HITACHI
Inspire the Next

Cosminexus V11 アプリケーションサーバ 機能解説 運用／監視／連携編

解説書

3021-3-J10-30

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。

Active Directory は，マイクロソフト 企業グループの商標です。

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

DB2 は，世界の多くの国で登録された International Business Machines Corporation の商標です。

Excel は，マイクロソフト 企業グループの商標です。

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

Intel は，Intel Corporation またはその子会社の商標です。

Internet Explorer は，マイクロソフト 企業グループの商標です。

Linux は，Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。

Microsoft は，マイクロソフト 企業グループの商標です。

Oracle および Java は，オラクルおよびその関連会社の登録商標です。

Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.

Red Hat 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。

Red Hat Enterprise Linux is a registered trademark of Red Hat, Inc. in the United States and other countries.

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。

SQL Server は，マイクロソフト 企業グループの商標です。

UNIX は，The Open Group の登録商標です。

Windows は，マイクロソフト 企業グループの商標です。

Windows Server は，マイクロソフト 企業グループの商標です。

すべての SPARC 商標は，米国 SPARC International,Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標がついた製品は，米国 Sun Microsystems,Inc. が開発したアーキテクチャに基づくものです。

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

2022 年 3 月 3021-3-J10-30

■ 著作権

All Rights Reserved. Copyright (C) 2020, 2022, Hitachi, Ltd.

変更内容

変更内容(3021-3-J10-30) uCosminexus Application Server 11-20，uCosminexus Client 11-20，uCosminexus Developer 11-20，uCosminexus Service Architect 11-20，uCosminexus Service Platform 11-20

追加・変更内容	変更個所
Java EE 8 に対応した。	1.1
アプリケーションサーバが対応している標準仕様に，JAX-RS 2.1，JPA 2.2，JSON-B 1.0 を追加した。	1.1.3
アプリケーションサーバ 11-20 での主な機能変更についての説明を追加した。	1.5

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

前書き　2
変更内容　4
はじめに　5

第 1 編　概要

1　アプリケーションサーバの機能　17
1.1　機能の分類　18
1.1.1　アプリケーションの実行基盤としての機能　20
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　21
1.1.3　機能とマニュアルの対応　22
1.2　システムの目的と機能の対応　25
1.2.1　システムの日常運用を支援する機能　25
1.2.2　システムの保守を支援する機能　26
1.2.3　J2EE アプリケーションの運用機能　26
1.2.4　システムの監査を支援する機能　27
1.2.5　JP1 連携による運用管理機能　27
1.2.6　クラスタソフトウェアとの連携による系切り替え機能　27
1.3　このマニュアルに記載している機能の説明　29
1.3.1　分類の意味　29
1.3.2　分類を示す表の例　29
1.4　Component Container 管理者によるシステムの運用（UNIX の場合）　31
1.5　アプリケーションサーバ 11-20 での主な機能変更　33
1.5.1　標準機能・既存機能への対応　33

第 2 編　運用・監視機能

2　システムの起動と停止　34
2.1　この章の構成　35
2.2　日常運用での起動と停止　36
2.3　論理サーバの起動・停止の仕組み　37
2.3.1　論理サーバの起動と稼働確認　37
2.3.2　論理サーバの停止　39
2.4　障害発生時の自動再起動　40
2.4.1　Management Server の自動再起動　40

2.4.2 運用管理エージェントの自動再起動 40
2.4.3 論理サーバの自動再起動 40
2.5 システムの稼働監視 43
2.5.1 システムの運用状況の監視 43
2.5.2 ステータスの監視 45
2.6 システムの起動と停止の設定 48
2.6.1 システムの起動 48
2.6.2 システムの停止 50
2.6.3 自動起動の設定 52
2.6.4 自動再起動の設定 55
2.6.5 自動停止の設定 58

3 稼働情報の監視（稼働情報収集機能） 65
3.1 この章の構成 66
3.2 稼働情報収集機能の概要 67
3.3 稼働情報ファイルの出力機能 68
3.3.1 稼働情報ファイルで収集できる情報の種類 71
3.3.2 稼働情報ファイルで収集できる情報 74
3.3.3 稼働情報ファイルの出力先とファイル面数 75
3.3.4 実行環境での設定（J2EE サーバの設定） 76
3.3.5 稼働情報ファイルの出力形式と出力内容 77
3.4 イベントの発行機能 96
3.4.1 しきい値を設定できる監視対象 96
3.4.2 イベントの発行方法 98
3.4.3 cosminexus.xml での定義 99
3.4.4 実行環境での設定（J2EE サーバの設定） 101

4 リソースの枯渇監視 103
4.1 この章の構成 104
4.2 リソース枯渇監視機能の概要 105
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 106
4.3.1 監視できるリソースの種類 106
4.3.2 cosminexus.xml での定義 108
4.3.3 実行環境での設定 109
4.3.4 リソース枯渇監視情報ファイルの出力先と出力情報の種類 112
4.3.5 出力形式と出力内容 114

5 J2EE アプリケーションの運用 122
5.1 この章の構成 123
5.2 J2EE アプリケーションの運用の概要 124

5.2.1 タイムアウトしたリクエストをキャンセルする 124
5.2.2 J2EE アプリケーションを停止する 124
5.2.3 J2EE アプリケーションを強制停止する 124
5.2.4 J2EE アプリケーションを入れ替える 125
5.2.5 J2EE アプリケーションからネットワークリソースにアクセスする 125
5.2.6 J2EE アプリケーションの運用の作業と参照先 125
5.3 J2EE アプリケーションの実行時間の監視とキャンセル 128
5.3.1 J2EE アプリケーションの実行時間の監視とキャンセルの概要 129
5.3.2 J2EE アプリケーション実行時間の監視とは 129
5.3.3 メソッドタイムアウトとは 131
5.3.4 メソッドキャンセルとは 132
5.3.5 タイムアウト値の設定例と設定値の有効範囲 136
5.3.6 使用するスレッド数 142
5.3.7 cosminexus.xml での定義 142
5.3.8 実装時の注意事項 143
5.3.9 実行環境での設定 147
5.3.10 J2EE アプリケーションの実行時間の監視とキャンセルの流れ 148
5.3.11 J2EE アプリケーションの実行状態の確認 148
5.3.12 タイムアウトが発生したリクエストのキャンセル 150
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 151
5.4 J2EE アプリケーションのサービスの閉塞 156
5.4.1 Web アプリケーションのサービスの閉塞と閉塞解除とは 156
5.4.2 システムごとに実行できるサービスの閉塞方法および J2EE アプリケーションの停止方法 160
5.4.3 負荷分散機を利用したサービス閉塞 162
5.5 J2EE アプリケーションの停止 165
5.5.1 J2EE アプリケーションの停止とは 165
5.5.2 閉塞処理とは 168
5.5.3 停止処理とは 170
5.5.4 強制停止処理とは 172
5.5.5 cosminexus.xml での定義 173
5.5.6 実行環境での設定 173
5.5.7 J2EE アプリケーションの停止 174
5.6 J2EE アプリケーションの入れ替え 181
5.6.1 J2EE アプリケーションの入れ替えとは 181
5.6.2 Web アプリケーションのサービスの部分閉塞による入れ替え 185
5.6.3 J2EE アプリケーションの入れ替えと保守 186
5.7 J2EE アプリケーションからのネットワークリソースへのアクセス 194
5.7.1 ネットワークリソースへのアクセスの概要 194
5.7.2 ネットワークリソースにアクセスするための設定 195

5.7.3 運用時のトラブルについて 197
5.8 J2EE アプリケーション運用時の注意事項 198

6 監査ログ出力機能 199
6.1 この章の構成 200
6.2 監査ログ出力機能の概要 201
6.3 監査ログとは 202
6.3.1 監査ログが出力される流れと主な出力情報 202
6.3.2 監査事象の定義 202
6.3.3 監査ログ出力機能を使用するシステムで必要な作業と参照先 203
6.4 監査ログの出力 205
6.4.1 監査ログの出力方式 205
6.4.2 監査ログの出力先 205
6.4.3 監査ログの出力形式 205
6.4.4 監査ログの出力項目 206
6.5 監査ログの保存 212
6.5.1 監査ログの出力に失敗した場合にシステムを自動で停止する流れ 212
6.5.2 監査ログを自動でアーカイブする流れ 213
6.6 監査ログを出力するコマンド・操作一覧 214
6.6.1 J2EE サーバで使用するコマンド一覧 214
6.6.2 バッチサーバで使用するコマンド一覧 216
6.6.3 性能解析トレース・CTM で使用するコマンド一覧 218
6.6.4 Management Server で使用するコマンド一覧 218
6.6.5 EJB クライアントアプリケーションで使用するコマンド一覧 221
6.6.6 HTTP Server で使用するコマンド・操作一覧（Windows の場合） 221
6.6.7 HTTP Server で使用するコマンド・操作一覧（UNIX の場合） 221
6.7 監査ログの出力ポイント 223
6.7.1 J2EE サーバで使用するコマンドの監査ログの出力ポイント 223
6.7.2 バッチサーバで使用するコマンドの監査ログの出力ポイント 237
6.7.3 性能解析トレース・CTM で使用するコマンドの監査ログの出力ポイント 242
6.7.4 Management Server で使用するコマンドの監査ログの出力ポイント 246
6.7.5 EJB クライアントアプリケーションで使用するコマンドの監査ログの出力ポイント 256
6.7.6 HTTP Server で使用するコマンド，および HTTP Server に対する操作の監査ログの出力ポイント（Windows の場合） 257
6.7.7 HTTP Server で使用するコマンド，および HTTP Server に対する操作の監査ログの出力ポイント（UNIX の場合） 259
6.8 アプリケーションの監査ログを出力するための実装 264
6.8.1 監査ログ出力用の API の実装例 264
6.8.2 監査ログ出力用の API 実装時の注意事項 265
6.9 監査ログ出力の設定 266

7 データベース監査証跡連携機能 267
7.1 この章の構成 268
7.2 データベース監査証跡連携機能の概要 269
7.3 データベース監査証跡との連携 270
7.3.1 データベース監査証跡とは 270
7.3.2 データベース監査証跡と連携して取得できる情報 270
7.4 データベース監査証跡への情報の出力 271
7.4.1 データベース監査証跡への出力の流れ 271
7.4.2 前提条件 271
7.4.3 データベース監査証跡に出力する情報 272
7.4.4 データベース監査証跡の出力個所 273
7.5 データベース監査証跡との連携機能の設定 274
7.5.1 実行環境での設定 274
7.5.2 データベースサーバでの設定 275
7.6 データベース監査証跡と連携した運用例 276
7.6.1 アプリケーションのユーザログの取得 276
7.6.2 データベースの監査証跡情報の取得 277
7.6.3 性能解析トレースの取得 277
7.6.4 情報の特定 278
7.7 ルートアプリケーション情報をデータベースに設定する場合の注意事項 279

8 運用管理コマンドによる稼働情報の出力 280
8.1 この章の構成 281
8.2 運用管理コマンドによる稼働情報の出力の概要 282
8.3 サーバの稼働情報の出力方法 284
8.4 稼働情報監視で確認できる項目 285

9 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 294
9.1 この章の構成 295
9.2 Management イベントの通知と Management アクションの概要 296
9.3 Management アクションの実行制御とは 298
9.4 Management イベントによる処理の自動実行の設定 301
9.4.1 Management イベントによる処理の自動実行の設定手順 301
9.4.2 実行環境での設定 302
9.4.3 Management イベント発行用メッセージ ID リストファイルの設定 303
9.4.4 Management イベント発行機能を利用する各機能の設定 305
9.4.5 Management アクション実行用プロパティファイルの設定 305
9.4.6 Management アクション実行コマンドの設定 308

10 CTM の稼働統計情報の収集 314
10.1 この章の構成 315
10.2 CTM の稼働統計情報の概要 316
10.2.1 CTM の稼働統計情報の収集方法 317
10.2.2 CTM の稼働統計情報の出力先と出力情報 318
10.2.3 CTM の稼働統計情報の出力例 321
10.3 稼働統計情報ファイルの利用方法 323

11 コンソールログの出力 324
11.1 この章の構成 325
11.2 コンソールログの出力対象 326
11.2.1 コンソールログの出力対象となる操作 326
11.2.2 コンソールログの出力対象となるプロセス 326
11.3 コンソールログの取得の設定 328

第 3 編 連携機能

12 JP1 と連携したシステムの運用 329
12.1 この章の構成 330
12.2 JP1 との連携 331
12.3 JP1 と連携するシステムでできること 332
12.3.1 システムの集中監視 332
12.3.2 システムの自動運転 333
12.3.3 JP1 で監視できるアプリケーションサーバのプロセス 333

