラーステータスコードに応じたユーザ独自のエラーページをレスポンスとして返したい場合は，Servlet 仕様に従ってアプリケーション単位で定義するか，または Web サーバのリバースプロキシ機能を介した Web サーバ連携を構成し，Web サーバ側の機能を使用してエラーページをカスタマイズしてください。

(e) 有効な HTTP メソッドの制限機能

アプリケーションサーバ 11-40 の NIO HTTP サーバには，09-87 のインプロセス HTTP サーバで提供していた「有効な HTTP メソッドの制限機能」がありません。

有効な HTTP メソッドの制限が必要な場合は，Web サーバのリバースプロキシ機能を介した Web サーバ連携を構成し，Web サーバ側の機能を使用して有効な HTTP メソッドを制限してください。

(f) リダイレクトによるリクエストの振り分け機能

アプリケーションサーバ 11-40 の NIO HTTP サーバには，09-87 のインプロセス HTTP サーバで提供していた「リダイレクトによるリクエストの振り分け機能」がありません。

URL 単位でリダイレクトが必要な場合は，Web サーバのリバースプロキシ機能を介した Web サーバ連携を構成し，Web サーバ側の機能を使用して URL のリダイレクトをしてください。

(2) 非サポートの拡張機能

アプリケーションサーバ 11-40 では，「EADs セッションフェイルオーバ機能」をサポートしていません。この機能は，V9 互換モードでも使用できません。

(3) 非サポートの互換機能

アプリケーションサーバ 09-87 以前から，前バージョンとの互換性のために残されていた次の互換機能は，アプリケーションサーバ 11-40 では使用できません。これらの機能は，V9 互換モードでも使用できません。

表 12-2 アプリケーションサーバ 11-40 では非サポートの互換機能

分類	機能名	代替機能
互換機能	ベーシックモード	J2EE サーバを使用してください。
	サーブレットエンジンモード (Web コンテナサーバ)	J2EE サーバを使用してください。
	簡易 Web サーバ機能	NIO HTTP サーバ機能を使用してください。 詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「7. NIO HTTP サーバ」を参照してください。
	EJB クライアントアプリケーションの ログのサブディレクトリ専有モード	デフォルトのサブディレクトリ共有モードを使用してください。
	メモリセッションフェイルオーバ機能	データベースセッションフェイルオーバ機能を使用してください。 詳細は，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「6. データベースセッションフェイルオーバ機能」を参照してください。
	複数の構築済み実行環境の切り替え	なし
	J2EE アプリケーションのテスト機能	なし
	JSP1.1 仕様および JSP1.2 仕様に準拠した JSP ソースのチェック (cjjsp2java)	なし
	論理サーバのアプリケーション管理の V7 互換モード	なし
セキュリティ管理機能	セッションフェイルオーバ機能を使用したログイン状態の引き継ぎ	なし
コマンド	Management Server で使用するコマンド (旧バージョンとの互換用のコマンド) • cmx_define_application コマンド • cmx_deploy_application コマンド • cmx_register_application コマンド • cmx_start_application コマンド • cmx_stop_application コマンド • cmx_undefine_application コマンド • cmx_undeploy_application コマンド • cmx_unregister_application コマンド	J2EE サーバのコマンドを使用してください。 詳細は，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「2. J2EE サーバで使用するコマンド」を参照してください。

分類	機能名	代替機能
	• cmx_define_resource コマンド • cmx_deploy_resource コマンド • cmx_register_resource コマンド • cmx_start_resource コマンド • cmx_stop_resource コマンド • cmx_undefine_resource コマンド • cmx_undeploy_resource コマンド • cmx_unregister_resource コマンド • cmx_add_serverref コマンド • cmx_delete_serverref コマンド	

(4) 非サポートになった Cosminexus TPBroker の機能

Windows 版アプリケーションサーバ 11-40 の Cosminexus TPBroker には，11-00 以前にあった tssetfw コマンドがありません。Cosminexus TPBroker の Windows ファイアウォール例外リストの登録については，netsh コマンドを使用してください。netsh コマンドの使用方法については，マニュアル「アプリケーションサーバ 機能解説 セキュリティ管理機能編」を参照してください。

12.3　アプリケーションの移行ガイド

ここでは，ユーザアプリケーションの移行手続きについて説明します。

12.3.1　代替機能への移行

アプリケーションサーバ 09-87 で使用していた機能は，アプリケーションサーバ 11-40 で動作させるために，代替機能への移行が必要な場合があります。次の表を確認して，必要な場合はアプリケーションを移行してください。表に記載されていない機能についてはアプリケーションサーバ 09-87 と互換性があります。

表 12–3　アプリケーションの移行が必要な場合

機能	移行が必要な場合	参照先
Servlet	Servlet 3.0 の非サポート API を使用していて，UnsupportedOperationException 例外がスローされることを期待していた場合	12.3.2(1)
	動的サーブレット定義 API を使用して，デフォルトサーブレット（URL パターンが”/”）にマッピングするように ServletRegistration の addMapping メソッドを呼び出しているが，正常にマッピングされることを期待していなかった場合	12.3.2(2)
CDI	CDI のアノテーションをアプリケーション内で使用しているが，beans.xml をアプリケーションアーカイブ内に格納していない状態で，CDI が動作することは期待していなかった場合	12.3.3(1)
	Portable Extension の実装クラスを持つライブラリをクラスパスに追加しているか，またはアプリケーション内に格納しているが，Portable Extension が動作することを期待していなかった場合	12.3.3(2)
JAX-RS	Cosminexus JAX-RS エンジンを使用していた場合（「cjjaxrs.jar」をクラスパスに追加していた場合）	12.3.4(1)
	Cosminexus JAX-RS エンジン独自のパラメタを使用していた場合	12.3.4(2)
	Cosminexus JAX-RS エンジン独自の RESTful Web サービス用クライアント API を使用していた場合	12.3.4(3)
JPA	CJPA プロバイダ独自のパラメタを使用していた場合（「cosminexus.jpa」から始まる名称のユーザプロパティ，または persistence.xml の<property>タグ指定値）	12.3.5(1)
	永続化先データベースに HiRDB 以外を使用していた場合	12.3.5(2)
JSF	JSF を使用していた場合（「cjsf.jar」をクラスパスに追加していた場合）	12.3.6(1)

12.3.2　Servlet の変更点

(1)　Servlet 3.0 の非サポート API の扱い

アプリケーションサーバ 11-40 では，09-87 で非サポートだった一部の Servlet 3.0 API がサポート対象になっています。そのため，アプリケーションサーバ 09-87 では該当する API を使用すると UnsupportedOperationException 例外がスローされる仕様となっていましたが，アプリケーションサーバ 11-40 では Servlet 3.0 仕様に従って動作します。

該当する API を次の表に示します。もし，これらの API から常に UnsupportedOperationException 例外がスローされることを期待しているアプリケーションだった場合は，例外がスローされない前提の実装となるようにアプリケーションの改修が必要です。

表 12–4　アプリケーションサーバ 11-40 でサポートしている Servlet 3.0 API

パッケージ	クラス	メソッド
javax.servlet	AsyncContext	すべてのメソッド
	AsyncListener	すべてのメソッド
	ServletRequest	startAsync
		isAsyncStarted
		isAsyncSupported
		getAsyncContext
		getDispatcherType
	AsyncEvent	すべてのメソッド
	ServletRequestWrapper	startAsync
		isAsyncStarted
		isAsyncSupported
		getAsyncContext
		getDispatcherType
	Registration	setAsyncSupported

また，アプリケーションサーバ 11-40 で引き続き非サポートとなる API についても，アプリケーションサーバ 09-87 では UnsupportedOperationException 例外がスローされていましたが，アプリケーションサーバ 11-40 ではデフォルトでは例外がスローされません。該当する API を次の表に示します。

表 12–5　アプリケーションサーバ 11-40 でも非サポートの Servlet 3.0 API

パッケージ	クラス	メソッド	webserver.servlet_api.unsupported.throwUnsupportedOperationException の指定値に対する動作	
			false（デフォルト）	true
javax.servlet	ServletContext	getJspConfigDescriptor	null が返ります	UnsupportedOperationException 例外がスローされます

もし，これらの API から常に UnsupportedOperationException 例外がスローされることを期待しているアプリケーションだった場合は，例外がスローされない前提の実装となるようにアプリケーションを改修するか，または次の定義を J2EE サーバ用ユーザプロパティに追加してください。

```
webserver.servlet_api.unsupported.throwUnsupportedOperationException=true
```

(2)　デフォルトサーブレットへマッピングする動的サーブレット追加処理について

アプリケーションサーバ 09-87 では，javax.servlet.ServletRegistration の addMapping メソッドの引数にデフォルトサーブレットへのマッピングを示す”/”を指定した場合，すでにアプリケーションサーバが提供するデフォルトサーブレットがマッピング済みであるためにマッピングの上書きができませんでした。

アプリケーションサーバ 11-40 では，同一 ServletContext に対して 1 度だけ，デフォルトサーブレットに対するマッピングを上書きできます。これによって，アプリケーションサーバが提供するデフォルトサーブレットの代わりにユーザ定義サーブレットをデフォルトサーブレットにマッピングできます。

もし，アプリケーションサーバ 11-40 の動作を期待しておらず，addMapping メソッドがマッピングの上書きを拒否することを期待しているアプリケーションだった場合，デフォルトサーブレットにマッピングする addMapping メソッドの呼び出しをしないようにアプリケーションの改修が必要です。アプリケーションを改修しないでアプリケーションサーバ 11-40 に移行すると，デフォルトサーブレットへのマッピングがユーザ定義のサーブレットへのマッピングに上書きされるおそれがあります。

12.3.3　CDI の変更点

(1)　beans.xml 省略時の CDI 有効／無効判定条件の変更

アプリケーションサーバ 11-40 では，beans.xml ファイルの扱いと CDI 有効／無効の判定条件は CDI 2.0 仕様に従います。

CDI 1.2 以降の仕様では，beans.xml が格納されていないアプリケーションアーカイブであっても，CDI のアノテーションが使用されていると判定されると自動的に CDI が有効化されます。そのため，アプリケーションサーバ 09-87 でデプロイできていたアプリケーションであっても，CDI が有効になることが

想定されていないアプリケーションの場合は，アプリケーションサーバ 11-40 ではデプロイに失敗するおそれがあります。

CDI 1.2 以降の仕様では，そのような事態を避けるために CDI 1.0 基準の判定条件に戻すオプションを提供するよう規定されています。アプリケーションサーバ 11-40 でもそのオプションを提供しています。CDI 有効／無効の判定条件を CDI 1.0 基準に戻すには，次の定義を J2EE サーバ用ユーザプロパティに追加してください。

```
ejbserver.javaee.cdi.beansXmlRequired=true
```

(2) Portable Extension API の扱いの変更

アプリケーションサーバ 09-87 では CDI の Portable Extension API は非サポートとなっていて，Portable Extension の実装クラスがクラスパス内に存在していても無視されていました。

アプリケーションサーバ 11-40 でもユーザアプリケーションによる Portable Extension API の使用は引き続き非サポートとなりますが，製品内部で CDI の Portable Extension API を使用するため Portable Extension の実装クラスが動作します。

Portable Extension の実装クラス（javax.enterprise.inject.spi.Extension インタフェースを実装したクラス）とそのサービス定義ファイル（META-INF/services/javax.enterprise.inject.spi.Extension ファイル）が格納された JAR ファイルは，J2EE サーバのクラスパスに追加したり，アプリケーション内に格納したりしないでください。Portable Extension の実装クラスが動作して，アプリケーションのデプロイに失敗したり，アプリケーションの動作が変わったりするおそれがあります。

12.3.4 JAX-RS の変更点

(1) JAX-RS エンジンの変更

アプリケーションサーバ 11-40 では，JAX-RS 2.1 をサポートする JAX-RS 2.1 エンジンを提供しています。従来のアプリケーションサーバ 09-87 以前の JAX-RS エンジンとは異なる JAR ファイルに格納されているため，JAX-RS エンジンを利用するために必要なクラスパスの設定が変わります。

アプリケーションサーバ 09-87 以前の場合

```
add.class.path=<cosminexus.home>¥jaxrs¥lib¥cjjaxrs.jar
```

アプリケーションサーバ 11-40 の場合

```
add.class.path=<cosminexus.home>¥CC¥javaee¥1100¥lib¥jaxrs-impl.jar
add.class.path=<cosminexus.home>¥CC¥javaee¥1100¥lib¥jaxrs-jackson.jar
```

アプリケーションサーバ 09-87 以前の JAX-RS エンジン（cjjaxrs.jar）がクラスパスに入ったままになっている場合，アプリケーションが正常に動作しなくなるため，必ずクラスパスの設定を見直してください。

(2) JAX-RS エンジンの Cosminexus 独自パラメタの廃止

アプリケーションサーバ 11-40 の JAX-RS 2.1 エンジンは，アプリケーションサーバ 09-87 以前の JAX-RS エンジンでサポートしていた Cosminexus 独自定義ファイルや独自パラメタをサポートしていません。マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「13.1　動作定義ファイル」に記載されている定義ファイルやパラメタはアプリケーションサーバ 11-40 では使用できません。

(3) RESTful Web サービス用クライアント API の廃止

JAX-RS 2.0 仕様では，新たにクライアント API が標準化されました。これに伴い，アプリケーションサーバ 09-87 以前の JAX-RS エンジンでサポートしていた Cosminexus 独自の RESTful Web サービス用クライアント API をサポートしていません。マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「25.　RESTful Web サービス用クライアント API のサポート範囲」に記載されている機能は，アプリケーションサーバ 11-40 では使用できません。

JAX-RS のクライアント API を使用するアプリケーションについては，JAX-RS 2.0 仕様書および JAX-RS 2.0 API 仕様に従い，JAX-RS 2.0 標準 API に置き換えてください。

JAX-RS 2.0 API 仕様 (Oracle 社提供)

https://docs.oracle.com/javaee/7/api/javax/ws/rs/client/package-summary.html

12.3.5 JPA の変更点

(1) CJPA プロバイダの廃止

従来の CJPA プロバイダは，JPA 2.2 API に対応していないため使用できません。代わりに，アプリケーションサーバ 11-40 では，JPA 2.2 をサポートする JPA プロバイダを新たに提供しています。しかし，従来の CJPA プロバイダで提供していた Cosminexus 独自機能は使用できません。次に示すパラメタは，指定されていても無視されます。

- persistence.xml の<property>タグに指定する「cosminexus.jpa」で始まるプロパティ
- J2EE サーバ用ユーザプロパティファイルに指定する「cosminexus.jpa」で始まるキー
- Smart Composer 機能で論理 J2EE サーバに指定する「cosminexus.jpa」で始まるパラメタ

(2) 永続化先としてサポートされるデータベースの変更

アプリケーションサーバ 11-40 の JPA サポート範囲では，永続化先として使用可能なデータベースは次の製品です。

- HiRDB
- Oracle
- PostgreSQL
- MySQL

それ以外のデータベースへの永続化が必要な場合は，JPA 2.2 と永続化先データベースに対応した JPA プロバイダライブラリ※を用意し，Cosminexus の JPA 2.2 プロバイダの代わりにクラスパスに追加して使用してください。

注※

EclipseLink，Hibernate JPA など

12.3.6 JSF の変更点

(1) JSF ライブラリの変更

アプリケーションサーバ 11-40 では，JSF 2.3 をサポートします。JSF 2.3 用ライブラリは，従来のアプリケーションサーバ 09-87 以前の JSF 2.1 用ライブラリとは異なる JAR ファイルに格納され，デフォルトでクラスパスに追加されています。そのため，JSF を利用するとき，次のクラスパスを追加する必要はなくなりました。

アプリケーションサーバ 09-87 以前の場合

```
add.class.path=<cosminexus.home>¥CC¥lib¥cjsf.jar
add.class.path=<cosminexus.home>¥CC¥lib¥cjstl.jar
```

アプリケーションサーバ 11-40 の場合

クラスパスの追加は不要です。

アプリケーションサーバ 09-87 以前の JSF 2.1 用ライブラリ（cjsf.jar）がクラスパスに入ったままになっている場合，アプリケーションが正常に動作しなくなるため，必ずクラスパスの設定を見直してください。

12.4　システム設計の移行ガイド

この節では，チューニングパラメタやリソース見積もり式などのシステム設計の変更点について説明します。

なお，各パラメタの単位，デフォルト値，指定可能範囲などについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

12.4.1　パフォーマンスチューニング

アプリケーションサーバ 11-40 では，J2EE アプリケーション実行基盤のパフォーマンスチューニングの観点のうち，次の観点が 09-87 以前と異なります。

- 同時実行数の最適化
- タイムアウトの設定

(1)　同時実行数の最適化

アプリケーションサーバ 09-87 以前では，Web コンテナのリクエスト処理スレッドは 1 つのコネクションに対して必ず 1 つのスレッドが割り当てられ，クライアントにレスポンス本文を送信し終えるまでリクエスト処理スレッドを占有していました。

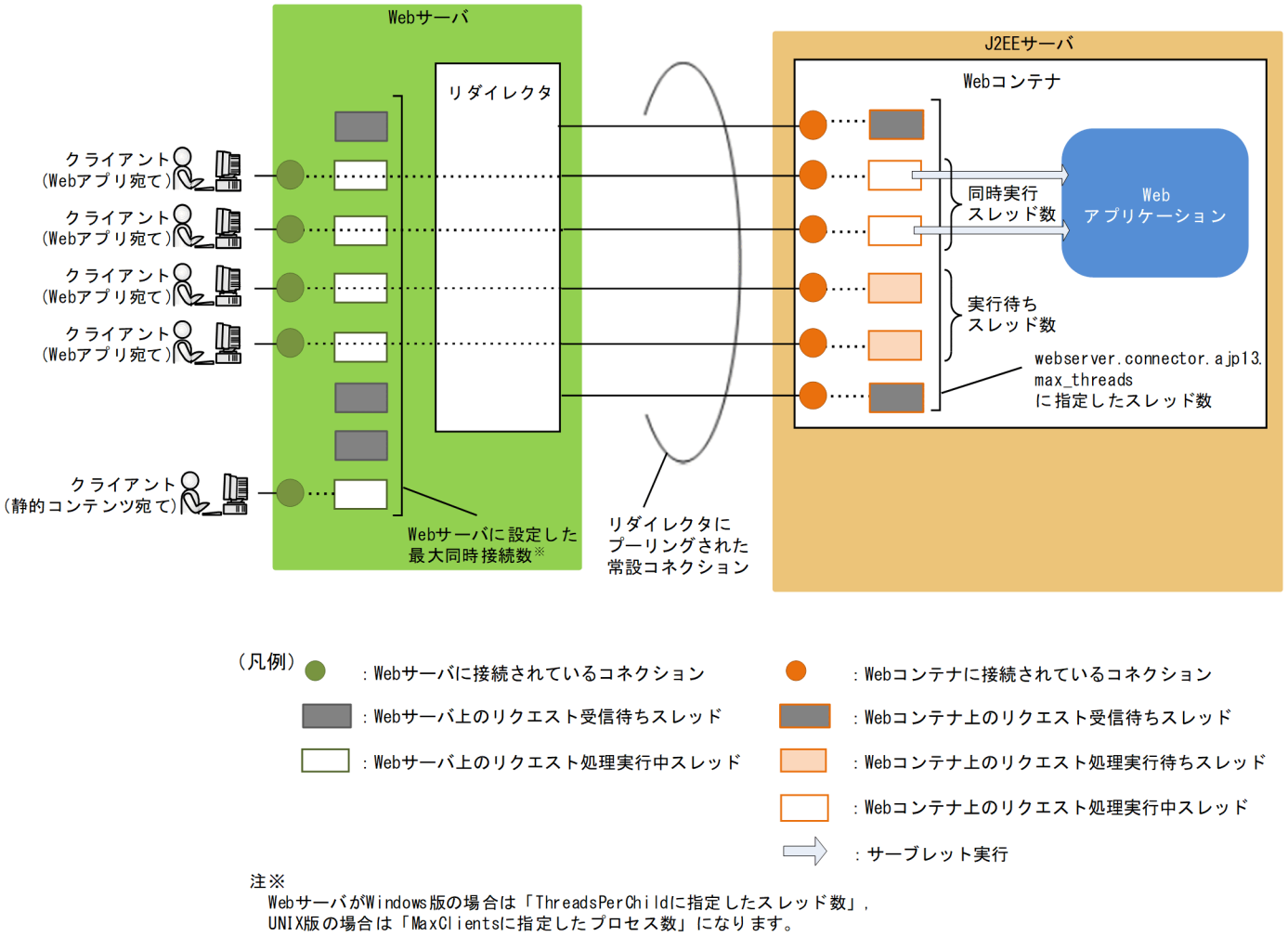
アプリケーションサーバ 11-40 では，Servlet 3.0 の非同期サーブレットや Servlet 3.1 の非同期 I/O API，WebSocket などのノンブロッキング I/O を前提とする標準仕様をサポートするため，リクエストを受けるコネクションの管理と，受け付けたリクエストを処理するスレッドの管理を分離し，1 つのコネクションが複数の処理スレッドを使用することや，リクエストの受付処理とレスポンスの送信処理を別々のスレッドに処理させることができます。

これに伴い，アプリケーションサーバ 11-40 では，09-87 以前の同時実行数制御の考え方とは異なる部分があります。以降では，Web サーバの構成ごとに同時実行数制御の設定ポイントの相違点を説明します。

(a)　リダイレクタによる Web サーバ連携構成から移行する場合

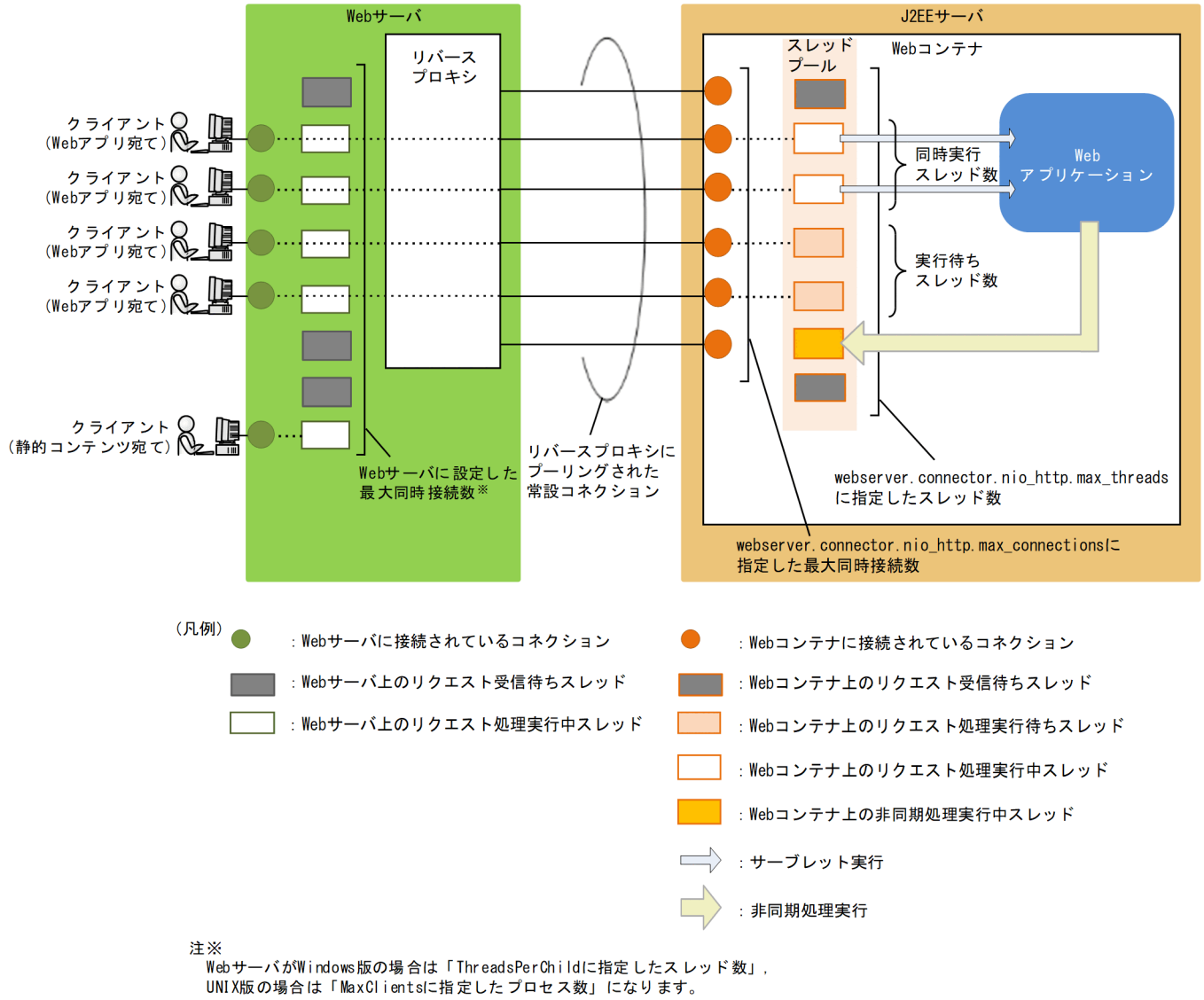
アプリケーションサーバ 09-87 以前で使用していたリダイレクタ機能では，Web サーバと Web コンテナ間を常設コネクションで接続し，Web コンテナ側では 1 つのコネクションごとに 1 つのリクエスト処理スレッドを占有していました。そのため，生成されるスレッド数は同時接続可能な最大コネクション数と常に同じ値になります。また，同時実行スレッド数制御によって，生成されたスレッドのうち同時に実行が可能なスレッドを制御することで，パフォーマンスの低下やリソースの枯渇を抑制できました。アプリケーションサーバ 09-87 のスレッド数制御の仕組みを次の図に示します。

図 12–3　アプリケーションサーバ 09-87 のスレッド数制御の仕組み（Web サーバ連携の場合）



アプリケーションサーバ 11-40 の NIO HTTP サーバ機能では，Web サーバと Web コンテナ間を常設コネクションで接続しても，Web コンテナ側のリクエスト処理スレッドはコネクションに占有されることはありません。Web コンテナ側がデータの受信を検知するたびにスレッドプールからスレッドを割り当てて処理します。また，同時実行スレッド数制御によって，サーブレットのリクエスト処理要求に対して割り当てられたスレッドのうち同時に実行できるスレッドを制御することで，アプリケーションサーバ 09-87 と同様にパフォーマンスの低下やリソースの枯渇を抑制できます。アプリケーションサーバ 11-40 のスレッド数制御の仕組みを次の図に示します。

図 12-4　アプリケーションサーバ 11-40 のスレッド数制御の仕組み（Web サーバ連携の場合）



NIO HTTP サーバ機能では，J2EE サーバ起動時にスレッドプールにあらかじめ生成しておくスレッド数と，動的に生成する最大スレッド数を定義できます。Web サーバ連携構成のアプリケーションサーバ 09-87 から移行する場合に推奨されるスレッド数の設定値を次の表に示します。ただし，WebSocket 機能を使用する場合はこの限りではありません。詳細は「12.4.1(1)(c)　WebSocket 機能を使用するために最大同時接続数を増やす場合の注意点」を参照してください。

表 12-6　推奨される同時実行数設定値（Web サーバ連携構成からの移行の場合）

設定先の NIO HTTP サーバのパラメタ	推奨される設定値	
	アプリケーションサーバ 09-87 以前の機能だけを使用する場合	アプリケーションサーバ 11-40 でサポートされている非同期処理を使用する場合
webserver.connector.nio_http.max_threads	webserver.connector.ajp13.max_threads と同じ値	webserver.connector.ajp13.max_threads の値

設定先の NIO HTTP サーバのパラメタ	推奨される設定値	
	アプリケーションサーバ 09-87 以前の機能だけを使用する場合	アプリケーションサーバ 11-40 でサポートされている非同期処理を使用する場合
		＋コネクション数を超えて同時実行される非同期処理※1 の最大同時実行数
webserver.connector.nio_http.min_threads	webserver.connector.ajp13.max_threads と同じ値	
webserver.connector.nio_http.max_connections	J2EE サーバに接続するコネクション数の総和※2	

注※1

「コネクション数を超えて同時実行される非同期処理」に含まれるものは次のとおりです。

- Servlet 3.0 の javax.servlet.AsyncContext の start メソッドの呼び出し
- WebSocket の javax.websocket.RemoteEndpoint.Async インタフェースの sendBinary メソッド，sendObject メソッド，sendText メソッドの呼び出し

注※2

「J2EE サーバに接続するコネクション数の総和」は，J2EE サーバの NIO HTTP サーバに接続する接続数の総和です。複数のプロセスから J2EE サーバに接続する場合，各プロセスからの最大接続数の総和を設定してください。

なお，Cosminexus HTTP Server のリバースプロキシから J2EE サーバへの最大接続数は，マニュアル「HTTP Server」の「4.7.4(5)　性能に関する注意事項」を参照してください。

(b)　インプロセス HTTP サーバ構成から移行する場合

アプリケーションサーバ 09-87 以前で使用していたインプロセス HTTP サーバ機能では，クライアントから接続要求があるたびに，スレッドプールにあらかじめ生成しておいたスレッドをコネクションごとに割り当て，クライアントとの接続が切れるまでそのスレッドを占有していました。また，同時実行スレッド数制御によって，生成されたスレッドのうち同時に実行できるスレッドを制御することで，パフォーマンスの低下やリソースの枯渇を抑制できました。

アプリケーションサーバ 11-40 の NIO HTTP サーバ機能では，Web コンテナ側のリクエスト処理スレッドはコネクションに占有されることなく，Web コンテナ側がデータの受信を検知するたびにスレッドプールからスレッドを割り当てて処理します。また，同時実行スレッド数制御によって，サーブレットのリクエスト処理要求に対して割り当てられたスレッドのうち同時に実行できるスレッドを制御することで，アプリケーションサーバ 09-87 と同様にパフォーマンスの低下やリソースの枯渇を抑制できます。

NIO HTTP サーバ機能では，J2EE サーバ起動時にスレッドプールにあらかじめ生成しておくスレッド数と，動的に生成する最大スレッド数を定義できます。インプロセス HTTP サーバ構成のアプリケーションサーバ 09-87 から移行する場合に推奨されるスレッド数の設定値を次の表に示します。ただし，WebSocket 機能を使用する場合はこの限りではありません。詳細は「12.4.1(1)(c)　WebSocket 機能を使用するために最大同時接続数を増やす場合の注意点」を参照してください。

表 12–7　推奨される同時実行数設定値（インプロセス HTTP サーバ構成からの移行の場合）

設定先の NIO HTTP サーバのパラメタ	アプリケーションサーバ 11-40 で推奨される設定値	
	アプリケーションサーバ 09-87 以前の機能だけを使用する場合	アプリケーションサーバ 11-40 でサポートされた非同期処理を使用する場合
webserver.connector.nio_http.max_threads	webserver.connector.inprocess_http.max_connections と同値	webserver.connector.inprocess_http.max_connections の値 ＋コネクション数を超えて同時実行される非同期処理※1 の最大同時実行数
webserver.connector.nio_http.min_threads	webserver.connector.inprocess_http.max_connections または webserver.connector.inprocess_http.max_spare_threads または webserver.connector.inprocess_http.init_threads のうち 最も小さい方の値	
webserver.connector.nio_http.max_connections※2	webserver.connector.inprocess_http.max_connections と同じ値※3	

注※1

「コネクション数を超えて同時実行される非同期処理」に含まれるものは次のとおりです。

- Servlet 3.0 の javax.servlet.AsyncContext の start メソッドの呼び出し
- WebSocket の javax.websocket.RemoteEndpoint.Async インタフェースの sendBinary メソッド，sendObject メソッド，sendText メソッドの呼び出し

注※2

クライアントと NIO HTTP サーバ間のコネクションを常設にする場合，webserver.connector.nio_http.max_connections の値は，クライアントから接続される数以上にしてください。

注※3

インプロセス HTTP サーバの接続元としてプロキシサーバを配置している構成からの移行で，複数のプロセスから J2EE サーバに接続する場合，各プロセスからの最大接続数の総和を設定してください。

なお，Cosminexus HTTP Server のリバースプロキシから J2EE サーバへの最大接続数は，マニュアル「HTTP Server」の「4.7.4(5)　性能に関する注意事項」を参照してください。