13 システムの集中監視（JP1/IM との連携） 337
13.1 この章の構成 338
13.2 システムの集中監視（JP1/IM との連携）の概要 339
13.3 システムの集中監視の仕組み 340
13.3.1 システムの集中監視に必要なプログラム 341
13.3.2 監視ツリーの自動生成 342
13.3.3 統合機能メニューからの運用管理ポータルの表示（Windows の場合） 348
13.3.4 障害の監視の仕組み 348
13.4 システムの集中監視のための設定 353
13.4.1 監視ツリーの自動生成の設定 353
13.4.2 障害監視の設定 358
13.4.3 監視するログファイルの設定 364
13.5 運用例 370
13.5.1 Web フロントシステムでの運用例 370

14 ジョブによるシステムの自動運転（JP1/AJS との連携） 377
14.1 この章の構成 378
14.2 ジョブによるシステムの自動運転（JP1/AJS との連携）の概要 379
14.2.1 ジョブによるシステムの自動運転の仕組み 379
14.2.2 ジョブによるシステムの自動運転に必要なプログラム 381
14.2.3 カスタムジョブによるジョブの定義（Windows の場合） 384
14.2.4 論理サーバおよび J2EE アプリケーションの起動と停止の自動化の仕組み 385
14.3 ジョブによるシステムの自動運転の設定 387
14.3.1 mngsvrutil コマンドの実行環境の設定 387
14.3.2 J2EE サーバの運用監視の設定 388
14.3.3 アプリケーションサーバの JP1/AJS 定義プログラムのインストール（Windows の場合） 389
14.3.4 アプリケーションサーバのカスタムジョブの登録（Windows の場合） 389
14.3.5 ジョブの定義 390
14.3.6 スケジュールの定義 397

15 クラスタソフトウェアとの連携 399
15.1 この章の構成 400
15.2 クラスタソフトウェアと連携して実現できる運用 401
15.2.1 実行系と待機系を 1:1 で運用するシステムとは（1:1 系切り替えシステム） 403
15.2.2 相互スタンバイで運用するシステムとは（相互系切り替えシステム） 404
15.2.3 待機系をリカバリ専用サーバとして運用するシステムとは（N:1 リカバリシステム） 404
15.2.4 実行系と待機系のアプリケーションサーバマシン（ホスト）を N:1 で運用するシステムとは（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） 404
15.3 クラスタソフトウェアと連携するための前提条件 406
15.3.1 系切り替えの管理対象となるサーバ 406
15.3.2 ライトトランザクション機能の利用 406
15.3.3 システムの運用方法 407
15.3.4 実行系の前提条件 407
15.3.5 待機系の前提条件 407

16 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） 410
16.1 この章の構成 411
16.2 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のシステム構成と動作 412
16.2.1 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のシステム構成例 412
16.2.2 系切り替えのタイミング 413
16.2.3 系切り替え処理の流れ 414
16.3 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の設定 416
16.3.1 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の設定手順 416
16.3.2 クラスタサーバの環境設定 417

16.3.3 設定ファイルの編集 419
16.3.4 クラスタの設定 421
16.3.5 アプリケーションサーバの設定 423
16.4 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の起動と停止 426

17 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） 427
17.1 この章の構成 428
17.2 1:1 系切り替えシステムの概要 429
17.3 1:1 系切り替えシステムのシステム構成と動作 430
17.3.1 1:1 系切り替えシステムのシステム構成例 430
17.3.2 系切り替えのタイミング 432
17.3.3 1:1 系切り替えシステムでの系切り替え処理の流れ 433
17.3.4 系切り替え時のシステムの動作 436
17.3.5 系切り替え時の情報の引き継ぎ 437
17.4 アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合） 439
17.4.1 アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定手順 439
17.4.2 クラスタサーバの環境設定 442
17.4.3 設定ファイルの編集 443
17.4.4 クラスタの設定 444
17.4.5 アプリケーションサーバの設定 445
17.5 運用管理サーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合） 447
17.5.1 運用管理サーバを対象にした 1:1 系切り替えシステムの設定手順 447
17.5.2 クラスタサーバの環境設定 450
17.5.3 設定ファイルの編集 450
17.5.4 現用系から予備系への Management Server の設定情報のコピー 451
17.5.5 クラスタの設定 453
17.6 アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合） 454
17.6.1 アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定手順 454
17.6.2 クラスタサーバの環境設定 457
17.6.3 設定ファイルの編集 458
17.6.4 HA モニタの環境設定 460
17.6.5 シェルスクリプトファイルの作成 460
17.6.6 サーバ対応の環境設定 463
17.6.7 LAN の状態設定 464
17.6.8 アプリケーションサーバの設定 465
17.7 運用管理サーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合） 467
17.7.1 運用管理サーバを対象にした 1:1 系切り替えシステムの設定手順 467
17.7.2 クラスタサーバの環境設定 470
17.7.3 設定ファイルの編集 470

17.7.4 現用系から予備系への Management Server の設定情報のコピー 471
17.7.5 HA モニタの環境設定 473
17.7.6 シェルスクリプトファイルの作成 473
17.7.7 サーバ対応の環境設定 475
17.7.8 LAN の状態設定 477
17.8 アプリケーションサーバを対象にした 1:1 系切り替えシステムの起動と停止（Windows の場合） 478
17.8.1 アプリケーションサーバを対象にした 1:1 系切り替えシステムの起動 478
17.8.2 アプリケーションサーバを対象にした 1:1 系切り替えシステムの停止 479
17.8.3 アプリケーションサーバを対象にした 1:1 系切り替えシステムで計画的に系を切り替えるときの起動と停止 480
17.8.4 アプリケーションサーバを対象にした 1:1 系切り替えシステムをメンテナンスする場合の起動と停止 480
17.9 運用管理サーバを対象にした 1:1 系切り替えシステムの起動と停止（Windows の場合） 484
17.9.1 運用管理サーバを対象にした 1:1 系切り替えシステムの起動 484
17.9.2 運用管理サーバを対象にした 1:1 系切り替えシステムの停止 485
17.9.3 運用管理サーバを対象にした 1:1 系切り替えシステムで計画的に系を切り替えるときの起動と停止 485
17.10 1:1 系切り替えシステムの起動と停止（UNIX の場合） 487
17.10.1 1:1 系切り替えシステムの起動 487
17.10.2 1:1 系切り替えシステムの停止 489
17.10.3 1:1 系切り替えシステムで計画的に系を切り替える場合の起動と停止 490
17.10.4 アプリケーションサーバの 1:1 系切り替えシステムをメンテナンスする場合の起動と停止 491
17.10.5 運用管理サーバの 1:1 系切り替えシステムをメンテナンスする場合の起動と停止 495
17.11 系切り替えシステムの設定内容の確認方法 497
17.11.1 J2EE サーバの設定 497
17.11.2 HTTP Server の設定 497
17.11.3 運用管理サーバの設定 498

18 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） 499
18.1 この章の構成 500
18.2 相互系切り替えシステムの概要 501
18.3 相互系切り替えシステムのシステム構成と動作 502
18.3.1 相互系切り替えシステムのシステム構成例 502
18.3.2 相互系切り替えシステムでの系切り替え処理の流れ 505
18.4 相互系切り替えシステムの設定（Windows の場合） 508
18.4.1 相互系切り替えシステムの設定手順 508
18.4.2 クラスタサーバの環境設定 511
18.4.3 設定ファイルの編集 513
18.4.4 クラスタの設定 514
18.4.5 アプリケーションサーバの設定 515

18.5 相互系切り替えシステムの設定（UNIX の場合） 518
18.5.1 相互系切り替えシステムの設定手順 518
18.5.2 クラスタサーバの環境設定 521
18.5.3 設定ファイルの編集 523
18.5.4 HA モニタの環境設定 525
18.5.5 シェルスクリプトファイルの作成 525
18.5.6 サーバ対応の環境設定 527
18.5.7 LAN の状態設定 529
18.5.8 アプリケーションサーバの設定 530
18.6 相互系切り替えシステムの起動と停止（Windows の場合） 533
18.6.1 相互系切り替えシステムの起動 533
18.6.2 相互系切り替えシステムの停止 534
18.6.3 相互系切り替えシステムで計画的に系を切り替える場合の起動と停止 535
18.7 相互系切り替えシステムの起動と停止（UNIX の場合） 536
18.7.1 相互系切り替えシステムの起動 537
18.7.2 相互系切り替えシステムの停止 539
18.7.3 相互系切り替えシステムで計画的に系を切り替える場合の起動と停止 541
18.7.4 相互系切り替えシステムをメンテナンスする場合の起動と停止 543

19 N:1 リカバリシステム（クラスタソフトウェアとの連携） 547
19.1 この章の構成 548
19.2 N:1 リカバリシステムの概要 549
19.3 N:1 リカバリシステムのシステム構成と動作 550
19.3.1 N:1 リカバリシステムのシステム構成例 550
19.3.2 リカバリ処理の流れ 551
19.3.3 系切り替え時の情報の引き継ぎ 553
19.4 N:1 リカバリシステムの設定（Windows の場合） 554
19.4.1 N:1 リカバリシステムの設定手順 554
19.4.2 クラスタサーバの環境設定 558
19.4.3 設定ファイルの編集 558
19.4.4 クラスタの設定 559
19.4.5 アプリケーションサーバの設定 564
19.5 N:1 リカバリシステムの設定（UNIX の場合） 566
19.5.1 N:1 リカバリシステムの設定手順 566
19.5.2 クラスタサーバの環境設定 573
19.5.3 設定ファイルの編集 573
19.5.4 HA モニタの環境設定 574
19.5.5 シェルスクリプトファイルの作成 574
19.5.6 サーバ対応の環境設定 579

19.5.7 LAN の状態設定 582
19.5.8 アプリケーションサーバの設定 583
19.6 N:1 リカバリシステムの起動と停止（Windows の場合） 585
19.6.1 N:1 リカバリシステムの起動 585
19.6.2 N:1 リカバリシステムの停止 586
19.7 N:1 リカバリシステムの起動と停止（UNIX の場合） 588
19.7.1 N:1 リカバリシステムの起動 588
19.7.2 N:1 リカバリシステムの停止 589

付録 591

付録 A クラスタソフトウェア連携時の TPBroker の設定（Windows の場合） 592
付録 A.1 システムの起動と停止 592
付録 A.2 ORB 機能使用時の設定 592
付録 B クラスタソフトウェア連携時の TPBroker の設定（UNIX の場合） 599
付録 B.1 システムの起動と停止 599
付録 B.2 ORB 機能使用時の設定 600
付録 C 保護区リストの内容 606
付録 D 各バージョンでの主な機能変更 647
付録 D.1 11-10 での主な機能変更 647
付録 D.2 11-00 での主な機能変更 648
付録 D.3 09-87 での主な機能変更 650
付録 D.4 09-80 での主な機能変更 650
付録 D.5 09-70 での主な機能変更 651
付録 D.6 09-60 での主な機能変更 652
付録 D.7 09-50 での主な機能変更 653
付録 D.8 09-00 での主な機能変更 657
付録 D.9 08-70 での主な機能変更 660
付録 D.10 08-53 での主な機能変更 663
付録 D.11 08-50 での主な機能変更 665
付録 D.12 08-00 での主な機能変更 668
付録 E 用語解説 672

索引 673

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。

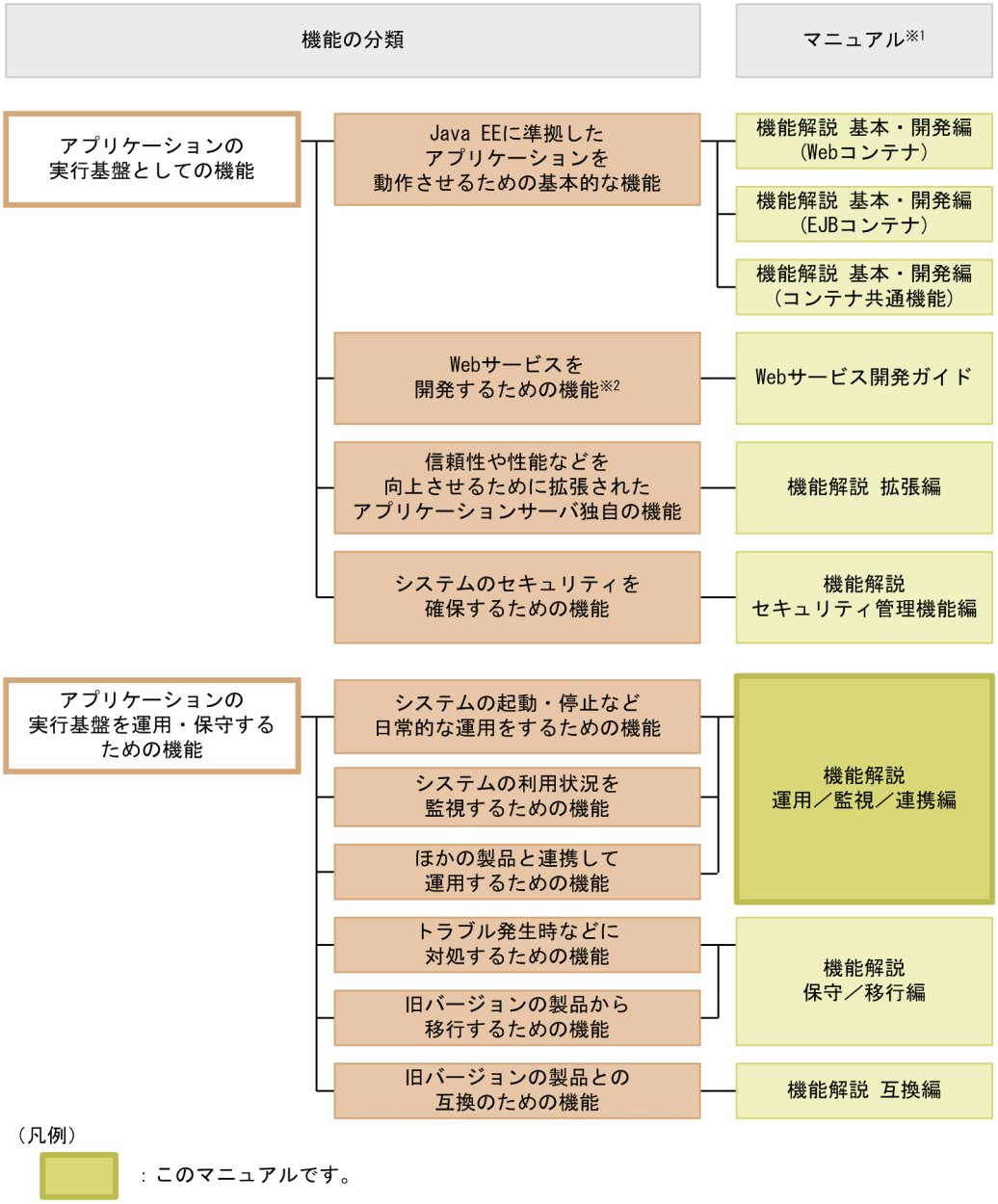
アプリケーションサーバの機能は，大きく分けて，次の二つに分類できます。

- 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	
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※2
	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）	
	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
	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。