(c)　WebSocket 機能を使用するために最大同時接続数を増やす場合の注意点

アプリケーションサーバ 11-40 の WebSocket 機能を使用する場合，多数の WebSocket クライアントとの接続を長時間維持するために最大同時接続数を増やす必要が生じることがあります。アプリケーションサーバ 09-87 以前は Cosminexus HTTP Server とインプロセス HTTP サーバともに，最大同時接続数の上限を「1024」としていましたが，アプリケーションサーバ 11-40 では WebSocket を考慮して上限値が 09-87 以前より大きくなっています。

しかし，最大同時接続数に合わせて NIO HTTP サーバの最大スレッド数を増やすと，スレッド数に比例してメモリ使用量も増加します。また，1 つのプロセスが生成できるスレッド数は OS によって上限があ

ります。最小スレッド数に対して確保できるメモリ容量や OS のスレッド数上限が不足している場合は，J2EE サーバの起動処理中に OutOfMemoryError が発生して J2EE サーバの起動に失敗するおそれがあります。最大スレッド数に対して確保できるメモリ容量や OS のスレッド数上限が不足している場合は，リクエスト受付処理中に OutOfMemoryError が発生して J2EE サーバがプロセスダウンするおそれがあります。

もし，NIO HTTP サーバの最小スレッド数や最大スレッド数を増やす場合には，メモリサイズ（物理メモリのサイズ，J2EE サーバプロセスの-Xmx 指定値）も適切に再見積もりしてください。

最大スレッド数の見積もり方法についてはマニュアル「アプリケーションサーバ システム設計ガイド」の「5.2.1　J2EE サーバが使用するリソースの見積もり」を参照してください。メモリ使用量については，マニュアル「アプリケーションサーバ システム設計ガイド」の「5.3.1　J2EE サーバが使用する仮想メモリの使用量の見積もり」を参照してください。

(2)　タイムアウトの設定

アプリケーションサーバ 11-40 では，タイムアウトの発生個所とタイムアウト値の設定パラメタがアプリケーションサーバ 09-87 以前と異なります。以降では，Web サーバの構成ごとにタイムアウトの設定ポイントの相違点を説明します。

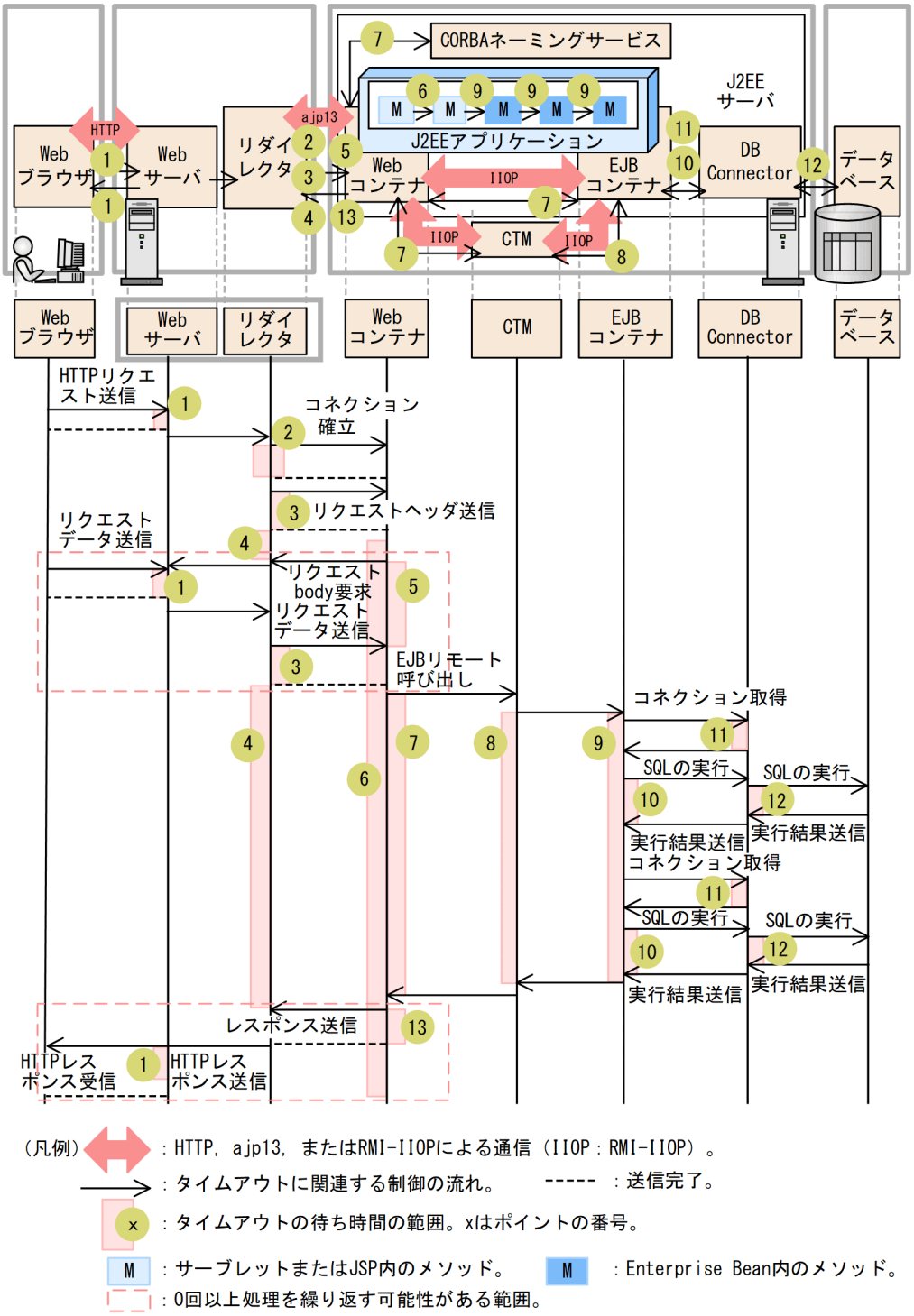
(a)　リダイレクタによる Web サーバ連携構成から移行する場合

アプリケーションサーバ 11-40 では，リダイレクタモジュール（mod_jk）をプロキシモジュール（mod_proxy）に，Web コンテナ側のリクエストの受け口が NIO HTTP サーバになっています。

そのため，アプリケーションサーバ 09-87 でリダイレクタと Web コンテナに設定していたタイムアウトは，アプリケーションサーバ 11-40 ではリバースプロキシと NIO HTTP サーバに設定するタイムアウトに変わります。

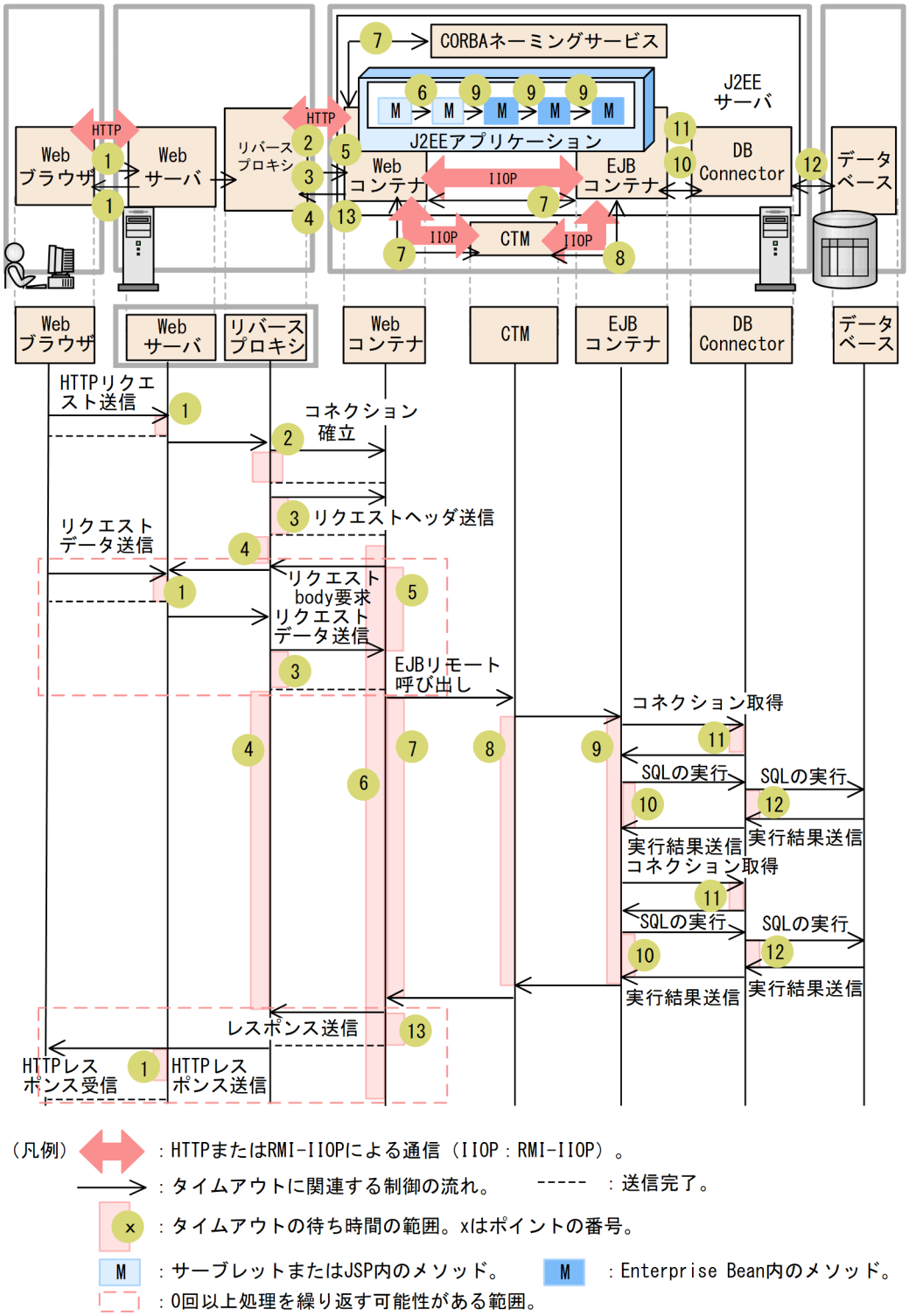
アプリケーションサーバ 09-87 でタイムアウトが設定できるポイントについて，次の図に示します。

図 12-5　アプリケーションサーバ 09-87 でタイムアウトが設定できるポイント（Web サーバ連携の場合）



アプリケーションサーバ 11-40 でタイムアウトが設定できるポイントについて，次の図に示します。

図 12-6　アプリケーションサーバ 11-40 でタイムアウトが設定できるポイント（Web サーバ連携の場合）



アプリケーションサーバ 09-87 からアプリケーションサーバ 11-40 への移行が必要なポイントについて，それぞれの相違点を次の表に示します。

表 12–8　各ポイントに設定するタイムアウトの相違点（Web サーバ連携構成からの移行）

ポイント	タイムアウトの種類	
	アプリケーションサーバ 09-87	アプリケーションサーバ 11-40
2	リダイレクタ側で設定する Web コンテナへのコネクション確立のタイムアウト。 設定対象 • mod_jk.conf （リダイレクタ動作定義ファイル） • 論理 Web サーバ（web-server） 設定個所 JkConnectTimeout	リバースプロキシ側で設定する Web コンテナ（NIO HTTP サーバ）へのコネクション確立のタイムアウト。 設定対象 • httpsd.conf • 論理 Web サーバ（web-server） 設定個所 ProxyPass の connectiontimeout キー， ProxyPass の timeout キー， または Timeout
3	リダイレクタ側で設定する Web コンテナへのリクエストヘッダおよびリクエストボディ送信のタイムアウト。 設定対象 • mod_jk.conf （リダイレクタ動作定義ファイル） • 論理 Web サーバ（web-server） 設定個所 JkSendTimeout	リバースプロキシ側で設定する Web コンテナ（NIO HTTP サーバ）へのリクエストヘッダおよびリクエストボディ送信のタイムアウト。 設定対象 • httpsd.conf • 論理 Web サーバ（web-server） 設定個所 ProxyPass の timeout キー，または Timeout
4	リダイレクタ側で設定する Web コンテナからのデータ受信のタイムアウト。 設定対象 • worker.properties （ワーカ定義ファイル） • 論理 Web サーバ（web-server） 設定個所 worker.<ワーカ名>.receive_timeout	リバースプロキシ側で設定する Web コンテナ（NIO HTTP サーバ）からのデータ受信のタイムアウト。 設定対象 • httpsd.conf • 論理 Web サーバ（web-server） 設定個所 ProxyPass の timeout キー，または Timeout
5	Web コンテナ側で設定するリダイレクタからのデータ受信のタイムアウト。 設定対象 • usrconf.properties（J2EE サーバ用ユーザプロパティ） • 論理 J2EE サーバ（j2ee-server） 設定個所 webserver.connector.ajp13.receive_timeout	Web コンテナ（NIO HTTP サーバ）側で設定するリバースプロキシからのデータ受信のタイムアウト。 設定対象 • usrconf.properties（J2EE サーバ用ユーザプロパティ） • 論理 J2EE サーバ（j2ee-server） 設定個所 webserver.connector.nio_http.receive_timeout
13	Web コンテナ側で設定するリダイレクタへのレスポンス送信のタイムアウト。 設定対象 • usrconf.properties（J2EE サーバ用ユーザプロパティ） • 論理 J2EE サーバ（j2ee-server）	Web コンテナ（NIO HTTP サーバ）側で設定するリバースプロキシへのデータ送信のタイムアウト。 設定対象 • usrconf.properties（J2EE サーバ用ユーザプロパティ） • 論理 J2EE サーバ（j2ee-server）

ポイント	タイムアウトの種類	
	アプリケーションサーバ 09-87	アプリケーションサーバ 11-40
	設定個所 webserver.connector.ajp13.send_timeout	設定個所 webserver.connector.nio_http.send_timeout

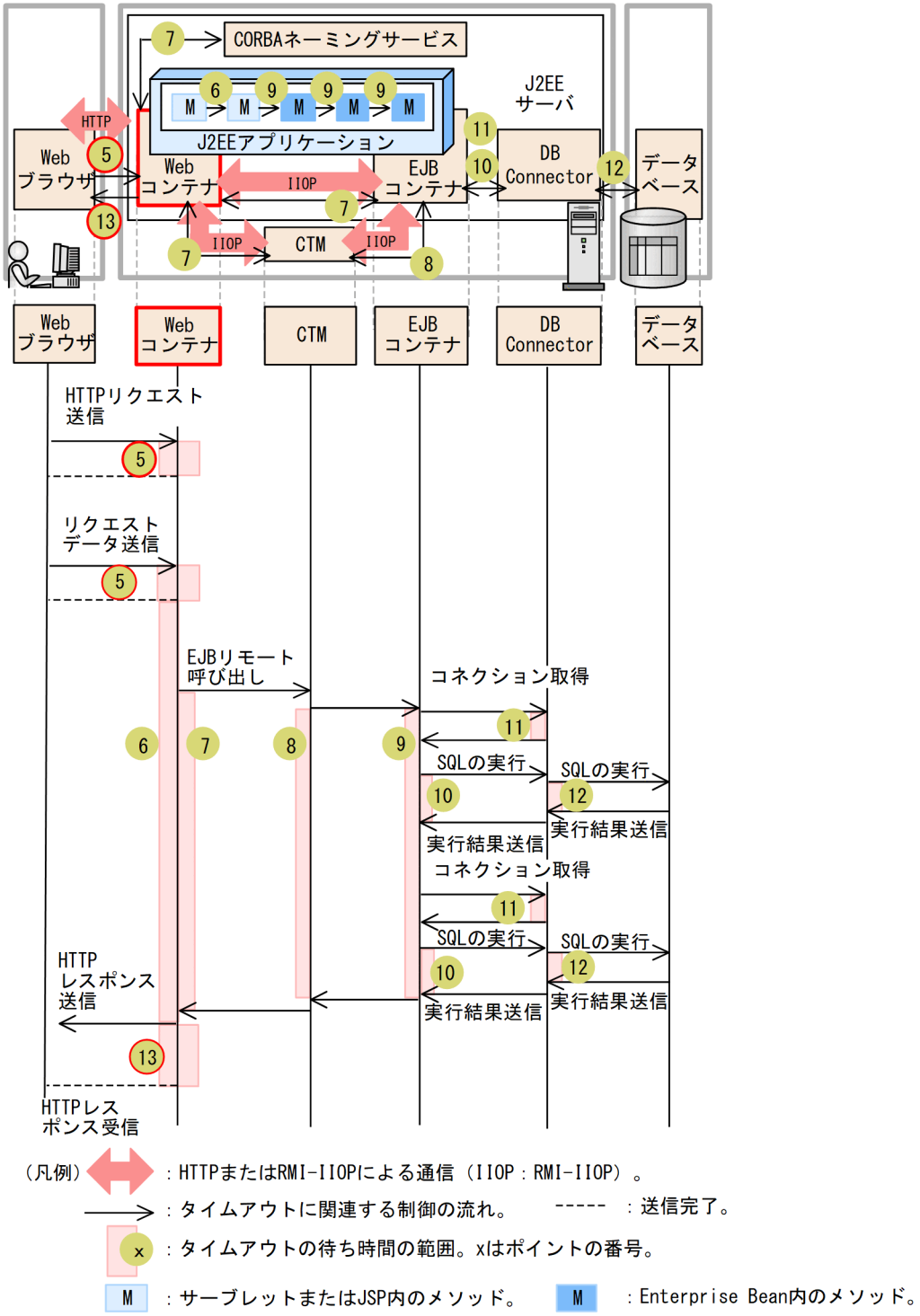
(b) インプロセス HTTP サーバ構成から移行する場合

アプリケーションサーバ 11-40 では，09-87 のインプロセス HTTP サーバが NIO HTTP サーバになります。

そのため，アプリケーションサーバ 09-87 でインプロセス HTTP サーバに設定していたタイムアウトは，アプリケーションサーバ 11-40 では NIO HTTP サーバに設定するタイムアウトに変わります。

タイムアウトが設定できるポイントについて，次の図に示します。

図 12-7　タイムアウトが設定できるポイント（インプロセス HTTP サーバの場合）



アプリケーションサーバ 09-87 からアプリケーションサーバ 11-40 への移行が必要なポイントについて，それぞれの相違点を次の表に示します。

表 12–9　各ポイントに設定するタイムアウトの相違点（インプロセス HTTP サーバ構成からの移行）

ポイント	タイムアウトの種類	
	アプリケーションサーバ 09-87	アプリケーションサーバ 11-40
5	Web コンテナ側で設定するクライアントからのリクエスト受信のタイムアウト。 設定対象 • usrconf.properties（J2EE サーバ用ユーザプロパティ） • 論理 J2EE サーバ（j2ee-server） 設定個所 webserver.connector.inprocess_http.receive_timeout	NIO HTTP サーバ側で設定するクライアントからのデータ受信のタイムアウト。 設定対象 • usrconf.properties（J2EE サーバ用ユーザプロパティ） • 論理 J2EE サーバ（j2ee-server） 設定個所 webserver.connector.nio_http.receive_timeout
13	Web コンテナ側で設定するクライアントへのレスポンス送信のタイムアウト。 設定対象 • usrconf.properties（J2EE サーバ用ユーザプロパティ） • 論理 J2EE サーバ（j2ee-server） 設定個所 webserver.connector.inprocess_http.send_timeout	NIO HTTP サーバ側で設定するクライアントへのデータ送信のタイムアウト。 設定対象 • usrconf.properties（J2EE サーバ用ユーザプロパティ） • 論理 J2EE サーバ（j2ee-server） 設定個所 webserver.connector.nio_http.send_timeout

12.4.2　使用するリソースの見積もり

アプリケーションサーバ 11-40 では，J2EE アプリケーション実行基盤の使用するリソースの見積もり式のうち，次の見積もり式が 09-87 以前と異なります。

- J2EE サーバが使用するリソースの見積もり

(1)　J2EE サーバが使用するリソースの見積もり

アプリケーションサーバ 11-40 では，インプロセス HTTP サーバが NIO HTTP サーバ機能に代わったことや，非同期処理をする機能が加わったことによって，J2EE サーバプロセスのスレッド数とファイルディスクリプタ数が 09-87 以前と異なります。

詳細は，マニュアル「アプリケーションサーバ システム設計ガイド」の「5.2.1　J2EE サーバが使用するリソースの見積もり」を参照してください。

12.5　システム保守情報の移行ガイド

この節では，ログや性能解析トレースなどの保守情報の扱いについて，アプリケーションサーバ 09-87 からの変更点を説明します。

12.5.1　出力先ログファイルの変更点

アプリケーションサーバ 11-40 では，一部の保守情報について出力先ログファイルが 09-87 以前と異なります。

表 12–10　保守情報出力先の変更点

出力元	分類	出力先	
		アプリケーションサーバ 09-87	アプリケーションサーバ 11-40
J2EE サーバ	インプロセス HTTP サーバ／NIO HTTP サーバのアクセスログ	<ejb.server.log.directory>¥http¥cjhttp_access.inprocess_http[n].log	<ejb.server.log.directory>¥cj_access_niohttp[n].log
	WebSocket アクセスログ	−	<ejb.server.log.directory>¥cj_access_websocket[n].log
	開発調査ログ	<ejb.server.log.directory>¥cjdevelopment[n].log	導入母体ログ出力機能によって，同等の内容がメッセージログと例外ログに出力されます。
リダイレクタ	リダイレクタのメッセージログ	<Application Server のインストールディレクトリ>¥CC¥web¥redirector¥logs または JkLogFileDir の指定値	廃止
	リダイレクタの保守用トレースログ	<Application Server のインストールディレクトリ>¥CC¥web¥redirector¥logs または JkLogFileDir の指定値	廃止

12.5.2　アクセスログの変更点

アプリケーションサーバ 11-40 の NIO HTTP サーバ機能には，従来のインプロセス HTTP サーバのアクセスログに相当するアクセスログ出力機能があります。アクセスログの設定先プロパティの変更点を次の表に示します。

表 12–11　NIO HTTP サーバのアクセスログに対するプロパティ推奨値

設定項目	アプリケーションサーバ 09-87 のプロパティ名	アプリケーションサーバ 11-40 のプロパティ名	アプリケーションサーバ 11-40 の推奨値
最大ファイルサイズ	webserver.logger.access_log.inprocess_http.filenum	ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filenum	アプリケーションサーバ 09-87 と同じ値
最大面数	webserver.logger.access_log.inprocess_http.filesize	ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filesize	アプリケーションサーバ 09-87 と同じ値
フォーマット	webserver.logger.access_log.<フォーマット名>	ejbserver.logger.access_log.nio_http.format	デフォルト値＋ 個別に指定していたフォーマット引数

12.5.3　性能解析トレースのトレース取得ポイントの変更点

アプリケーションサーバ 11-40 では，性能解析トレースの一部の取得ポイントが 09-87 以前と異なります。09-87 で性能解析トレース機能を使用していた場合は，次の表に従ってイベント ID を読み替えてください。

表 12–12　性能解析トレースのトレース取得ポイントの変更点

出力元	トレース取得ポイント	イベント ID		レベル
		アプリケーションサーバ 09-87	アプリケーションサーバ 11-40	
Web コンテナ（Web サーバ連携）	リクエスト取得・リクエストヘッダ解析完了直後	0x8200	0x8236	A
	リクエスト処理完了直後	0x8300	0x8336	A
Web コンテナ（インプロセス HTTP サーバ）	リクエスト取得時／リクエストヘッダ解析完了直後	0x8211	0x8236※1	A／B
	リクエスト処理完了直後	0x8311	0x8336※1	A／B

出力元	トレース取得ポイント	イベント ID		レベル
		アプリケーションサーバ 09-87	アプリケーションサーバ 11-40	
	Web クライアントからのデータ読み込み開始直前	0x8212	0x8237	B
	Web クライアントからのデータ読み込み完了直後	0x8312	0x8337	B
	Web クライアントへのデータ書き込み開始直前	0x8213	0x8238	B
	Web クライアントへのデータ書き込み完了直後	0x8313	0x8338	B
リダイレクタ	すべての取得ポイント	0x8000～0x8104	廃止※2	A／B

注※1

アプリケーションサーバ 09-87 ではレベル B の場合はレベル A より詳細な情報が出力されましたが，アプリケーションサーバ 11-40 ではレベル A とレベル B とで出力内容は同じです。

注※2

HTTP Server のアクセスログやリクエストログにルートアプリケーション情報が出力されますので，それらのログと突き合わせることで代替してください。

12.5.4 メッセージの変更点

アプリケーションサーバ 11-40 では，メッセージログに出力されるメッセージ ID や内容が 09-87 以前と異なります。同等の条件で出力されるアプリケーションサーバ 11-40 のメッセージ ID を次の表に示します。09-87 で該当するメッセージを監視していた場合は，次の表に従ってメッセージ ID を読み替えてください。

表 12–13 メッセージ ID の変更点

出力元	アプリケーションサーバ 09-87 でのメッセージ		アプリケーションサーバ 11-40 の読み替え先 ID	ログレベル
	ID	内容		
Web コンテナ	KDJE39236-I	Web サーバ連携を開始します。	なし	Error
	KDJE39237-I	インプロセス HTTP サーバを開始します。Web サーバ連携機能は無効となります。	KDJE39562-I	Error
	KDJE39238-W	webserver.connector.inprocess_http.permitted.hosts に指定されたホスト名は，名前解決できませんでした。	KDJE39563-W	Error

出力元	アプリケーションサーバ 09-87 でのメッセージ		アプリケーションサーバ 11-40 の読み替え先 ID	ログレベル
	ID	内容		
	KDJE39239-W	インプロセス HTTP サーバへのアクセスを許可されていないホストからのアクセスが拒否されました。	KDJE39564-W	Error
	KDJE39240-W	webserver.connector.inprocess_http.bind_host に指定されたホスト（{0}）は解決できませんでした。	KDJE39565-W	Error
	KDJE39241-E	インプロセス HTTP サーバを指定されたポート番号（{0}）で起動できませんでした。	KDJE39566-E	Error
	KDJE39242-W	webserver.connector.inprocess_http.bind_host に指定されたホスト（{0}）は解決できませんでした。	KDJE39567-W	Error
	KDJE39243-E	送信されたリクエストの HTTP メソッドは許可されていません。	なし	Error
	KDJE39244-E	送信されたリクエストのリクエストラインが最大サイズを超えています。	なし	Error
	KDJE39245-E	送信されたリクエストのリクエストヘッダが最大サイズを超えています。	KDJE39568-E	Error
	KDJE39246-E	送信されたリクエストのリクエストボディが最大サイズを超えています。	KDJE39569-E	Error
	KDJE39247-E	送信されたリクエストに含まれる HTTP ヘッダの数が最大数を超えています。	KDJE39570-E	Error
	KDJE39256-W	リダイレクト機能のレスポンスボディとして送信するファイルの読み込みができませんでした。	なし	Error
	KDJE39257-E	リダイレクト機能のレスポンスボディとして送信するファイルの読み込み中にエラーが発生しました。	なし	Error
	KDJE39258-W	リダイレクト機能のレスポンスボディを送信中にエラーが発生しました。	なし	Information
	KDJE39259-W	ファイルの読み込みができないため，エラーコンテンツカスタマイズ機能のレスポンスボディとして送信するファイルはクライアントに送信されませんでした。	なし	Error
	KDJE39260-E	エラーコンテンツカスタマイズ機能のレスポンスボディとして送信するファイルの読み込み中にエラーが発生しました。	なし	Error
	KDJE39261-W	エラーコンテンツカスタマイズ機能のレスポンスボディを送信中にエラーが発生しました。	なし	Information
	KDJE39262-E	Web クライアントとの接続数が上限を超えたため，リクエスト処理を拒否します。	なし	Error
	KDJE39263-W	リクエスト処理スレッドが不足しています。	なし	Error
	KDJE39265-E	クライアントからのリクエストヘッダ読み込み中にタイムアウトが発生しました。	KDJE39576-E	Warning

出力元	アプリケーションサーバ 09-87 でのメッセージ		アプリケーションサーバ 11-40 の読み替え先 ID	ログレベル
	ID	内容		
	KDJE39267-W	アクセスされたリクエストを拒否しました。 インプロセス HTTP サーバでは，"/ejb/"または"/web/"で始まるリクエストは使えません。	KDJE39571-W	Error
	KDJE39268-E	リダイレクト機能のレスポンスボディとして送信するファイルの読み込みができませんでした。	なし	Error
	KDJE39269-E	HTTP リクエストのボディ情報が大き過ぎるため，インプロセス HTTP サーバはアクセスされたリクエストを拒否しました。	なし	Error
	KDJE39270-W	レスポンスがコミット済みのため，エラーページのカスタマイズを行えませんでした。	なし	Error
	KDJE39274-E	クライアントとのコネクション確立時に障害が発生しました。	KDJE39572-E	Error
	KDJE39276-W	クライアントとのコネクション確立時に障害が発生しましたが，クライアントとのコネクション確立に成功しました。	KDJE39573-W	Error
	KDJE39277-E	不正なリクエストヘッダを受信したため，インプロセス HTTP サーバはアクセスされたリクエストを拒否しました。	KDJE39574-E	Error
	KDJE39514-W	幾つかのリクエスト処理スレッドが終了していません。	KDJE39575-W	Error
	KDJE39561-W	Servlet ContainerInitializer 検索範囲絞り込み機能が有効になっています。	なし	Error
リダイレクタ	KDJE41000 ～KDJE41041	すべてのメッセージ	なし※	－

注※

HTTP Server のエラーログで代替してください。

12.6　パラメタ読み替えリファレンス

この節では，アプリケーションサーバ 11-00 以降にだけ存在するパラメタの設定値を，アプリケーションサーバ 09-87 での設定値からどのように算出すべきかについて説明します。

12.6.1　J2EE サーバ用ユーザプロパティ定義

アプリケーションサーバ 11-00 以降にだけ存在するパラメタのうち，J2EE サーバ用ユーザプロパティファイルに指定するプロパティの推奨値を次の表に示します。プロパティの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

表 12–14　J2EE サーバ用ユーザプロパティの移行先プロパティ

アプリケーションサーバ 11-40 のプロパティ名	アプリケーションサーバ 11-40 での推奨値	
	Web サーバ連携の場合	インプロセス HTTP サーバの場合
webserver.connector.nio_http.max_connections	12.4.1(1)(a)を参照	12.4.1(1)(b)を参照
webserver.connector.nio_http.max_threads		
webserver.connector.nio_http.min_threads		
webserver.connector.nio_http.receive_timeout	12.4.1(2)(a)を参照	12.4.1(2)(b)を参照
webserver.connector.nio_http.send_timeout		
webserver.connector.nio_http.backlog	webserver.connector.ajp13.backlog と同じ値	webserver.connector.inprocess_http.backlog と同じ値
webserver.connector.nio_http.bind_host	リバースプロキシの転送先 URL のホスト名が「localhost」の場合は"localhost"。 それ以外の場合は，Web サーバと通信可能な IP アドレスまたはホスト名	webserver.connector.inprocess_http.bind_host と同じ値
webserver.connector.nio_http.hostname_lookups	false（デフォルト値）	webserver.connector.inprocess_http.hostname_lookups と同じ値
webserver.connector.nio_http.limit.max_headers	100（デフォルト値）	webserver.connector.inprocess_http.limit.max_headers と同じ値
webserver.connector.nio_http.limit.max_request_body	-1（デフォルト値）	webserver.connector.inprocess_http.limit.max_request_body と同じ値

アプリケーションサーバ 11-40 のプロパティ名	アプリケーションサーバ 11-40 での推奨値	
	Web サーバ連携の場合	インプロセス HTTP サーバの場合
webserver.connector.nio_http.limit.max_request_header	16384（デフォルト値）	webserver.connector.inprocess_http.limit.max_request_header と同じ値
webserver.connector.nio_http.keep_alive.max_requests	0（デフォルト値）	webserver.connector.inprocess_http.persistent_connection.max_requests と同じ値
webserver.connector.nio_http.keep_alive.timeout	0（デフォルト値）	webserver.connector.inprocess_http.persistent_connection.timeout と同じ値
webserver.connector.nio_http.port	Web サーバと通信可能な任意のポート番号	webserver.connector.inprocess_http.port と同じ値
webserver.connector.nio_http.response.header.server	"CosminexusComponentContainer"（デフォルト値）	webserver.connector.inprocess_http.response.header.server と同じ値
webserver.connector.nio_http.idle_thread_timeout	60（デフォルト値）	60（デフォルト値）
webserver.connector.nio_http.permitted.hosts	リバースプロキシの転送先 URL のホスト名が localhost の場合 "localhost" それ以外の場合 Web サーバの IP アドレスまたはホスト名	webserver.connector.inprocess_http.permitted.hosts と同じ値
webserver.connector.nio_http.max_servlet_execute_threads	webserver.connector.ajp13.max_threads と同じ値	webserver.connector.inprocess_http.max_execute_threads と同じ値
ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filenum	16（デフォルト値）	webserver.logger.access_log.inprocess_http.filenum と同じ値
ejbserver.logger.channels.define.NIOHTTPAccessLogFile.filesize	4194304（デフォルト値）	webserver.logger.access_log.inprocess_http.filesize と同じ値
ejbserver.logger.access_log.nio_http.format	%h %{X-Forwarded-For}i %l %u %d %rootap "%r" %s %b %D %S（デフォルト値）	webserver.logger.access_log.inprocess_http.usage_format の値が”common”または”combined”だった場合 デフォルト値 それ以外の場合 デフォルト値に加え，webserver.logger.access_log.<フォーマット名>に個別に指定していたフォーマット引数

アプリケーションサーバ 11-40 のプロパティ名	アプリケーションサーバ 11-40 での推奨値	
	Web サーバ連携の場合	インプロセス HTTP サーバの場合
		<フォーマット名>は webserver.logger.access_log.inprocess_http.usage_format に指定していた文字列
webserver.context.stop_asyncwait_timeout	30（デフォルト値）	
webserver.servlet_api.unsupported.throwUnsupportedOperationException	12.3.2(1)を参照	

12.6.2 リダイレクタの定義

アプリケーションサーバ 09-87 でリダイレクタによる Web サーバ連携構成を使用していた場合は，リダイレクタによる Web サーバ連携機能が HTTP Server のリバースプロキシ機能に置き換わります。従来のリダイレクタに指定していたパラメタから，リバースプロキシ機能のディレクティブへの読み替えを次の表に示します。

表 12–15 リダイレクタの定義ファイルの移行先

アプリケーションサーバ 09-87 での指定先		アプリケーションサーバ 11-40 で対応する指定先
指定先ファイル	パラメタ名	
mod_jk.conf（リダイレクタ動作定義ファイル）	JkConnectTimeout	12.4.1(2)(a)を参照
	JkSendTimeout	
	JkPrfId	HWSPrfId ディレクティブ
	JkRequestRetryCount	なし（常にリトライはしない）
	上記以外	なし
worker.properties（ワーカ定義ファイル）	worker.<ワーカ名>.host	ProxyPass ディレクティブの第 2 引数に指定する URL のホスト名
	worker.<ワーカ名>.port	ProxyPass ディレクティブの第 2 引数に指定する URL のポート番号
	worker.<ワーカ名>.receive_timeout	12.4.1(2)(a)を参照
	上記以外	なし

12.7　廃止されたパラメタリファレンス

この節では，アプリケーションサーバ 11-40 で廃止されているパラメタについて説明します。

12.7.1　J2EE サーバで使用するファイル

(1)　usrconf.properties（J2EE サーバ用ユーザプロパティファイル）

アプリケーションサーバ 11-40 では指定できないキーを次の表に示します。

表 12–16　J2EE サーバ用ユーザプロパティファイルの廃止パラメタ

廃止されたアプリケーションサーバ 09-87 のキー名	備考
webserver.connector.inprocess_http から始まるキー	インプロセス HTTP サーバ機能は使用できません。
webserver.logger.access_log から始まるキー	
webserver.logger.communication_trace.inprocess_http.filenum	
webserver.logger.thread_trace.inprocess_http.filenum	
webserver.container.thread_control.enabled	指定値は無視され，常に true 相当の動きになります。
webserver.connector.ajp12 から始まるキー	リダイレクタによる Web サーバ連携機能は使用できません。
webserver.connector.ajp13 から始まるキー	
ejbserver.server.eheap.ajp13.enabled	
webserver.eadssfo.から始まるキー	EADs セッションフェイルオーバ機能は使用できません。
webserver.jsp.el2_2.enabled	指定値は無視され，常に EL 3.0 で動作します。
cosminexus.jpa.から始まるキー	CJPA プロバイダ独自パラメタは使用できません。
ejbserver.server.j2ee.feature	1.3basic などの旧バージョン互換の動作モードは指定できません。

12.7.2　Web サーバ連携で使用するファイル

次のファイルは使用できません。

- HTTP Server 用リダイレクタ動作定義ファイル（mod_jk.conf）
- ワーカ定義ファイル（workers.properties）
- Microsoft IIS 用リダイレクタ動作定義ファイル（isapi_redirect.conf）

- Microsoft IIS 用マッピング定義ファイル（usrworkermap.properties）

12.7.3　JPA で使用するファイル

(1)　persistence.xml

persistence.xml の<property>タグに指定するプロパティとして，次のプロパティは指定できません。

- cosminexus.jpa から始まるプロパティ

12.7.4　Smart Composer 機能で指定するパラメタ

(1)　論理 J2EE サーバで指定するパラメタ

「表 12-16　J2EE サーバ用ユーザプロパティファイルの廃止パラメタ」にあるパラメタは，論理 J2EE サーバのパラメタとしても指定できません。

(2)　論理 Web サーバで指定するパラメタ

次のパラメタは論理 Web サーバのパラメタとして指定できません。

- HTTP Server 用リダイレクタ動作定義を設定するパラメタ
- ワーカ定義を設定するパラメタ

付録

付録 A　snapshot ログの収集対象一覧

ここでは，snapshot ログの一括収集を実行する場合に収集対象となる構成ソフトウェアの実行環境ディレクトリについて説明します。なお，収集した ZIP ファイルを展開すると，収集元と同じディレクトリ構成で展開されます。

snapshot ログの一括収集時，アプリケーションサーバで構築したシステムの構成ソフトウェア，ライブラリのログファイルおよび定義ファイルが収集されます。収集対象は，次のファイルを編集すると変更できます。snapshot ログ取得の設定については，「3.3.3　snapshot ログ収集の設定（J2EE アプリケーションを実行するシステム）」または「3.3.4　snapshot ログ収集の設定（バッチアプリケーションを実行するシステム）」を参照してください。

トラブルシューティングに必要な資料と snapshot ログの資料の種類との対応を次の表に示します。

表 A-1　トラブルシューティングに必要な資料と snapshot ログの資料の種類との対応

トラブルシューティングに必要な資料		snapshot ログの資料の種類
アプリケーションサーバのログ	メッセージログ	メッセージログ
	ユーザログ	そのほかのログ
	例外ログ	そのほかのログ
	保守用ログ	保守ログ
EJB クライアントアプリケーションのシステムログ	メッセージログ	メッセージログ
	ユーザログ	そのほかのログ
	例外ログ	そのほかのログ
	保守用ログ	保守ログ
性能解析トレース		性能，障害解析トレース
JavaVM のスレッドダンプ		ダンプ
JavaVM の GC のログ		そのほかのログ
メモリダンプ		ダンプ
JavaVM ログファイル		そのほかのログ
エラーリポートファイル		ダンプ
OS の状態・ログ		そのほか
OS の統計情報		そのほか
定義情報		定義情報
作業ディレクトリ		ユーザデータ
リソースの設定		ユーザデータ
Web サーバのログ		メッセージログ

トラブルシューティングに必要な資料	snapshot ログの資料の種類
	そのほかのログ
	アクセスログ
	内部インタフェース
JavaVM のスタックトレース	そのほかのログ
明示管理ヒープ機能のイベントログ	そのほかのログ

注意事項

snapshot ログの一括収集は，デフォルトの状態では構成ソフトウェアのインストール時にデフォルトで作成されたディレクトリを対象に実行します。ログの出力先および作業ディレクトリを変更している場合は，収集先をカスタマイズしてください。

付録 A.1　snapshot ログの収集対象一覧の概要

ここでは，snapshot ログの資料の格納場所や，一覧で説明している収集方法，ディレクトリおよびファイルの名称について説明します。

(1)　資料の格納場所

snapshot ログは，snapshotlog.conf，snapshotlog.2.conf，および snapshotlog.param.conf のファイルに分けて収集されます。

- snapshotlog.conf
  一次送付資料の収集対象が記載されているファイルです。ファイルの格納場所を次に示します。
  - Windows の場合
    <Application Server のインストールディレクトリ>¥manager¥config¥snapshotlog.conf
  - UNIX の場合
    /opt/Cosminexus/manager/config/snapshotlog.conf
- snapshotlog.2.conf
  二次送付資料の収集対象が記載されているファイルです。ファイルの格納場所を次に示します。
  - Windows の場合
    <Application Server のインストールディレクトリ>¥manager¥config¥snapshotlog.2.conf
  - UNIX の場合
    /opt/Cosminexus/manager/config/snapshotlog.2.conf
- snapshotlog.param.conf
  定義送付資料の収集対象が記載されているファイルです。ファイルの格納場所を次に示します。

- Windows の場合
  <Application Server のインストールディレクトリ>¥manager¥config¥snapshotlog.param.conf
- UNIX の場合
  /opt/Cosminexus/manager/config/snapshotlog.param.conf

(2) snapshot ログの収集方法

snapshot ログを収集する方法には，次の種類があります。snapshot ログの収集方法を次の表に示します。

表 A–2 snapshot ログの収集方法

項番	収集方法	分類
1	運用管理ポータルの「論理サーバの起動／停止」アンカーの［snapshot ログ］画面から収集する	収集方法 A
2	運用管理コマンド（mngsvrutil collect snapshot コマンド）を実行する	収集方法 A
3	論理サーバの障害を検知して自動収集する	収集方法 A
4	標準の snapshotlog.conf，snapshotlog.2.conf，snapshotlog.param.conf を使用して snapshotlog コマンドを実行する	収集方法 B

(3) snapshot ログの収集可否や収集に関する設定変更

表 A-5 から表 A-45 では，mngsvrutil コマンドまたは snapshotlog コマンドでの収集可否や設定変更可否などを記号で分類しています。それぞれの記号について次の表で説明します。

表 A–3 表 A-5 から表 A-45 の表中の記号の定義

表中の記号	デフォルトでの収集	ログの出力先や定義ファイルの格納先の変更	ログの出力先や定義ファイルの格納先の変更をした場合の収集
◎	収集される	できる	できる
○	収集される	できない	できない
△	収集される	できる	できるが制限がある※1
×	収集されない	できない	できない※2

注※1

snapshot ログ収集対象定義ファイル（snapshotlog.conf，snapshotlog.2.conf，または snapshotlog.param.conf）を編集して，収集対象ディレクトリに設定する必要があります。

注※2

ログの出力先や定義ファイルの格納先を変更できません。また，snapshot ログ収集対象定義ファイル（snapshotlog.conf，snapshotlog.2.conf，または snapshotlog.param.conf）を編集して，収集対象ディレクトリに設定する必要があります。

また，表 A-5 から表 A-45 の資料列では，mngsvrutil コマンドまたは snapshotlog コマンドを実行した場合に，収集される資料を示しています。

ほかの方法で資料を収集した場合，または mngsvrutil collect snapshot 2 を実行した場合は，表の資料列の内容に関係なくすべての資料が収集されます。

資料列の表記および記号について説明します。

- 一次：一次送付資料

  snapshotconf.conf を指定した snapshotlog コマンド，または mngsvrutil collect snapshot 1 を実行した場合に収集される資料です。
- 二次：二次送付資料

  snapshotconf.2.conf を指定した snapshotlog コマンド，または mngsvrutil collect snapshot 2 を実行した場合に収集される資料です。
- 定義：定義送付資料

  snapshotlog.param.conf を指定した snapshotlog コマンドを実行した場合に収集される資料です。
- 表中の記号

  ○：収集される

  －：収集されない

(4) 収集対象の記述規則

収集対象のディレクトリおよびファイルの共通の記述方法を次に示します。

- 収集対象中の「*」は 0 文字以上の任意の文字列を示します。
- Windows での「¥*¥*¥*」や UNIX での「/*/*/*」のような記述は，何階層下のファイルを収集するかを示します。例えば，Windows での「log¥*¥*」や UNIX での「log/*/*」の場合は，log ディレクトリの 2 階層下のファイルが収集対象になります。

また，収集対象のディレクトリおよびファイルの意味を次の表に示します。

表 A-4 収集対象のディレクトリおよびファイルの意味

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
<Application Server のインストールディレクトリ>	アプリケーションサーバのインストールディレクトリパス名	• Windows の場合 アプリケーションサーバのインストール時に決定する。 • UNIX の場合 /opt/Cosminexus	${cosminexus.home}
<CTM 識別子>	ctmstart コマンドの-CTMID オプション名	－	.+
<CTM スプールディレクトリ(ctmspool)>	環境変数 CTMSPOOL に指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CTM/spool	&{ctmspool}

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
<EJB クライアントの定義ファイル格納ディレクトリ>	環境変数 CJCLUSRCONFDIR に指定したディレクトリパス名，または cjclstartap コマンドを実行したディレクトリパス名	−	<ejb.client.definition.file.dir>
<EJB クライアントログサブディレクトリ 1(ejb.client.ejb.log)>	Java アプリケーション用の usrconf.cfg の ejb.client.ejb.log キーに指定したサブディレクトリ名	system	.+
<EJB クライアントログサブディレクトリ 1(ejbserver..client.ejb.log)>	Java アプリケーション用の usrconf.properties の ejbserver.client.ejb.log キーに指定したディレクトリパス名	−	.+
<EJB クライアントログサブディレクトリ 2(ejb.client.log.appid)>	Java アプリケーション用の usrconf.cfg の ejb.client.log.appid キーに指定したサブディレクトリ名	ejbcl	.+
<EJB クライアントログサブディレクトリ 2(ejbserver..client.log.appid)>	Java アプリケーション用の usrconf.properties の ejbserver.client.log.appid キーに指定したディレクトリパス名	−	.+
<EJB クライアントログ出力ディレクトリ (ejb.client.log.directory)>	Java アプリケーション用の usrconf.cfg の ejb.client.log.directory キーに指定したディレクトリパス名	−	<ejb.client.log.directory>
<EJB クライアントログ出力ディレクトリ (ejbserver.client.log.directory)>	Java アプリケーション用の usrconf.properties の ejbserver.client.log.directory キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/client/logs	<ejbserver.client.log.directory>
<HCSC サーバのプロパティ (methodtrace-filepath)>	HCSC サーバ定義コマンド cscsvconfig で methodtrace-filepath プロパティに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs/csc/maintenance	<methodtrace-filepath>
<HCSC サーバのプロパティ (requesttrace-filepath)>	HCSC サーバ定義コマンド cscsvconfig で requesttrace-filepath プロパティに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs/csc	<requesttrace-filepath>

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
<HCSC リポジトリルート(cscmng.repository.root)>	HCSC-Manager 定義 cscmng.properties の cscmng.repository.root キーに指定したリポジトリルート名	<Application Server のインストールディレクトリ>/CSC/repository	<cscmng.repository.root>
<HCSC ログ出力ディレクトリ(cscmng.log.dir)>	HCSC-Manager 定義 cscmng.properties の cscmng.log.dir キーに指定したログディレクトリパス名	<Application Server のインストールディレクトリ>/CSC/log/manager	<cscmng.log.dir>
<HWS アクセスログディレクトリ(HttpsdCustomLogFileDir)>	httpsd.conf の CustomLog ディレクティブに指定したファイル名のディレクトリパス名部分	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>/logs	&{hws.logfile.dir}
<HWS のインストールディレクトリ>	HTTP Server のインストールディレクトリパス名	Windows の場合 <Application Server のインストールディレクトリ>/httpsd UNIX の場合 /opt/hitachi/httpsd	${hws.home}
<HWS エラーログディレクトリ(HttpsdErrorLogFileDir)>	httpsd.conf の ErrorLogr ディレクティブに指定したファイル名のディレクトリパス名部分	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>/logs	&{hws.logfile.dir}
<HWS キャッシュサーバの作業ディレクトリ(SSLCacheServerRunDir)>	httpsd.conf の SSLCacheServerRunDir ディレクティブに指定したディレクトリパス名	<HWS サーバのルートディレクトリ(ServerRoot)>	<SSLCacheServerRunDir>
<HWS コアダンプ出力ディレクトリ(CoreDumpDirectory)>	httpsd.conf の CoreDumpDirectory ディレクティブに指定したディレクトリパス名	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>	&{core.dump.directory}
<HWS サーバのルートディレクトリ(ServerRoot)>	httpsd.conf の ServerRoot ディレクティブに指定したディレクトリパス名	<HWS のインストールディレクトリ>	<ServerRoot>
<HWS 内部トレースディレクトリ(HttpsdTraceLogFileDir)>	httpsd.conf の HWSTraceLogFile ディレクティブに指定したファイル名のディレクトリパス名部分	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>/logs	&{hws.logfile.dir}
<HWS プロセス ID ファイル(PidFile)>	httpsd.conf の PidFile ディレクティブに指定したファイル名	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>/logs/httpd.pid	<PidFile>

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
<HWS 用リダイレクタログ出力ディレクトリ (JkLogFileDir)>	mod_jk.conf の JkLogFileDir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/web/redirector/servers/<サーバ名称>/logs	&{jklogfiledir}
<HWS 用リダイレクタトレースログ出力ディレクトリ (JkTraceLogFileDir)>	mod_jk.conf の JkTraceLogFileDir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/web/redirector/servers/<サーバ名称>/logs	&{jktracelogfiledir}
<HWS リクエストログディレクトリ (HttpsdRequestLogFileDir)>	httpsd.conf の HWSRequestLog ディレクティブに指定したファイル名のディレクトリパス名部分	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>/logs	&{hws.logfile.dir}
<IIS アクセスログディレクトリ (IIS_log_dir)>	Microsoft IIS のアクセスログ出力ディレクトリパス名	<システムルートディレクトリ (systemroot)>/system32/LogFiles/W3SVC1	<IIS_log_dir>
<IIS 用リダイレクタトレースログ出力ディレクトリ (trace_log_file_dir)>	isapi_redirect.conf の trace_log_file_dir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/web/redirector/servers/<サーバ名称>/logs	&{jktracelogfiledir}
<IIS 用リダイレクタログ出力ディレクトリ (log_file_dir)>	isapi_redirect.conf の log_file_dir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/web/redirector/servers/<サーバ名称>/logs	&{jklogfiledir}
<J2EE サーバ作業ディレクトリ (ejb.public.directory)>	J2EE サーバ用の usrconf.cfg の ejb.public.directory キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/server/public	&{ejb.public.directory}
<J2EE サーバログ出力ディレクトリ (ejb.server.log.directory)>	J2EE サーバ用の usrconf.cfg の ejb.server.log.directory キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs	&{ejb.server.log.directory}
<JAXR トレースファイル名 (com.cosminexus.xml.registry.trace.file_path)>	システムプロパティの com.cosminexus.xml.registry.trace.file_path キーに指定したファイルパス名	${cosminexus.home}/c4web/logs/JAXRAPITrace	<com.cosminexus.xml.registry.trace.file_path>
<JSP 用一時ディレクトリ (webserver.work.directory)>	J2EE サーバの usrconf.properties の webserver.work.directory キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/server/repository/<サーバ名称>/web	&{webserver.work.directory}

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
<Management イベント発行用メッセージ ID リストファイル (manager.mevent.message_id.list)>	mevent.<サーバ名称>.properties の manager.mevent.message_id.list キーに指定したファイルパス名	−	<manager.mevent.message_id.list>
<Manager ログ出力ディレクトリ (com.cosminexus.manager.log.dir)>	manager.cfg の com.cosminexus.manager.log.dir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/manager/log	${com.cosminexus.manager.log.dir}
<PRF 識別子>	cprfstart コマンドの-PRFID オプション名	−	.+
<PRF スプールディレクトリ (prfspool)>	環境変数 PRFSPOOL に指定したディレクトリパス名	<Application Server のインストールディレクトリ>/PRF/spool	&{prfspool}
<RM インストールディレクトリ>	環境変数 HRMDIR に指定したディレクトリパス名	<Application Server のインストールディレクトリ>/RM	<HRMDIR>
<TP1Connector ログ出力ディレクトリ (jp.co.hitachi_system.tp1connector.logdestination)>	システムプロパティの jp.co.hitachi_system.tp1connector.logdestination キーに指定したディレクトリパス名	<ユーザホームディレクトリ (user.home)>	<jp.co.hitachi_system.tp1connector.logdestination>
<UNIX の syslog(syslog)>	UNIX の syslog ファイルパス名	• AIX の場合 /var/adm/ras/syslog* • Linux の場合 /var/log/messages*	<syslog>
<VBROKER_ADM ディレクトリ(vbroker_adm)>	環境変数 VBROKER_ADM に指定したディレクトリパス名	<Application Server のインストールディレクトリ>/TPB/adm	<VBROKER_ADM>
<Web コンテナサーバログ出力ディレクトリ (web.server.log.directory)>	J2EE サーバ用の usrconf.cfg の web.server.log.directory キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/logs	<web.server.log.directory>
<Windows のイベントログ (EventLog)>	Windows のイベントログを wmic コマンドなどで収集したファイル	−	<EventLog>
<Windows のクラッシュダンプ出力ディレクトリ (CrashDumpDir)>	Windows のクラッシュダンプ出力先ディレクトリパス名	<ユーザの Application Data ディレクトリ (LOCALAPPDATA)>/CrashDumps	<CrashDumpDir>
<XML Security - Core トレース出力ディレクトリ	J2EE サーバ用の usrconf.cfg の	<Application Server のインストールディレクトリ>/CC/	<com.cosminexus.xml.security.logging.trace_dir>

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
(com.cosminexus.xml.security.logging.trace_dir)>	com.cosminexus.xml.security.logging.trace_dir キーに指定したディレクトリパス名	server/public/ejb/<サーバ名称>	
<インプロセス HTTP サーバアクセスログファイル (webserver.logger.access_log.inprocess_http.filename)>	J2EE サーバ用の usrconf.properties の webserver.logger.access_log.inprocess_http.filename キーに指定したファイルパス名	<Application Server のインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs/http/cjhttp_access.inprocess_http	&{webserver.logger.access_log.inprocess_http.filename}.
<稼働情報ファイル出力先ディレクトリ (ejbserver.management.stats_file.dir)>	J2EE サーバ用の usrconf.properties の ejbserver.management.stats_file.dir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/stats	&{ejbserver.management.stats_file.dir}
<監査ログ出力ディレクトリ (auditlog.directory)>	監査ログ定義ファイルの auditlog.directory キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/auditlog	<auditlog.directory>
<監査ログメッセージ出力ディレクトリ (auditlog.raslog.message.directory)>	監査ログ定義ファイルの auditlog.raslog.message.directory キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/auditlog	<auditlog.raslog.message.directory>
<監査ログ例外出力ディレクトリ (auditlog.raslog.exception.directory)>	監査ログ定義ファイルの auditlog.raslog.exception.directory キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/auditlog	<auditlog.raslog.exception.directory>
<コンテキストルート>	サーバで実行している J2EE web アプリケーションのルート名	−	.+
<サーバ管理コマンドログ出力ディレクトリ (admin_ejb.server.log.directory)>	サーバ管理コマンド用オプション定義ファイルの USRCONF_JVM_ARGS キーの-Dejbserver.log.directory オプションに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/admin/logs	<admin_ejb.server.log.directory>
<サーバ名称>	Management Server で設定した論理サーバ名または実サーバ名	−	.+
<システムドライブ (SystemDrive)>	環境変数 SystemDrive に指定したディレクトリパス名	−	%{SystemDrive}
<システムルートディレクトリ(systemroot)>	環境変数 SystemRoot に指定したディレクトリパス名	−	%{SystemRoot}

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
<ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory1)>	J2EE サーバの usrconf.properties の ejbserver.distributedtx.ots.status.directory1 キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/otsstatus	&{ejbserver.distributedtx.ots.status.directory1}
<統合ユーザ管理のトレースファイル名(com.cosminexus.admin.auth.trace.prefix)>	ua.conf の com.cosminexus.admin.auth.trace.prefix オプションに指定したファイルパス名	−	<com.cosminexus.admin.auth.trace.prefix>
<バッチアプリケーションの定義ファイル格納ディレクトリ>	環境変数 CJBATCHUSRCONFDIR に指定したディレクトリパス名，または cjexecjob コマンド，cjkilljob コマンドもしくは cjlistjob コマンドを実行したディレクトリパス名	−	<batch.application.definition.file.dir>
<バッチアプリケーションログ出力ディレクトリ(batch.log.directory)>	バッチアプリケーション用の usrconf.cfg の batch.log.directory に指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/batch/logs	<batch.log.directory>
<ユーザの Application Data ディレクトリ(LOCALAPPDATA)>	環境変数 LOCALAPPDATA に指定したディレクトリパス名	−	%{LOCALAPPDATA}
<ユーザホームディレクトリ(user.home)>	実行ユーザのホームディレクトリパス名 • 収集方法が A の場合 運用管理エージェント起動ユーザのホームディレクトリ • 収集方法が B の場合 snapshotlog コマンドの実行ユーザのホームディレクトリ	−	${user.home}および<user.home>
<予備ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory2)>	J2EE サーバの usrconf.properties の ejbserver.distributedtx.ots.status.directory2 キーに指定したディレクトリパス名	−	&{ejbserver.distributedtx.ots.status.directory2}
<08-50 モードの仮想サーバマネージャログディレクトリ(vmx.log.dir)>	vmx.properties の vmx.log.dir キーに指定したディレクトリパス名※	<Application Server のインストールディレクトリ>/manager/vmx/log	<vmx.log.dir>

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
<08-50 モードの仮想サーバマネージャの処理データ格納ディレクトリ (vmx.spool.dir)>	vmx.properties の vmx.spool.dir キーに指定したディレクトリパス名※	<Application Server のインストールディレクトリ>/manager/vmx/spool	<vmx.spool.dir>
<仮想サーバマネージャの処理データ格納ディレクトリ (vmi.spool.dir)>	vmi.properties の vmi.spool.dir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/manager/vmi/spool	<vmi.spool.dir>
<サーバ通信エージェントログ出力ディレクトリ (sinaviagent.log.dir)>	sinaviagent.cfg の sinaviagent.log.dir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/sinagent/log	<sinaviagent.log.dir>
<サーバ通信エージェント処理データ格納ディレクトリ (sinaviagent.spool.dir)>	sinaviagent.cfg の sinaviagent.spool.dir キーに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/sinagent/spool	<sinaviagent.spool.dir>
<FTP アダプタインストールディレクトリ (ftpadp.home)>	FTP アダプタのインストールディレクトリパス名	−	<ftpadp.home>
<FTP アダプタコマンドメッセージログ出力先ディレクトリ (ftpadp.command.messagelog.filepath)>	FTP アダプタの運用コマンドのメッセージログ出力先ディレクトリパス名	<FTP アダプタインストールディレクトリ (ftpadp.home)>/log	<ftpadp.command.messagelog.filepath>
<HCSC メールアダプタ運用コマンドメッセージログ出力先ディレクトリ (mailadp.command.messagelog.filepath)>	Service Coordinator 本体に組み込まれるメールアダプタの運用コマンド実行ログディレクトリパス名	<Application Server のインストールディレクトリ>/CSC/log/adapter/command	<mailadp.command.messagelog.filepath>
<HCSC メールアダプタの保守ログ出力先ディレクトリ (mailadp.methodtrace.filepath)>	メールアダプタの実行環境プロパティファイルで mailadp.methodtrace.filepath プロパティに指定したディレクトリパス名	<Application Server のインストールディレクトリ>/CC/server/public/ejb/<サーバ名称>/logs/CSCADP/MAILADP/maintenance/	<mailadp.methodtrace.filepath>
<CTM レギュレータの設定ファイル(ctm.RegOption)>	CTM レギュレータの設定ファイルのパス名	−	&{ctm.RegOption}
<OTM ゲートウェイの設定ファイル (ctm.TSCGwOption)>	OTM ゲートウェイの設定ファイルのパス名	−	&{ctm.TSCGwOption}
<一時 PRF トレースファイル出力用ディレクトリ (adminagent.prftrace_dir)>	一時 PRF トレースファイルの出力先ディレクトリパス名	<Application Server インストールディレクトリ>/manager/tmp	${adminagent.prftrace_dir}

ディレクトリまたはファイル名	内容	デフォルト値	置換後の値
<明示管理ヒープ機能適用除外設定ファイル名(jvm.exmemexcludeclass.File)>	JavaVM 拡張オプション-XX:ExplicitMemoryExcludeClassListFile に指定したファイルパス名	<Application Server のインストールディレクトリ>/jdk/usrconf/exmemexcludeclass.cfg	<jvm.exmemexcludeclass.File>
<明示管理ヒープ機能適用除外設定の無効化対象設定ファイル名(jvm.exmemnotexcludeclass.File)>	JavaVM 拡張オプション-XX:ExplicitMemoryNotExcludeClassListFile に指定したファイルパス名	<Application Server のインストールディレクトリ>/jdk/usrconf/exmemnotexcludeclass.cfg	<jvm.exmemnotexcludeclass.File>
<HWSWebSocket ログディレクトリ(HttpsdWebSocketLogFileDir)>	httpsd.conf の HWSWebSocketLog ディレクティブに指定したファイル名のディレクトリパス名	<HWS インストールディレクトリ>/servers/HWS_<サーバ名称>/logs	&{hws.logfile.dir}
<CSCHTTP アダプタ運用コマンドメッセージログ出力先ディレクトリ(httpadp.command.messagelog.filepath)>	CSC 本体に組み込まれる CSCHTTP アダプタの運用コマンド実行ログディレクトリパス名	<Application Server のインストールディレクトリ>/CSC/log/adapter/command	<httpadp.command.messagelog.filepath>

(凡例) －：デフォルト値なし

注※　このバージョンでは指定できません。常にデフォルト値になります。

付録 A.2　Component Container

Component Container に関連する収集対象を次の表に示します。

表 A-5　Component Container に関連する収集対象（Windows の場合）

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
J2EE サーバ	メッセージログ	稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjmessage*.log	○	－	－	◎	△
		ログ稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjlogger.log	○	－	－	◎	△
		J2EE リソースアダプタとしてデプロイして使用するリソース	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/*.log	○	－	－	◎	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		アダプタの稼働ログ						
		J2EE アプリケーションに含めて使用するリソースアダプタの稼働ログ（通常モード）	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/<J2EE アプリケーション名>/*.log	○	−	−	◎	△
		J2EE アプリケーションに含めて使用するリソースアダプタの稼働ログ（テストモード）	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/TEST#<J2EE アプリケーション名>/*.log	○	−	−	◎	△
		Web サーブレットログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/web_servlet*.log	○	−	−	◎	△
	そのほかのログ	ユーザ出力ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/user_out*.log	○	−	−	◎	△
		ユーザエラーログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/user_err*.log	○	−	−	◎	△
		JavaVM の保守情報および GC のログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/javalog*.log	○	−	−	◎	△
		JavaVM の明示管理ヒープ機能イベントログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/ehjavalog*.log	○	−	−	◎	△
		開発調査ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjdevelopment*.log	○	−	−	◎	△
		障害発生時の例外情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjexception*.log	○	−	−	◎	△
		JavaVM イベントログ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/hs_err*	○	−	−	◎	◎