- システムの日常運用を支援する機能
- システムの保守を支援する機能
- J2EE アプリケーションの運用機能
- システムの監査を支援する機能
- JP1 連携による運用管理機能
- クラスタソフトウェアとの連携による系切り替え機能

機能ごとに，次の項目への対応を示しています。

- **信頼性**
  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。

1.2.1　システムの日常運用を支援する機能

システムの日常運用を支援する機能の目的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　システムの日常運用を支援する機能の目的

機能名	目的		参照先
	信頼性	運用・保守	
システムの起動と停止	−	○	2 章
稼働情報の監視（稼働情報収集機能）	−	○	3 章
リソースの枯渇監視	−	○	4 章
運用管理コマンドによる稼働情報の出力	−	○	8 章
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	−	○	9 章

機能名	目的		参照先
	信頼性	運用・保守	
CTM の稼働統計情報の収集※	－	○	10 章

（凡例） ○：対応する。　－：対応しない。

注※

J2EE アプリケーションの実行環境での日常運用を支援する機能です。バッチアプリケーションの実行環境では提供していません。

1.2.2 システムの保守を支援する機能

システムの保守を支援する機能の目的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 システムの保守を支援する機能の目的

機能名	目的		参照先
	信頼性	運用・保守	
コンソールログの出力	－	○	11 章

（凡例） ○：対応する。　－：対応しない。

1.2.3 J2EE アプリケーションの運用機能

J2EE アプリケーションの運用機能の目的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 J2EE アプリケーションの運用機能の目的

機能名	目的		参照先
	信頼性	運用・保守	
J2EE アプリケーションの実行時間の監視とキャンセル	－	○	5.3
J2EE アプリケーションのサービスの閉塞	－	○	5.4
J2EE アプリケーションの停止	－	○	5.5
J2EE アプリケーションの入れ替え	－	○	5.6
J2EE アプリケーションからのネットワークリソースへのアクセス	－	○	5.7

（凡例）　○：対応する。　　－：対応しない。

1.2.4　システムの監査を支援する機能

システムの監査を支援する機能の目的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 1-5　システムの監査を支援する機能の目的

機能名	目的		参照先
	信頼性	運用・保守	
監査ログ出力機能	○	○	6 章
データベース監査証跡連携機能	○	○	7 章

（凡例）　○：対応する。

1.2.5　JP1 連携による運用管理機能

JP1 連携による運用管理機能の目的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 1-6　JP1 連携による運用管理機能の目的

機能名	目的		参照先
	信頼性	運用・保守	
JP1 と連携したシステムの運用	－	○	12 章
システムの集中監視（JP1/IM との連携）	－	○	13 章
ジョブによるシステムの自動運転（JP1/AJS との連携）	－	○	14 章

（凡例）　○：対応する。　　－：対応しない。

1.2.6　クラスタソフトウェアとの連携による系切り替え機能

クラスタソフトウェアとの連携による系切り替え機能の目的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 1-7　クラスタソフトウェアとの連携による系切り替え機能の目的

機能名	目的		参照先
	信頼性	運用・保守	
クラスタソフトウェアとの連携	○	○	15 章
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	○	○	16 章
1:1 系切り替えシステム	○	○	17 章
相互系切り替えシステム	○	○	18 章
N:1 リカバリシステム	○	○	19 章

（凡例）　○：対応する。

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，各機能の説明について，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　Component Container 管理者によるシステムの運用（UNIX の場合）

UNIX の場合，J2EE サーバやバッチサーバのセットアップ，起動・停止などはスーパーユーザで実施しますが，スーパーユーザ以外の一般ユーザでも権限を与えることでこれらの操作ができるようになります。このユーザを **Component Container 管理者**といいます。Component Container 管理者を設定することで，スーパーユーザ以外のユーザを運用管理者にできます。

Component Container 管理者の設定は，アプリケーションサーバの構成ソフトウェアである Component Container をインストールしたあとに実施します。なお，Component Container 管理者の設定後，Component Container 管理者ができる操作は，一部の操作を除き，スーパーユーザでは操作できなくなるので注意してください。

Component Container 管理者を設定した場合に，スーパーユーザおよび Component Container 管理者が実施できる操作について，次の表に示します。

表 1-8　スーパーユーザおよび Component Container 管理者が実施できる操作

操作	スーパーユーザ	Component Container 管理者
Component Container のインストール	○	×
Component Container 管理者のセットアップ（cjenvsetup コマンド）	○	×
作業ディレクトリおよびユーザ定義ファイルの移行（cjenvupdate コマンド）	×	○
J2EE サーバまたはバッチサーバのセットアップ／アンセットアップ（cjsetup コマンド）	×	○
リバースプロキシの設定	×	○
J2EE サーバまたはバッチサーバの起動／停止（cjstartsv コマンド，cjstopsv コマンド）	×	○
サーバ管理コマンドの実行	○	○
スレッドダンプの取得（cjdumpsv コマンド）	○	○
OS 状態情報の取得（cjgetsysinfo コマンド）	○	○
バッチ実行コマンド（cjexecjob コマンド）※	○	○
バッチ強制停止コマンド（cjkilljob コマンド）※	○	○
バッチ一覧表示コマンド（cjlistjob コマンド）※	○	○

（凡例）　○：できる。　　×：できない。

注※　バッチアプリケーションの場合に使用できるコマンドです。

参考

Component Container 管理者を設定しない場合は，スーパーユーザが運用管理者となるため，表中の Component Container 管理者の操作はスーパーユーザが実施します。

1.5　アプリケーションサーバ 11-20 での主な機能変更

この節では，アプリケーションサーバ 11-20 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 11-20 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。

1.5.1　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 1-9　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

2 システムの起動と停止

この章では，日常運用で行うシステムの起動・停止，および障害発生時の自動再起動について説明します。

2.1　この章の構成

この章では，日常運用で行うシステムの起動と停止や，障害発生時の自動再起動の仕組み，およびシステムの稼働監視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 2-1　この章の構成（システムの起動と停止）

分類	タイトル	参照先
解説	日常運用での起動と停止	2.2
	論理サーバの起動・停止の仕組み	2.3
	障害発生時の自動再起動	2.4
	システムの稼働監視	2.5
設定	システムの起動と停止の設定	2.6

注　「実装」,「運用」,「注意事項」について，この機能固有の説明はありません。

2.2　日常運用での起動と停止

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ためには，アプリケーションサーバの構成ソフトウェアの複数のサーバプロセスを適切な順序で起動する必要があります。システムの構成によっては，同じ種類のサーバプロセスを複数起動することもあります。また，サービスを開始するためには，J2EE アプリケーションの開始も必要です。

アプリケーションサーバのシステムでは，J2EE サーバや Web サーバで構成される業務サービスを提供する，閉じた部分系を**サービスユニット**として管理します。Smart Composer 機能のコマンドを使用すると，論理サーバをサービスユニット単位で，またはすべてのサービスユニットを一括で起動・停止できます。また，Web システム内のすべてのサービスユニットを一括で起動・停止することもできます。

Smart Composer 機能では，運用管理に Management Server を使用するため，次の流れでシステムを起動します。停止の場合は，起動と逆の流れになります。論理サーバの起動と停止の仕組みについては，「2.3 論理サーバの起動・停止の仕組み」を参照してください。また，システムの起動・停止の設定については，「2.6　システムの起動と停止の設定」を参照してください。

1. 運用管理エージェントの起動
2. Management Server の起動
3. サービスユニット（論理サーバ）の起動（Smart Composer 機能のコマンド）
4. リソースアダプタの開始（サーバ管理コマンド）
5. J2EE アプリケーションの開始（サーバ管理コマンド）

Smart Composer 機能を使用したシステムの起動・停止方法については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の次の個所を参照してください。

- 4.1.24　システムを起動する（CUI 利用時）
- 4.1.34　システムを停止する（CUI 利用時）

2.3　論理サーバの起動・停止の仕組み

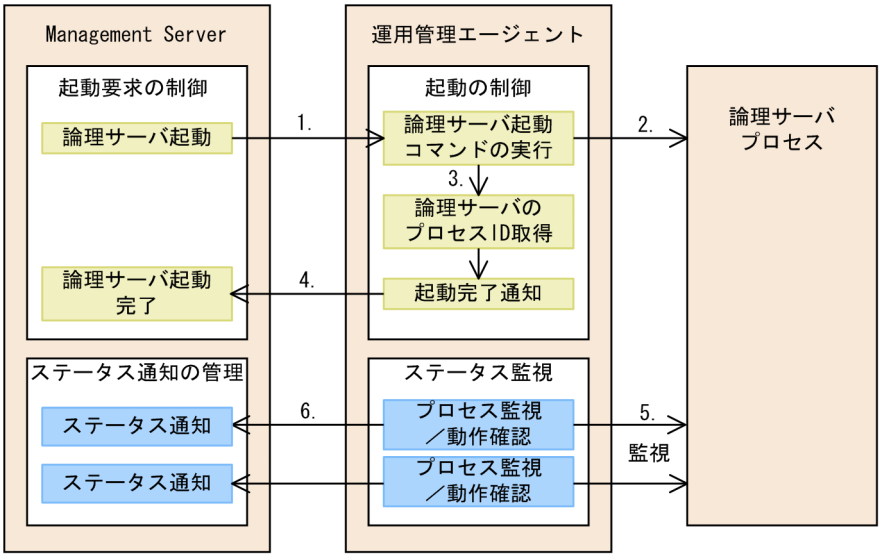
アプリケーションサーバでは，Management Server を利用して論理サーバを起動・停止，および稼働確認を行います。

ここでは，Management Server を利用した論理サーバの起動と稼働確認の仕組み，および停止の仕組みについて説明します。

2.3.1　論理サーバの起動と稼働確認

論理サーバの起動と稼働確認について次の図に示します。

図 2-1　論理サーバの起動と稼働確認



1. Management Server は運用管理エージェントに論理サーバの起動要求を出します。
2. 運用管理エージェントは，論理サーバの起動コマンドを実行して，要求された論理サーバを起動します。
3. 運用管理エージェントで論理サーバプロセスのプロセス ID を取得します。
4. 運用管理エージェントは，Management Server に論理サーバの起動完了を通知します。
5. 運用管理エージェントは，論理サーバのプロセス監視および動作確認を実施します。
   論理サーバプロセスのプロセス ID を使用してプロセスがあるかどうかを確認し，プロセスの存在が確認できると，論理サーバの動作確認を実施します。
6. 運用管理エージェントは，論理サーバのステータスを Management Server に通知します。

論理サーバの種類ごとのプロセス確認および動作確認の内容を，次の表に示します。なお，表中の動作確認の項目がない論理サーバについては，プロセスの存在確認だけが実施されます。