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
			<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/replay_pid*.log	○	−	−	◎	◎
		J2EE アプリケーションのユーザログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/user/*	○	−	−	◎	△
	保守ログ	保守情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjmaintenance*.log	○	−	−	◎	△
		コンソールメッセージ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjconsole*.log	○	−	−	◎	△
		EJB コンテナの保守情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjejbcontainer*.log	○	−	−	◎	△
		Web コンテナの保守情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjwebcontainer*.log	○	−	−	◎	△
		起動プロセス標準出力情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjstdout*.log	○	−	−	◎	△
		起動プロセス標準エラー情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjstderr*.log	○	−	−	◎	△
		終了プロセス情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cj_shutdown*.log	○	−	−	◎	△
	ダンプ	スレッドダンプ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/javacore*	○	−	−	◎	◎
	統計情報	稼働情報ファイル	<稼働情報ファイル出力先ディレクトリ(ejbserver.management.stats_file.dir)>/*	−	○	−	◎	△
		メモリ監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjmemorywatch*.log	○	−	−	◎	△
		スレッド監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjthreadwatch*.log	○	−	−	◎	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		スレッドダンプ監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjthreaddumpwatch*.log	○	−	−	◎	△
		HTTP リクエスト実行待ちキュー監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjrequestqueuewatch*.log	○	−	−	◎	△
		HTTP セッション数監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjhttpsessionwatch*.log	○	−	−	◎	△
		コネクションプール監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjconnectionpoolwatch*.log	○	−	−	◎	△
	そのほか	J2EE サーバの起動，停止または異常終了を示すログ	<Windows のイベントログ(EventLog)>	○	−	−	×	×
	定義情報	J2EE サーバ用オプション定義ファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/usrconf.cfg	○	−	○	○	○
		J2EE サーバ用ユーザプロパティファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/usrconf.properties	○	−	○	○	○
		J2EE サーバ用セキュリティポリシーファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/server.policy	○	−	○	○	○
		MNG で作成した各種定義ファイルのバックアップ	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/*	−	○	−	○	○
		保護区リストファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/criticalList.cfg	○	−	○	○	○
		リソース設定情報	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/import/**	−	○	−	×	×

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
			<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/rars/**	−	○	−	×	×
		保守情報	<Application Server のインストールディレクトリ>/CC/server/version/**	○	−	○	○	○
	ユーザデータ	EJB サーバ作業ディレクトリの内容	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/**	−	○	−	×	×
		Web コンテナ作業ディレクトリの内容	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/**	−	○	−	×	×
		JSP 用一時ディレクトリの内容	<JSP 用一時ディレクトリ(webserver.work.directory)>/**	−	○	−	×	×
サーバ管理コマンド	メッセージログ	稼働ログ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/cjmessage*.log	○	−	−	△	△
		ログ稼働ログ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/cjlogger.log	○	−	−	△	△
		互換モードの稼働ログ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*message*.log	○	−	−	△	△
	そのほかのログ	障害発生時の例外情報	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/cjexception*.log	○	−	−	△	△
		互換モードの障害発生時の例外情報	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*exception*.log	○	−	−	△	△
	保守ログ	保守情報	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/CC/maintenance/cjmaintenance*.log	○	−	−	△	△
		コンソールメッセージ	<サーバ管理コマンドログ出力ディレクトリ	○	−	−	△	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
			(admin_ejb.server.log.directory)>/CC/maintenance/cjconsole*.log					
		サーバ管理コマンドの保守情報	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/CC/maintenance/cjserveradmin*.log	○	−	−	△	△
		そのほかのログ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*/*	○	−	−	△	△
			<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*/*/*	○	−	−	△	△
			<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*/*/*/*	−	○	−	△	△
	定義情報	サーバ管理コマンド用定義ファイル	<Application Server のインストールディレクトリ>/CC/admin/usrconf/*	○	−	○	○	○
バッチアプリケーション	メッセージログ	cjexecjob コマンド，cjkilljob コマンドおよび cjlistjob コマンドの稼働ログ	<バッチアプリケーションログ出力ディレクトリ(batch.log.directory)>/cjmessage*.log	○	−	−	△	△
	保守ログ	そのほかのログ	<バッチアプリケーションログ出力ディレクトリ(batch.log.directory)>/*/*	○	−	−	△	△
	定義情報	バッチアプリケーション用オプション定義ファイル	<バッチアプリケーションの定義ファイル格納ディレクトリ>/usrconf.cfg	○	−	−	×	×
		バッチアプリケーション用ユーザプロパティファイル	<バッチアプリケーションの定義ファイル格納ディレクトリ>/usrconf.properties	○	−	−	×	×
リソースアダプタバージョンアップコマンド	メッセージログ	稼働ログ	<Application Server のインストールディレクトリ>/CC/logs/cjrarupdatemessage*.log	○	−	−	○	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
(cjrarupdate)	そのほかのログ	障害発生時の例外情報	<Application Server のインストールディレクトリ>/CC/logs/cjrarupdateexception*.log	○	−	−	○	○
	保守ログ	保守情報	<Application Server のインストールディレクトリ>/CC/logs/cjrarupdatemaintenance*.log	○	−	−	○	○
インプロセス HTTP サーバ	アクセスログ	インプロセス HTTP サーバの処理結果	<インプロセス HTTP サーバアクセスログファイル(webserver.logger.access_log.inprocess_http.filename)>*.log	○	−	−	◎	△
	モジュールトレース	スレッドトレースの情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/http/maintenance/thr/cjhttp_thr.*.inprocess_http.mm	○	−	−	◎	△
	通信トレース	通信トレースの情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/http/maintenance/comm/cjhttp_comm.*.inprocess_http.mm	○	−	−	◎	△
NIO HTTP サーバ	アクセスログ	NIO HTTP サーバ（HTTP）の処理結果	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cj_access_niohttp*.log	○	−	−	◎	△
		NIO HTTP サーバ（WebSocket）の処理結果	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cj_access_websocket*.log	○	−	−	◎	△
移行コマンド(cjenvupdate)	メッセージログ	cjenvupdate コマンドの稼働ログ	<Application Server のインストールディレクトリ>/CC/logs/cjenvupdatemessage*.log	○	−	−	○	○
	そのほかのログ	cjenvupdate コマンドの例外情報	<Application Server のインストールディレクトリ>/CC/logs/cjenvupdateexception*.log	○	−	−	○	○
	保守ログ	cjenvupdate コマンドの保守情報	<Application Server のインストールディレクトリ>/CC/logs/cjenvupdatemaintenance*.log	○	−	−	○	○
インプロセストランザクションサービス	そのほか	インプロセストランザクションサービスのステータスファイル	<ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory1)>/*	−	○	−	◎	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
			<ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory1)>/*/*	−	○	−	◎	△
		インプロセストランザクションサービスの予備ステータスファイル	<予備ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory2)>/*	−	○	−	◎	×
			<予備ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory2)>/*/*	−	○	−	◎	×
EJB クライアント	メッセージログ	稼働ログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/cjclmessage*.log	○	−	−	△	△
			<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/cjclmessage*.log	○	−	−	×	×
		cjclstartap コマンドの稼働ログ	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/cjclstartap*.log	○	−	−	×	×
		cjcldellog コマンドの稼働ログ	<Application Server のインストールディレクトリ>/CC/client/logs/cjcldellog.log	○	−	−	○	○
	そのほかのログ	ユーザ出力ログ	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/user_out*.log	○	−	−	×	×
		ユーザエラーログ	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライア	○	−	−	×	×

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
			ントログサブディレクトリ2(ejb.client.log.appid)>/user_err*.log					
		JavaVMの保守情報，GCのログ	<EJBクライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJBクライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJBクライアントログサブディレクトリ2(ejb.client.log.appid)>/javalog*.log	○	−	−	×	×
		障害発生時の例外情報	<EJBクライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/cjclexception*.log	○	−	−	△	△
			<EJBクライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJBクライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJBクライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjclmaintenance*.log	○	−	−	×	×
		EJBクライアントアプリケーションのユーザログ	<EJBクライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/user/*	○	−	−	△	△
			<EJBクライアントログ出力ディレクトリ(ejb.client.log.directory)>/user/*	○	−	−	×	×
	保守ログ	保守情報	<EJBクライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/cjclmaintenance*.log	○	−	−	△	△
			<EJBクライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJBクライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJBクライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjclmaintenance*.log	○	−	−	×	×

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		EJB コンテナの保守情報	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjejbcontainer*.log	○	−	−	×	×
		起動プロセス標準出力情報	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjstdout*.log	○	−	−	×	×
		起動プロセス標準エラー情報	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjstderr*.log	○	−	−	×	×
		ログの稼働情報	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/cjlogger.log	○	−	−	△	△
			<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/cjlogger.log	○	−	−	×	×
	定義情報	EJB クライアント用オプション定義ファイル	<EJB クライアントの定義ファイル格納ディレクトリ>/usrconf.cfg	○	−	−	×	×
		EJB クライアント用ユーザプロパティファイル	<EJB クライアントの定義ファイル格納ディレクトリ>/usrconf.properties	○	−	−	×	×
リダイレクタ(Web サーバ)	メッセージログ	HWS 用リダイレクタのメッセージログ	<HWS 用リダイレクタログ出力ディレクトリ(JkLogFileDir)>/hws_redirect*.log	○	−	−	◎	△
		旧バージョン互換 HWS 用リダイレクタのメッセージログ	<HWS 用リダイレクタログ出力ディレクトリ(JkLogFileDir)>/hws_redirect*.log	○	−	−	◎	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		旧バージョン互換 IIS 用リダイレクタのメッセージログ	<IIS 用リダイレクタログ出力ディレクトリ(log_file_dir)>/isapi_redirect*.log	○	−	−	△	△
	保守ログ	HWS 用リダイレクタの保守用トレースログ	<HWS 用リダイレクタトレースログ出力ディレクトリ(JkTraceLogFileDir)>/hws_rd_trace*.log	○	−	−	◎	△
		旧バージョン互換 HWS 用リダイレクタの保守用トレースログ	<HWS 用リダイレクタトレースログ出力ディレクトリ(JkTraceLogFileDir)>/hws_rd_trace*.log	○	−	−	◎	△
		旧バージョン互換 I IS 用リダイレクタの保守用トレースログ	<IIS 用リダイレクタトレースログ出力ディレクトリ(trace_log_file_dir)>/iis_rd_trace*.log	○	−	−	△	△
	定義情報	HWS 用リダイレクタ動作定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/servers/<サーバ名称>/mod_jk.conf	○	−	○	○	○
		ワーカ定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/servers/<サーバ名称>/workers.properties	○	−	○	○	○
		旧バージョン互換 HWS 用リダイレクタ動作定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/mod_jk.conf	○	−	○	○	○
		旧バージョン互換 IIS 用リダイレクタ動作定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/isapi_redirect.conf	○	−	○	○	○
		旧バージョン互換用ワーカ定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/workers.properties	○	−	○	○	○
		旧バージョン互換用マッピング定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/uriworkermap.properties	○	−	○	○	○
TP1/ Message Queue-Access	保守ログ	API，および Application Server インタフェースに関するメソッドト	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/*.log	○	−	−	◎	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		レース，メッセージログ						
	API トレース	API トレースファイル	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/mqc.api*	○	−	−	◎	◎
TP1 Connector	メッセージログ	J2EE リソースアダプタとしてデプロイして使用する TP1 Connector の稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/*.log	○	−	−	◎	△
		J2EE アプリケーションに含めて使用する TP1 Connector の稼働ログ（通常モード）	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/<J2EE アプリケーション名>/*.log	○	−	−	◎	△
		Non-managed 環境で使用する場合の TP1 Connector の稼働ログ	<TP1Connector ログ出力ディレクトリ(jp.co.hitachi_system.tp1connector.logdestination)>/tp1connector*.log	○	−	−	△	△
TP1/Client/J	そのほかのログ	デバッグトレース情報	<ユーザホームディレクトリ(user.home)>/TP1clientJ/dcClt*.dmp	−	○	−	△	△
Manager	メッセージログ	運用管理エージェントのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagent*.log	○	−	−	◎	○
		運用管理エージェントの起動・停止コマンドのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagentctl.exe.*.log	○	−	−	◎	○
		統合メッセージログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/message/mngmessage*.log	○	−	−	◎	○
		運用管理エージェントサービスのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagentsv.exe.*.log	○	−	−	◎	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		運用監視エージェントのログ・トレース	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngagent*.*.log	○	−	−	◎	○
		Management Server サービスのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvr.exe.*.log	○	−	−	◎	○
		Management Server サービス起動・停止コマンドのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvrctl.exe.*.log	○	−	−	◎	○
		Management Server のログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvr*.log	○	−	−	◎	○
	そのほかのログ	運用管理エージェントの標準エラー出力	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagent.err.*.log	○	−	−	◎	○
		運用管理エージェントの標準出力	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagent.out.*.log	○	−	−	◎	○
		運用管理エージェントの標準エラー出力コマンドライン	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagent.err	○	−	−	◎	○
		コンソールログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/processConsole*.log	○	−	−	◎	○
		運用管理エージェントサービスの標準出力	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagentsv.exe.out	○	−	−	◎	○
		運用管理エージェントサービスの標準エラー出力	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagentsv.exe.err	○	−	−	◎	○
		Management Server サービスの標準エラー出力	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvr.exe.err	○	−	−	◎	○
		Management Server サービスの標準出力	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvr.exe.out	○	−	−	◎	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		明示管理ヒープ機能で使用する自動配置用設定ファイル	<Application Server インストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/auto_explicit_memory.cfg	○	−	○	△	△
		ユーザ拡張性能解析トレースで使用するユーザ拡張性能解析トレース設定ファイル	<Application Server インストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/userprf.cfg	○	−	○	△	△
		セットアップウィザードなどのセットアップコマンドの定義ファイル	<Application Server インストールディレクトリ>/manager/setup/config/*	○	−	○	○	○
		セットアップウィザードなどのセットアップコマンドのログ	<Application Server インストールディレクトリ>/manager/setup/log/*.log	○	−	−	△	△
	保守ログ	コマンド保守ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/mngcmd*.log	−	○	−	○	○
		運用管理エージェントの保守ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/adminagent*.log	○	−	−	○	○
		運用管理エージェントが行うRMI 処理での保守ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/mngrmi*.log	−	○	−	○	○
		Management Server の保守ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/mngsvr*.log	○	−	−	○	○
	定義情報	運用管理エージェントプロパティファイル	<Application Server のインストールディレクトリ>/manager/config/adminagent.properties	○	−	○	○	○
		運用管理エージェント用オプション定義ファイル	<Application Server のインストールディレクトリ>/manager/config/adminagentuser.cfg	○	−	○	○	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		運用管理エージェント設定ファイル	<Application Server のインストールディレクトリ>/manager/config/adminagent.xml	○	−	○	○	○
		運用監視エージェントプロパティファイル	<Application Server のインストールディレクトリ>/manager/config/mngagent.<サーバ名称>.properties	○	−	○	○	○
		Management Server 環境設定ファイル	<Application Server のインストールディレクトリ>/manager/config/mserver.properties	○	−	○	○	○
		Management Server 用オプション定義ファイル	<Application Server のインストールディレクトリ>/manager/config/mserver.cfg	○	−	○	○	○
		Management Server 用環境変数定義ファイル	<Application Server のインストールディレクトリ>/manager/config/mserverenv.cfg	○	−	○	○	○
		Manager 設定ファイル	<Application Server のインストールディレクトリ>/manager/config/manager.cfg	○	−	○	○	○
		Management アクション実行用プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/maction.properties	○	−	○	○	○
		Management イベント発行用プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/mevent.<サーバ名称>.properties	−	○	○	○	○
		Management イベント発行用メッセージ ID リストファイル	<Management イベント発行用メッセージ ID リストファイル (manager.mevent.message_id.list)>	−	○	○	×	×
		mngsvrutil コマンドのクライアント側定義ファイル	<ユーザホームディレクトリ (user.home)>/.mngsvrutilrc	−	○	○	△	△
		mngsvrutil コマンドのサーバ側定義ファイル	<Application Server のインストールディレクトリ>/manager/config/mngsvrutil.properties	○	−	○	○	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		mngsvrutil コマンドのクライアント側共通定義ファイル	<Application Server のインストールディレクトリ>/manager/config/mngsvrutilcl.properties	○	−	○	○	○
		JP1/IM 連携用モニタ起動コマンドの設定ファイル	<ユーザホームディレクトリ(user.home)>/.mngsvrmonitorrc	−	○	○	△	△
		JP1/IM 連携用システムログメッセージマッピングファイル	<Application Server のインストールディレクトリ>/manager/config/mserver.jp1event.system.mapping.properties	○	−	○	○	○
			<Application Server のインストールディレクトリ>/manager/config/manager.jp1event.system.mapping.properties	○	−	○	○	○
			<Application Server のインストールディレクトリ>/manager/config/manager.<サーバ名称>.jp1event.system.mapping.properties	○	−	○	○	○
	性能，障害解析トレース	PRF トレースなど	<一時 PRF トレースファイル出力用ディレクトリ(adminagent.prftrace_dir)>/*.zip	−	○	−	◎	×
	内部インタフェーストレース	統合トレースログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/trace/mngtrace*.log	○	−	−	○	○
	そのほか	OS の状態情報など	<Application Server のインストールディレクトリ>/manager/tmp/*	○	−	−	○	○
Smart Composer	定義情報	サーバ設定プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/cmxserver.properties	○	−	○	○	○
		クライアント設定プロパティファイル	<ユーザホームディレクトリ(user.home)>/.cmxrc	−	○	○	△	△
		クライアント共通設定プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/cmxclient.properties	○	−	○	○	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		負荷分散機定義プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/lb.properties	○	−	○	○	○
統合ユーザ管理	定義情報	JAAS のコンフィグレーションファイル	<Application Server のインストールディレクトリ>/manager/config/jaas.conf	○	−	○	○	○
		統合ユーザ管理のコンフィグレーションファイル	<Application Server のインストールディレクトリ>/manager/config/ua.conf	○	−	○	○	○
	API トレース	統合ユーザ管理のトレース	<統合ユーザ管理のトレースファイル名(com.cosminexus.admin.auth.trace.prefix)>.*.log	−	○	−	×	×
08-50 モードの仮想サーバマネージャ※	定義情報	仮想サーバマネージャプロパティファイル	<Application Server のインストールディレクトリ>/manager/vmx/config/vmx.properties	○	−	○	×	○
		仮想サーバマネージャのクライアント共通設定プロパティファイル	<Application Server のインストールディレクトリ>/manager/vmx/config/vmxclient.properties	○	−	○	×	○
		VMware vCenter Server の接続処理のオプション定義ファイル	<Application Server のインストールディレクトリ>/manager/vmx/config/vmx_avcs_usrconf.cfg	○	−	○	×	○
		VMware vCenter Server の接続処理のプロパティファイル	<Application Server のインストールディレクトリ>/manager/vmx/config/vmx_avcs_usrconf.properties	○	−	○	×	○
		VMware vCenter Server の接続処理の共通定義ファイル	<Application Server のインストールディレクトリ>/manager/vmx/config/vmx_avcs_cjwconf.properties	○	−	○	×	○
	そのほかのログ	仮想サーバマネージャで使用する cjclstartap コマンドのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/vmx_avcs_logs/**	−	○	−	×	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		仮想サーバから収集したログ情報	<08-50 モードの仮想サーバマネージャログディレクトリ(vmx.log.dir)>/mngunit/*/*.zip	−	○	−	×	△
仮想サーバマネージャ	定義情報	仮想サーバマネージャプロパティファイル	<Application Server のインストールディレクトリ>/manager/vmi/config/vmi.properties	○	−	○	×	○
		仮想サーバマネージャのクライアント設定プロパティファイル	<ユーザホームディレクトリ(user.home)>/.vmirc	−	○	○	△	△
		仮想サーバマネージャのクライアント共通設定プロパティファイル	<Application Server のインストールディレクトリ>/manager/vmi/config/vmiclient.properties	○	−	○	×	○
		負荷分散機接続設定プロパティファイル	<Application Server のインストールディレクトリ>/manager/vmi/config/lb/*.properties	○	−	○	×	○
サーバ通信エージェント	メッセージログ	サーバ通信エージェントのログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/sinaviagent*.log	○	−	−	×	○
		サーバ通信エージェントのサービスのログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/sinaviagentsv*.log	○	−	−	×	△
		snactl コマンドのログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/snactl*.log	○	−	−	×	△
	そのほかのログ	サーバ通信エージェント起動時のエラー情報	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/sinaviagent.err	○	−	−	×	△
		サーバ通信エージェントの標準出力	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/sinaviagent.out	○	−	−	×	△
		サーバ通信エージェントが起動したコマンドのプロセスのコンソール出力情報	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/processConsole*.log	○	−	−	×	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
	保守ログ	サーバ通信エージェントの保守ログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/maintenance/sinaviagent*.log	−	○	−	×	△
		サーバ通信エージェントサービスの保守ログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/maintenance/sinaviagentsv*.log	−	○	−	×	△
		snactl コマンドの保守ログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/maintenance/snactl*.log	−	○	−	×	△
		サーバ通信エージェントの Javalog	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/maintenance/sinaviagent.javalog*.log	−	○	−	×	△
	定義情報	サーバ通信エージェント用オプション定義ファイル	<Application Server のインストールディレクトリ>/sinagent/config/sinaviagent.cfg	○	−	○	×	○
		サーバ通信エージェントプロパティファイル	<Application Server のインストールディレクトリ>/sinagent/config/sinaviagent.properties	○	−	○	×	○
Web Services - Base	メッセージログ	メッセージログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjmessage*.log	○	−	−	◎	△
			<Web コンテナサーバログ出力ディレクトリ(web.server.log.directory)>/cjweb_message*.log	○	−	−	△	△
			<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/cjclmessage*.log	○	−	−	△	△
		サーバトレース，クライアントトレース，デフォルトトレース，アプリケーションログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/WS/*.log	○	−	−	◎	△
			<Web コンテナサーバログ出力ディレクトリ(web.server.log.directory)>/WS/*.log	○	−	−	△	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		クライアントトレース，デフォルトトレース，提供コマンドトレース，アプリケーションログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/WS/*.log	○	−	−	△	△
		JAXR トレース	<JAXR トレースファイル名(com.cosminexus.xml.registry.trace.file_path)>*.log	○	−	−	△	△
	定義情報	共通定義ファイル	<Application Server のインストールディレクトリ>/c4web/conf/c4webcom.cfg	○	−	○	○	○
		サーバ定義ファイル	<Application Server のインストールディレクトリ>/c4web/conf/c4websv.cfg	○	−	○	○	○
		サービスデプロイ定義	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/server-config.xml	○	−	○	○	○
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/server-config.xml	○	−	○	○	○
		クライアント定義ファイル	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/classes/c4webcl.properties	○	−	○	○	○
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/classes/c4webcl.properties	○	−	○	○	○
この製品の JAX-WS 機能	メッセージログ	cjwsimport コマンドの稼働ログ	<Application Server のインストールディレクトリ>/jaxws/logs/cjwsimport*.log	○	−	−	○	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		apt コマンドの稼働ログ	<Application Server のインストールディレクトリ>/jaxws/logs/cjwapt*.log	○	−	−	○	○
		サービスおよびクライアントの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJW/cjwmessage*.log	○	−	−	◎	△
		クライアントの稼働ログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/CJW/cjwmessage*.log	○	−	−	△	△
		サービスおよびクライアントの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJR/cjrmessage*.log	○	−	−	◎	△
	そのほかのログ	サービスおよびクライアントの通信ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJW/cjwtransport*.log	○	−	−	◎	△
		クライアントの通信ログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/CJW/cjwtransport*.log	○	−	−	△	△
		サービスおよびクライアントの通信ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJR/cjrtransport*.log	○	−	−	◎	△
	例外ログ	cjwsimport コマンドの例外ログ	<Application Server のインストールディレクトリ>/jaxws/logs/cjwsimportex*.log	○	−	−	○	○
		apt コマンドの例外ログ	<Application Server のインストールディレクトリ>/jaxws/logs/cjwaptex*.log	○	−	−	○	○
		サービスおよびクライアントの例外ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJW/cjwexception*.log	○	−	−	◎	△
		クライアントの例外ログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ	○	−	−	△	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
			1(ejbserver.client.ejb.log)>/<サブディレクトリ 2(ejbserver.client.log.appid)>/CJW/cjwexception*.log					
		サービスおよびクライアントの例外ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJR/cjrexception*.log	○	−	−	◎	△
	定義情報	動作定義ファイル	<Application Server のインストールディレクトリ>/jaxws/conf/*.properties	○	−	○	○	○
		プロセス別の動作定義ファイル	<プロセス別動作定義ファイル(com.cosminexus.jaxws.confpath)>	○	−	○	×	×
		動作定義ファイル	<Application Server インストールディレクトリ>/jaxrs/conf/*.properties	○	−	○	△	△
		プロセス別の動作定義ファイル	<プロセス別動作定義ファイル(com.cosminexus.jaxrs.confpath)>	○	−	○	×	×
	定義ファイル	web.xml	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/*/WEB-INF/web.xml	−	○	○	◎	◎
		cosminexus-jaxws.xml	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/*/WEB-INF/cosminexus-jaxws.xml	−	○	○	◎	◎
		ハンドラチェイン設定ファイル	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/**	−	○	○	×	×
		WSDL	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/*/WEB-INF/wsdl/*	−	○	○	◎	◎
		カタログファイル	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/*/WEB-INF/classes/META-INF/jax-ws-catalog.xml	−	○	○	◎	◎
			<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/apps/*/*/*/jax-ws-catalog.xml	−	○	○	◎	◎

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
CJPA プロバイダ	メッセージログ	CJPA プロバイダの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjpa/cjpaoperation*.log	○	−	−	◎	△
		PRF デーモンおよび PRF コマンドのログ	<PRF スプールディレクトリ(prfspool)>/log/<PRF 識別子>/ctmlog*	○	−	−	◎	△
		J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/*.log	○	−	−	◎	△
		メッセージログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjmessage*.log	○	−	−	◎	△
	そのほかのログ	障害発生時の例外情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjexception*.log	○	−	−	◎	△
	定義情報	J2EE サーバ用オプション定義ファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/*.cfg	○	−	○	○	○
		J2EE サーバ用ユーザプロパティファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/*.properties	○	−	○	○	○
		J2EE サーバ用セキュリティポリシーファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/*.policy	○	−	○	○	○
	ユーザデータ	EJB サーバ作業ディレクトリの内容	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/**	−	○	−	×	×
		Web コンテナ作業ディレクトリの内容	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/**	−	○	−	×	×
CJMS プロバイダ	メッセージログ	CJMSP ブローカーのログ	<Application Server のインストールディレクトリ>/CC/cjmsp/var/instances/<インスタンス名>/log/*.log	○	−	−	△	△

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
		管理コマンドのログ	<Application Server のインストールディレクトリ>/CC/cjmsp/var/admin/log/*.log	○	−	−	△	△
		CJMSP リソースアダプタのログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjms/<リソースアダプタ名>/*.log	○	−	−	◎	△
	定義情報	CJMSP ブローカー共通プロパティファイル	<Application Server のインストールディレクトリ>/CC/cjmsp/lib/props/broker/commonconfig.properties	○	−	○	○	○
		CJMSP ブローカー個別プロパティファイル	<Application Server のインストールディレクトリ>/CC/cjmsp/var/instances/<インスタンス名>/props/config.properties	○	−	○	△	△
		管理コマンド定義ファイル	<Application Server のインストールディレクトリ>/CC/cjmsp/var/admin/config/admin.properties	○	−	○	△	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注※　08-50 モードの仮想サーバマネージャは，このバージョンでは使用できません。デフォルトの収集先が収集対象です。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-6　Component Container に関連する収集対象（UNIX の場合）

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
J2EE サーバ	メッセージログ	稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjmessage*.log	○	−	−	◎	△
		ログ稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjlogger.log	○	−	−	◎	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/*.log	○	−	−	◎	△
		J2EE アプリケーションに含めて使用するリソースアダプタの稼働ログ（通常モード）	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/<J2EE アプリケーション名>/*.log	○	−	−	◎	△
		J2EE アプリケーションに含めて使用するリソースアダプタの稼働ログ（テストモード）	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/TEST#<J2EE アプリケーション名>/*.log	○	−	−	◎	△
		Web サーブレットログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/web_servlet*.log	○	−	−	◎	△
	そのほかのログ	ユーザ出力ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/user_out*.log	○	−	−	◎	△
		ユーザエラーログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/user_err*.log	○	−	−	◎	△
		JavaVM の保守情報および GC のログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/javalog*.log	○	−	−	◎	△
		JavaVM の明示管理ヒープ機能イベントログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/ehjavalog*.log	○	−	−	◎	△
		開発調査ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjdevelopment*.log	○	−	−	◎	△
		障害発生時の例外情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjexception*.log	○	−	−	◎	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		JavaVM イベントログ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/hs_err*	○	−	−	◎	◎
			<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/replay_pid*.log	○	−	−	◎	◎
		J2EE アプリケーションのユーザログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/user/*	○	−	−	◎	△
	保守ログ	保守情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjmaintenance*.log	○	−	−	◎	△
		コンソールメッセージ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjconsole*.log	○	−	−	◎	△
		EJB コンテナの保守情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjejbcontainer*.log	○	−	−	◎	△
		Web コンテナの保守情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjwebcontainer*.log	○	−	−	◎	△
		起動プロセス標準出力情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjstdout*.log	○	−	−	◎	△
		起動プロセス標準エラー情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cjstderr*.log	○	−	−	◎	△
		終了プロセス情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CC/maintenance/cj_shutdown*.log	○	−	−	◎	△
	ダンプ	スレッドダンプ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/javacore*	○	−	−	◎	◎
		core ダンプ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/core*	−	○	−	◎	◎
	統計情報	稼働情報ファイル	<稼働情報ファイル出力先ディレクトリ(ejbserver.management.stats_file.dir)>/*	−	○	−	◎	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		メモリ監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjmemorywatch*.log	○	−	−	◎	△
		ファイルディスクリプタ監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjfiledescriptorwatch*.log	○	−	−	◎	△
		スレッド監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjthreadwatch*.log	○	−	−	◎	△
		スレッドダンプ監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjthreaddumpwatch*.log	○	−	−	◎	△
		HTTP リクエスト実行待ちキュー監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjrequestqueuewatch*.log	○	−	−	◎	△
		HTTP セッション数監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjhttpsessionwatch*.log	○	−	−	◎	△
		コネクションプール監視ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/watch/cjconnectionpoolwatch*.log	○	−	−	◎	△
	定義情報	J2EE サーバ用オプション定義ファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/usrconf.cfg	○	−	○	○	○
		J2EE サーバ用ユーザプロパティファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/usrconf.properties	○	−	○	○	○
		J2EE サーバ用セキュリティポリシーファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/server.policy	○	−	○	○	○
		MNG で作成した各種定義ファイルのバックアップ	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/*	−	○	−	○	○
		保護区リストファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/criticalList.cfg	○	−	○	○	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		リソース設定情報	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/import/**	−	○	−	×	×
			<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/rars/**	−	○	−	×	×
		保守情報	<Application Server のインストールディレクトリ>/CC/server/version/**	○	−	○	○	○
	そのほか	J2EE サーバの起動，停止または異常終了を示すログ	<UNIX の syslog(syslog)>	○	−	−	△	△
	ユーザデータ	EJB サーバ作業ディレクトリの内容	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/**	−	○	−	×	×
		Web コンテナ作業ディレクトリの内容	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/**	−	○	−	×	×
		JSP 用一時ディレクトリの内容	<JSP 用一時ディレクトリ(webserver.work.directory)>/**	−	○	−	×	×
サーバ管理コマンド	メッセージログ	稼働ログ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/cjmessage*.log	○	−	−	△	△
		ログ稼働ログ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/cjlogger.log	○	−	−	△	△
		互換モードの稼働ログ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*message*.log	○	−	−	△	△
	そのほかのログ	障害発生時の例外情報	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/cjexception*.log	○	−	−	△	△
		互換モードの障害発生時の例外情報	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*exception*.log	○	−	−	△	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
	保守ログ	保守情報	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/CC/maintenance/cjmaintenance*.log	○	−	−	△	△
		コンソールメッセージ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/CC/maintenance/cjconsole*.log	○	−	−	△	△
		サーバ管理コマンドの保守情報	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/CC/maintenance/cjserveradmin*.log	○	−	−	△	△
		そのほかのログ	<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*/*	○	−	−	△	△
			<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*/*/*	○	−	−	△	△
			<サーバ管理コマンドログ出力ディレクトリ(admin_ejb.server.log.directory)>/*/*/*/*	−	○	−	△	△
	定義情報	サーバ管理コマンド用定義ファイル	<Application Server のインストールディレクトリ>/CC/admin/usrconf/*	○	−	−	○	○
バッチアプリケーション	メッセージログ	cjexecjob コマンド，cjkilljob コマンドおよび cjlistjob コマンドの稼働ログ	<バッチアプリケーションログ出力ディレクトリ(batch.log.directory)>/cjmessage*.log	○	−	−	△	△
	保守ログ	そのほかのログ	<バッチアプリケーションログ出力ディレクトリ(batch.log.directory)>/*/*	○	−	−	△	△
	定義情報	バッチアプリケーション用オプション定義ファイル	<バッチアプリケーションの定義ファイル格納ディレクトリ>/usrconf.cfg	○	−	−	×	×
		バッチアプリケーション用	<バッチアプリケーションの定義ファイル格納ディレクトリ>/usrconf.properties	○	−	−	×	×