表 2-2　論理サーバの稼働確認方法

論理サーバの種類		プロセスの起動方法※	論理サーバの稼働確認方法	
			プロセス存在確認	動作確認
論理パフォーマンストレーサ		間接起動	パフォーマンストレーサの提供コマンドから取得したプロセス ID でプロセスの存在を確認する。	－
論理スマートエージェント		直接起動	起動コマンドのプロセス ID でプロセスの存在を確認する。	－
論理ネーミングサービス		直接起動	起動コマンドのプロセス ID でプロセスの存在を確認する。	ルートコンテキストを取得できることを確認する。
論理 CTM ドメインマネジャ		間接起動	CTM ドメインマネジャの提供コマンドで取得したプロセス ID でプロセスの存在を確認する。	－
論理 CTM	ネーミングサービス	直接起動	起動コマンドのプロセス ID でプロセスの存在を確認する。	ルートコンテキストを取得できることを確認する。
	CTM デーモン	間接起動	CTM デーモンの提供コマンドで取得したプロセス ID でプロセスの存在を確認する。	－
論理 J2EE サーバ		直接起動	起動コマンドのプロセス ID でプロセスの存在を確認する。	RMI による呼び出しに応答することを確認する。
論理 Web サーバ		間接起動	Web サーバが生成する httpd.pid ファイルから取得した制御プロセスのプロセス ID でプロセスの存在を確認する。	HTTP Server 動作確認用 URL に対して HTTP アクセスで正しいレスポンスが受信できるかどうかを確認する。応答は Web サーバのサーバプロセスが行う。
論理ユーザサーバ		直接起動	起動コマンドのプロセス ID でプロセスの存在を確認する。	論理ユーザサーバ定義ファイルに isAlive コマンドが定義されている場合は，isAlive コマンドを使用してプロセスが稼働しているかを確認する。定義されていない場合は，動作確認は成功したものと見なす。
		間接起動	getProcessID コマンドで取得したプロセス ID でプロセスの存在を確認する。	論理ユーザサーバ定義ファイルに isAlive コマンドが定義されている場合は，isAlive コマンドを使用する。定義されていない場合は，動作確認は成功したものと見なす。

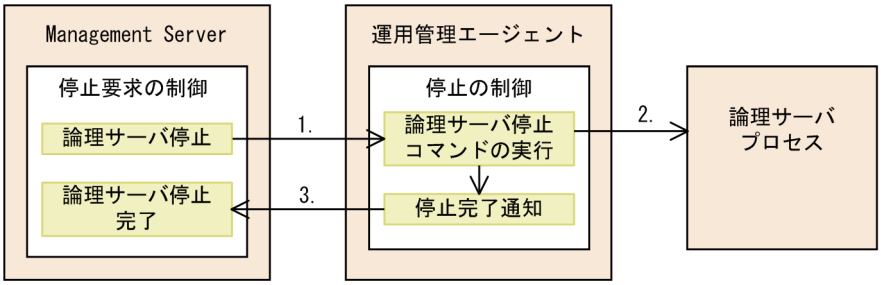
（凡例）　－：なし

注※　直接起動では，実行したコマンドそのものが監視対象となります。間接起動では，コマンドが起動したサービスやプロセスが監視対象となります。コマンドを実行して監視対象のサービスやプロセスが起動されると，コマンドは終了します。

2.3.2　論理サーバの停止

論理サーバの停止について次の図に示します。

図 2-2　論理サーバの停止



1. Management Server は運用管理エージェントに論理サーバの停止要求を出します。
2. 運用管理エージェントは，論理サーバの停止コマンドを実行して，要求された論理サーバを停止します。
3. 運用管理エージェントは Management Server に論理サーバの停止完了を通知します。

2.4　障害発生時の自動再起動

障害が発生した場合に，Management Server，運用管理エージェント，および論理サーバを自動再起動できます。ここでは，プロセスごとに自動再起動について説明します。

2.4.1　Management Server の自動再起動

ここでは，障害が発生した場合の Management Server の自動再起動について説明します。

障害が発生した場合に，停止した Management Server を自動再起動できます。Management Server のプロセスが障害によってダウンした場合などに自動再起動されるため，運用を継続できるようになります。また，自動再起動を設定すると，ホストの起動と同時に Management Server も自動起動するようになります。自動再起動の設定には，mngautorun コマンドを使用します。自動再起動の設定方法については，「2.6.4　自動再起動の設定」を参照してください。

注意事項

Management Server が停止している場合で，稼働中の論理サーバのプロセスが異常終了またはハングアップを検知すると，そのあと Management Server が起動しても，自動停止処理や自動再起動処理は実行されません。

2.4.2　運用管理エージェントの自動再起動

ここでは，障害が発生した場合の運用管理エージェントの自動再起動について説明します。

障害が発生した場合に，停止した運用管理エージェントを自動再起動できます。運用管理エージェントのプロセスが障害によってダウンした場合などに自動再起動されるため，運用を継続できるようになります。また，自動再起動を設定すると，ホストの起動と同時に運用管理エージェントも自動起動するようになります。自動再起動の設定には，mngautorun コマンドを使用します。自動再起動の設定方法については，「2.6.4　自動再起動の設定」を参照してください。

2.4.3　論理サーバの自動再起動

ここでは，障害が発生した場合の論理サーバの自動再起動について説明します。

障害が発生した場合，Management Server を使用して，停止した論理サーバを自動再起動できます。

自動再起動は，Management Server によって正しく起動された論理サーバから，起動後の状態として「障害」が検出された場合に実行されます。「障害」とは，停止要求を受け付けていないのに論理サーバが停止したことが検出された状態です。

運用管理エージェントでは，論理サーバに対してプロセス監視および動作確認をしています。プロセスのダウン，およびハングアップなどの論理サーバの障害を検出すると，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，障害検知時コマンドおよび snapshot ログ収集を実行してトラブルシューティング用の資料を収集したあとで，論理サーバを自動再起動します。

ユーザからの起動要求に対する起動処理の途中で障害が発生した場合は，自動再起動をしないでユーザに起動の失敗が通知されます。

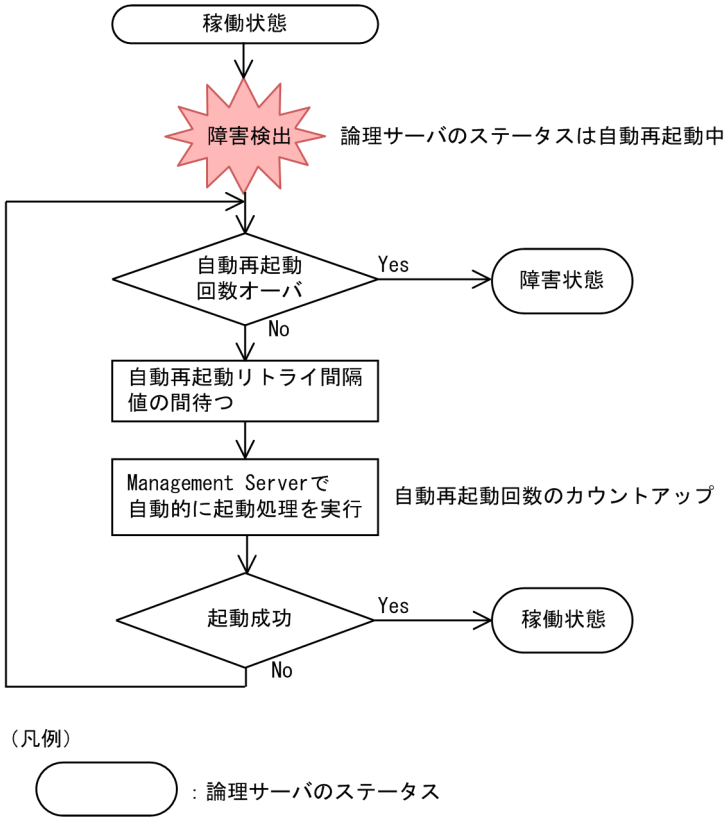
自動再起動は，システムの構築時に設定した，自動再起動回数および自動再起動リトライ間隔に従って実行されます。なお，自動再起動回数として「0 回」を設定している場合，自動再起動は実行されません。

自動再起動回数を超えたときに，<Manager のログ出力ディレクトリ>/mngsvr<ファイル面数>.log にメッセージが出力され，論理サーバのステータスは障害状態になります。

mserver.properties（Management Server 環境設定ファイル）の com.cosminexus.mngsvr.logical_server_abnormal_stop.exit キーに true を設定している場合，論理サーバのステータスが異常停止状態に遷移したタイミングで Management Server は停止します。異常停止状態とは自動再起動回数を超えたとき，または自動再起動回数の設定が 0 で障害を検知したとき，論理サーバが停止した状態です。

障害発生時の自動再起動の流れと論理サーバのステータスについて次の図に示します。

図 2-3　障害発生時の自動再起動



なお，前提となる論理サーバで自動再起動をした場合には，前提となる論理サーバの自動再起動の完了後に，該当する論理サーバを前提として設定している論理サーバも再起動されます。

ポイント

障害検出について

論理サーバの稼働状態の確認は，運用管理エージェントでの論理サーバのプロセス監視および動作確認で実施しています。例えば，プロセス監視では，論理サーバプロセスのプロセス ID が存在するかどうかを確認します。プロセス ID が存在しない場合は，運用管理エージェントはプロセスダウンを検知し，Management Server に異常を通知します。

なお，プロセス監視および動作確認の内容は，論理サーバの種類によって異なります。詳細については，「2.3　論理サーバの起動・停止の仕組み」を参照してください。

注意事項

論理サーバの起動に成功した直後に論理サーバのプロセスが終了すると，障害検出と自動再起動が繰り返し実行されることがあります。この場合，adminagent.properties の adminagent.<サーバ種別>.watch.start_time キー（論理サーバの起動コマンドを実行してから動作確認を開始するまでの時間）に，論理サーバのプロセスが起動されてから終了するまでの時間以上の値を設定することで回避できます。

論理 J2EE サーバは，J2EE アプリケーションの起動時間を考慮して J2EE サーバの起動監視時間を設定してください。論理サーバの自動再起動時に J2EE アプリケーションの起動に時間が掛かり，自動再起動に失敗するおそれがあります。J2EE サーバ起動後に J2EE アプリケーションを起動する運用であっても，論理サーバの自動再起動前に J2EE アプリケーションが開始されていた場合，自動再起動時に J2EE サーバの起動と同時に J2EE アプリケーションが開始されます。ただし，論理 J2EE サーバの環境設定で「コマンドオプションの追加 」※に-nostartapp を指定して運用している場合は，論理サーバの自動再起動時に J2EE サーバの起動と同時に J2EE アプリケーションが開始されません。

注※　SmartComposer の場合，additional.startcmd パラメタで指定します。

2.5　システムの稼働監視

この節では，システム運用時に必要な，論理サーバのステータスの監視やシステムの稼働情報の監視について説明します。

2.5.1　システムの運用状況の監視

システムの運用を開始したら，論理サーバのステータスやシステムの稼働情報を適宜確認して，安定した稼働状態を保っていくことが必要です。稼働情報に出力されたシステム性能を基に，チューニングが必要かどうかも判断する必要があります。

ここでは，システムの運用状況を監視する作業について説明します。システムの運用監視について次の表に示します。

表 2-3　システムの運用監視

作業内容	手段	作業概要	参照先マニュアル	参照個所
サービスユニットのステータス監視	Smart Composer 機能のコマンド（cmx_list_status コマンド）	Web システム中のサービスユニットのステータスを確認して，システムが正常に稼働しているかどうかを監視できます。	このマニュアル	2.5.2(1)
論理サーバのステータス監視	運用管理コマンド（mngsvrutil）	システムを構成するホストおよびサーバのステータスを確認して，システムが正常に稼働しているかどうかを監視できます。この情報を基に，必要に応じて起動，停止，再起動ができます。 論理サーバのステータスは，次の単位で確認できます。 • 運用管理ドメイン単位 • ホスト単位 • 論理サーバ単位		2.5.2(2)
システムの稼働監視	稼働情報収集機能※1※2	J2EE サーバまたはバッチサーバの稼働状態を監視して，稼働情報ファイルに出力できます。 システムの性能を監視することで，システム内の問題個所を発見したり，パフォーマンスチューニングに役立てたりすることができます。		3.2
	運用管理コマンド（mngsvrutil）	J2EE サーバまたはバッチサーバの稼働状態を監視して，CSV 形式または SNMP 連携用形式のファイルに出力できます。		8.2

作業内容	手段	作業概要	参照先マニュアル	参照個所
		システムの性能を監視することで，システム内の問題個所を発見したり，パフォーマンスチューニングに役立てたりすることができます。		
	リソース枯渇監視機能 ※1※3	リソースの使用率または使用数の推移を監視して，使用率または使用数がしきい値を超えた場合，メッセージを出力できます。リソースの使用率または使用数の推移は，リソース枯渇監視情報として定期的に出力できます。しきい値を超えた場合，出力されているリソース枯渇監視情報を基に原因を分析して適切に対処することで，トラブルの発生を未然に防げます。また，リソース枯渇監視情報は，トラブル発生後の要因の解析にも役立ちます。		4.2
	運用管理コマンド（mngsvrutil）	運用管理ドメイン内の CTM※4 の稼働統計情報を確認します。次の単位で確認できます。 • 運用管理ドメイン内のすべての CTM • 運用管理ドメイン内の指定した CTM，または指定したホスト内の CTM		10.2
トランザクション情報の確認	サーバ管理コマンド（cjlisttrn）	稼働中の J2EE サーバでのトランザクションの状態や，停止中の J2EE サーバでの未決着のトランザクションの有無などの情報を確認します。	機能解説 基本・開発編(コンテナ共通機能)	3.15.9

注※1　システム構築時に設定した一定の間隔ごとにファイルが出力されるため，サーバ管理コマンドや運用管理コマンドなどのツールを使用して出力する必要はありません。

注※2　稼働情報ファイルに出力される FullGC 回数，または URL グループ単位の実行待ちリクエスト数がしきい値を超えたときに，メッセージが出力されます（**しきい値イベント**）。このメッセージを利用して **Management イベント**を発行し，対処を **Management アクション**として自動化できます。しきい値イベントの設定については，「3.4.4　実行環境での設定（J2EE サーバの設定）」を参照してください。また，Management イベントによる処理の自動実行の設定については，「9.4　Management イベントによる処理の自動実行の設定」を参照してください。

注※3　リソースの使用状況がしきい値を超えた場合，**Management イベント**を発行できます。Management イベントを利用すると，対処を **Management アクション**として自動化できます。Management イベントによる処理の自動実行の設定については，「9.4　Management イベントによる処理の自動実行の設定」を参照してください。

注※4　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「2.2.1　製品と構成ソフトウェアの対応」を参照してください。

2.5.2　ステータスの監視

アプリケーションサーバでは，ステータス監視によって，システムが正常に稼働しているかどうかを監視できます。ステータス監視には次の 2 種類があります。

- サービスユニットのステータス監視
  サービスユニットのステータスを監視できます。
- 論理サーバのステータス監視
  論理サーバのステータスを監視できます。

> **参考**
>
> サーバ管理コマンドを使用して，サーバ上で動作するアプリケーションのステータス監視，および稼働中の J2EE サーバやバッチサーバのトランザクション情報の表示などができます。

(1)　サービスユニットのステータス監視

Smart Composer 機能のコマンド（cmx_list_status コマンド）を使用して，サービスユニットの稼働状態（ステータス）を監視できます。cmx_list_status 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」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。サービスユニットのステータス監視では，特定のユニット，または Web システム中のすべてのユニットの稼働状態を CSV 形式にファイル出力できます。

(2)　論理サーバのステータス監視

論理サーバのステータス監視では，運用管理コマンド（mngsvrutil）を使用して論理サーバの稼働状態（ステータス）を監視できます。ステータス情報は標準出力に出力したり，CSV 形式または SNMP 連携様式にファイル出力したりできます。

ここでは，論理サーバの稼働状態（ステータス）の監視方法と，監視できる項目について説明します。

> **参考**
>
> Management Server および運用管理エージェントのステータスを確認するには，次の方法があります。
>
> - 運用管理コマンド（mngsvrutil）のサブコマンド「check」
> - adminagentcheck コマンド
>
> 運用管理コマンド，および adminagentcheck 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)」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 コマンド編」の「adminagentcheck（運用管理エージェントの稼働確認)」を参照してください。

(a) ステータスの監視方法

論理サーバのステータスは，運用管理コマンド（mngsvrutil）で監視できます。

論理サーバのステータスを監視するには，運用管理コマンド（mngsvrutil）にサブコマンド「list」を指定して実行します。これによって，論理サーバのステータス情報を標準出力に出力したり，CSV 形式または SNMP 連携用形式にファイル出力したりできます。

「list」の引数に指定する値によって，ステータスを出力する対象を指定できます。

運用管理コマンドおよびそのサブ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「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。

次に，実行形式と実行例を示します。

- 運用管理ドメイン内の論理サーバ名とステータス情報を出力する場合

  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> list status
  ```

  実行例

  ```
  mngsvrutil -m mnghost -u user01 -p pw1 list status
  ```

- 指定した論理サーバにインポートされている J2EE アプリケーション名とステータス情報を出力する場合

  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 <論理サーバ名> list appStatus
  ```

  実行例

  ```
  mngsvrutil -m mnghost -u user01 -p pw1 -t J2EEserver1 list appStatus
  ```

(b) ステータス監視で確認できる項目

ステータス監視では，論理サーバの起動／停止状態が確認できます。ステータスは，すべての論理サーバで確認できます。

ステータスは，次の単位で確認できます。

- 運用管理ドメイン単位

  運用管理ドメインに含まれるすべての論理サーバのステータスをまとめて確認できます。

- ホスト単位

  選択したホスト内のすべての論理サーバのステータスをまとめて確認できます。

- 論理サーバ単位

  選択した論理サーバのステータスを確認できます。種類ごとに確認することも，個々の論理サーバだけを確認することもできます。

確認できるステータスの種類と意味を次に示します。

表 2-4　稼働状況のステータスの種類と意味

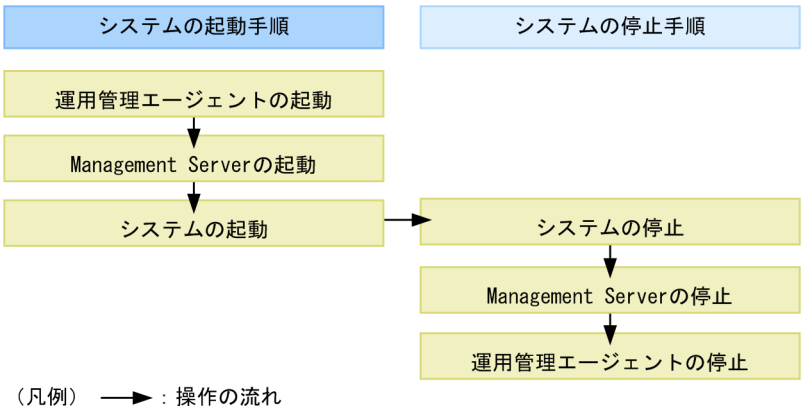
稼働状況のステータス	説明
停止	初期状態および停止要求を受け付け停止処理が完了したあとの，論理サーバが停止した状態です。または，通信障害の回復後に運用管理エージェントが停止状態であることを確認できた状態です。
起動中	起動要求を受け付けてから論理サーバが稼働状態になるまでの起動処理中の状態です。
稼働中	起動要求を受け付け起動処理が完了したあとの，論理サーバが稼働している状態です。または，通信障害の回復後に運用管理エージェントが稼働状態であることを確認できた状態です。
停止中	停止要求を受け付けてから論理サーバが停止状態になるまでの停止処理中の状態です。
強制停止中	強制停止要求を受け付けてから論理サーバが停止状態になるまでの強制停止処理中の状態です。
異常停止	停止要求を受け付けていない状態で，論理サーバの停止を検出した状態です。
回復中	異常停止状態で起動要求を受け付けてから稼働状態になるまでの，論理サーバの起動処理中の状態です。
通信障害	運用管理エージェントとの通信で障害が発生し，ステータスの表示ができない状態です。
自動停止中	運用管理エージェントから論理サーバの異常通知（プロセスはあるが動作していない状態の通知）を受けて論理サーバの強制停止中の状態です。
自動再起動中	自動再起動指定ありの論理サーバで，稼働中状態に運用管理エージェントから論理サーバの停止通知を受けて自動再起動処理中の状態です。
計画停止中	計画停止要求を受け付けてから，論理 Web サーバが停止状態になるまでの計画停止処理中の状態です。このステータスに遷移するのは，論理 Web サーバの場合だけです。

2.6　システムの起動と停止の設定

この節では，構築したシステムを起動，停止する手順と，自動で起動，停止する場合の設定について説明します。

構築したシステムのプログラムやサーバは，次の図に示す手順で起動，停止します。

図 2-4　システムの起動と停止の手順



プログラムや，Web システム内のサーバは，自動または手動で起動，停止できます。システムの運用方法に合わせて起動，停止の方法を検討し，必要な設定をしてください。

2.6.1　システムの起動

通常の運用では，運用管理エージェントと Management Server は自動起動する方法をお勧めします。運用管理エージェントと Management Server に自動起動を設定すると，運用管理エージェントの起動から Management Server の起動までの一連の操作を，ホスト起動と同時に処理できるようになります。自動起動の設定方法については，「2.6.3　自動起動の設定」を参照してください。

プログラムやサーバの起動手順を次に示します。

1. 運用管理エージェントの起動
2. Management Server の起動
3. システムの起動

プログラムやサーバの起動方法を次に示します。

(1)　運用管理エージェントの起動

手動起動する場合は，アプリケーションサーバのホストごとに，adminagentctl コマンドを使用して運用管理エージェントを起動します。

コマンドの処理と運用管理エージェントの起動処理の状態は，同期させることをお勧めします。この場合，コマンド実行時に同期実行オプション（-sync オプション）を指定します。このオプションを指定すると，コマンドの処理が終了した時点で運用管理エージェントは稼働状態となります。また，同期実行時には，コマンドを実行してから運用管理エージェントが稼働状態となるまでの待ち時間（同期実行のタイムアウト時間）を，-timeout オプションで指定できます。

起動方法を，OS ごとに示します。

Windows の場合

```
<Application Serverのインストールディレクトリ>¥manager¥bin¥adminagentctl start -sync -timeou
t <同期実行のタイムアウト時間>
```

参考

Windows の場合はサービスからも起動できます。

UNIX の場合

```
# /opt/Cosminexus/manager/bin/adminagentctl start -sync -timeout <同期実行のタイムアウト時間
>
```

参考

UNIX の場合，adminagentctl コマンドで-daemon オプションを指定，または daemon コマンドを使用すると，運用管理エージェントをデーモンプロセスとして起動できます。daemon コマンドを使用した場合，コマンドの処理と運用管理エージェントの起動処理の状態は同期できません。

(2) Management Server の起動

手動起動する場合は，Management Server をセットアップしたホストで，mngsvrctl コマンドに，引数「start」を指定して Management Server を起動します。

注意事項

名前解決のできないホスト名または存在しない IP アドレスを指定してホストを定義した場合，Management Server の起動に時間が掛かることがあります。

コマンドの処理と Management Server の起動処理の状態は，同期させることをお勧めします。この場合，コマンド実行時に同期実行オプション（-sync オプション）を指定します。このオプションを指定すると，コマンドの処理が終了した時点で Management Server は稼働状態となります。また，同期実行時には，コマンドを実行してから Management Server が稼働状態となるまでの待ち時間（同期実行のタイムアウト時間）を，-timeout オプションで指定できます。

起動方法を，OS ごとに示します。

Windows の場合

```
<Application Serverのインストールディレクトリ>¥manager¥bin¥mngsvrctl start -sync -timeout <同期実行のタイムアウト時間>
```

参考

Windows の場合はサービスからも起動できます。

UNIX の場合

```
# /opt/Cosminexus/manager/bin/mngsvrctl start -sync -timeout <同期実行のタイムアウト時間>
```

参考

UNIX の場合，mngsvrctl コマンドで-daemon オプションを指定，または daemon コマンドを使用すると，Management Server をデーモンプロセスとして起動できます。daemon コマンドを使用した場合，コマンドの処理と Management Server の起動処理の状態は同期できません。

(3)　システムの起動

Smart Composer 機能のコマンドで，Web システムまたはサービスユニット単位に一括起動します。Web システム単位で起動すると，Web システム内にあるすべてのサービスユニット中の全論理サーバを起動します。

参考

論理サーバの起動順序は，デフォルトの設定を使用することをお勧めします。なお，論理サーバの起動順序は，簡易構築定義ファイルで各論理サーバの<configuration>タグ内に mstartup.no パラメタで設定できます。

2.6.2　システムの停止

プログラムやサーバの停止手順と停止方法を次に示します。なお，UNIX の場合は，運用管理エージェントと Management Server を自動停止できます。自動停止する場合は，ホストと同時に停止するようにします。自動停止の設定方法については，「2.6.5　自動停止の設定」を参照してください。

プログラムやサーバの停止手順を次に示します。

1. システムの停止

2. Management Server の停止
3. 運用管理エージェントの停止

プログラムやサーバの停止方法を次に示します。

(1) システムの停止

Smart Composer 機能のコマンドで，Web システムまたはサービスユニット単位に一括停止します。

(2) Management Server の停止

手動停止する場合は，Management Server をセットアップしたホストで，mngsvrctl コマンドに，引数「stop」を指定して Management Server を停止します。

コマンドの処理と Management Server の停止処理の状態は，同期させることをお勧めします。この場合，コマンド実行時に同期実行オプション（-sync オプション）を指定します。このオプションを指定すると，コマンドの処理が終了した時点で Management Server は停止状態となります。また，同期実行時には，コマンドを実行してから Management Server が停止状態となるまでの待ち時間（同期実行のタイムアウト時間）を，-timeout オプションで指定できます。

停止方法を，OS ごとに示します。

Windows の場合

```
<Application Serverのインストールディレクトリ>¥manager¥bin¥mngsvrctl stop -sync -timeout <同期実行のタイムアウト時間>
```

参考

Windows の場合はサービスからも停止できます。

UNIX の場合

```
# /opt/Cosminexus/manager/bin/mngsvrctl stop -sync -timeout <同期実行のタイムアウト時間>
```