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		ユーザプロパティファイル						
リソースアダプタバージョンアップコマンド(cjrarupdate)	メッセージログ	稼働ログ	<Application Server のインストールディレクトリ>/CC/logs/cjrarupdatemessage*.log	○	−	−	○	○
	そのほかのログ	障害発生時の例外情報	<Application Server のインストールディレクトリ>/CC/logs/cjrarupdateexception*.log	○	−	−	○	○
	保守ログ	保守情報	<Application Server のインストールディレクトリ>/CC/logs/cjrarupdatemaintenance*.log	○	−	−	○	○
インプロセス HTTP サーバ	アクセスログ	インプロセス HTTP サーバの処理結果	<インプロセス HTTP サーバアクセスログファイル(webserver.logger.access_log.inprocess_http.filename)>*.log	○	−	−	◎	△
	モジュールトレース	スレッドトレースの情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/http/maintenance/thr/cjhttp_thr.*.inprocess_http.mm	○	−	−	◎	△
	通信トレース	通信トレースの情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/http/maintenance/comm/cjhttp_comm.*.inprocess_http.mm	○	−	−	◎	△
NIO HTTP サーバ	アクセスログ	NIO HTTP サーバ（HTTP）の処理結果	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cj_access_niohttp*.log	○	−	−	◎	△
		NIO HTTP サーバ（WebSocket）の処理結果	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cj_access_websocket*.log	○	−	−	◎	△
移行コマンド(cjenvupdate)	メッセージログ	cjenvupdate コマンドの稼働ログ	<Application Server のインストールディレクトリ>/CC/logs/cjenvupdatemessage*.log	○	−	−	○	○
	そのほかのログ	cjenvupdate コマンドの例外情報	<Application Server のインストールディレクトリ>/CC/logs/cjenvupdateexception*.log	○	−	−	○	○
	保守ログ	cjenvupdate コマンドの保守情報	<Application Server のインストールディレクトリ>/CC/logs/cjenvupdatemaintenance*.log	○	−	−	○	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
CC 管理者変更コマンド	メッセージログ	cjenvsetup コマンドの稼働ログ	<Application Server のインストールディレクトリ>/CC/logs/cjenvsetupmessage*.log	○	−	−	○	○
	そのほか	cjenvsetup コマンドの実行前ファイル構成情報	<Application Server のインストールディレクトリ>/CC/logs/before_cjenvsetup_files*.txt	○	−	−	○	○
		cjenvsetup コマンドの実行後ファイル構成情報	<Application Server のインストールディレクトリ>/CC/logs/after_cjenvsetup_files*.txt	○	−	−	○	○
インプロセストランザクションサービス	そのほか	インプロセストランザクションサービスのステータスファイル	<ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory1)>/*	−	○	−	◎	△
			<ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory1)>/*/*	−	○	−	◎	△
		インプロセストランザクションサービスの予備ステータスファイル	<予備ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory2)>/*	−	○	−	◎	×
			<予備ステータスファイルのディレクトリ(ejbserver.distributedtx.ots.status.directory2)>/*/*	−	○	−	◎	×
EJB クライアント	メッセージログ	稼働ログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/cjclmessage*.log	○	−	−	△	△
			<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/cjclmessage*.log	○	−	−	×	×

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		cjclstartap コマンドの稼働ログ	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/cjclstartap*.log	○	−	−	×	×
		cjcldellog コマンドの稼働ログ	<Application Server のインストールディレクトリ>/CC/client/logs/cjcldellog.log	○	−	−	○	○
	そのほかのログ	ユーザ出力ログ	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/user_out*.log	○	−	−	×	×
		ユーザエラーログ	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/user_err*.log	○	−	−	×	×
		JavaVM の保守情報，GC のログ	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/javalog*.log	○	−	−	×	×
		障害発生時の例外情報	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/cjclexception*.log	○	−	−	△	△
			<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/cjclexception*.log	○	−	−	×	×
		EJB クライアントアプリケー	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/user/*	○	−	−	△	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		ションのユーザログ	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/user/*	○	−	−	×	×
	保守ログ	保守情報	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/cjclmaintenance*.log	○	−	−	△	△
			<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjclmaintenance*.log	○	−	−	×	×
		EJB コンテナの保守情報	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjejbcontainer*.log	○	−	−	×	×
		起動プロセス標準出力情報	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjstdout*.log	○	−	−	×	×
		起動プロセス標準エラー情報	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/<EJB クライアントログサブディレクトリ1(ejb.client.ejb.log)>/<EJB クライアントログサブディレクトリ2(ejb.client.log.appid)>/maintenance/cjstderr*.log	○	−	−	×	×
		ログの稼働情報	<EJB クライアントログ出力ディレクトリ(ejb.client.log.directory)>/cjlogger.log	○	−	−	×	×
	定義情報	EJB クライアント用オプション定義ファイル	<EJB クライアントの定義ファイル格納ディレクトリ>/usrconf.cfg	○	−	−	×	×

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		EJB クライアント用ユーザプロパティファイル	<EJB クライアントの定義ファイル格納ディレクトリ>/usrconf.properties	○	−	−	×	×
リダイレクタ(Web サーバ)	メッセージログ	HWS 用リダイレクタのメッセージログ	<HWS 用リダイレクタログ出力ディレクトリ(JkLogFileDir)>/hws_redirect*.log	○	−	−	◎	△
		旧バージョン互換 HWS 用リダイレクタのメッセージログ	<HWS 用リダイレクタログ出力ディレクトリ(JkLogFileDir)>/hws_redirect*.log	○	−	−	◎	△
	保守ログ	HWS 用リダイレクタの保守用トレースログ	<HWS 用リダイレクタトレースログ出力ディレクトリ(JkTraceLogFileDir)>/hws_rd_trace*.log	○	−	−	◎	△
		旧バージョン互換 HWS 用リダイレクタの保守用トレースログ	<HWS 用リダイレクタトレースログ出力ディレクトリ(JkTraceLogFileDir)>/hws_rd_trace*.log	○	−	−	◎	△
	定義情報	HWS 用リダイレクタ動作定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/servers/<サーバ名称>/mod_jk.conf	○	−	○	○	○
		ワーカ定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/servers/<サーバ名称>/workers.properties	○	−	○	○	○
		旧バージョン互換 HWS 用リダイレクタ動作定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/mod_jk.conf	○	−	−	○	○
		旧バージョン互換用ワーカ定義ファイル	<Application Server のインストールディレクトリ>/CC/web/redirector/workers.properties	○	−	−	○	○
TP1/Message Queue-Access	保守ログ	API，および Application Server インタフェースに関するメソッドトレース，メッセージログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/*.log	○	−	−	◎	△
	API トレース	API トレースファイル	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/mqc.api*	○	−	−	◎	◎

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
TP1 Connector	メッセージログ	J2EE リソースアダプタとしてデプロイして使用する TP1 Connector の稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/*.log	○	−	−	◎	△
		J2EE アプリケーションに含めて使用する TP1 Connector の稼働ログ（通常モード）	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/<J2EE アプリケーション名>/*.log	○	−	−	◎	△
		Non-managed 環境で使用する場合の TP1 Connector の稼働ログ	<TP1Connector ログ出力ディレクトリ(jp.co.hitachi_system.tp1connector.logdestination)>/tp1connector*.log	○	−	−	△	△
TP1/Client/J	そのほかのログ	デバッグトレース情報	<ユーザホームディレクトリ(user.home)>/TP1clientJ/dcClt*.dmp	−	○	−	△	△
Manager	メッセージログ	統合メッセージログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/message/mngmessage*.log	○	−	−	◎	○
		運用管理エージェントのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagent*.log	○	−	−	◎	○
		運用管理エージェントの起動・停止コマンドのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagentctl.*.log	○	−	−	◎	○
		運用監視エージェントのログ・トレース	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngagent*.*.log	○	−	−	◎	○
		Management Server 起動コマンドのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvrctlstart.*.log	○	−	−	◎	○
		Management Server 停止コマンドのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvrctlstop.*.log	○	−	−	◎	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		Management Server セットアップコマンドのログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvrctlsetup.*.log	○	−	−	◎	○
		Management Server のログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/mngsvr*.log	○	−	−	◎	○
	そのほかのログ	運用管理エージェントの標準エラー出力	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagent.err.*.log	○	−	−	◎	○
		運用管理エージェントの標準出力	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagent.out.*.log	○	−	−	◎	○
		運用管理エージェントの標準エラー出力コマンドライン	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/adminagent.err	○	−	−	◎	○
		コンソールログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/processConsole*.log	○	−	−	◎	○
		明示管理ヒープ機能で使用する自動配置用設定ファイル	<Application Server インストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/auto_explicit_memory.cfg	○	−	○	△	△
		ユーザ拡張性能解析トレースで使用するユーザ拡張性能解析トレース設定ファイル	<Application Server インストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/userprf.cfg	○	−	○	△	△
		セットアップウィザードなどのセットアップコマンドの定義ファイル	<Application Server インストールディレクトリ>/manager/setup/config/*	○	−	○	○	○
		セットアップウィザードなどのセットアップコマンドのログ	<Application Server インストールディレクトリ>/manager/setup/log/*.log	○	−	−	△	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
	保守ログ	コマンド保守ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/mngcmd*.log	−	○	−	○	○
		運用管理エージェントが行うRMI 処理での保守ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/mngrmi*.log	−	○	−	○	○
		運用管理エージェントの保守ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/adminagent*.log	○	−	−	○	○
		mngenvsetup コマンドの実行ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/mngenvsetup*.log	−	○	−	○	○
		Management Server の保守ログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/maintenance/mngsvr*.log	○	−	−	○	○
	定義情報	運用管理エージェントプロパティファイル	<Application Server のインストールディレクトリ>/manager/config/adminagent.properties	○	−	○	○	○
		運用管理エージェント自動起動用設定ファイル	<Application Server のインストールディレクトリ>/manager/config/AdminAgentrc	○	−	○	○	○
		運用管理エージェント用オプション定義ファイル	<Application Server のインストールディレクトリ>/manager/config/adminagentuser.cfg	○	−	○	○	○
		運用管理エージェント設定ファイル	<Application Server のインストールディレクトリ>/manager/config/adminagent.xml	○	−	○	○	○
		運用監視エージェントプロパティファイル	<Application Server のインストールディレクトリ>/manager/config/mngagent.<サーバ名称>.properties	○	−	○	○	○
		Management Server 環境設定ファイル	<Application Server のインストールディレクトリ>/manager/config/mserver.properties	○	−	○	○	○
		Management Server 用オプション定義ファイル	<Application Server のインストールディレクトリ>/manager/config/mserver.cfg	○	−	○	○	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		Management Server 用環境変数定義ファイル	<Application Server のインストールディレクトリ>/manager/config/mserverenv.cfg	○	−	○	○	○
		Manager 設定ファイル	<Application Server のインストールディレクトリ>/manager/config/manager.cfg	○	−	○	○	○
		Management アクション実行用プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/maction.properties	○	−	○	○	○
		Management イベント発行用プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/mevent.<サーバ名称>.properties	−	○	○	○	○
		Management イベント発行用メッセージ ID リストファイル	<Management イベント発行用メッセージ ID リストファイル (manager.mevent.message_id.list)>	−	○	○	×	×
		mngsvrutil コマンドのクライアント側定義ファイル	<ユーザホームディレクトリ (user.home)>/.mngsvrutilrc	−	○	○	△	△
		mngsvrutil コマンドのサーバ側定義ファイル	<Application Server のインストールディレクトリ>/manager/config/mngsvrutil.properties	○	−	○	○	○
		mngsvrutil コマンドのクライアント側共通定義ファイル	<Application Server のインストールディレクトリ>/manager/config/mngsvrutilcl.properties	○	−	○	○	○
		JP1/IM 連携用システムログメッセージマッピングファイル	<Application Server のインストールディレクトリ>/manager/config/mserver.jp1event.system.mapping.properties	○	−	○	○	○
			<Application Server のインストールディレクトリ>/manager/config/manager.jp1event.system.mapping.properties	○	−	○	○	○
			<Application Server のインストールディレクトリ>/manager/config/manager.<サーバ名称	○	−	○	○	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
			>.jp1event.system.mapping.properties					
	性能，障害解析トレース	PRF トレースなど	<一時 PRF トレースファイル出力用ディレクトリ(adminagent.prftrace_dir)>/*.zip	−	○	−	◎	×
	内部インタフェーストレース	統合トレースログ	<Manager ログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/trace/mngtrace*.log	○	−	−	○	○
	そのほか	OS の状態情報など	<Application Server のインストールディレクトリ>/manager/tmp/*	○	−	−	○	○
Smart Composer	定義情報	サーバ設定プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/cmxserver.properties	○	−	○	○	○
		クライアント設定プロパティファイル	<ユーザホームディレクトリ(user.home)>/.cmxrc	−	○	○	△	△
		クライアント共通設定プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/cmxclient.properties	○	−	○	○	○
		負荷分散機定義プロパティファイル	<Application Server のインストールディレクトリ>/manager/config/lb.properties	○	−	○	○	○
統合ユーザ管理	定義情報	JAAS のコンフィグレーションファイル	<Application Server のインストールディレクトリ>/manager/config/jaas.conf	○	−	○	○	○
		統合ユーザ管理のコンフィグレーションファイル	<Application Server のインストールディレクトリ>/manager/config/ua.conf	○	−	○	○	○
	API トレース	統合ユーザ管理のトレース	<統合ユーザ管理のトレースファイル名(com.cosminexus.admin.auth.trace.prefix)>.*.log	−	○	−	×	×
08-50 モードの仮想サーバマネージャ※	定義情報	仮想サーバマネージャプロパティファイル	<Application Server のインストールディレクトリ>/manager/vmx/config/vmx.properties	○	−	○	×	○
		仮想サーバマネージャのクライアント共通設	<Application Server のインストールディレクトリ>/manager/vmx/config/vmxclient.properties	○	−	○	×	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
		定プロパティファイル						
		VMware vCenter Serverの接続処理のオプション定義ファイル	<Application Serverのインストールディレクトリ>/manager/vmx/config/vmx_avcs_usrconf.cfg	○	−	○	×	○
		VMware vCenter Serverの接続処理のプロパティファイル	<Application Serverのインストールディレクトリ>/manager/vmx/config/vmx_avcs_usrconf.properties	○	−	○	×	○
		VMware vCenter Serverの接続処理の共通定義ファイル	<Application Serverのインストールディレクトリ>/manager/vmx/config/vmx_avcs_cjwconf.properties	○	−	○	×	○
	そのほかのログ	仮想サーバマネージャで使用するcjclstartapコマンドのログ	<Managerログ出力ディレクトリ(com.cosminexus.manager.log.dir)>/vmx_avcs_logs/**	−	○	−	×	○
		仮想サーバから収集したログ情報	<08-50モードの仮想サーバマネージャログディレクトリ(vmx.log.dir)>/mngunit/*/*.zip	−	○	−	×	△
仮想サーバマネージャ	定義情報	仮想サーバマネージャプロパティファイル	<Application Serverのインストールディレクトリ>/manager/vmi/config/vmi.properties	○	−	○	×	○
		仮想サーバマネージャのクライアント設定プロパティファイル	<ユーザホームディレクトリ(user.home)>/.vmirc	−	○	○	△	△
		仮想サーバマネージャのクライアント共通設定プロパティファイル	<Application Serverのインストールディレクトリ>/manager/vmi/config/vmiclient.properties	○	−	○	×	○
		負荷分散機接続設定プロパティファイル	<Application Serverのインストールディレクトリ>/manager/vmi/config/lb/*.properties	○	−	○	×	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
サーバ通信エージェント	メッセージログ	サーバ通信エージェントのログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/sinaviagent*.log	○	−	−	×	○
		サーバ通信エージェントのサービスのログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/sinaviagentsv*.log	○	−	−	×	△
		snactl コマンドのログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/snactl*.log	○	−	−	×	△
	そのほかのログ	サーバ通信エージェント起動時のエラー情報	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/sinaviagent.err	○	−	−	×	△
		サーバ通信エージェントの標準出力	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/sinaviagent.out	○	−	−	×	△
		サーバ通信エージェントが起動したコマンドのプロセスのコンソール出力情報	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/processConsole*.log	○	−	−	×	△
	保守ログ	サーバ通信エージェントの保守ログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/maintenance/sinaviagent*.log	−	○	−	×	△
		snactl コマンドの保守ログ	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/maintenance/snactl*.log	−	○	−	×	△
		サーバ通信エージェントのJavalog	<サーバ通信エージェントログ出力ディレクトリ(sinaviagent.log.dir)>/maintenance/sinaviagent.javalog*.log	−	○	−	×	△
	定義情報	サーバ通信エージェント用オプション定義ファイル	<Application Server のインストールディレクトリ>/sinagent/config/sinaviagent.cfg	○	−	○	×	○
		サーバ通信エージェントプロパティファイル	<Application Server のインストールディレクトリ>/sinagent/config/sinaviagent.properties	○	−	○	×	○
Web Services - Base	メッセージログ	メッセージログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjmessage*.log	○	−	−	◎	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
			<Web コンテナサーバログ出力ディレクトリ(web.server.log.directory)>/cjweb_message*.log	○	−	−	△	△
			<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/cjclmessage*.log	○	−	−	△	△
		サーバトレース，クライアントトレース，デフォルトトレース，アプリケーションログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/WS/*.log	○	−	−	◎	△
			<Web コンテナサーバログ出力ディレクトリ(web.server.log.directory)>/WS/*.log	○	−	−	△	○
		クライアントトレース，デフォルトトレース，提供コマンドトレース，アプリケーションログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/WS/*.log	○	−	−	△	△
		JAXR トレース	<JAXR トレースファイル名(com.cosminexus.xml.registry.trace.file_path)>*.log	○	−	−	△	△
	定義情報	共通定義ファイル	<Application Server のインストールディレクトリ>/c4web/conf/c4webcom.cfg	○	−	○	○	○
		サーバ定義ファイル	<Application Server のインストールディレクトリ>/c4web/conf/c4websv.cfg	○	−	○	○	○
		サービスデプロイ定義	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/server-config.xml	○	−	○	○	○
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/	○	−	○	○	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
			webapps/<コンテキストルート>/WEB-INF/server-config.xml					
		クライアント定義ファイル	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/classes/c4webcl.properties	○	−	○	○	○
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/classes/c4webcl.properties	○	−	○	○	○
この製品の JAX-WS 機能	メッセージログ	サービスおよびクライアントの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJW/cjwmessage*.log	○	−	−	◎	△
		クライアントの稼働ログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/CJW/cjwmessage*.log	○	−	−	△	△
		サービスおよびクライアントの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJR/cjrmessage*.log	○	−	−	◎	△
	そのほかのログ	サービスおよびクライアントの通信ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJW/cjwtransport*.log	○	−	−	◎	△
		クライアントの通信ログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/CJW/cjwtransport*.log	○	−	−	△	△
		サービスおよびクライアントの通信ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJR/cjrtransport*.log	○	−	−	◎	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
	例外ログ	サービスおよびクライアントの例外ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJW/cjwexception*.log	○	−	−	◎	△
		クライアントの例外ログ	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/CJW/cjwexception*.log	○	−	−	△	△
		サービスおよびクライアントの例外ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CJR/cjrexception*.log	○	−	−	◎	△
	定義情報	動作定義ファイル	<Application Server のインストールディレクトリ>/jaxws/conf/*.properties	○	−	○	○	○
		プロセス別の動作定義ファイル	<プロセス別動作定義ファイル(com.cosminexus.jaxws.confpath)>	○	−	○	×	×
		動作定義ファイル	<Application Server インストールディレクトリ>/jaxrs/conf/*.properties	○	−	○	△	△
		プロセス別の動作定義ファイル	<プロセス別動作定義ファイル(com.cosminexus.jaxrs.confpath)>	○	−	○	×	×
	定義ファイル	web.xml	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/*/WEB-INF/web.xml	−	○	○	◎	◎
		cosminexus-jaxws.xml	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/*/WEB-INF/cosminexus-jaxws.xml	−	○	○	◎	◎
		ハンドラチェイン設定ファイル	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/**	−	○	○	×	×
		WSDL	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/*/WEB-INF/wsdl/*	−	○	○	◎	◎
		カタログファイル	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/*/WEB-INF/classes/META-INF/jax-ws-catalog.xml	−	○	○	◎	◎

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
			<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/apps/*/*/*/jax-ws-catalog.xml	−	○	○	◎	◎
CJPA プロバイダ	メッセージログ	CJPA プロバイダの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjpa/cjpaoperation*.log	○	−	−	◎	△
		PRF デーモンおよび PRF コマンドのログ	<PRF スプールディレクトリ(prfspool)>/log/<PRF 識別子>/ctmlog*	○	−	−	◎	△
		J2EE リソースアダプタとしてデプロイして使用するリソースアダプタの稼働ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/connectors/*.log	○	−	−	◎	△
		メッセージログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjmessage*.log	○	−	−	◎	△
	そのほかのログ	障害発生時の例外情報	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjexception*.log	○	−	−	◎	△
	定義情報	J2EE サーバ用オプション定義ファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/*.cfg	○	−	○	○	○
		J2EE サーバ用ユーザプロパティファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/*.properties	○	−	○	○	○
		J2EE サーバ用セキュリティポリシーファイル	<Application Server のインストールディレクトリ>/CC/server/usrconf/ejb/<サーバ名称>/*.policy	○	−	○	○	○
	ユーザデータ	EJB サーバ作業ディレクトリの内容	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/**	−	○	−	×	×
		Web コンテナ作業ディレクトリの内容	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/web/<サーバ名称>/**	−	○	−	×	×
CJMS プロバイダ	メッセージログ	CJMSP ブローカーのログ	<Application Server のインストールディレクトリ>/CC/cjmsp/var/	○	−	−	△	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
			instances/<インスタンス名>/log/*.log					
		管理コマンドのログ	<Application Server のインストールディレクトリ>/CC/cjmsp/var/admin/log/*.log	○	−	−	△	△
		CJMSP リソースアダプタのログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/cjms/<リソースアダプタ名>/*.log	○	−	−	◎	△
	定義情報	CJMSP ブローカー共通プロパティファイル	<Application Server のインストールディレクトリ>/CC/cjmsp/lib/props/broker/commonconfig.properties	○	−	○	○	○
		CJMSP ブローカー個別プロパティファイル	<Application Server のインストールディレクトリ>/CC/cjmsp/var/instances/<インスタンス名>/props/config.properties	○	−	○	△	△
		管理コマンド定義ファイル	<Application Server のインストールディレクトリ>/CC/cjmsp/var/admin/config/admin.properties	○	−	○	△	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注※　08-50 モードの仮想サーバマネージャは，このバージョンでは使用できません。デフォルトの収集先が収集対象です。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.3　Component Transaction Monitor

Component Transaction Monitor に関連する収集対象を次の表に示します。

表 A-7　Component Transaction Monitor に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	CTM デーモンおよび CTM コマンドのログ	<CTM スプールディレクトリ(ctmspool)>/log/<CTM 識別子>/ctmlog*	○	－	－	◎	△
	CTM ドメインマネジャのログ	<CTM スプールディレクトリ(ctmspool)>/log/ctmdmlog*	○	－	－	◎	△
ダンプ	スレッドダンプ	<Application Server のインストールディレクトリ>/TPB/logj/javacore*	○	－	－	○	○
定義情報	CTM レギュレータの設定ファイル	<CTM レギュレータの設定ファイル(ctm.RegOption)>	○	－	－	◎	×
	OTM ゲートウェイの設定ファイル	<OTM ゲートウェイの設定ファイル(ctm.TSCGwOption)>	○	－	－	◎	×

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-8　Component Transaction Monitor に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	CTM デーモンおよび CTM コマンドのログ	<CTM スプールディレクトリ(ctmspool)>/log/<CTM 識別子>/ctmlog*	○	－	－	◎	△
	CTM ドメインマネジャのログ	<CTM スプールディレクトリ(ctmspool)>/log/ctmdmlog*	○	－	－	◎	△
ダンプ	スレッドダンプ	<Application Server のインストールディレクトリ>/TPB/logj/javacore*	○	－	－	○	○

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
	core ダンプ	<Application Server のインストールディレクトリ>/TPB/logj/core*	−	○	−	○	○
定義情報	CTM レギュレータの設定ファイル	<CTM レギュレータの設定ファイル(ctm.RegOption)>	○	−	−	◎	×
	OTM ゲートウェイの設定ファイル	<OTM ゲートウェイの設定ファイル(ctm.TSCGwOption)>	○	−	−	◎	×

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.4　Developer's Kit for Java

Developer's Kit for Java に関連する収集対象を次の表に示します。

表 A-9　Developer's Kit for Java に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
ダンプ	クラッシュダンプ	<Windows のクラッシュダンプファイル(CrashDumpFile)>	−	○	−	△	△
		<Windows のクラッシュダンプ出力ディレクトリ(CrashDumpDir)>/*.dmp	−	○	−	△	△
定義情報	ユーザ拡張性能解析トレース設定ファイル	<ユーザ拡張性能解析トレース設定ファイル(jvm.userprf.File)>	○	−	○	△	△
	明示管理ヒープ機能適用除外設定ファイル	<明示管理ヒープ機能適用除外設定ファイル名(jvm.exmemexcludeclass.File)>	○	−	○	△	△

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
	明示管理ヒープ機能適用除外設定の無効化対象設定ファイル	<明示管理ヒープ機能適用除外設定の無効化対象設定ファイル名(jvm.exmemnotexcludeclass.File)>	○	－	○	△	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-10　Developer's Kit for Java に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
定義情報	ユーザ拡張性能解析トレース設定ファイル	<ユーザ拡張性能解析トレース設定ファイル(jvm.userprf.File)>	○	－	○	△	△
	明示管理ヒープ機能適用除外設定ファイル	<明示管理ヒープ機能適用除外設定ファイル名(jvm.exmemexcludeclass.File)>	○	－	○	△	△
	明示管理ヒープ機能適用除外設定の無効化対象設定ファイル	<明示管理ヒープ機能適用除外設定の無効化対象設定ファイル名(jvm.exmemnotexcludeclass.File)>	○	－	○	△	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.5　Performance Tracer

Performance Tracer に関連する収集対象を次の表に示します。

表 A-11　Performance Tracer に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	PRF デーモンおよび PRF コマンドのログ	<PRF スプールディレクトリ(prfspool)>/log/<PRF 識別子>/ctmlog*	○	－	－	◎	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-12　Performance Tracer に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	PRF デーモンおよび PRF コマンドのログ	<PRF スプールディレクトリ(prfspool)>/log/<PRF 識別子>/ctmlog*	○	－	－	◎	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.6　Web Services - Security

Web Services - Security に関連する収集対象を次の表に示します。

表 A-13　Web Services - Security に関連する収集対象（Windows の場合）

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
Web Services - Security	メッセージログ	コマンドトレース，クライアントトレース	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/<サブディレクトリ2(ejbserver.client.log.appid)>/WS/*.log	○	−	−	△	△
		サーバトレース，クライアントトレース	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/WS/*.log	○	−	−	◎	△
			<Web コンテナサーバログ出力ディレクトリ(web.server.log.directory)>/WS/*.log	○	−	−	△	△
	定義情報	環境設定ファイル	<Application Server のインストールディレクトリ>/wss/conf/cwsscfg.properties	○	−	○	○	○
		クライアントデプロイ定義	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/classes/client-config.xml	○	−	○	○	○
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/classes/client-config.xml	○	−	○	○	○
		ポリシー定義ファイル	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/classes/policy-config.xml	○	−	○	○	○

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/classes/policy-config.xml	○	−	○	○	○
		機能定義ファイル	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/classes/security-config.xml	○	−	○	○	○
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/classes/security-config.xml	○	−	○	○	○
XML Security - Core	保守ログ	メソッド呼び出しなどのトレース保守情報	<XML Security - Core トレース出力ディレクトリ(com.cosminexus.xml.security.logging.trace_dir)>/*.log	○	−	−	△	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-14　Web Services - Security に関連する収集対象（UNIX の場合）

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
Web Services - Security	メッセージログ	コマンドトレース，クライアントトレース	<EJB クライアントログ出力ディレクトリ(ejbserver.client.log.directory)>/<サブディレクトリ1(ejbserver.client.ejb.log)>/	○	−	−	△	△

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
			<サブディレクトリ2(ejbserver.client.log.appid)>/WS/*.log					
		サーバトレース，クライアントトレース	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/WS/*.log	○	−	−	◎	△
			<Web コンテナサーバログ出力ディレクトリ(web.server.log.directory)>/WS/*.log	○	−	−	△	△
	定義情報	環境設定ファイル	<Application Server のインストールディレクトリ>/wss/conf/cwsscfg.properties	○	−	○	○	○
		クライアントデプロイ定義	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/classes/client-config.xml	○	−	○	○	○
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/classes/client-config.xml	○	−	○	○	○
		ポリシー定義ファイル	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート>/WEB-INF/classes/policy-config.xml	○	−	○	○	○
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/classes/policy-config.xml	○	−	○	○	○
		機能定義ファイル	<Application Server のインストールディレクトリ>/CC/server/public/web/<サーバ名称>/<コンテキストルート	○	−	○	○	○

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
			>/WEB-INF/classes/security-config.xml					
			<Application Server のインストールディレクトリ>/CC/web/containers/<サーバ名称>/webapps/<コンテキストルート>/WEB-INF/classes/security-config.xml	○	−	○	○	○
XML Security - Core	保守ログ	メソッド呼び出しなどのトレース保守情報	<XML Security - Core トレース出力ディレクトリ(com.cosminexus.xml.security.logging.trace_dir)>/*.log	○	−	−	△	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.7　HTTP Server

HTTP Server に関連する収集対象を次の表に示します。

表 A-15　HTTP Server に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	エラーログ	<HWS エラーログディレクトリ(HttpsdErrorLogFileDir)>/error*	○	−	−	◎	△
	ログ一式(旧バージョン互換用)	<HWS のインストールディレクトリ>/logs/*	○	−	−	○	○
そのほかのログ	リクエストログ	<HWS リクエストログディレクトリ(HttpsdRequestLogFileDir)>/hwsrequest*	−	○	−	◎	△

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
	プロセス ID ファイル	<HWS プロセス ID ファイル (PidFile)>	○	−	−	△	△
アクセスログ	アクセスログ	<HWS アクセスログディレクトリ (HttpsdCustomLogFileDir)>/access*	−	○	−	◎	△
定義情報	定義ファイル	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>/conf/*.conf	○	−	○	○	○
	定義ファイル一式(旧バージョン互換用)	<HWS のインストールディレクトリ>/conf/*.conf	○	−	○	○	○
内部インタフェーストレース	内部トレース	<HWS 内部トレースディレクトリ (HttpsdTraceLogFileDir)>/hws.trc*	−	○	−	◎	△
WebSocket ログ	WebSocket ログ	<HWSWebSocket ログディレクトリ (HttpsdWebSocketLogFileDir)>/hws_websocket_log*	−	○	−	◎	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-16　HTTP Server に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	エラーログ	<HWS エラーログディレクトリ (HttpsdErrorLogFileDir)>/error*	○	−	−	◎	△
	ログ一式(旧バージョン互換用)	<HWS のインストールディレクトリ>/logs/*	○	−	−	○	○

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
そのほかのログ	リクエストログ	<HWS リクエストログディレクトリ(HttpsdRequestLogFileDir)>/hwsrequest*	−	○	−	◎	△
	プロセス ID ファイル	<HWS プロセス ID ファイル(PidFile)>	○	−	−	△	△
アクセスログ	アクセスログ	<HWS アクセスログディレクトリ(HttpsdCustomLogFileDir)>/access*	−	○	−	◎	△
ダンプ	core ダンプ	<HWS コアダンプ出力ディレクトリ(CoreDumpDirectory)>/core*	−	○	−	◎	△
	gcache サーバ core ダンプ	<HWS キャッシュサーバの作業ディレクトリ(SSLCacheServerRunDir)>/core*	−	○	−	△	△
定義情報	定義ファイル	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>/conf/*.conf	○	−	○	○	○
	定義ファイル一式(旧バージョン互換用)	<HWS のインストールディレクトリ>/conf/*.conf	○	−	○	○	○
内部インタフェーストレース	内部トレース	<HWS 内部トレースディレクトリ(HttpsdTraceLogFileDir)>/hws.trc*	−	○	−	◎	△
WebSocket ログ	WebSocket ログ	<HWSWebSocket ログディレクトリ(HttpsdWebSocketLogFileDir)>/hws_websocket_log*	−	○	−	◎	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.8　Microsoft Internet Information Service

Microsoft Internet Information Service の情報に関連する収集対象を次の表に示します。

表 A-17　Microsoft Internet Information Service の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	アクセスログ	<IIS アクセスログディレクトリ(IIS_log_dir)>/*	－	○	－	△	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.9　HCSC サーバ

HCSC サーバの情報に関連する収集対象を次の表に示します。

表 A-18　HCSC サーバの情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	HCSC-Manager ログ	<HCSC ログ出力ディレクトリ(cscmng.log.dir)>/*	○	－	－	△	△
	ユーザ認証情報管理コマンドログ	<ユーザ認証情報管理コマンドのメッセージログファイルの出力先ディレクトリ(authinfo.command.messagelog.filepath)>/*	○	－	－	△	△
アクセスログ	リクエストトレース	<HCSC サーバのプロパティ(requesttrace-filepath)>/*	○	－	－	△	△
保守ログ	メソッドトレース	<HCSC サーバのプロパティ(methodtrace-filepath)>/*	○	－	－	△	△

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
定義情報	HCSC サーバの定義情報	<Application Server のインストールディレクトリ>/CSC/system/msg/*	○	−	○	○	○
	HCSC-Manager 定義ファイル	<Application Server のインストールディレクトリ>/CSC/config/manager/*	○	−	○	○	○
	HCSC-Messaging 定義ファイル	<Application Server のインストールディレクトリ>/CSC/config/msg/*	○	−	○	○	○
	リポジトリ	<HCSC リポジトリルート(cscmng.repository.root)>/**	○	−	−	○	○
その他のログ	ビジネスプロセスのアクティビティトレース	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/csc/*	−	○	−	◎	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-19　HCSC サーバの情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	HCSC-Manager ログ	<HCSC ログ出力ディレクトリ(cscmng.log.dir)>/*	○	−	−	△	△
アクセスログ	リクエストトレース	<HCSC サーバのプロパティ(requesttrace-filepath)>/*	○	−	−	△	△
保守ログ	メソッドトレース	<HCSC サーバのプロパティ(methodtrace-filepath)>/*	○	−	−	△	△
定義情報	HCSC サーバの定義情報	<Application Server のインストールディレクトリ>/CSC/system/msg/*	○	−	○	○	○

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
	HCSC-Manager 定義ファイル	<Application Server のインストールディレクトリ>/CSC/config/manager/*	○	−	○	○	○
	HCSC-Messaging 定義ファイル	<Application Server のインストールディレクトリ>/CSC/config/msg/*	○	−	○	○	○
	リポジトリ	<HCSC リポジトリルート(cscmng.repository.root)>/**	○	−	−	○	○
その他のログ	ビジネスプロセスのアクティビティトレース	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/csc/*	−	○	−	◎	△

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.10　HCSC サーバ（FTP 受付）

HCSC サーバ（FTP 受付）の情報に関連する収集対象を次の表に示します。

表 A-20　HCSC サーバ（FTP 受付）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
定義情報	FTP 受付の定義情報	<Application Server のインストールディレクトリ>/CSC/config/ftprecp/*	○	−	○	○	○
	FTP 受付の共通定義情報	<Application Server のインストールディレクトリ>/CSC/config/ftprecp/common/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-21　HCSC サーバ（FTP 受付）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
定義情報	FTP 受付の定義情報	<Application Server のインストールディレクトリ>/CSC/config/ftprecp/*	○	－	○	○	○
	FTP 受付の共通定義情報	<Application Server のインストールディレクトリ>/CSC/config/ftprecp/common/*	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.11　HCSC サーバ（TP1 アダプタ）

HCSC サーバ（TP1 アダプタ）の情報に関連する収集対象を次の表に示します。

表 A-22　HCSC サーバ（TP1 アダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	トレース一式	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<	－	○	－	◎	◎

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
		サーバ名称>/logs/CSCADP/TP1ADP/maintenance/*/*					
定義情報	TP1 アダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/TP1/config/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-23　HCSC サーバ（TP1 アダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	トレース一式	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/TP1ADP/maintenance/*/*	−	○	−	◎	◎
定義情報	TP1 アダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/TP1/config/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.12　HCSC サーバ（ファイルアダプタ）

HCSC サーバ（ファイルアダプタ）の情報に関連する収集対象を次の表に示します。

表 A-24　HCSC サーバ（ファイルアダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	トレース一式	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/FFADP/maintenance/*/*	−	○	−	◎	◎
定義情報	ファイルアダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/File/config/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-25　HCSC サーバ（ファイルアダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	トレース一式	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/FFADP/maintenance/*/*	−	○	−	◎	◎
定義情報	ファイルアダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/File/config/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合

「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.13　HCSC サーバ（Object Access アダプタ）

HCSC サーバ（Object Access アダプタ）の情報に関連する収集対象を次の表に示します。

表 A-26　HCSC サーバ（Object Access アダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	トレース一式	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/OAADP/maintenance/*/*	－	○	－	◎	◎
定義情報	OA アダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/OA/config/*	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-27　HCSC サーバ（Object Access アダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	トレース一式	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/OAADP/maintenance/*/*	－	○	－	◎	◎

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
定義情報	OA アダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/OA/config/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.14　HCSC サーバ（Message Queue アダプタ）

HCSC サーバ（Message Queue アダプタ）の情報に関連する収集対象を次の表に示します。

表 A-28　HCSC サーバ（Message Queue アダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	トレース一式	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/MQADP/maintenance/*/*	−	○	−	◎	◎
定義情報	MQ アダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/MQ/config/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-29　HCSC サーバ（Message Queue アダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	トレース一式	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/MQADP/maintenance/*/*	－	○	－	◎	◎
定義情報	MQ アダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/MQ/config/*	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.15　HCSC サーバ（FTP アダプタ）

HCSC サーバ（FTP アダプタ）の情報に関連する収集対象を次の表に示します。

表 A-30　HCSC サーバ（FTP アダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
通信トレース	FTP アダプタのプロトコルトレース	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/FTPADP/*/*	－	○	－	◎	△
保守ログ	FTP アダプタの保守ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/FTPADP/maintenance/*/*	－	○	－	◎	△

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	FTP アダプタの運用コマンドのメッセージログ	<FTP アダプタコマンドメッセージログ出力先ディレクトリ(ftpadp.command.messagelog.filepath)>/*	−	○	−	△	△
定義情報	FTP アダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/FTP/config/*	○	−	○	○	○
	FTP アダプタの共通定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/FTP/config/common/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-31　HCSC サーバ（FTP アダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
通信トレース	FTP アダプタのプロトコルトレース	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/FTPADP/*/*	−	○	−	◎	△
保守ログ	FTP アダプタの保守ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/FTPADP/maintenance/*/*	−	○	−	◎	△
メッセージログ	FTP アダプタの運用コマンドのメッセージログ	<FTP アダプタコマンドメッセージログ出力先ディレクトリ(ftpadp.command.messagelog.filepath)>/*	−	○	−	△	△
定義情報	FTP アダプタの定義情報	<Application Server インストールディレクトリ>/CSC/custom-adapter/FTP/config/*	○	−	○	○	○

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
	FTP アダプタの共通定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/FTP/config/common/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.16　HCSC サーバ（SFTP アダプタ）

HCSC サーバ（SFTP アダプタ）の情報に関連する収集対象を次の表に示します。

表 A-32　HCSC サーバ（SFTP アダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
通信トレース	SFTP アダプタのプロトコルトレース	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/SFTPADP/*/*	−	○	−	◎	◎
保守ログ	SFTP アダプタの保守ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/SFTPADP/maintenance/*/*	−	○	−	◎	◎
定義情報	SFTP アダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/SFTP/config/*	○	−	○	○	○
	SFTP アダプタの共通定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/SFTP/config/common/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない

- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-33　HCSC サーバ（SFTP アダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
通信トレース	SFTP アダプタのプロトコルトレース	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/SFTPADP/*/*	－	○	－	◎	◎
保守ログ	SFTP アダプタの保守ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/SFTPADP/maintenance/*/*	－	○	－	◎	◎
定義情報	SFTP アダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/SFTP/config/*	○	－	○	○	○
	SFTP アダプタの共通定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/SFTP/config/common/*	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.17　HCSC サーバ（ファイル操作アダプタ）

HCSC サーバ（ファイル操作アダプタ）の情報に関連する収集対象を次の表に示します。

表 A-34　HCSC サーバ（ファイル操作アダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
保守ログ	ファイル操作アダプタの保守ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/ADPFOP/maintenance/*/cscadpfopmnt_*.log	○	－	－	◎	△
例外ログ	ファイル操作アダプタの例外ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/ADPFOP/maintenance/*/cscadpfopexp_*.log	○	－	－	◎	△
定義情報	ファイル操作アダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/config/adpfop/*	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-35　HCSC サーバ（ファイル操作アダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
保守ログ	ファイル操作アダプタの保守ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/ADPFOP/maintenance/*/cscadpfopmnt_*.log	○	－	－	◎	△
例外ログ	ファイル操作アダプタの例外ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/ADPFOP/maintenance/*/cscadpfopexp_*.log	○	－	－	◎	△

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
定義情報	ファイル操作アダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/config/adpfop/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.18　HCSC サーバ（FTP インバウンドアダプタ）

HCSC サーバ（FTP インバウンドアダプタ）の情報に関連する収集対象を次の表に示します。

表 A-36　HCSC サーバ（FTP インバウンドアダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	FTP インバウンドアダプタのメッセージログ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/csc/inbound-adapter/ftp/<リソースアダプタ名称>/*	○	−	−	◎	◎
	FTP インバウンドアダプタの運用コマンドのメッセージログ	<Application Server のインストールディレクトリ>/CSC/inbound-adapter/ftp/logs/<リソースアダプタ名称>/*	○	−	−	○	○
定義情報	FTP インバウンドアダプタの定義ファイル	<Application Server のインストールディレクトリ>/CSC/inbound-adapter/ftp/config/<リソースアダプタ名称>/<サーバ名称>/*.xml	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合

○：収集される
－：収集されない

- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-37　HCSC サーバ（FTP インバウンドアダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	FTP インバウンドアダプタのメッセージログ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/csc/inbound-adapter/ftp/<リソースアダプタ名称>/*	○	－	－	◎	◎
	FTP インバウンドアダプタの運用コマンドのメッセージログ	<Application Server のインストールディレクトリ>/CSC/inbound-adapter/ftp/logs/<リソースアダプタ名称>/*	○	－	－	○	○
定義情報	FTP インバウンドアダプタの定義ファイル	<Application Server のインストールディレクトリ>/CSC/inbound-adapter/ftp/config/<リソースアダプタ名称>/*.xml	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.19　HCSC サーバ（メールアダプタ）

HCSC サーバ（メールアダプタ）の情報に関連する収集対象を次の表に示します。

表 A-38　HCSC サーバ（メールアダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	メールアダプタ運用コマンドメッセージログ	<HCSC メールアダプタ運用コマンドメッセージログ出力先ディレクトリ(mailadp.command.messagelog.filepath)>/*	○	－	－	△	△
保守ログ	メールアダプタの保守ログ	<HCSC メールアダプタの保守ログ出力先ディレクトリ(mailadp.methodtrace.filepath)>/*/*	－	○	－	◎	△
定義情報	メールアダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/config/mail/*	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-39　HCSC サーバ（メールアダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	メールアダプタ運用コマンドメッセージログ	<HCSC メールアダプタ運用コマンドメッセージログ出力先ディレクトリ(mailadp.command.messagelog.filepath)>/*	○	－	－	△	△
保守ログ	メールアダプタの保守ログ	<HCSC メールアダプタの保守ログ出力先ディレクトリ(mailadp.methodtrace.filepath)>/*/*	－	○	－	◎	△
定義情報	メールアダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/config/mail/*	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.20　HCSC サーバ（HTTP アダプタ）

HCSC サーバ（HTTP アダプタ）の情報に関連する収集対象を次の表に示します。

表 A-40　HCSC サーバ（HTTP アダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	HTTP アダプタ運用コマンドメッセージログ	<CSCHTTP アダプタ運用コマンドメッセージログ出力先ディレクトリ(httpadp.command.messagelog.filepath)>/*	○	－	－	△	△
保守ログ	HTTP アダプタの保守ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/ADPHTTP/maintenance/*/cscadphttpmnt_*.log	－	○	－	◎	△
例外ログ	HTTP アダプタの例外ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/ADPHTTP/maintenance/*/cscadphttpexp_*.log	○	－	－	◎	△
定義情報	HTTP アダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/HTTP/config/**	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合

「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-41　HCSC サーバ（HTTP アダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	収集対象	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	HTTP アダプタ運用コマンドメッセージログ	<CSCHTTP アダプタ運用コマンドメッセージログ出力先ディレクトリ(httpadp.command.messagelog.filepath)>/*	○	−	−	△	△
保守ログ	HTTP アダプタの保守ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/ADPHTTP/maintenance/*/cscadphttpmnt_*.log	−	○	−	◎	△
例外ログ	HTTP アダプタの例外ログ	<J2EE サーバログ出力ディレクトリ(ejb.server.log.directory)>/CSCADP/ADPHTTP/maintenance/*/cscadphttpexp_*.log	○	−	−	◎	△
定義情報	HTTP アダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/HTTP/config/**	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.21　HCSC サーバ（コマンドアダプタ）

HCSC サーバ（コマンドアダプタ）の情報に関連する収集対象を次の表に示します。

表 A-42　HCSC サーバ（コマンドアダプタ）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
保守ログ	コマンドアダプタの保守ログ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/ADPCMD/maintenance/*/cscadpcmdmnt_*.log	−	○	−	◎	◎
例外ログ	コマンドアダプタの例外ログ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/ADPCMD/maintenance/*/cscadpcmdexp_*.log	○	−	−	◎	◎
定義情報	コマンドアダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/Command/config/*	○	−	○	○	○
		<Application Server のインストールディレクトリ>/CSC/custom-adapter/Command/config/common/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-43　HCSC サーバ（コマンドアダプタ）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
保守ログ	コマンドアダプタの保守ログ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/CSCADP/ADPCMD/maintenance/*/cscadpcmdmnt_*.log	−	○	−	◎	◎
例外ログ	コマンドアダプタの例外ログ	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<	○	−	−	◎	◎

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
		サーバ名称>/logs/CSCADP/ADPCMD/maintenance/*/cscadpcmdexp_*.log					
定義情報	コマンドアダプタの定義情報	<Application Server のインストールディレクトリ>/CSC/custom-adapter/Command/config/*	○	−	○	○	○
		<Application Server のインストールディレクトリ>/CSC/custom-adapter/Command/config/common/*	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.22　HCSC サーバ（ファイルイベント受付）

HCSC サーバ（ファイルイベント受付）の情報に関連する収集対象を次の表に示します。

表 A-44　HCSC サーバ（ファイルイベント受付）の情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	ファイルイベントトレース	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/csc/rcp/fileevent/*/fileeventtrace_*.log	−	○	−	◎	◎
定義情報	ファイルイベント受付の定義情報	<Application Server のインストールディレクトリ>/CSC/custom-reception/fileevent/config/*	○	−	○	○	○
		<Application Server のインストールディレクトリ>/CSC/custom-	○	−	○	○	○

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
		reception/fileevent/config/common/*					

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-45　HCSC サーバ（ファイルイベント受付）の情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	ファイルイベントトレース	<J2EE サーバ作業ディレクトリ(ejb.public.directory)>/ejb/<サーバ名称>/logs/csc/rcp/fileevent/*/fileeventtrace_*.log	－	○	－	◎	◎
定義情報	ファイルイベント受付の定義情報	<Application Server のインストールディレクトリ>/CSC/custom-reception/fileevent/config/*	○	－	○	○	○
		<Application Server のインストールディレクトリ>/CSC/custom-reception/fileevent/config/common/*	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.23　監査ログ

監査ログの情報に関連する収集対象を次の表に示します。

表 A-46　監査ログの情報に関連する収集対象（Windows の場合）

分類	資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
				一次	二次	定義	A	B
なし	メッセージログ	監査ログのメッセージログ	<監査ログメッセージ出力ディレクトリ(auditlog.raslog.message.directory)>/rasmessage*.log	○	−	−	△	△
	そのほかのログ	監査ログ	<監査ログ出力ディレクトリ(auditlog.directory)>/*	−	○	−	△	△
		監査ログの例外情報	<監査ログ例外出力ディレクトリ(auditlog.raslog.exception.directory)>/rasexception*.log	○	−	−	△	△
	保守ログ	そのほかのログ	<監査ログメッセージ出力ディレクトリ(auditlog.raslog.message.directory)>/*/*	○	−	−	△	△
			<監査ログ例外出力ディレクトリ(auditlog.raslog.exception.directory)>/*/*	○	−	−	△	△
	定義情報	監査ログ定義ファイル	<Application Server のインストールディレクトリ>/common/conf/auditlog.properties	○	−	○	△	△
ネーミングサービス	そのほか	ネーミングサービスの起動，停止または異常終了を示すログ	<Windows のイベントログ(EventLog)>	○	−	−	×	×
	ダンプ	スレッドダンプ	<Application Server のインストールディレクトリ>/TPB/logj/javacore*	○	−	−	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-47　監査ログの情報に関連する収集対象（UNIX の場合）

分類	資料の種類	収集ファイル	収集対象	資料			収集方法	
				一次	二次	定義	A	B
なし	メッセージログ	監査ログのメッセージログ	<監査ログメッセージ出力ディレクトリ(auditlog.raslog.message.directory)>/rasmessage*.log	○	−	−	△	△
	そのほかのログ	監査ログ	<監査ログ出力ディレクトリ(auditlog.directory)>/*	−	○	−	△	△
		監査ログの例外情報	<監査ログ例外出力ディレクトリ(auditlog.raslog.exception.directory)>/rasexception*.log	○	−	−	△	△
	保守ログ	そのほかのログ	<監査ログメッセージ出力ディレクトリ(auditlog.raslog.message.directory)>/*/*	○	−	−	△	△
			<監査ログ例外出力ディレクトリ(auditlog.raslog.exception.directory)>/*/*	○	−	−	△	△
	定義情報	監査ログ定義ファイル	<Application Server のインストールディレクトリ>/common/conf/auditlog.properties	○	−	○	△	△
ネーミングサービス	そのほか	ネーミングサービスの起動，停止または異常終了を示すログ	<UNIX の syslog(syslog)>	○	−	−	△	△
	ダンプ	スレッドダンプ	<Application Server のインストールディレクトリ>/TPB/logj/javacore*	○	−	−	○	○
		core ダンプ	<Application Server のインストールディレクトリ>/TPB/logj/core*	−	○	−	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される

－：収集されない

- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 A.24　そのほかの情報

そのほかの情報に関連する収集対象を次の表に示します。

表 A-48　そのほかの情報に関連する収集対象（Windows の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	cosmienv コマンドのログ	<Application Server のインストールディレクトリ>/env/log/*	○	－	－	○	○
定義情報	ホスト定義ファイル(Windows 用)	<システムルートディレクトリ(systemroot)>/system32/drivers/etc/hosts	○	－	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  －：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

表 A-49　そのほかの情報に関連する収集対象（UNIX の場合）

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
メッセージログ	cosmienv コマンドのログ	<Application Server のインストールディレクトリ>/env/log/*	○	－	－	○	○
定義情報	ホスト定義ファイル(UNIX 用)	/etc/hosts	○	－	○	○	○

資料の種類	収集ファイル	デフォルトの収集先	資料			収集方法	
			一次	二次	定義	A	B
そのほか	プログラムプロダクト情報	/etc/.hitachi/pplistd/pplistd	○	−	○	○	○

（凡例）一次：一次送付資料　二次：二次送付資料　定義：定義送付資料　A：mngsvrutil collect snapshot コマンド　B：snapshotlog コマンド

- 資料の場合
  ○：収集される
  −：収集されない
- 収集方法の場合
  「付録 A.1(3)　snapshot ログの収集可否や収集に関する設定変更」を参照してください。

注　収集方法 A および収集方法 B については，「付録 A.1(2)　snapshot ログの収集方法」を参照してください。表内の収集対象のディレクトリおよびファイルの意味については，「付録 A.1(4)　収集対象の記述規則」を参照してください。

付録 B　データベースと接続中にトラブルが発生したコネクションの特定

データベース関連のトラブルに対処するためには，どのコネクションを使用してデータベースと接続しているかを特定することが重要です。

ここでは，データベース（HiRDB または Oracle）と接続中にトラブルが発生した場合に，アプリケーションサーバから出力される情報（性能解析トレースやログなど）とデータベースから出力される情報（トレース情報や pdls コマンドの実行結果，動的パフォーマンスビューの内容など）を使用して，トラブルが発生したコネクションを特定する方法について説明します。

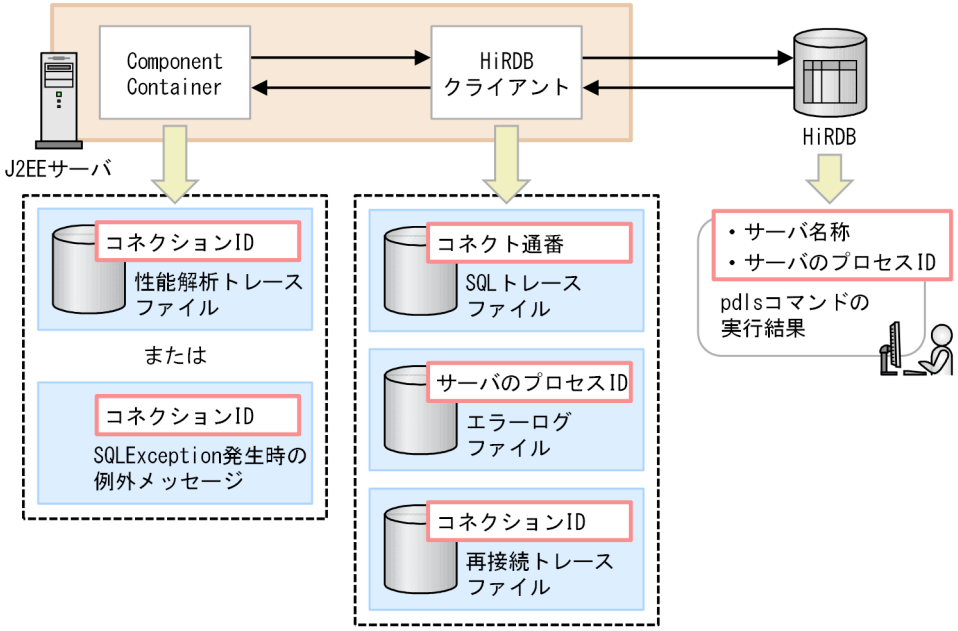
アプリケーションサーバでは，HiRDB および Oracle との接続に使用しているコネクションを一意に識別するための接続情報（**コネクション ID**）を性能解析トレースに出力しています。J2EE サーバからデータベースに至るまで，およびデータベースから J2EE サーバに処理結果が返却されるまでの一連の処理では，関連する構成ソフトウェアのログやトレース情報などにコネクション ID やデータベースサーバで割り振られるコネクト通番などが出力されます。これらの情報を突き合わせて確認することで，トラブルが発生したコネクションを特定できます。

HiRDB を使用する場合，および Oracle を使用する場合のコネクション ID の出力の概要，およびトラブルが発生したコネクションの特定に使用する情報を次に示します。

- **HiRDB を使用する場合**

  コネクション ID の出力の概要を次に示します。

  図 B-1　コネクション ID の出力の概要（HiRDB の場合）



  トラブルが発生したコネクションの特定に使用する情報を次の表に示します。

表 B-1　トラブルが発生したコネクションの特定に使用する情報（HiRDB の場合）

項番	出力元	情報の種類	参照先
1	Component Container	性能解析トレースファイル	付録 B.1
2		SQLException 発生時の例外メッセージ	
3	HiRDB クライアント	SQL トレースファイル	付録 B.2
4		エラーログ情報	
5		再接続トレース	
6	HiRDB サーバ	pdls コマンドの実行結果	付録 B.3

コネクション ID には次に示す情報が含まれています。

- **サーバ名称**

  フロントエンドサーバ名（HiRDB/Parallel Server 使用時）またはシングルサーバ名（HiRDB/Single Server 使用時）が表示されます。

- **コネクト通番**

  サーバ名称に表示された HiRDB サーバで割り振られるコネクト通番が表示されます。

- **サーバのプロセス ID**

  サーバ名称に表示された HiRDB サーバのプロセス ID が表示されます。

コネクション ID の出力形式と出力例を次に示します。

コネクション ID の出力形式（HiRDB の場合）

```
サーバ名称：コネクト通番：サーバのプロセスID
```

コネクション ID の出力例（HiRDB の場合）

```
fes01:15:2351
```

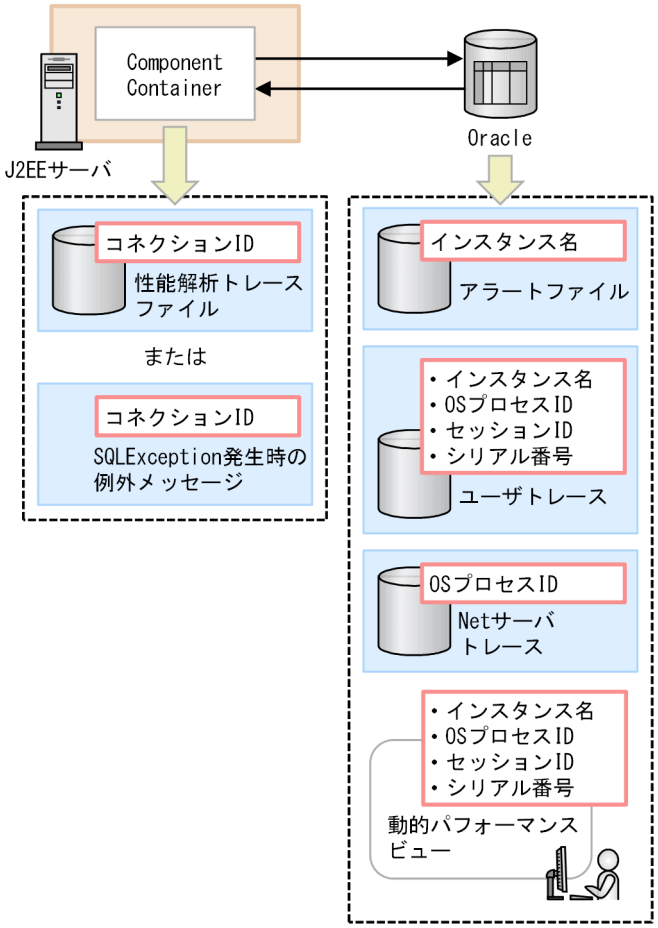
参考

グローバルトランザクションの処理中に，エラーが発生してコネクションが切断されると自動的に再接続されます。しかし，この再接続処理は再接続トレースには出力されません。この場合，再接続トレースに出力されるコネクション ID と実際の接続情報に不整合が生じます。ただし，基本的にグローバルトランザクションの処理中にコネクションが切断されることはありません。

- Oracle を使用する場合

  コネクション ID の出力の概要を次に示します。

図 B-2　コネクション ID の出力の概要（Oracle の場合）



トラブルが発生したコネクションの特定に使用する情報を次の表に示します。

表 B-2　トラブルが発生したコネクションの特定に使用する情報（Oracle の場合）

項番	出力元	情報の種類	参照先
1	Component Container	性能解析トレースファイル	付録 B.1
2		SQLException 発生時の例外メッセージ	
3	Oracle サーバ	アラートファイル	付録 B.4
4		ユーザトレース	
5		Net サーバトレース	
6		動的パフォーマンスビュー	

コネクション ID には次に示す情報が含まれています。

- **インスタンス名**

  Oracle サーバのインスタンス名が表示されます。

- **セッション ID**

  Oracle サーバで割り振られるセッション ID が表示されます。

- **セッションシリアル番号**

Oracle サーバで割り振られるセッションのシリアル番号が表示されます。

- OS プロセス ID

  OS のプロセス ID が表示されます。

コネクション ID の出力形式（Oracle の場合）

```
インスタンス名：セッションID：セッションシリアル番号：OSプロセス名
```

コネクション ID の出力例（Oracle の場合）

```
ORCL:17:5:920
```

コネクション ID の生成には Oracle の動的パフォーマンスビューを使用するため，Oracle に接続するユーザが動的パフォーマンスビューの参照権を持っている必要があります。次のどちらかの方法で Oracle に接続するユーザに参照権を設定してください。

- 「GRANT SELECT_CATALOG_ROLE TO <ユーザ名>;」を実行する。
- 「GRANT SELECT ON V_$INSTANCE TO <ユーザ名>;」，「GRANT SELECT ON V_$PROCESS TO <ユーザ名>;」，および「GRANT SELECT ON V_$SESSION TO <ユーザ名>;」を実行する。

また，Oracle を使用する場合，DB Connector のプロパティ定義で，プロパティ項目「ConnectionIDUpdate」の値に「true」を設定すると，コネクションを取得するたびにコネクション ID を生成することができます。ただし，コネクションを取得するたびにコネクション ID 生成用の SQL が発行されるため，性能に影響を与える可能性があります。再接続が行われない環境では「false」を設定してください。DB Connector のプロパティ定義の設定方法については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「4.2.2　DB Connector のプロパティ定義」を参照してください。

注意事項

Oracle の透過的アプリケーションフェイルオーバーを使用している場合，PRF トレースに出力されるコネクション ID と実際に接続しているコネクション ID が異なることがあります。これは Oracle 内でコネクションが再接続されているためです。PRF トレースに出力されるコネクション ID と実際に接続しているコネクション ID が異なる場合，コネクション ID で Oracle のトレース情報を追跡することはできません。

付録 B.1　Component Container

トランザクション処理の状態（正常または異常）によって，Component Container から出力されるコネクション ID の出力先が異なります。

正常の場合

性能解析トレースファイルに出力されます。

異常の場合

SQLException 発生時の例外メッセージに出力されます。

性能解析トレースファイルには，次の三つのタイミングでコネクション ID が出力されます。

- データベースとのコネクション取得時

  HiRDB を使用している場合，DataSource.getConnection()メソッドまたは DataSource.getConnection(String username, String password)メソッドの終了直前に，取得したコネクションのコネクション ID を出力します。

  Oracle を使用している場合，DataSource.getConnection()メソッドの終了直前に，取得したコネクションのコネクション ID を出力します。

- データベースのコネクション解放時

  Connection.close()メソッドの開始直後に，このメソッドに対応する getConnection メソッドで取得したコネクションのコネクション ID を出力します。

- アソシエーション機能によってデータベースとのコネクションが差し替えられた時

  ManagedConnection.associateConnection()メソッドの呼び出し時に，アソシエーション機能によって差し替えられた先のコネクションのコネクション ID を出力します。

上記のタイミングで取得したコネクション ID は，次の表に示すイベント ID のトレース情報のインタフェース名に出力されています。

表 B-3　コネクション ID が出力される性能解析トレース

イベント ID	説明
0x8C01	DataSource.getConnection()メソッドの終了直前の処理で出力された情報です。
0x8C03	DataSource.getConnection(String username, String password)メソッドの終了直前の処理で出力された情報です。
0x8C20	Connection.close()メソッドの開始直後の処理で出力された情報です。
0x8C40	ManagedConnection.associateConnection()メソッドの呼び出し時の処理で出力された情報です。

なお，トレース取得ポイントと取得できるトレース情報の詳細については，「8.12　DB Connector，JCA コンテナのトレース取得ポイント」を参照してください。

性能解析トレース，および SQLException 発生時の例外メッセージの取得方法と出力形式について説明します。

(1)　性能解析トレースファイル

PRF トレースファイルに出力された，クライアントからデータベースなどの EIS に至るまで，およびその処理結果がクライアントに返却されるまでのリクエストの一連の処理で出力されるトレース情報を CSV 形式で編集出力したファイルです。