(3) 運用管理エージェントの停止

手動停止する場合は，アプリケーションサーバのホストごとに，adminagentctl コマンドを使用して運用管理エージェントを停止します。

コマンドの処理と運用管理エージェントの停止処理の状態は，同期させることをお勧めします。この場合，コマンド実行時に同期実行オプション（-sync オプション）を指定します。このオプションを指定すると，コマンドの処理が終了した時点で運用管理エージェントは停止状態となります。また，同期実行時には，コマンドを実行してから運用管理エージェントが停止状態となるまでの待ち時間（同期実行のタイムアウト時間）を，-timeout オプションで指定できます。

停止方法を，OS ごとに示します。

Windows の場合

```
<Application Serverのインストールディレクトリ>¥manager¥bin¥adminagentctl stop -sync -timeout
 <同期実行のタイムアウト時間>
```

参考

Windows の場合はサービスからも停止できます。

UNIX の場合

```
# /opt/Cosminexus/manager/bin/adminagentctl stop -sync -timeout <同期実行のタイムアウト時間>
```

2.6.3　自動起動の設定

ホストの起動と同時に，Management Server および運用管理エージェントを自動起動できます。Management Server および運用管理エージェントの自動起動の設定には，mngautorun コマンドを使用します。mngautorun 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「mngautorun（自動起動および自動再起動の設定／設定解除）」を参照してください。

mngautorun コマンドで設定する，Management Server および運用管理エージェントの自動起動処理は，同期実行させることをお勧めします。この場合，コマンド実行時に同期実行オプション（-sync オプション）を指定します。このオプションを指定すると，自動起動時に，Management Server および運用管理エージェントの起動を同期実行します。また，自動起動時に，Management Server および運用管理エージェントの起動を同期実行するまでの待ち時間（同期実行のタイムアウト時間）を，-timeout オプションで指定できます。

注意事項

09-00 より前に自動起動を設定して運用している場合に，ここで説明する設定を有効にするときは，09-00 より前の自動起動の設定を解除する必要があります。設定変更手順については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「10.3　旧バージョンから 11-20 までの仕様変更の確認」を参照してください。

UNIX の場合，09-00 より前に自動起動を設定して運用しているときは，同期実行を有効にできません。09-00 より前の自動起動の設定を解除し，mngautorun コマンドを使用して自動起動を設定する際に，同期実行を有効にしてください。

mngautorun コマンドの実行形式と実行例を次に示します。

実行形式

```
mngautorun [once] {server|agent|both} [-runlevel <ランレベル>]※ [-sync [-timeout <同期実行のタイムアウト時間>]]
```

注※　ランレベルは，UNIX の場合に指定できます。Windows の場合は指定できません。

実行例

- Management Server をホスト起動時に自動起動する場合

  ```
  mngautorun once server
  ```

- 運用管理エージェントをホスト起動時に自動起動する場合

  ```
  mngautorun once agent
  ```

- Management Server および運用管理エージェントをホスト起動時に自動起動する場合

  ```
  mngautorun once both
  ```

- Management Server および運用管理エージェントをホスト起動時に同期実行で自動起動する場合

  ```
  mngautorun once both -sync
  ```

mngautorun コマンドの実行時に，Management Server および運用管理エージェントに設定される自動起動の内容を次に示します。

(1)　Management Server に設定される自動起動の内容

mngautorun コマンドの実行時に，Management Server に設定される自動起動の内容を OS ごとに説明します。

- Windows の場合

  Management Server のサービスで，ホスト起動時にサービスを起動するように次の内容が設定されます。

  - スタートアップの種類：自動

  また，同一ホスト上で Management Server および運用管理エージェントの両方が起動する場合には，運用管理エージェントが先に起動するように依存関係が設定されます。

- UNIX の場合

  mngautorun コマンドの once オプションを使用して（once オプションおよび respawn オプションの両方を省略した場合も含む）自動起動を設定※し，OS を起動や再起動することで，Management Server を自動起動することができます。

  注※　/etc/inittab ファイルに Management Server の起動コマンドを once オプションを指定して登録します。ただし，Linux の場合は，/etc/systemd/system 下のディレクトリに CoMS.service（Management Server を自動起動，自動停止するサービス）を登録します。

(2) 運用管理エージェントに設定される自動起動の内容

mngautorun コマンドの実行時に，運用管理エージェントに設定される自動起動の内容を OS ごとに説明します。

参考

自動起動処理では，運用管理エージェントを起動する前に，運用管理エージェント自動起動用設定ファイル（/opt/Cosminexus/manager/config/AdminAgentrc）を読み込みます。このファイルに次の設定を記述することで，運用管理エージェントから起動する論理サーバに設定を引き継ぐことができます。

- リソースの制御設定（ulimit）
- ファイル作成時のパーミッション（umask）
- 環境変数

運用管理エージェント自動起動用設定ファイルにはこれらの設定以外を記述しないでください。運用管理エージェント自動起動用設定ファイルにこれらの設定以外を記述した場合の動作は保証しません。

運用管理エージェント自動起動用設定ファイルはシェルスクリプトとして実行されます。このため，運用管理エージェント自動起動用設定ファイルの記載内容を変更する場合は十分な動作確認を実施してください。

- Windows の場合

  運用管理エージェントのサービスで，ホスト起動時にサービスを起動するように次の内容が設定されます。

  - スタートアップの種類：自動

  また，同一ホスト上で Management Server および運用管理エージェントの両方が起動する場合には，運用管理エージェントが先に起動するように依存関係が設定されます。

- UNIX の場合

  mngautorun コマンドの once オプションを使用して（once オプションおよび respawn オプションの両方を省略した場合も含む）自動起動を設定※し，OS を起動や再起動することで，運用管理エージェントを自動起動することができます。

  注※　/etc/inittab ファイルに運用管理エージェントの起動コマンドを once オプションを指定して登録します。ただし，Linux の場合は，/etc/systemd/system 下のディレクトリに CoAA.service（運用管理エージェントを自動起動，自動停止するサービス）を登録します。

2.6.4　自動再起動の設定

障害が発生した場合に，停止した Management Server および運用管理エージェントを自動再起動できます。自動再起動を設定すると，自動起動の設定も有効になります。Management Server および運用管理エージェントの自動再起動の設定には，mngautorun コマンドを使用します。mngautorun 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「mngautorun（自動起動および自動再起動の設定／設定解除）」を参照してください。

Windows の場合，mngautorun コマンドで設定する，Management Server および運用管理エージェントの自動再起動処理は，同期実行させることをお勧めします。この場合，コマンド実行時に同期実行オプション（-sync オプション）を指定します。このオプションを指定すると，自動再起動時に，Management Server および運用管理エージェントの起動を同期実行します。また，自動再起動時に，Management Server および運用管理エージェントの起動を同期実行するまでの待ち時間（同期実行のタイムアウト時間）を，-timeout オプションで指定できます。

UNIX の場合，自動再起動処理に対して同期実行オプションを指定できません。

注意事項

09-00 より前に自動起動を設定して運用している場合

ここで説明する設定を有効にするときは，09-00 より前の自動起動の設定を解除する必要があります。設定変更手順については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「10.3　旧バージョンから 11-20 までの仕様変更の確認」を参照してください。

JavaVM の異常終了時に自動再起動するための設定

JavaVM の異常終了時にユーザダンプを取得したあと，アプリケーションサーバの自動再起動を正しく動作させるには，次のどちらかのレジストリキーにレジストリ値（DontShowUI ：1）を設定して，ユーザダンプ取得時の応答要求を抑止することをお勧めします。

「¥¥HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥Windows Error Reporting」

「¥¥HKEY_CURRENT_USER¥SOFTWARE¥Microsoft¥Windows¥Windows Error Reporting」（上記のレジストリキーがない場合に使用します）

mngautorun コマンドの実行形式と実行例を次に示します。

実行形式

```
mngautorun respawn {server|agent|both} [-runlevel <ランレベル>][※1] [-sync[※2] [-timeout <同期実行のタイムアウト時間>[※2]]]
```

注※1　ランレベルは，UNIX の場合に指定できます。Windows の場合は指定できません。

注※2　同期実行オプションおよび同期実行のタイムアウト時間は，Windows の場合に指定できます。UNIX の場合は指定できません。

実行例

- Management Server を自動再起動する場合

  ```
  mngautorun respawn server
  ```

- 運用管理エージェントを自動再起動する場合

  ```
  mngautorun respawn agent
  ```

- Management Server および運用管理エージェントを自動再起動する場合

  ```
  mngautorun respawn both
  ```

- Management Server および運用管理エージェントの自動再起動を同期実行する場合※

  ```
  mngautorun respawn both -sync
  ```

注※　Windows の場合に指定できます。UNIX の場合は指定できません。

mngautorun コマンドの実行時に，Management Server および運用管理エージェントに設定される自動再起動の内容を次に示します。

(1) Management Server に設定される自動再起動の内容

mngautorun コマンドの実行時に，Management Server に設定される自動再起動の内容を OS ごとに説明します。

- Windows の場合

  Management Server のサービスで，サービスの回復機能にエラー発生時にサービスを再起動するように次の内容が設定されます。

  - 最初のエラー：サービスを再起動する
  - エラーカウントのリセット：0（エラーカウントをリセットしません）
  - サービスの再起動：0（異常終了後に即時に再起動を実行します）

  なお，「エラーカウントのリセット」，「サービスの再起動」の値は，手動で設定を変更できます。

- UNIX の場合

  mngautorun コマンドの respawn オプションを使用して自動再起動を設定※し，OS を起動や再起動することで Management Server を自動再起動することができます。

  Management Server のサービスを監視し，サービス終了時に再起動するように設定されます。

注※　/etc/inittab ファイルに Management Server の起動コマンドを respawn オプションを指定して登録します。ただし，Linux の場合は，/etc/systemd/system 下のディレクトリに CoMS.service（Management Server を自動起動，自動停止，自動再起動するサービス）を登録します。

注意事項

UNIX の場合は，Management Server のサービスで障害が発生しているかどうかに関係なく，mngsvrctl コマンドなどで Management Server のサービスを停止した場合も自動再起動されます。

(2)　運用管理エージェントに設定される自動再起動の内容

mngautorun コマンドの実行時に，運用管理エージェントに設定される自動再起動の内容を OS ごとに説明します。

参考

自動再起動処理では，運用管理エージェントを起動する前に，運用管理エージェント自動起動用設定ファイル（/opt/Cosminexus/manager/config/AdminAgentrc）を読み込みます。このファイルに次の設定を記述することで，運用管理エージェントから起動する論理サーバに設定を引き継ぐことができます。

- リソースの制御設定（ulimit）
- ファイル作成時のパーミッション（umask）
- 環境変数

運用管理エージェント自動起動用設定ファイルにはこれらの設定以外を記述しないでください。運用管理エージェント自動起動用設定ファイルにこれらの設定以外を記述した場合の動作は保証しません。

運用管理エージェント自動起動用設定ファイルはシェルスクリプトとして実行されます。このため，運用管理エージェント自動起動用設定ファイルの記載内容を変更する場合は十分な動作確認を実施してください。

- Windows の場合

  運用管理エージェントのサービスで，サービスの回復機能にエラー発生時にサービスを再起動するように次の内容が設定されます。

  - 最初のエラー：サービスを再起動する
  - エラーカウントのリセット：0（エラーカウントをリセットしません）
  - サービスの再起動：0（異常終了後に即時に再起動を実行します）

  なお，「エラーカウントのリセット」，「サービスの再起動」の値は，手動で設定を変更できます。

注意事項

親プロセスである adminagentsv プロセスがダウンして自動再起動されたあと，サービスまたはコマンドから運用管理エージェントを停止しても，次のキーの設定は有効ならないため，論理サーバは停止されません。

- ＜Application Server のインストールディレクトリ＞¥manager¥config¥adminagent.properties の adminagent.finalization.stop_servers キー

- UNIX の場合

  mngautorun コマンドの respawn オプションを使用して自動再起動を設定※し，OS を起動や再起動することで運用管理エージェントを自動再起動することができます。

  運用管理エージェントのサービスを監視し，サービス終了時に再起動するように設定されます。

  注※　/etc/inittab ファイルに運用管理エージェントの起動コマンドを respawn オプションを指定して登録します。ただし，Linux の場合は，/etc/systemd/system 下のディレクトリに CoAA.service（運用管理エージェントを自動起動，自動停止，自動再起動するサービス）を登録します。

注意事項

UNIX の場合は，運用管理エージェントのサービスで障害が発生しているかどうかに関係なく，adminagentctl コマンドなどで運用管理エージェントのサービスを停止した場合も自動再起動されます。

2.6.5　自動停止の設定

UNIX の場合は，ホストの停止と同時に Management Server および運用管理エージェントを自動停止できます。

(1)　Management Server の自動停止の設定

Management Server の自動停止を設定する手順を OS ごとに次に示します。

注意事項

Management Server を自動再起動するように設定している場合は，この手順では停止できません。各論理サーバを停止してからホストを停止してください。ただし，Red Hat Enterprise Linux Server 7 以降は，Management Server を自動再起動するように設定している場合でも，この手順で停止できます。