コネクション ID が出力される条件

次の条件をすべて満たしている場合，性能解析トレースファイルにコネクション ID が出力されます。

- 使用しているデータベースが，次のどれかである。

  HiRDB

  Oracle

- 論理パフォーマンストレーサが起動中である。
- リソースアダプタとして DB Connector を使用している。

(a) 留意事項

性能解析トレースを参照するときに留意することを次に示します。

- コネクションの解放に関する留意事項

  同一コネクションに対して Connection.close()メソッドを複数回呼び出した場合，その数だけ性能解析トレースが出力されます。

- アソシエーション機能を使用する場合の留意事項

  アソシエーション機能によってデータベースとのコネクションが差し替えられた場合，getConnection()メソッドで出力されたコネクション ID とは異なる物理コネクションでデータベースに接続されます。この場合，性能解析トレースには，ManagedConnection.associateConnection()メソッドの呼び出し時に，アソシエーション機能によって差し替えられた先の物理コネクションに対応するコネクション ID が出力されます。したがって，実際にデータベースに接続しているコネクション ID を特定するためには，イベント ID が「0x8C40」の性能解析トレースに出力されたコネクション ID も追跡する必要があります。

  なお，ManagedConnection.associateConnection()メソッドを呼び出しても，コネクションが差し替えられない場合があります。この場合，コネクション ID は性能解析トレースに出力されません。

  > **参考**
  >
  > アソシエーション機能は，通常，1:1 で対応する論理コネクションと物理コネクションの対応を差し替えて，複数の論理コネクションで一つの物理コネクションを共有する機能です。

- 自動再接続機能を使用する場合の留意事項

  HiRDB の自動再接続機能を使用する場合，性能解析トレースファイルに出力されるコネクション ID と，実際の接続に使用しているコネクションのコネクション ID が異なることがあります。この場合，HiRDB クライアントの再接続トレースも参照する必要があります。

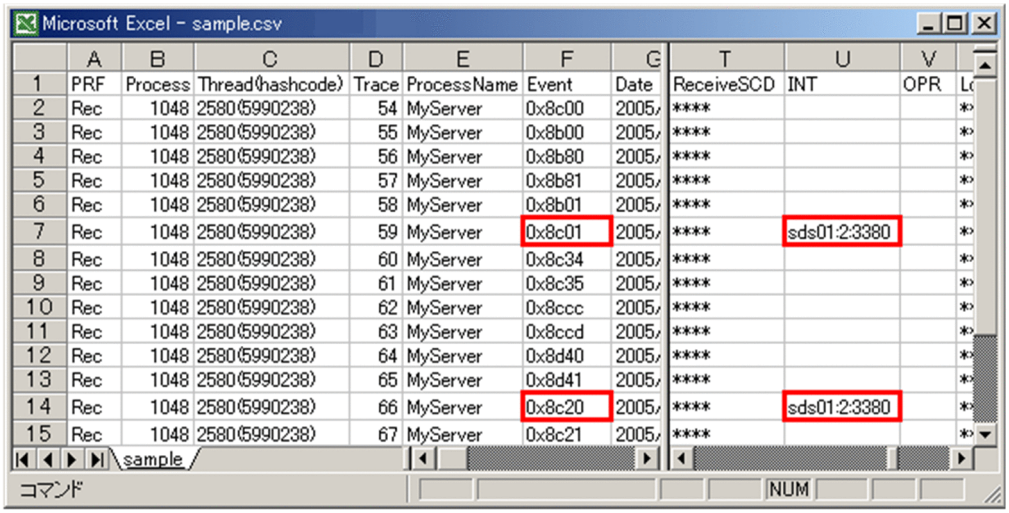
(b) 取得方法

運用管理コマンド（mngsvrutil）を実行して性能解析トレースファイルを取得します。性能解析トレースファイルは，「<Manager のログ出力ディレクトリ>¥prf」（Windows の場合），または「<Manager のログ出力ディレクトリ>/prf」（UNIX の場合）に出力されます。取得方法については，「7.3.1 性能解析トレースファイルの収集方法」を参照してください。

(c)　出力形式

性能解析トレースの出力例を次に示します。コネクション ID はインタフェース名（「INT」欄）に出力されます。

図 B-3　性能解析トレースの出力例



	A	B	C	D	E	F	G	T	U	V
1	PRF	Process	Thread(hashcode)	Trace	ProcessName	Event	Date	ReceiveSCD	INT	OPR
2	Rec	1048	2580(5990238)	54	MyServer	0x8c00	2005/	****		
3	Rec	1048	2580(5990238)	55	MyServer	0x8b00	2005/	****		
4	Rec	1048	2580(5990238)	56	MyServer	0x8b80	2005/	****		
5	Rec	1048	2580(5990238)	57	MyServer	0x8b81	2005/	****		
6	Rec	1048	2580(5990238)	58	MyServer	0x8b01	2005/	****		
7	Rec	1048	2580(5990238)	59	MyServer	0x8c01	2005/	****	sds01:2:3380	
8	Rec	1048	2580(5990238)	60	MyServer	0x8c34	2005/	****		
9	Rec	1048	2580(5990238)	61	MyServer	0x8c35	2005/	****		
10	Rec	1048	2580(5990238)	62	MyServer	0x8ccc	2005/	****		
11	Rec	1048	2580(5990238)	63	MyServer	0x8ccd	2005/	****		
12	Rec	1048	2580(5990238)	64	MyServer	0x8d40	2005/	****		
13	Rec	1048	2580(5990238)	65	MyServer	0x8d41	2005/	****		
14	Rec	1048	2580(5990238)	66	MyServer	0x8c20	2005/	****	sds01:2:3380	
15	Rec	1048	2580(5990238)	67	MyServer	0x8c21	2005/	****		

(2)　SQLException 発生時の例外メッセージ

データベースアクセスまたは JDBC ドライバでエラーが発生した場合に，例外として SQLException がスローされたことを示すメッセージです。

コネクション ID が出力される条件

使用しているデータベースが次のどれかである場合，SQLException 発生時の例外メッセージにコネクション ID が出力されます。

- HiRDB
- Oracle

(a)　留意事項

SQLException 発生時の例外メッセージを参照するときに留意することを次に示します。

- 出力されるコネクション ID に関する留意事項

  SQLException 発生時の例外メッセージに出力されるコネクション ID は常に最新です。HiRDB の自動再接続機能によって再接続が実行された場合，再接続後のコネクション ID が出力されます。

(b)　取得方法

SQLException 発生時に，次のログファイルに例外メッセージが出力されます。

- Windows の場合
  - <作業ディレクトリ>¥ejb¥<サーバ名称>¥logs¥cjexception[n].log

- <作業ディレクトリ>¥ejb¥<サーバ名称>¥logs¥connectors¥<リソースアダプタの表示名>[n].log

- UNIX の場合
  - <作業ディレクトリ>/ejb/<サーバ名称>/logs/cjexception[n].log
  - <作業ディレクトリ>/ejb/<サーバ名称>/logs/connectors/<リソースアダプタの表示名>[n].log

ログファイル名の[n]の部分には，面の番号（1 から面数（最大 16）まで）が付きます。

(c) 出力形式

HiRDB の場合

メッセージ KFDJ00001-E の ErrMsg の最後にコネクション ID が出力されます。出力例を次に示します。背景色付きの**太字**はコネクション ID です。

```
JP.co.Hitachi.soft.DBPSV_Driver.SQLException: KFDJ00001-E Error occurred at server.
[JdbcDbpsvResultSQLExecute.SQLExecute]
OperationType : 2002
ReturnCode    : -100
ErrorCode     : -404
WarningInfo   : 0
ErrorMsg      : KFPA11404-E Input data too long for column or assignment target in variable
 1 [HiRDB_CONNECTION_ID(sds01:7:2988)]
```

Oracle の場合

メッセージ ORA-00942 の ErrMsg の最後にコネクション ID が出力されます。出力例を次に示します。背景色付きの**太字**はコネクション ID です。

```
JP.co.Hitachi.soft.DBPSV_Driver.SQLException: KFDJ00001-E Error occurred at server.
[JdbcDbpsvResultSQLExecute.SQLPrepare]
OperationType : 2002
ReturnCode    : -200
ErrorCode     : 942
WarningInfo   : 0
ErrorMsg      : ORA-00942: 表またはビューが存在しません。[ORACLE_CONNECTION_ID(ORCL:17:5:
920)]
PreparedSQL   : selectSectionID     : 2
```

付録 B.2 HiRDB クライアント

HiRDB クライアントでは，SQL トレースファイル，エラーログファイル，および再接続トレースファイルに，コネクション ID やコネクト通番などの情報が出力されます。

SQL トレースファイル，エラーログファイル，および再接続トレースファイルの取得方法と出力形式について説明します。

(1) SQL トレースファイル

実行した UAP の SQL トレース情報を出力したトレースファイルです。SQL の実行終了時に出力されます。

UAP 実行時に SQL エラーが発生した場合，SQL トレースファイルを参照すると，エラーの原因となる SQL を特定できます。

コネクション ID が出力される条件

次の条件を満たしている場合，SQL トレースファイルにコネクション ID が出力されます。

- クライアント環境定義の PDCLTPATH および PDSQLTRACE に値を指定している。

なお，クライアント環境定義の設定方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(a) 取得方法

クライアント環境定義の PDCLTPATH に指定したディレクトリに格納されます。

(b) 出力形式

次に示す形式で，サーバ名称，コネクト通番，およびサーバのプロセス ID が出力されます。

```
（省略）
   :
CONNECTION STATUS：
  CURHOST（接続先ホスト名）  CURPORT（接続ポート番号）  SRVNAME（サーバ名称）
  CNCTNO（コネクト通番）  SVRPID（サーバのプロセスID）  CLTPID（UAPのプロセスID）  CLTTID（U
APのスレッド番号）
   :
（省略）
```

SQL トレースファイルには，SQL 文，SQL 実行時刻や SQL 文中の変数に設定した値なども出力されます。出力形式の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(2) エラーログファイル

クライアントと HiRDB サーバ間の通信処理中，または X/Open で規定した XA インタフェースでエラーが発生した場合のエラー情報を出力したログファイルです。SQL 実行時，通信処理時，または X/Open で規定した XA インタフェース関数実行時でエラーが発生したときに出力されます。

コネクション ID が出力される条件

次の条件を満たしている場合，エラーログファイルにコネクション ID が出力されます。

- クライアント環境定義の PDCLTPATH および PDUAPERLOG に値を指定している。

なお，設定方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(a) 取得方法

クライアント環境定義の PDCLTPATH に指定したディレクトリに格納されます。

(b) 出力形式

次に示す形式で，SQL トレースにサーバのプロセス ID が出力されます。

```
エラーログ先頭識別子（'>>'または'>'）　UAPのプロセスID　UAPのスレッド番号　サーバのプロセスID　エラーログカウンタ　（以降省略）
```

出力形式の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

エラーログ情報には，接続情報のほか，エラー取得時刻，SQLCODE やエラーが発生した SQL のオペレーションコードなどが出力されます。

(3) 再接続トレースファイル

HiRDB クライアントの自動再接続機能で再接続が実行された場合，HiRDB が内部的に管理している接続ハンドルの値，再接続前の接続情報，再接続後の接続情報，および再接続時刻を出力したトレースファイルです。自動再接続機能で自動的に接続が実行された場合に出力されます。

コネクション ID が出力される条件

次の条件を満たしている場合，再接続トレースファイルにコネクション ID が出力されます。

- クライアント環境定義の PDCLTPATH および PDSQLTRACE に値を指定している。
- HiRDB の自動再接続機能を使用している。

なお，設定方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(a) 取得方法

クライアント環境定義 PDCLTPATH に指定したディレクトリに再接続トレースファイルが格納されます。格納されるファイル名称は，pdrcnct1.trc および pdrcnct2.trc です。

(b) 出力形式

再接続トレースは次の形式で出力されます。

```
接続ハンドルの値　再接続結果（SまたはF）　再接続開始日時 - 再接続完了日時　再接続前の接続情報 => 再接続後の接続情報
```

コネクション ID は再接続後の接続情報として出力されます。

出力形式の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

再接続トレースファイルの出力例を次に示します。背景色付きの**太字**はコネクション ID です。

```
40004250 S 2004/04/12 11:10:36.766 - 2004/04/12 11:10:41.846 sds:9:23763 => sds:10:23750
40004250 S 2004/04/12 11:11:07.491 - 2004/04/12 11:11:12.547 sds:10:23750 => sds:11:23765
40004850 F 2004/04/12 11:17:58.285 - 2004/04/12 11:18:23.395 sds:14:23751 =>
40005050 S 2004/04/12 11:27:35.098 - 2004/04/12 11:27:40.152 sds:1:24414 => sds:2:24418
```

付録 B.3　HiRDB サーバ

HiRDB サーバでは，pdls コマンドの実行結果を参照してサーバの状態を確認します。

pdls コマンドの実行結果に表示されるサーバ名称やサーバのプロセス ID と，コネクション ID に含まれる情報を突き合わせて，接続した HiRDB サーバの状態を確認します。

pdls コマンドの実行形式と，実行結果の出力形式について説明します。

(1)　pdls コマンドの実行結果

HiRDB サーバの排他制御の状態，プロセスの状態，通信制御情報などを表示できるコマンドです。

(a)　実行形式

次に示す pdls コマンドを実行して，サーバの状態を確認します。

- サーバの排他制御の状態表示する場合

```
pdls -d lck
```

- サーバのプロセスの状態表示する場合

```
pdls -d prc
```

- サーバの通信制御情報の表示する場合

```
pdls -d rpc
```

pdls コマンドの詳細については，マニュアル「HiRDB コマンドリファレンス」を参照してください。

(b)　出力形式

実行結果の「SVID」欄にサーバ名称が出力されて，「PID」欄にサーバのプロセス ID が出力されます。

出力形式の詳細については，マニュアル「HiRDB コマンドリファレンス」を参照してください。

付録 B.4　Oracle サーバ

Oracle サーバでは，次の方法で Oracle サーバのエラー情報，サーバの状態などの情報が出力されます。

- アラートファイルの取得
- ユーザトレースの取得
- Net サーバトレースファイルの取得
- 動的パフォーマンスビューの使用

(1) アラートファイル

アラートファイルでは，性能解析トレースのインスタンス名をキーにして，次の情報を確認できます。

- 発生したすべての内部エラー，ブロック破損エラー，およびデッドロックエラー
- CREATE/ALTER/DROP 文，STARTUP/SHUTDOWN 文，ARCHIVELOG 文などの管理，操作
- 共有サーバとディスパッチャプロセスの機能に関するメッセージ，およびエラー
- マテリアライズドビューの自動リフレッシュ中に発生したエラー
- データベースおよびインスタンス起動時の，すべての初期化パラメタの値

アラートファイルは，次の場所に出力されます。

- Windows の場合
  <BACKGROUND_DUMP_DEST で指定した場所※>¥ALERT_<インスタンス名>.LOG
- UNIX の場合
  <BACKGROUND_DUMP_DEST で指定した場所※>/ALERT_<インスタンス名>.LOG

注※　Oracle の初期化パラメタ USER_DUMP_DEST で設定します。USER_DUMP_DEST の詳細については，Oracle のドキュメントを参照してください。

出力されたアラートファイルの中から，性能解析トレースに出力されたインスタンス名とアラートファイル名に含まれるインスタンス名を突き合わせて，目的のアラートファイルを特定してください。

(2) ユーザトレース

ユーザトレースでは，性能解析トレースのインスタンス名，OS プロセス ID，セッション ID，セッションシリアル番号をキーにして，次の情報を確認できます。

- サーバプロセスで発生したエラー情報
- SQL 分の実行計画と統計情報

(3) Net サーバトレースファイル

Oracle サーバでは，Net サーバトレースファイルに，実行されたネットワークイベントの詳細が出力されます。Net サーバトレースファイルのファイル名にはサーバの OS プロセス ID が含まれています。

Net サーバトレースファイルは次の場所に出力されます。

- Windows の場合
  <sqlnet.ora ファイルで指定した場所※>¥CLI_<OS プロセス ID>.TRC
- UNIX の場合
  <sqlnet.ora ファイルで指定した場所※>/CLI_<OS プロセス ID>.TRC

注※　Net サーバトレースファイルの出力先は，Oracle の sqlnet.ora ファイルで設定します。sqlnet.ora ファイルの記述形式を次に示します。なお，sqlnet.ora ファイルの詳細については，Oracle のドキュメントを参照してください。

```
TRACE_LEVEL_CLIENT=16
TRACE_DIRECTORY_CLIENT=<ディレクトリ名>
TRACE_UNIQUE_CLIENT=ON
TRACE_TIMESTAMP_CLIENT=ON
```

出力された Net サーバトレースファイルの中から，性能解析トレースの OS プロセス ID と Net サーバトレースファイル名に含まれる OS プロセス ID 名を突き合わせて，目的の Net サーバトレースファイルを特定してください。

なお，Net サーバトレースファイルの出力内容の詳細については，Oracle のドキュメントを参照してください。

注意事項

Net サーバトレースファイルは，大量のディスク領域を消費するため，システムの性能が低下する場合があります。必要なときにだけ Net クライアントトレースを参照してください。

(4)　動的パフォーマンスビュー

動的パフォーマンスビューを調査して，セッションを特定したり，プロセスの詳細についての情報を参照したりできます。動的パフォーマンスビューは，性能解析トレースファイルに出力されるコネクション ID と突き合わせて調査します。

動的パフォーマンスビューと性能解析トレースのコネクション ID の関係を次の表に示します。

表 B-4　動的パフォーマンスビューと性能解析トレースのコネクション ID の関係

項番	動的パフォーマンスビューに表示される項目	性能解析トレースのコネクション ID の項目
1	V$INSTANCE　INSTANCE_NAME	インスタンス名
2	V$SESSION　SID	セッション ID
3	V$SESSION　SERIAL#	セッションシリアル番号
4	V$PROCESS　SPID	OS プロセス ID

動的パフォーマンスビューの詳細については，Oracle のドキュメントを参照してください。

付録 C　障害発生時の CMR 用の表の回復

ここでは，CMR を含むアプリケーションをデプロイしたあと J2EE サーバで障害が発生した場合に，障害発生前に使用していた CMR 用の表を回復させる手順について説明します。なお，CMR 用の表の詳細については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「9.6.3　CMP2.x とデータベースのマッピング」を参照してください。

CMR を含むアプリケーションがデプロイされ，運用していた状態で保守などのために J2EE サーバを停止，起動したときに障害が発生すると，CMR を含むアプリケーションがデプロイされた状態で立ち上がらないことが考えられます。障害を解決し，再度 CMR を含むアプリケーションをデプロイしようとしても，作成する表と同名の表がデータベース上にある場合，デプロイができません（アプリケーション間で表の共有を避けるため）。

ここで，SQL 生成を再度実行すると，新しい CMR 用の表名を使用した SQL が生成され，新しい CMR 用の表を使用するようにデプロイができます。しかし，J2EE サーバを停止する以前に使用していた関係を使用したい場合，データベースに残っている表を使うようにデプロイする必要があります。

簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に ejbserver.ejb.cmp20.cmr.use.existing_table パラメタで指定します。
ejbserver.ejb.cmp20.cmr.use.existing_table は，アプリケーションの起動障害発生時，それまで使用していた関係の情報を回復させるためのオプションです。true を指定すると，データベース既存の表を使用するようにデプロイができます。このオプションを使用して，既存の表を使用する場合の手順を次に示します。

1. 障害発生前に使用していた CMR 用の表がデータベース上に残っていることを，SQL を使用して確認してください。
2. J2EE サーバを停止します（アプリケーションがデプロイ状態で立ち上がることに失敗した原因の対処をしてください）。
3. 簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に ejbserver.ejb.cmp20.cmr.use.existing_table パラメタで次のように指定します。

   <param-name>タグ

   　ejbserver.ejb.cmp20.cmr.use.existing_table

   <param-value>タグ

   　true
4. J2EE サーバを起動します。
5. デプロイ状態での開始に失敗した CMR を含むアプリケーションを再度デプロイします。

注意事項

ここで SQL 生成を再実行しないでください。再実行すると新しい表名で SQL が生成され，以前の表が使用できなくなります。

6. J2EE サーバを停止します。

7. 簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に ejbserver.ejb.cmp20.cmr.use.existing_table パラメタで false を指定するか，またはこのオプションを設定しない状態に戻します。

8. 再度 J2EE サーバを起動します。

付録 D　各バージョンでの主な機能変更

ここでは，11-40 よりも前のアプリケーションサーバの各バージョンでの主な機能の変更について，変更目的ごとに説明します。11-40 での主な機能変更については，「1.4　アプリケーションサーバ 11-40 での主な機能変更」を参照してください。

説明内容は次のとおりです。

- アプリケーションサーバの各バージョンで変更になった主な機能と，その概要を説明しています。機能の詳細については「参照先マニュアル」の「参照個所」の記述を確認してください。「参照先マニュアル」および「参照個所」には，その機能についての 11-40 のマニュアルでの主な記載個所を記載しています。
- 「参照先マニュアル」に示したマニュアル名の「アプリケーションサーバ」は省略しています。

付録 D.1　11-30 での主な機能変更

(1)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D–1　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JDBC4.3 への対応	JDBC4.3 に対応しました。	アプリケーションサーバ & BPM/ESB 基盤 概説	4.6.2
接続できるデータベースに MySQL，PostgreSQL を追加	DB Connector を使用して接続できるデータベースに，MySQL，PostgreSQL を追加しました。	機能解説 基本・開発編 (コンテナ共通機能)	3.6
パッケージ名変換機能の追加	jakarta パッケージ名を前提とするアプリケーションをアプリケーションサーバで動作させる機能を追加しました。	機能解説 基本・開発編 (コンテナ共通機能)	20 章

付録 D.2　11-20 での主な機能変更

(1)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D–2　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JSF 2.3 への対応	JSF 2.3 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	3 章
JAX-RS 2.1 への対応	JAX-RS 2.1 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	4 章
WebSocket 1.1 への対応	WebSocket 1.1 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	5 章
Servlet 4.0 への対応	Servlet 4.0 に対応しました。これに伴い，NIO HTTP サーバで HTTP/2 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	8 章
CDI Managed Bean での JPA 利用	CDI Managed Bean での JPA 利用に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	5 章
JPA 2.2 への対応	JPA 2.2 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	6 章
CDI 2.0 への対応	CDI 2.0 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	9 章
WEB-INF/lib 内の CDI への対応	WAR ファイルの WEB-INF/lib 以下の JAR ファイルから CDI を利用できるようにしました。	機能解説 基本・開発編 (コンテナ共通機能)	9 章
BV 2.0 への対応	BV 2.0 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	10 章
JSON-P 1.1 への対応	JSON-P 1.1 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	12 章
JSON-B 1.0 への対応	JSON-B 1.0 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	13 章

付録 D.3　11-10 での主な機能変更

(1)　導入・構築の容易性強化

導入・構築の容易性強化を目的として変更した項目を次の表に示します。

表 D–3　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
ライブラリ競合回避機能の追加	クラス・リソースをロードするときの検索順序を変更し，ユーザアプリケーションに含まれるライブラリを優先して参照できるようにする機能を追加しました。	機能解説 基本・開発編 (コンテナ共通機能)	付録 B.4

(2)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D–4　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JSP2.3 への対応	JSP2.3 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	8 章
コンテナ管理の EntityManager への対応	JPA 2.1 に対応したコンテナ管理の EntityManager に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	5 章
Interceptors 1.2 への対応	Interceptors 1.2 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	15 章

(3)　信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 D–5　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
セッションマネージャの指定機能の追加	クラウド環境利用時に HTTP セッションのセッションフェイルオーバを利用できるようにする機能を追加しました。	機能解説 基本・開発編 (Web コンテナ)	2.22

付録 D.4　11-00 での主な機能変更

(1)　導入・構築の容易性強化

導入・構築の容易性強化を目的として変更した項目を次の表に示します。

表 D–6　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
開発環境の Windows Server 対応	クラウド環境上にアプリケーション開発環境を構築できるよう，uCosminexus Developer のサポート OS に Windows Server OS を追加しました。	−	−

(2)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D-7　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Servlet 3.0/3.1 への対応	Servlet 3.0 の非同期サーブレット，および Servlet 3.1 の非同期 I/O 系 API に対応しました。	機能解説 基本・開発編 (Web コンテナ)	8.1
EL 3.0 への対応	EL 3.0 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	2.3.3
JSF 2.2 への対応	JSF 2.2 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	3 章
JAX-RS 2.0 への対応	JAX-RS 2.0 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	4 章
WebSocket 1.0 への対応	WebSocket 1.0 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	5 章
NIO HTTP サーバ機能の追加	従来のリダイレクタ機能やインプロセス HTTP サーバ機能に代わり，非同期サーブレットや WebSocket などのノンブロッキング I/O 処理に対応したインプロセスの HTTP サーバとして，NIO HTTP サーバ機能を追加しました。	機能解説 基本・開発編 (Web コンテナ)	7 章
JPA 2.1 への対応	JPA 2.1 に対応し，JPA 2.1 対応の JPA プロバイダを利用できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	6 章
CDI 1.2 への対応	CDI 1.2 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	9 章
BV 1.1 への対応	Bean Validation 1.1 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	10 章
Java Batch 1.0 への対応	Batch Applications for the Java Platform （Java Batch） 1.0 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	11 章
JSON-P 1.0 への対応	Java API for JSON Processing（JSON-P） 1.0 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	12 章
Concurrency Utilities 1.0 への対応	Concurrency Utilities for Java EE 1.0 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	14 章
WebSocket 通信への対応	HTTP Server から J2EE サーバに WebSocket 通信を中継する機能を追加しました。	HTTP Server	4.15

(3)　信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 D-8　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
暗号化通信モジュールの変更	HTTP Server の暗号化通信モジュールとして mod_ssl を採用しました。	HTTP Server	5 章，付録 D

(4)　そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 D–9　そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
V9 互換モードの追加	Version 9 以前の J2EE サーバから移行するユーザ向けに，Version 9 との互換性を維持するための V9 互換モードを追加しました。	このマニュアル	10.3.4

付録 D.5　09-87 での主な機能変更

(1)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D–10　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Java SE 11 への対応	Java SE 11 の機能が使用できるようになりました。	このマニュアル	9 章

付録 D.6　09-80 での主な機能変更

(1)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D–11　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JAX-RS 機能におけるラムダ式の使用	web.xml のサーブレット初期化パラメタに指定したパッケージとそのサブパッケージに含まれるクラスで，ラムダ式が使用できるようになりました。	Web サービス開発ガイド	11.2
Java SE 9 への対応	Java SE 9 の機能が使用できるようになりました。	このマニュアル	9 章

(2)　信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 D–12　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Web サーバの Apache2.4 のサポート	Web サーバのベースバージョンとして Apache2.4 をサポートしました。	HTTP Server	6 章，付録 C
SSL 通信での楕円曲線暗号の利用	楕円曲線暗号を利用した SSL 通信ができるようになりました。	HTTP Server	5 章，付録 C
SSL ライブラリの変更	SSL 機能を提供する SSL ライブラリを OpenSSL に変更しました。	HTTP Server	5 章，付録 C

付録 D.7　09-70 での主な機能変更

(1)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D–13　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
運用管理ポータルでの JSP コンパイルバージョンの追加	J2EE サーバでの JSP から生成されたサーブレットのコンパイル方法に「JDK1.7 の仕様に従ったコンパイル」と「JDK7 の仕様に従ったコンパイル」を追加する。	運用管理ポータル操作ガイド	10.8.4
		リファレンス 定義編 (サーバ定義)	4.11.2
JDK8 でのメタスペース対応	JavaVM の起動で使用している Permanent 領域用のオプションを Metaspace 領域用のオプションに変更する。	システム構築・運用ガイド	付録 A.2
		運用管理ポータル操作ガイド	10.8.7
		リファレンス 定義編 (サーバ定義)	5.2.1，5.2.2，付録 A.2
統合ユーザ管理でのユーザ認証の SHA-2 対応	統合ユーザ管理でのユーザ認証のハッシュアルゴリズムとして SHA-224，SHA-256，SHA-384，SHA-512 を追加する。	機能解説 セキュリティ管理機能編	5.3.1，5.3.9，5.10.7，11.4.3，12.4.3，12.5.3，13.2，14.2.2
Red Hat Enterprise Linux Server 7 での自動起動と自動再起動および自動停止の追加	Red Hat Enterprise Linux Server 7 での Management Server と運用管理エージェントの自動起動と自動再起動および自動停止方法を追加する。	機能解説 運用／監視／連携編	2.6.3，2.6.4，2.6.5

項目	変更の概要	参照先マニュアル	参照個所
		リファレンス コマンド編	7.2

(2) 運用性の維持・向上

運用性の維持・向上を目的として変更した項目を次の表に示します。

表 D–14 運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
V9.7 へのバージョンアップ対応	バージョンアップ時の JavaVM の起動で使用している Permanent 領域用のオプションを Metaspace 領域用のオプションに変更する手順を追加する。	このマニュアル	10.3.1, 10.3.2, 10.3.5
WAR による運用	WAR ファイルだけで構成された WAR アプリケーションを J2EE サーバにデプロイできるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.2.1
		機能解説 基本・開発編 (コンテナ共通機能)	18.9
		リファレンス コマンド編	cjimport war (WAR アプリケーションのインポート)
明示管理ヒープ機能での Explicit メモリブロックの強制解放	javagc コマンドで，Explicit メモリブロックの解放処理を任意のタイミングで実行できるようになりました。	機能解説 拡張編	7.6.1, 7.9
		リファレンス コマンド編	javagc (GC の強制発生)

(3) そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 D–15 そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
snapshot ログの収集対象	snapshot ログの収集対象として JavaVM イベントログと Management Server のスレッドダンプを追加する。	このマニュアル	付録 A.2
cjenvsetup コマンドのログ出力	Component Container 管理者のセットアップ（cjenvsetup コマンド）の実行情報がメッセージログに出力されるようになりました。	システム構築・運用ガイド	4.1.4
		機能解説 保守／移行編	4.20

項目	変更の概要	参照先マニュアル	参照個所
		リファレンス コマンド編	cjenvsetup (Component Container 管理者のセットアップ)
明示管理ヒープ機能のイベントログへの CPU 時間の出力	Explicit メモリブロック解放処理に掛かった CPU 時間が，明示管理ヒープ機能のイベントログに出力されるようになりました。	このマニュアル	5.11.3
ユーザ拡張性能解析トレースの機能拡張	ユーザ拡張性能解析トレースで，次の機能を追加しました。 • トレース対象の指定方法を通常のメソッド単位の指定方法に加えて，パッケージ単位またはクラス単位で指定できるようになりました。 • 使用できるイベント ID の範囲を拡張しました。 • ユーザ拡張性能解析トレース設定ファイルに指定できる行数の制限を緩和しました。 • ユーザ拡張性能解析トレース設定ファイルでトレース取得レベルを指定できるようになりました。	このマニュアル	7.5.2, 7.5.3, 8.25.1
Session Bean の非同期呼び出し使用時の情報解析向上	PRF トレースのルートアプリケーション情報を使用して，呼び出し元と呼び出し先のリクエストを突き合わせることができるようになりました。	機能解説 基本・開発編 (EJB コンテナ)	2.17.3

付録 D.8　09-60 での主な機能変更

(1)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D-16　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
G1GC への対応	G1GC を選択できるようになりました。	システム設計ガイド	7.15
		リファレンス 定義編 (サーバ定義)	14.5
圧縮オブジェクトポインタ機能への対応	圧縮オブジェクトポインタ機能を使用できるようになりました。	このマニュアル	9.18

(2)　信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 D–17　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
ファイナライズ滞留解消機能の追加	ファイナライズ処理の滞留を解消でき，OS 資源の解放遅れなどの発生を抑止できるようになりました。	このマニュアル	9.16

(3)　そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 D–18　そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
ログファイルの非同期出力機能の追加	ログのファイル出力を非同期でできるようになりました。	リファレンス 定義編(サーバ定義)	14.2

付録 D.9　09-50 での主な機能変更

(1)　開発生産性の向上

開発生産性の向上を目的として変更した項目を次の表に示します。

表 D–19　開発生産性の向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Eclipse セットアップの簡略化	GUI を利用して Eclipse 環境をセットアップできるようになりました。	アプリケーション開発ガイド	1.1.5, 2.4
ユーザ拡張性能解析トレースを使ったデバッグ支援	ユーザ拡張性能解析トレース設定ファイルを開発環境で作成できるようになりました。	アプリケーション開発ガイド	1.1.3, 6.4

(2)　導入・構築の容易性強化

導入・構築の容易性強化を目的として変更した項目を次の表に示します。

表 D–20　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
仮想化環境でのシステム構成パターンの拡充	仮想化環境で使用できるティアの種類（http-tier，j2ee-tier および ctm-tier）が増えました。これによって，次のシステム構成パターンが構築できるようになりました。	仮想化システム構築・運用ガイド	1.1.2

項目	変更の概要	参照先マニュアル	参照個所
	• Web サーバと J2EE サーバを別のホストに配置するパターン • フロントエンド（サーブレット，JSP）とバックエンド（EJB）を分けて配置するパターン • CTM を使用するパターン		

(3)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D–21　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JDBC 4.0 仕様への対応	DB Connector で JDBC 4.0 仕様の HiRDB Type4 JDBC Driver，および SQL Server の JDBC ドライバに対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	3.6.3
Portable Global JNDI 名での命名規則の緩和	Portable Global JNDI 名に使用できる文字を追加しました。	機能解説 基本・開発編 (コンテナ共通機能)	2.4.3
Servlet 3.0 仕様への対応	Servlet 3.0 の HTTP Cookie の名称，および URL のパスパラメタ名の変更が，Servlet 2.5 以前のバージョンでも使用できるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.7
Bean Validation と連携できるアプリケーションの適用拡大	CDI やユーザアプリケーションでも Bean Validation を使って検証できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	10 章
JavaMail への対応	JavaMail 1.4 に準拠した API を使用したメール送受信機能を利用できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	8 章
javacore コマンドが使用できる OS の適用拡大	javacore コマンドを使って，Windows のスレッドダンプを取得できるようになりました。	リファレンス コマンド編	javacore (スレッドダンプの取得／Windows の場合)

(4)　信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 D–22　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
コードキャッシュ領域の枯渇回避	システムで使用しているコードキャッシュ領域のサイズを確認して，領域が枯渇する前にしきい値を変更して領域枯渇するのを回避できるようになりました。	システム設計ガイド	7.2.6
		このマニュアル	5.7.2, 5.7.3

項目	変更の概要	参照先マニュアル	参照個所
		リファレンス 定義編(サーバ定義)	14.1, 14.2, 14.4
明示管理ヒープ機能の効率的な適用への対応	自動解放処理時間を短縮し，明示管理ヒープ機能を効率的に適用するための機能として，Explicit ヒープに移動するオブジェクトを制御できる機能を追加しました。 • Explicit メモリブロックへのオブジェクト移動制御機能 • 明示管理ヒープ機能適用除外クラス指定機能 • Explicit ヒープ情報へのオブジェクト解放率情報の出力	システム設計ガイド	7.14.6
		機能解説 拡張編	7.2.2, 7.6.5, 7.10, 7.13.1, 7.13.3
		このマニュアル	5.5
クラス別統計情報の出力範囲拡大	クラス別統計情報を含んだ拡張スレッドダンプに，static フィールドを基点とした参照関係を出力できるようになりました。	このマニュアル	9.6

(5) 運用性の維持・向上

運用性の維持・向上を目的として変更した項目を次の表に示します。

表 D-23 運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
EADs セッションフェイルオーバ機能のサポート	EADs と連携してセッションフェイルオーバ機能を実現する EADs セッションフェイルオーバ機能をサポートしました。	機能解説 拡張編	5 章
WAR による運用	WAR ファイルだけで構成された WAR アプリケーションを J2EE サーバにデプロイできるようになりました。	機能解説 基本・開発編(Web コンテナ)	2.2.1
		機能解説 基本・開発編(コンテナ共通機能)	18.9
		リファレンス コマンド編	cjimportwar(WAR アプリケーションのインポート)
運用管理機能の同期実行による起動と停止	運用管理機能（Management Server および運用管理エージェント）の起動および停止を，同期実行するオプションを追加しました。	機能解説 運用／監視／連携編	2.6.1, 2.6.2, 2.6.3, 2.6.4
		リファレンス コマンド編	adminagentctl(運用管理

項目	変更の概要	参照先マニュアル	参照個所
			エージェントの起動と停止)，mngautorun（自動起動および自動再起動の設定／設定解除)，mngsvrctl (Management Server の起動／停止／セットアップ)
明示管理ヒープ機能での Explicit メモリブロックの強制解放	javagc コマンドで，Explicit メモリブロックの解放処理を任意のタイミングで実行できるようになりました。	機能解説 拡張編	7.6.1，7.9
		リファレンス コマンド編	javagc (GC の強制発生)

(6) そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 D-24 そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
定義情報の取得	snapshotlog（snapshot ログの収集）コマンドで定義ファイルだけを収集できるようになりました。	このマニュアル	2.3
		リファレンス コマンド編	snapshotlog (snapshot ログの収集)
cjenvsetup コマンドのログ出力	Component Container 管理者のセットアップ（cjenvsetup コマンド）の実行情報がメッセージログに出力されるようになりました。	システム構築・運用ガイド	4.1.4
		このマニュアル	4.20
		リファレンス コマンド編	cjenvsetup (Compo

項目	変更の概要	参照先マニュアル	参照個所
			nent Container 管理者のセットアップ)
BIG-IP v11 のサポート	使用できる負荷分散機の種類に BIG-IP v11 が追加になりました。	システム構築・運用ガイド	4.7.2
		仮想化システム構築・運用ガイド	2.1
明示管理ヒープ機能のイベントログへの CPU 時間の出力	Explicit メモリブロック解放処理に掛かった CPU 時間が，明示管理ヒープ機能のイベントログに出力されるようになりました。	このマニュアル	5.11.3
ユーザ拡張性能解析トレースの機能拡張	ユーザ拡張性能解析トレースで，次の機能を追加しました。 • トレース対象の指定方法を通常のメソッド単位の指定方法に加えて，パッケージ単位またはクラス単位で指定できるようになりました。 • 使用できるイベント ID の範囲を拡張しました。 • ユーザ拡張性能解析トレース設定ファイルに指定できる行数の制限を緩和しました。 • ユーザ拡張性能解析トレース設定ファイルでトレース取得レベルを指定できるようになりました。	このマニュアル	7.5.2, 7.5.3, 8.25.1
Session Bean の非同期呼び出し使用時の情報解析向上	PRF トレースのルートアプリケーション情報を使用して，呼び出し元と呼び出し先のリクエストを突き合わせることができるようになりました。	機能解説 基本・開発編 (EJB コンテナ)	2.17.3

付録 D.10　09-00 での主な機能変更

(1)　導入・構築の容易性強化

導入・構築の容易性強化を目的として変更した項目を次の表に示します。

表 D–25　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
仮想化環境での構築・運用の操作対象単位の変更	仮想化環境の構築・運用時の操作対象単位が仮想サーバから仮想サーバグループへ変更になりました。仮想サーバグループの情報を定義したファイルを使用して，複数の仮想サーバを管理ユニットへ一括で登録できるようになりました。	仮想化システム構築・運用ガイド	1.1.2

項目	変更の概要	参照先マニュアル	参照個所
セットアップウィザードによる構築環境の制限解除	セットアップウィザードを使用して構築できる環境の制限が解除されました。ほかの機能で構築した環境があってもアンセットアップされて，セットアップウィザードで構築できるようになりました。	システム構築・運用ガイド	2.2.7
構築環境の削除手順の簡略化	Management Server を使用して構築したシステム環境を削除する機能（mngunsetup コマンド）の追加によって，削除手順を簡略化しました。	システム構築・運用ガイド	4.1.37
		運用管理ポータル操作ガイド	3.6，5.4
		リファレンス コマンド編	mngunsetup（Management Server の構築環境の削除）

(2)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D-26　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Servlet 3.0 への対応	Servlet 3.0 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	8 章
EJB 3.1 への対応	EJB 3.1 に対応しました。	機能解説 基本・開発編 (EJB コンテナ)	2 章
JSF 2.1 への対応	JSF 2.1 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	3 章
JSTL 1.2 への対応	JSTL 1.2 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	3 章
CDI 1.0 への対応	CDI 1.0 に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	9 章
Portable Global JNDI 名の利用	Portable Global JNDI 名を利用したオブジェクトのルックアップができるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	2.4
JAX-WS 2.2 への対応	JAX-WS 2.2 に対応しました。	Web サービス開発ガイド	1.1, 16.1.5, 16.1.7, 16.2.1, 16.2.6, 16.2.10, 16.2.12, 16.2.13,

項目	変更の概要	参照先マニュアル	参照個所
			16.2.14, 16.2.16, 16.2.17, 16.2.18, 16.2.20, 16.2.22, 19.1, 19.2.3, 37.2, 37.6.1, 37.6.2, 37.6.3
JAX-RS 1.1 への対応	JAX-RS 1.1 に対応しました。	Web サービス開発ガイド	1.1, 1.2.2, 1.3.2, 1.4.2, 1.5.1, 1.6，2.3, 11 章, 12 章, 13 章, 17 章, 24 章, 39 章

(3) 信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 D–27 信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
SSL/TLS 通信での TLSv1.2 の使用	RSA BSAFE SSL-J を使用して，TLSv1.2 を含むセキュリティ・プロトコルで SSL/TLS 通信ができるようになりました。	機能解説 セキュリティ管理機能編	7.3

(4) 運用性の維持・向上

運用性の維持・向上を目的として変更した項目を次の表に示します。

表 D–28 運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Web コンテナ全体の実行待ちキューの総和の監視	Web コンテナ全体の実行待ちキューの総和を稼働情報に出力して監視できるようになりました。	機能解説 運用／監視／連携編	3 章

項目	変更の概要	参照先マニュアル	参照個所
アプリケーションの性能解析トレース（ユーザ拡張トレース）の出力	ユーザが開発したアプリケーションの処理性能を解析するための性能解析トレースを，アプリケーションの変更をしないで出力できるようになりました。	このマニュアル	7 章
仮想化環境でのユーザスクリプトを使用した運用	任意のタイミングでユーザ作成のスクリプト（ユーザスクリプト）を仮想サーバ上で実行できるようになりました。	仮想化システム構築・運用ガイド	7.8
運用管理ポータル改善	運用管理ポータルの次の画面で，手順を示すメッセージを画面に表示するように変更しました。 • ［設定情報の配布］画面 • Web サーバ，J2EE サーバおよび SFO サーバの起動画面 • Web サーバクラスタと J2EE サーバクラスタの一括起動，一括再起動および起動画面	運用管理ポータル操作ガイド	10.10.1, 11.9.2, 11.10.2, 11.10.4, 11.10.6, 11.11.2, 11.12.2, 11.12.4, 11.12.6
運用管理機能の再起動機能の追加	運用管理機能（Management Server および運用管理エージェント）で自動再起動が設定できるようになり，運用管理機能で障害が発生した場合でも運用が継続できるようになりました。また，自動起動の設定方法も変更になりました。	機能解説 運用／監視／連携編	2.4.1, 2.4.2, 2.6.3, 2.6.4
		リファレンス コマンド編	mngautorun（自動起動および自動再起動の設定／設定解除）

(5) そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 D-29 そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
ログ出力時のファイル切り替え単位の変更	ログ出力時に，日付ごとに出力先のファイルを切り替えられるようになりました。	このマニュアル	3.2.1
Web サーバの名称の変更	アプリケーションサーバに含まれる Web サーバの名称を HTTP Server に変更しました。	HTTP Server	－
BIG-IP の API（SOAP アーキテクチャ）を使用した直接接続への対応	BIG-IP（負荷分散機）で API（SOAP アーキテクチャ）を使用した直接接続に対応しました。 また，API を使用した直接接続を使用する場合に，負荷分散機の接続環境を設定する方法が変更になりました。	システム構築・運用ガイド	4.7.3，付録 J
		仮想化システム構築・運用ガイド	2.1，付録 C

項目	変更の概要	参照先マニュアル	参照個所
		機能解説 セキュリティ管理機能編	8.2，8.4，8.5，8.6，18.2.1，18.2.2，18.2.3

（凡例）－：マニュアル全体を参照する

付録 D.11　08-70 での主な機能変更

(1)　導入・構築の容易性強化

導入・構築の容易性強化を目的として変更した項目を次の表に示します。

表 D-30　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
運用管理ポータル改善	運用管理ポータルの画面で，リソースアダプタの属性を定義するプロパティ（Connector 属性ファイルの設定内容）の設定，および接続テストができるようになりました。また，運用管理ポータルの画面で，J2EE アプリケーション（ear ファイルおよび zip ファイル）を Management Server にアップロードできるようになりました。	ファーストステップガイド	3.5
		運用管理ポータル操作ガイド	－
page/tag ディレクティブの import 属性暗黙インポート機能の追加	page/tag ディレクティブの import 属性暗黙インポート機能を使用できるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.3.7
仮想化環境での JP1 製品に対する環境設定の自動化対応	仮想サーバへのアプリケーションサーバ構築時に，仮想サーバに対する JP1 製品の環境設定を，フックスクリプトで自動的に設定できるようになりました。	仮想化システム構築・運用ガイド	7.7.2
統合ユーザ管理機能の改善	ユーザ情報リポジトリでデータベースを使用する場合に，データベース製品の JDBC ドライバを使用して，データベースに接続できるようになりました。 DABroker Library の JDBC ドライバによるデータベース接続はサポート外になりました。 簡易構築定義ファイルおよび運用管理ポータルの画面で，統合ユーザ管理機能に関する設定ができるようになりました。 また，Active Directory の場合，DN で日本語などの 2 バイト文字に対応しました。	機能解説 セキュリティ管理機能編	5 章，14.2.2
		運用管理ポータル操作ガイド	3.5，10.8.1
HTTP Server 設定項目の拡充	簡易構築定義ファイルおよび運用管理ポータルの画面で，HTTP Server の動作環境を定義するディレクティブ（httpsd.conf の設定内容）を直接設定できるようになりました。	システム構築・運用ガイド	4.1.21

項目	変更の概要	参照先マニュアル	参照個所
		運用管理ポータル操作ガイド	10.9.1
		リファレンス 定義編(サーバ定義)	4.10

（凡例）－：マニュアル全体を参照する

(2) 標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更した項目を次の表に示します。

表 D–31 標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
ejb-jar.xml の指定項目の追加	ejb-jar.xml に，クラスレベルインターセプタおよびメソッドレベルインターセプタを指定できるようになりました。	機能解説 基本・開発編(EJB コンテナ)	2.15
パラレルコピーガーベージコレクションへの対応	パラレルコピーガーベージコレクションを選択できるようになりました。	リファレンス 定義編(サーバ定義)	14.5
Connector 1.5 仕様に準拠した Inbound リソースアダプタのグローバルトランザクションへの対応	Connector 1.5 仕様に準拠したリソースアダプタで Transacted Delivery を使用できるようにしました。これによって，Message-driven Bean を呼び出す EIS がグローバルトランザクションに参加できるようになりました。	機能解説 基本・開発編(コンテナ共通機能)	3.16.3
TP1 インバウンドアダプタの MHP への対応	TP1 インバウンドアダプタを使用してアプリケーションサーバを呼び出す OpenTP1 のクライアントとして，MHP を使用できるようになりました。	機能解説 基本・開発編(コンテナ共通機能)	4 章
cjrarupdate コマンドの FTP インバウンドアダプタへの対応	cjrarupdate コマンドでバージョンアップできるリソースアダプタに FTP インバウンドアダプタを追加しました。	リファレンス コマンド編	2.2

(3) 信頼性の維持・向上

信頼性の維持・向上を目的として変更した項目を次の表に示します。

表 D–32 信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
データベースセッションフェイルオーバ機能の改善	性能を重視するシステムで，グローバルセッション情報を格納したデータベースのロックを取得しないモードを選択できるようになりました。また，データベースを更新しない，参照専用のリクエストを定義できるようになりました。	機能解説 拡張編	6 章

項目	変更の概要	参照先マニュアル	参照個所
OutOfMemory ハンドリング機能の対象となる処理の拡大	OutOfMemory ハンドリング機能の対象となる処理を追加しました。	このマニュアル	2.5.4
		リファレンス 定義編(サーバ定義)	14.2
HTTP セッションで利用する Explicit ヒープの省メモリ化機能の追加	HTTP セッションで利用する Explicit ヒープのメモリ使用量を抑止する機能を追加しました。	機能解説 拡張編	7.11

(4) 運用性の維持・向上

運用性の維持・向上を目的として変更した項目を次の表に示します。

表 D-33 運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
仮想化環境での JP1 製品を使用したユーザ認証への対応（クラウド運用対応）	JP1 連携時に，JP1 製品の認証サーバを利用して，仮想サーバマネージャを使用するユーザを管理・認証できるようになりました。	仮想化システム構築・運用ガイド	1.2.2，3 章，4 章，5 章，6 章，7.9

(5) そのほかの目的

そのほかの目的で変更した項目を次の表に示します。

表 D-34 そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
負荷分散機への API（REST アーキテクチャ）を使用した直接接続の対応	負荷分散機への接続方法として，API（REST アーキテクチャ）を使用した直接接続に対応しました。 また，使用できる負荷分散機の種類に ACOS（AX2500）が追加になりました。	システム構築・運用ガイド	4.7.2，4.7.3
		仮想化システム構築・運用ガイド	2.1
		リファレンス 定義編(サーバ定義)	4.2.4
snapshot ログ収集時のタイムアウトへの対応と収集対象の改善	snapshot ログの収集が指定した時間で終了（タイムアウト）できるようになりました。一次送付資料として収集される内容が変更になりました。	このマニュアル	付録 A

付録 D.12 08-53 での主な機能変更

(1) 導入・構築の容易性強化

導入・構築の容易性強化を目的として変更された項目を次の表に示します。

表 D–35　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
さまざまなハイパーバイザに対応した仮想化環境の構築	さまざまなハイパーバイザを使用して実現する仮想サーバ上に，アプリケーションサーバを構築できるようになりました。 また，複数のハイパーバイザが混在する環境にも対応しました。	仮想化システム構築・運用ガイド	2 章，3 章，5 章

(2)　標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更された項目を次の表に示します。

表 D–36　標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
トランザクション連携に対応した OpenTP1 からの呼び出し	OpenTP1 からアプリケーションサーバ上で動作する Message-driven Bean を呼び出すときに，トランザクション連携ができるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	4 章
JavaMail	POP3 に準拠したメールサーバと連携して，JavaMail 1.3 に準拠した API を使用したメール受信機能を利用できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	8 章

(3)　信頼性の維持・向上

信頼性の維持・向上を目的として変更された項目を次の表に示します。

表 D–37　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JavaVM のトラブルシュート機能強化	JavaVM のトラブルシュート機能として，次の機能が使用できるようになりました。 • OutOfMemoryError 発生時の動作を変更できるようになりました。 • JIT コンパイル時に，C ヒープ確保量の上限値を設定できるようになりました。 • スレッド数の上限値を設定できるようになりました。 • 拡張 verbosegc 情報の出力項目を拡張しました。	このマニュアル	4 章，5 章，9 章

(4)　運用性の維持・向上

運用性の維持・向上を目的として変更された項目を次の表に示します。

表 D–38　運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
JP1/ITRM への対応	IT リソースを一元管理する製品である JP1/ITRM に対応しました。	仮想化システム構築・運用ガイド	1.3，2.1

(5)　そのほかの目的

そのほかの目的で変更された項目を次の表に示します。

表 D–39　そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
Microsoft IIS 7.0 および Microsoft IIS 7.5 への対応	Web サーバとして Microsoft IIS 7.0 および Microsoft IIS 7.5 に対応しました。	－	－
HiRDB Version 9 および SQL Server 2008 への対応	データベースとして次の製品に対応しました。 • HiRDB Server Version 9 • HiRDB/Developer's Kit Version 9 • HiRDB/Run Time Version 9 • SQL Server 2008 また，SQL Server 2008 に対応する JDBC ドライバとして，SQL Server JDBC Driver に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	3 章

（凡例）－：該当なし。

付録 D.13　08-50 での主な機能変更

(1)　導入・構築の容易性強化

導入・構築の容易性強化を目的として変更された項目を次の表に示します。

表 D–40　導入・構築の容易性強化を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Web サービスプロバイダ側での web.xml の指定必須タグの変更	Web サービスプロバイダ側での web.xml で，listener タグ，servlet タグおよび servlet-mapping タグの指定を必須から任意に変更しました。	リファレンス 定義編 (サーバ定義)	2.2.3
論理サーバのネットワークリソース使用	J2EE アプリケーションからほかのホスト上にあるネットワークリソースやネットワークドライブにアクセスするための機能を追加しました。	機能解説 運用／監視／連携編	1.2.3，5.2，5.7
サンプルプログラムの実行手順の簡略化	一部のサンプルプログラムを EAR 形式で提供することによって，サンプルプログラムの実行手順を簡略化しました。	ファーストステップガイド	3.5

項目	変更の概要	参照先マニュアル	参照個所
		システム構築・運用ガイド	付録 L
運用管理ポータルの画面の動作の改善	画面の更新間隔のデフォルトを「更新しない」から「3秒」に変更しました。	運用管理ポータル操作ガイド	7.4.1
セットアップウィザードの完了画面の改善	セットアップウィザード完了時の画面に，セットアップで使用した簡易構築定義ファイルおよび Connector 属性ファイルが表示されるようになりました。	システム構築・運用ガイド	2.2.6
仮想化環境の構築	ハイパーバイザを使用して実現する仮想サーバ上に，アプリケーションサーバを構築する手順を追加しました。	仮想化システム構築・運用ガイド	3 章，5 章

(2) 標準機能・既存機能への対応

標準機能・既存機能への対応を目的として変更された項目を次の表に示します。

表 D–41 標準機能・既存機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
OpenTP1 からの呼び出しへの対応	OpenTP1 からアプリケーションサーバ上で動作する Message-driven Bean を呼び出せるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	4 章
JMS への対応	JMS 1.1 仕様に対応した CJMS プロバイダ機能を使用できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	7 章
Java SE 6 への対応	Java SE 6 の機能が使用できるようになりました。	このマニュアル	5.5, 5.8.1
ジェネリクスの使用への対応	EJB でジェネリクスを使用できるようになりました。	機能解説 基本・開発編 (EJB コンテナ)	4.2.18

(3) 信頼性の維持・向上

信頼性の維持・向上を目的として変更された項目を次の表に示します。

表 D–42 信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
明示管理ヒープ機能の使用性向上	自動配置設定ファイルを使用して，明示管理ヒープ機能を容易に使用できるようになりました。	システム設計ガイド	7.2, 7.7.3, 7.11.4, 7.12.1
		機能解説 拡張編	7 章
データベースセッションフェイルオーバ機能の URI 単位での抑止	データベースセッションフェイルオーバ機能を使用する場合に，機能の対象外にするリクエストを URI 単位で指定できるようになりました。	機能解説 拡張編	5.6.1

項目	変更の概要	参照先マニュアル	参照個所
仮想化環境での障害監視	仮想化システムで，仮想サーバを監視し，障害の発生を検知できるようになりました。	仮想化システム構築・運用ガイド	－

(4) 運用性の維持・向上

運用性の維持・向上を目的として変更された項目を次の表に示します。

表 D-43 運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
管理ユーザアカウントの省略	運用管理ポータル，Management Server のコマンド，または Smart Composer 機能のコマンドで，ユーザのログイン ID およびパスワードの入力を省略できるようになりました。	システム構築・運用ガイド	4.1.15
		運用管理ポータル操作ガイド	2.2，7.1.1，7.1.2，7.1.3，8.1，8.2.1，付録 E.2
		リファレンス コマンド編	1.4，mngsvrctl (Management Server の起動／停止／セットアップ)，mngsvrutil (Management Server の運用管理コマンド)，8.3，cmx_admin_passwd (Management Server の管理ユーザアカウントの設定)
仮想化環境の運用	仮想化システムで，複数の仮想サーバを対象にした一括起動・一括停止，スケールイン・スケールアウトなどを実行する手順を追加しました。	仮想化システム構築・運用ガイド	4 章，6 章

(5) そのほかの目的

そのほかの目的で変更された項目を次の表に示します。

表 D-44 そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
Tenured 領域内不要オブジェクト統計機能	Tenured 領域内で不要となったオブジェクトだけを特定できるようになりました。	このマニュアル	9.8
Tenured 増加要因の基点オブジェクトリスト出力機能	Tenured 領域内不要オブジェクト統計機能を使って特定した，不要オブジェクトの基点となるオブジェクトの情報を出力できるようになりました。		9.9

項目	変更の概要	参照先マニュアル	参照個所
クラス別統計情報解析機能	クラス別統計情報を CSV 形式で出力できるようになりました。		9.10
論理サーバの自動再起動回数オーバー検知によるクラスタ系切り替え	Management Server を系切り替えの監視対象としているクラスタ構成の場合，論理サーバが異常停止状態(自動再起動回数をオーバーした状態，または自動再起動回数の設定が 0 のときに障害を検知した状態)になったタイミングでの系切り替えができるようになりました。	機能解説 運用／監視／連携編	16.2.2, 16.3.3, 16.3.4, 18.4.3, 18.5.3
ホスト単位管理モデルを対象とした系切り替えシステム	クラスタソフトウェアと連携したシステム運用で，ホスト単位管理モデルを対象にした系切り替えができるようになりました。		16 章
ACOS（AX2000，BS320）のサポート	使用できる負荷分散機の種類に ACOS（AX2000，BS320）が追加になりました。	システム構築・運用ガイド	4.7.2, 4.7.3, 4.7.5, 4.7.6，付録 J，付録 J.2
		リファレンス 定義編(サーバ定義)	4.2.4, 4.3.2, 4.3.4, 4.3.5, 4.3.6, 4.7.1
CMT でトランザクション管理をする場合に Stateful Session Bean（SessionSynchronization）に指定できるトランザクション属性の追加	CMT でトランザクション管理をする場合に，Stateful Session Bean（SessionSynchronization）にトランザクション属性として Supports，NotSupported および Never を指定できるようになりました。	機能解説 基本・開発編(EJB コンテナ)	2.7.3
OutOfMemoryError 発生時の運用管理エージェントの強制終了	JavaVM で OutOfMemoryError が発生したときに，運用管理エージェントが強制終了するようになりました。	このマニュアル	2.5.5
スレッドの非同期並行処理	TimerManager および WorkManager を使用して，非同期タイマ処理および非同期スレッド処理を実現できるようになりました。	機能解説 拡張編	－

付録 D.14　08-00 での主な機能変更

(1)　開発生産性の向上

開発生産性の向上を目的として変更された項目を次の表に示します。

表 D–45　開発生産性の向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
ほかのアプリケーションサーバ製品からの移行容易化	ほかのアプリケーションサーバ製品からの移行を円滑に実施するため，次の機能を使用できるようになりました。 • HTTP セッションの上限が例外で判定できるようになりました。 • JavaBeans の ID が重複している場合や，カスタムタグの属性名と TLD の定義で大文字・小文字が異なる場合に，トランスレーションエラーが発生することを抑止できるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.3, 2.7.5
cosminexus.xml の提供	アプリケーションサーバ独自の属性を cosminexus.xml に記載することによって，J2EE アプリケーションを J2EE サーバにインポート後，プロパティの設定をしないで開始できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	16.3

(2)　標準機能への対応

標準機能への対応を目的として変更された項目を次の表に示します。

表 D–46　標準機能への対応を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
Servlet 2.5 への対応	Servlet 2.5 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	2.2, 2.5.4, 2.6，8 章
JSP 2.1 への対応	JSP 2.1 に対応しました。	機能解説 基本・開発編 (Web コンテナ)	2.3.1, 2.3.3, 2.5, 2.6，8 章
JSP デバッグ	MyEclipse を使用した開発環境で JSP デバッグができるようになりました。※	機能解説 基本・開発編 (Web コンテナ)	2.4
タグライブラリのライブラリ JAR への格納と TLD のマッピング	タグライブラリをライブラリ JAR に格納した場合に，Web アプリケーション開始時に Web コンテナによってライブラリ JAR 内の TLD ファイルを検索し，自動的にマッピングできるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.3.4
application.xml の省略	J2EE アプリケーションで application.xml が省略できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	16.4
アノテーションと DD の併用	アノテーションと DD を併用できるようになり，アノテーションで指定した内容を DD で更新できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	17.5

項目	変更の概要	参照先マニュアル	参照個所
アノテーションの Java EE 5 標準準拠（デフォルトインターセプタ）	デフォルトインターセプタをライブラリ JAR に格納できるようになりました。また，デフォルトインターセプタから DI できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	16.4
@Resource の参照解決	@Resource でリソースの参照解決ができるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	17.4
JPA への対応	JPA 仕様に対応しました。	機能解説 基本・開発編 (コンテナ共通機能)	6 章

注※　09-00 以降では，WTP を使用した開発環境で JSP デバッグ機能を使用できます。

(3)　信頼性の維持・向上

信頼性の維持・向上を目的として変更された項目を次の表に示します。

表 D–47　信頼性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
セッション情報の永続化	HTTP セッションのセッション情報をデータベースに保存して引き継げるようになりました。	機能解説 拡張編	5 章，6 章
FullGC の抑止	FullGC の要因となるオブジェクトを Java ヒープ外に配置することで，FullGC 発生を抑止できるようになりました。	機能解説 拡張編	7 章
クライアント性能モニタ	クライアント処理に掛かった時間を調査・分析できるようになりました。	－	－

（凡例）－：09-00 で削除された機能です。

(4)　運用性の維持・向上

運用性の維持・向上を目的として変更された項目を次の表に示します。

表 D–48　運用性の維持・向上を目的とした変更

項目	変更の概要	参照先マニュアル	参照個所
運用管理ポータルでのアプリケーション操作性向上	アプリケーションおよびリソースの操作について，サーバ管理コマンドと運用管理ポータルの相互運用ができるようになりました。	運用管理ポータル操作ガイド	1.1.3

(5)　そのほかの目的

そのほかの目的で変更された項目を次の表に示します。

表 D-49　そのほかの目的による変更

項目	変更の概要	参照先マニュアル	参照個所
無効な HTTP Cookie の削除	無効な HTTP Cookie を削除できるようになりました。	機能解説 基本・開発編 (Web コンテナ)	2.7.4
ネーミングサービスの障害検知	ネーミングサービスの障害が発生した場合に，EJB クライアントが，より早くエラーを検知できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	2.9
コネクション障害検知タイムアウト	コネクション障害検知タイムアウトのタイムアウト時間を指定できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	3.15.1
Oracle11g への対応	データベースとして Oracle11g が使用できるようになりました。	機能解説 基本・開発編 (コンテナ共通機能)	3 章
バッチ処理のスケジューリング	バッチアプリケーションの実行を CTM によってスケジューリングできるようになりました。	機能解説 拡張編	4 章
バッチ処理のログ	バッチ実行コマンドのログファイルのサイズ，面数，ログの排他処理失敗時のリトライ回数とリトライ間隔を指定できるようになりました。	リファレンス 定義編 (サーバ定義)	3.2.5
snapshot ログ	snapshot ログの収集内容が変更されました。	このマニュアル	付録 A.1，付録 A.2
メソッドキャンセルの保護区公開	メソッドキャンセルの対象外となる保護区リストの内容を公開しました。	機能解説 運用／監視／連携編	付録 C
統計前の GC 選択機能	クラス別統計情報を出力する前に，GC を実行するかどうかを選択できるようになりました。	このマニュアル	9.7
Survivor 領域の年齢分布情報出力機能	Survivor 領域の Java オブジェクトの年齢分布情報を JavaVM ログファイルに出力できるようになりました。	このマニュアル	9.11
ファイナライズ滞留解消機能	JavaVM のファイナライズ処理の状態を監視して，処理の滞留を解消できるようになりました。	－	－
サーバ管理コマンドの最大ヒープサイズの変更	サーバ管理コマンドが使用する最大ヒープサイズが変更されました。	リファレンス 定義編 (サーバ定義)	5.2.1，5.2.2
推奨しない表示名を指定された場合の対応	J2EE アプリケーションで推奨しない表示名を指定された場合にメッセージが出力されるようになりました。	メッセージ(構築／運用／開発用)	KDJE42374-W

（凡例）－：09-00 で削除された機能です。

付録 E　用語解説

マニュアルで使用する用語について

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 用語解説」を参照してください。

索引

記号

数字

C

D

E

G

H

I

J

L

M

N

O

P

R

S

T

U

V

え

お

か

き

く

こ

さ

し

に

は

る

れ

ろ