(a) AIX の停止手順

ホスト停止時に Management Server を自動停止させるためには，次に示す作業が必要です。

1. Management Server を停止させるスクリプトファイルを作成します。

2. /etc/rc.shutdown スクリプトに，Management Server を停止させるための処理を追加します。

ここでは，Management Server を停止させるスクリプトファイルの作成方法と，/etc/rc.shutdown スクリプトへの Management Server の停止処理の追加方法について説明します。

Management Server を停止させるスクリプトファイルの作成

Management Server を停止させるスクリプトファイルを，/etc/下に任意のファイル名で作成します（例：/etc/MngSvrStop）。なお，スクリプトファイルの権限は「755」に設定してください。

スクリプトファイルの例を次に示します。

```
#!/bin/sh
BIN_PATH=/opt/Cosminexus/manager/bin
# Management Server停止
if [ -x $BIN_PATH/mngsvrctl ] ; then
  $BIN_PATH/mngsvrctl stop
fi
exit 0
```

この例では，最後の"exit 0"で停止処理に失敗した場合でもシャットダウン処理が中断されないようにしています。エラー発生時にシャットダウンを中断する場合は，それぞれのコマンド実行後にリターンコードをチェックして 0 以外のリターンコードを返すようにしてください。

なお，ホスト停止時に論理サーバを停止させるには，論理サーバを停止させるスクリプトを，Management Server を停止させるスクリプトの前に追加してください。

スクリプトファイルの例を次に示します。

```
#!/bin/sh
BIN_PATH=/opt/Cosminexus/manager/bin
#論理サーバ停止
if [ -x $BIN_PATH/mngsvrutil ] ; then
  $BIN_PATH/mngsvrutil -m mnghost:28080 -u user1 -p user1 -t mnghost -k host -s stop serv
er
fi
# Management Server停止
if [ -x $BIN_PATH/mngsvrctl ] ; then
  $BIN_PATH/mngsvrctl stop
fi
exit 0
```

mngsvrutil コマンドの各種オプションは，運用環境に合わせて指定してください。

/etc/rc.shutdown スクリプトへの Management Server の停止処理の追加

/etc/rc.shutdown スクリプトで実行する処理として，/etc/rc.shutdown スクリプトに Management Server を停止させるためのスクリプトファイルを追加します。同一ホスト上で Management Server と運用管理エージェントを停止する場合，Management Server が運用管理エージェントよりも先に停止するように順番を設定してください。

/etc/rc.shutdown スクリプトへの Management Server の停止処理の追加例を次に示します。なお，この例は，Management Server を停止させるためのスクリプトファイルを/etc/MngSvrStop に保存した場合の例です。

```
if [ -x /etc/MngSvrStop ]; then
  /etc/MngSvrStop
fi
```

(b)　Linux の停止手順

ホスト停止時に Management Server を自動停止させるための設定手順について説明します。また，ホスト停止時に論理サーバを自動停止させるための設定手順についても説明します。

ホスト停止時に Management Server を自動停止させるための設定手順

Management Server を自動起動設定（mngautorun コマンド）すれば，自動停止も行われます。

mngautorun コマンドで「-sync」指定した場合，Management Server の自動停止のタイムアウト時間はデフォルトで 120 秒となります。Management Server の自動停止のタイムアウト時間（-timeout）を設定する場合は次の手順で設定してください。

1. /etc/systemd/system ディレクトリに格納されている CoMS.service ファイルを開き，ExecStop= オプションの指定にタイムアウト時間を追加します。

   指定例を次に示します。

   （例）

   CoMS.service の変更前

   ```
   ExecStop= /opt/Cosminexus/manager/bin/mngsvrctl stop -sync
   ```

   CoMS.service の変更後

   ```
   ExecStop= /opt/Cosminexus/manager/bin/mngsvrctl stop -sync -timeout 120
   ```

   注　背景色付きの太字個所が編集個所になります。また，斜体個所は運用環境に合わせて設定してください。

2. 1.で変更した CoMS.service ファイルを適用するために，次のコマンドを実行します。

   ```
   # systemctl reenable CoMS.service
   ```

   注意事項

   mngautorun コマンドを再度実行すると，この手順で CoMS.service に設定した内容はリセットされます。その場合は再度この手順を実施してください。

また，同一ホスト上で運用管理エージェントと Management Server を停止する場合は，両方を自動起動設定（mngautorun コマンドの both 指定）します。このとき同期起動（mngautorun コマンドの-sync 指定）するように設定した場合は， ホスト停止時またはホスト再起動時に Management Server が運用管理エージェントより先に停止します。

ホスト停止時に論理サーバを自動停止させるための設定手順

ホスト停止時に論理サーバを自動停止させるための設定手順を次に示します。なお，この手順は，Management Server を停止するホストだけで運用管理ドメインを構成していることを前提とした手順です。

1. /etc/systemd/system ディレクトリに格納されている CoMS.service ファイルをテキストエディタで開き，ExecStop=オプションに設定されているコマンド構文をコピーして控えます。
2. 任意の場所およびファイル名で論理サーバと Management Server を停止させるためのスクリプトファイルを作成します（例：/opt/Cosminexus/manager/bin/LS_MNG_stop）。スクリプトファイルの作成例を次に示します。

```
#!/bin/sh

ret=0

/opt/Cosminexus/manager/bin/mngsvrutil -m mnghost:28080 -u user1 -p user1 -t mnghost -
k host -s stop server
ERROR=$?
if [ $ERROR -ne 0 ] ; then
    ret=1
fi

/opt/Cosminexus/manager/bin/mngsvrctl stop -sync
ERROR=$?
if [ $ERROR -ne 0 ] ; then
    ret=1
fi

exit $ret
```

   なお，mngsvrutil コマンドの各種オプションは，運用環境に合わせて指定してください。また，Management Server の停止コマンド構文は，1.で控えたコマンド構文を指定してください。

3. 作成した論理サーバおよび Management Server 停止スクリプトの権限を変更します。

```
# chmod 755 /opt/Cosminexus/manager/bin/LS_MNG_stop
```

4. /etc/systemd/system ディレクトリに格納されている CoMS.service ファイルをテキストエディタで再度開き，ExecStop=オプションの指定を，2.で作成したスクリプトのパスに変更します。

   CoMS.service ファイルの変更例を次に示します。

   変更前（mngautorun -sync 指定ありの場合の内容）

```
ExecStop=/opt/Cosminexus/manager/bin/mngsvrctl stop -sync
```

   変更後（2.で作成したスクリプト場合）

```
ExecStop=/opt/Cosminexus/manager/bin/LS_MNG_stop
```

5. 運用管理エージェントと Management Server を非同期起動（mngautorun both 指定あり，-sync 指定なし）で設定している場合は，CoMS.service ファイルの After=オプションに” CoAA.service”を追加します。

CoMS.service ファイルの変更例を次に示します。

CoMS.service の変更前（mngautorun -sync 指定なしの場合の内容）

```
After=network.target multi-user.target
```

変更後

```
After=CoAA.service network.target multi-user.target
```

6. 変更した CoMS.service ファイルを適用するために，次のコマンドを実行します。

```
# systemctl reenable CoMS.service
```

> **注意事項**
>
> mngautorun コマンドを再度実行すると，この手順で CoMS.service に設定した内容はリセットされます。その場合は再度この手順を実施してください。

ホスト停止時の論理サーバの自動停止を解除する手順

上記で設定したホスト停止時の論理サーバの停止を解除する場合は，次の手順を実施してください。

1. 上記で作成した論理サーバと Management Server を停止するスクリプトファイル（例：/opt/Cosminexus/manager/bin/LS_MNG_stop）をテキストエディタで開き，Management Server の停止コマンド構文をコピーして控えます。
2. /etc/systemd/system ディレクトリに格納されている CoMS.service ファイルを開き，ExecStop=オプションの指定を，1.で控えたコマンド構文に変更します。

   CoMS.service ファイルの変更例を次に示します。

   変更前

   ```
   ExecStop=/opt/Cosminexus/manager/bin/LS_MNG_stop
   ```

   変更後

   ```
   ExecStop=/opt/Cosminexus/manager/bin/mngsvrctl stop -sync
   ```

3. 運用管理エージェントと Management Server を非同期起動（mngautorun both 指定あり，-sync 指定なし）で設定している場合は， CoMS.service ファイルの After=オプションの”CoAA.service”を削除します。

   CoMS.service ファイルの変更例を次に示します。

   変更前

   ```
   After=CoAA.service network.target multi-user.target
   ```

   変更後

   ```
   After=network.target multi-user.target
   ```

4. 変更した CoMS.service ファイルを適用するために，次のコマンドを実行します。

```
# systemctl reenable CoMS.service
```

5. 論理サーバと Management Server を停止するスクリプトファイル（例：/opt/Cosminexus/manager/bin/LS_MNG_stop）を削除してください。

(2) 運用管理エージェントの自動停止の設定

運用管理エージェントの自動停止を設定する手順を OS ごとに次に示します。

注意事項

運用管理エージェントを自動再起動するように設定している場合は，この手順では停止できません。各論理サーバを停止してからホストを停止してください。ただし，Red Hat Enterprise Linux Server 7 以降は，運用管理エージェントを自動再起動するように設定している場合でも，この手順で停止できます。

(a) AIX の停止手順

ホスト停止時に運用管理エージェントを自動停止させるためには，次に示す作業が必要です。

1. 運用管理エージェントを停止させるスクリプトファイルを作成します。
2. /etc/rc.shutdown スクリプトに，運用管理エージェントを停止させるための処理を追加します。

ここでは，運用管理エージェントを停止させるスクリプトファイルの作成方法と，/etc/rc.shutdown スクリプトへの運用管理エージェントの停止処理の追加方法について説明します。

運用管理エージェントを停止させるスクリプトファイルの作成

運用管理エージェントを停止させるスクリプトファイルを，/etc/下に任意のファイル名で作成します（例：/etc/AdminAgentStop）。なお，スクリプトファイルの権限は「755」に設定してください。

スクリプトファイルの例を次に示します。

```
#!/bin/sh
BIN_PATH=/opt/Cosminexus/manager/bin
# 運用管理エージェント停止
if [ -x $BIN_PATH/adminagentctl ] ; then
  $BIN_PATH/adminagentctl stop
fi
exit 0
```

なお，この例では，最後の"exit 0"で停止処理に失敗した場合でもシャットダウン処理が中断されないようにしています。エラー発生時にシャットダウンを中断する場合は，それぞれのコマンド実行後にリターンコードをチェックして 0 以外のリターンコードを返すようにしてください。

/etc/rc.shutdown スクリプトへの運用管理エージェントの停止処理の追加

/etc/rc.shutdown スクリプトで実行する処理として，/etc/rc.shutdown スクリプトに運用管理エージェントを停止させるためのスクリプトファイルを追加します。

/etc/rc.shutdown スクリプトへの運用管理エージェントの停止処理の追加例を次に示します。なお，この例は，運用管理エージェントを停止させるためのスクリプトファイルを/etc/AdminAgentStop に保存した場合の例です。

```
if [ -x /etc/AdminAgentStop ]; then
  /etc/AdminAgentStop
fi
```

(b)　Linux の停止手順

運用管理エージェントを自動起動設定（mngautorun コマンド）すれば，自動停止も行われます。

mngautorun コマンドで「-sync」指定した場合，運用管理エージェントの自動停止のタイムアウト時間はデフォルトで 120 秒となります。運用管理エージェントの自動停止のタイムアウト時間（-timeout）を設定する場合は次の手順で設定してください。

1. /etc/systemd/system ディレクトリに格納されている CoAA.service ファイルを開き，ExecStop=オプションの指定にタイムアウト時間を追加します。

   指定例を次に示します。

   (例)

   CoAA.service の変更前

   ```
   ExecStop=/opt/Cosminexus/manager/bin/adminagentctl stop -sync
   ```

   CoAA.service の変更後

   ```
   ExecStop=/opt/Cosminexus/manager/bin/adminagentctl stop -sync -timeout 120
   ```

   注　背景色付きの太字個所が編集個所になります。また，斜体個所は運用環境に合わせて設定してください。

2. 1.で変更した CoAA.service ファイルを適用するために，次のコマンドを実行します。

   ```
   # systemctl reenable CoAA.service
   ```

   注意事項

   mngautorun コマンドを再度実行すると，この手順で CoAA.service に設定した内容はリセットされます。その場合は再度この手順を実施してください。

また，同一ホスト上で運用管理エージェントと Management Server を停止する場合は，両方を自動起動設定（mngautorun コマンドの both 指定）します。このとき同期起動（mngautorun コマンドの-sync 指定）するように設定した場合は， ホスト停止時またはホスト再起動時に運用管理エージェントが Management Server より後に停止します。

3 稼働情報の監視（稼働情報収集機能）

この章では，サーバ性能やリソースの情報などの稼働情報を収集する機能について説明します。稼働情報収集機能を利用した稼働情報の監視では，稼働情報を稼働情報ファイルに出力できます。また，稼働情報収集機能およびイベントの発行機能を利用することで，監視対象の稼働状態の異常を検知できます。

3.1　この章の構成

稼働情報収集機能と参照先を次の表に示します。

表 3–1　稼働情報収集機能と参照先

機能	参照先
稼働情報収集機能の概要	3.2
稼働情報ファイルの出力機能	3.3
イベントの発行機能	3.4

3.2　稼働情報収集機能の概要

この節では，**稼働情報ファイル**を使用した，J2EE サーバ，およびバッチサーバの稼働情報の監視について説明します。

稼働情報収集機能では，J2EE サーバやバッチサーバの稼働状態を定期的に監視して，サーバ性能やリソースの情報などの稼働情報を稼働情報ファイルとして取得できます。稼働情報ファイルとは，J2EE サーバ，およびバッチサーバ内の稼働情報が，定期的に出力されるファイルです。稼働情報ファイルは，J2EE サーバの機能ごとに出力されます。稼働情報は，J2EE サーバ，およびバッチサーバの稼働状態を確認したり，稼働実績を統計情報として参照したり，J2EE サーバ，およびバッチサーバの設定パラメタをチューニングしたりするのに利用できます。

なお，稼働情報ファイルに出力される FullGC 回数，または URL グループ単位の実行待ちリクエスト数を監視して，しきい値を超えた場合にメッセージを出力できます（しきい値イベント）。また，出力されたメッセージを利用して Management イベントを発行し，このアラートに対応して必要な処理を自動実行するように設定できます。Management イベントによる処理の自動実行の設定については，「9.4 Management イベントによる処理の自動実行の設定」を参照してください。

稼働情報を監視することで，次の機能を使用できます。

- **稼働情報ファイルの出力**

  取得した稼働情報を稼働情報ファイルとして出力できます。稼働情報ファイルの出力については，「3.3 稼働情報ファイルの出力機能」を参照してください。

- **イベントの発行**

  監視対象にしきい値を設定し，監視対象がしきい値を超えた時にイベントを発行できます。イベントの発行については，「3.4　イベントの発行機能」を参照してください。

なお，稼働情報を監視する場合，通常の運用では，ここで説明している稼働情報ファイルを使用した稼働監視を行ってください。稼働情報ファイルで取得できる情報より詳細な情報を取得したい場合だけ，運用管理コマンドを使用した稼働情報の監視を行ってください。運用管理コマンドを使用した稼働情報の監視は，「8.2　運用管理コマンドによる稼働情報の出力の概要」を参照してください。

3.3　稼働情報ファイルの出力機能

この節では，取得した稼働情報を稼働情報ファイルとして出力する機能について説明します。

この節の構成を次の表に示します。

表 3-2　この節の構成（稼働情報ファイルの出力機能）

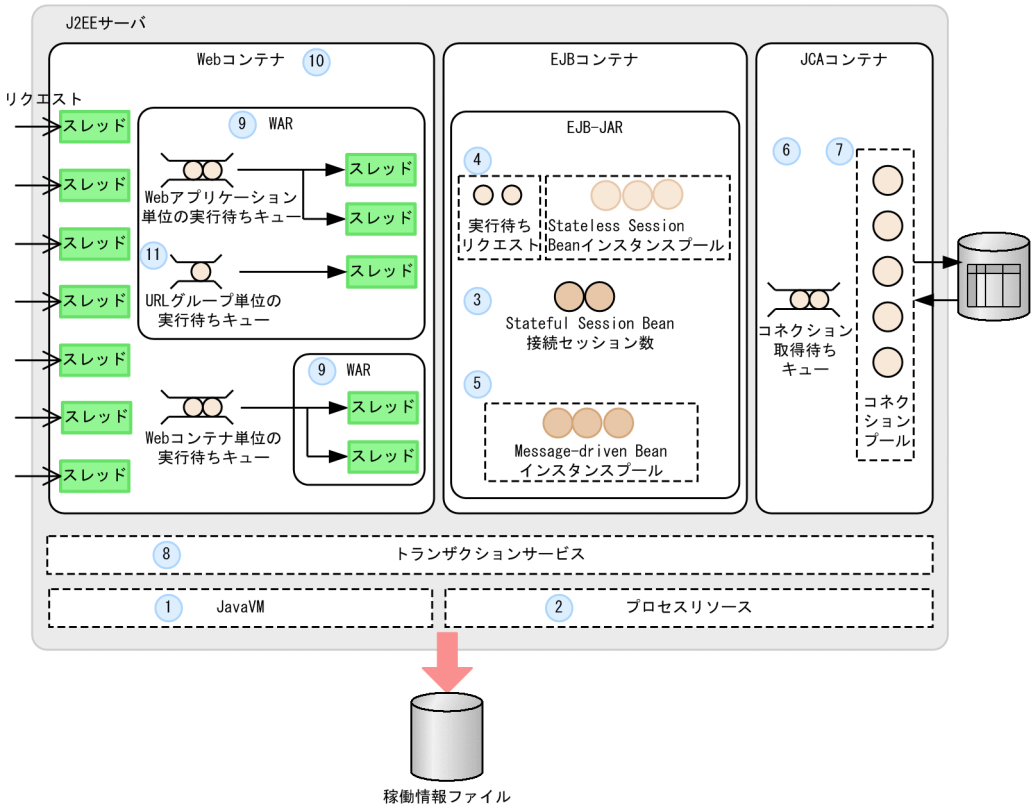
分類	タイトル	参照先
解説	稼働情報ファイルで収集できる情報の種類	3.3.1
	稼働情報ファイルで収集できる情報	3.3.2
	稼働情報ファイルの出力先とファイル面数	3.3.3
設定	実行環境での設定（J2EE サーバの設定）	3.3.4
運用	稼働情報ファイルの出力形式と出力内容	3.3.5

注　「実装」,「注意事項」について，この機能固有の説明はありません。

稼働情報ファイルの出力機能では，J2EE サーバやバッチサーバの各機能単位で出力される稼働情報を指定された稼働情報収集間隔ごとに収集し，テキストファイルに出力します。

稼働情報を出力する機能を次の図に示します。

図 3-1　稼働情報の種類と稼働情報を出力する J2EE サーバの機能



稼働情報を出力する機能を次に示します。なお，各機能の番号は，図中の数字と対応しています。

1. JavaVM
2. プロセスリソース
3. Stateful Session Bean※
4. Stateless Session Bean※
5. Message-driven Bean※
6. DB Connector
7. JCA リソース
8. トランザクションサービス
9. Web アプリケーション※
10. Web コンテナ※
11. URL グループ※

注※　バッチサーバの場合は該当しません。

出力される稼働情報を次の表に示します。

表 3-3　出力される稼働情報

稼働情報			稼働情報を出力する機能単位
分類	種別	情報	
サーバ性能	実行結果情報	受け付けリクエスト数※	Web アプリケーション
			URL グループ
		応答済みリクエスト数※	Web アプリケーション
			URL グループ
		セッション数※	Web アプリケーション
		実行待ちリクエスト数の上限からあふれたリクエスト数※	Web コンテナ
			Web アプリケーション
			URL グループ
		受け付けメッセージ数※	Message-driven Bean
		決着済みトランザクション数	トランザクションサービス
		トランザクションロールバック数	トランザクションサービス
		コネクション取得失敗数	JCA リソース
		プールされた PreparedStatement 数	DBConnector

稼働情報			稼働情報を出力する機能単位
分類	種別	情報	
		プールされた CallableStatement 数	DBConnector
		PrepareStatement メソッドが呼び出された回数	DBConnector
		PrepareCall メソッドが呼び出された回数	DBConnector
		プール内 PreparedStatement ヒット回数	DBConnector
		プール内 CallableStatement ヒット回数	DBConnector
サーバリソース	流量制御リソース情報	同時実行スレッド数※	Web コンテナ
			Web アプリケーション
			URL グループ
		実行待ちリクエスト数※	Web コンテナ
			Web アプリケーション
			URL グループ
		全体実行待ちリクエスト数※	Web コンテナ
			Web アプリケーション
		プールされたインスタンス数※	Stateless Session Bean
		プール内の使用中インスタンス数※	Stateless Session Bean
		実行待ちリクエスト数※	Stateless Session Bean
		接続セッション数※	Stateful Session Bean
		プールされたインスタンス数※	Message-driven Bean
		プール内の使用中インスタンス数※	Message-driven Bean
		プールされたコネクション数	JCA リソース
		プール内の使用中コネクション数	JCA リソース
		コネクション取得待ちスレッド数	JCA リソース
	OS リソース情報	JavaVM ヒープサイズ	JavaVM
		CopyGC 回数	JavaVM
		FullGC 回数	JavaVM
		ロードされているクラス数	JavaVM

稼働情報			稼働情報を出力する機能単位
分類	種別	情報	
		JavaVM の稼働中のスレッド数	JavaVM
		モニタロックのためにブロック状態であるスレッド数	JavaVM
		Explicit ヒープサイズ	JavaVM
		Explicit ヒープ領域の Explicit メモリブロックの個数	JavaVM
		Explicit メモリブロックの最大サイズ	JavaVM
		HTTP セッションで取得した Explicit メモリブロックの最大サイズ	JavaVM
		HTTP セッションで取得した Explicit メモリブロックの個数	JavaVM
		HTTP セッションで取得した Explicit ヒープ領域を除いたコンテナが管理している Explicit ヒープサイズ	JavaVM
		ユーザアプリケーションおよび JavaVM が管理している Explicit ヒープサイズ	JavaVM
		スレッド数	プロセスリソース
		ファイルディスクリプタ数	プロセスリソース

注※　バッチサーバの場合は該当しません。

3.3.1　稼働情報ファイルで収集できる情報の種類

稼働情報ファイルを使用すると，J2EE サーバ，およびバッチサーバの機能ごとの稼働情報を収集できます。稼働情報ファイルには，稼働情報として，サーバ性能，およびサーバリソースに関する情報が出力されます。稼働情報ファイルの対象となる J2EE サーバ，およびバッチサーバの機能，およびそれぞれの機能で取得できる稼働情報の種類を次に示します。

表 3-4　機能ごとに取得できる稼働情報の種類

機能	機能の説明	稼働情報の種類
JavaVM	J2EE サーバが使用する JavaVM	• JavaVM ヒープサイズ • CopyGC 回数 • FullGC 回数

機能	機能の説明	稼働情報の種類
		• ロードされているクラス数 • 稼働中のスレッド数 • モニタロックのためにブロック状態であるスレッド数 • Explicit ヒープサイズ • Explicit ヒープ領域の Explicit メモリブロックの個数 • Explicit メモリブロックの最大サイズ • HTTP セッションで取得した Explicit メモリブロックの最大サイズ • HTTP セッションで取得した Explicit メモリブロックの個数 • HTTP セッションで取得した Explicit ヒープ領域を除いたコンテナが管理している Explicit ヒープサイズ • ユーザアプリケーションおよび JavaVM が管理している Explicit ヒープサイズ
プロセスリソース	J2EE サーバが使用するプロセスリソース	• スレッド数 • ファイルディスクリプタ数
Stateful Session Bean	J2EE サーバ上で動作する Stateful Session Bean	• 接続セッション数
Stateless Session Bean	J2EE サーバ上で動作する Stateless Session Bean	• プールされたインスタンス数 • プール内の使用中インスタンス数 • 実行待ちリクエスト数
Message-driven Bean	J2EE サーバ上で動作する Message-driven Bean	• プールされたインスタンス数 • プール内の使用中インスタンス数 • 受け付けメッセージ数
DB Connector	J2EE サーバ上で動作する DB Connector	• プールされた PreparedStatement 数 • プールされた CallableStatement 数 • PrepareStatement メソッドが呼び出された回数 • PrepareCall メソッドが呼び出された回数 • プール内 PreparedStatement ヒット回数 • プール内 CallableStatement ヒット回数
JCA リソース	リソースアダプタが使用する JCA リソース	• プールされたコネクション数 • プール内の使用中コネクション数 • コネクション取得待ちスレッド数 • コネクション取得失敗数
トランザクションサービス	J2EE サーバで使用されるトランザクションサービス	• 決着済みトランザクション数 • トランザクションロールバック数
Web アプリケーション	J2EE サーバ上で動作する Web アプリケーション	• 同時実行スレッド数 • Web アプリケーション単位の実行待ちリクエスト数 • Web アプリケーション単位の全体実行待ちリクエスト数

機能	機能の説明	稼働情報の種類
		• Web アプリケーション単位の実行待ちリクエスト数の上限からあふれたリクエスト数 • 受け付けリクエスト数 • 応答済みリクエスト数 • セッション数
Web コンテナ	J2EE サーバ上で動作する Web コンテナ	• 同時実行スレッド数 • Web コンテナ単位またはデフォルトの実行待ちリクエスト数 • Web コンテナ単位の全体実行待ちリクエスト数 • Web コンテナ単位の実行待ちリクエスト数の上限からあふれたリクエスト数
URL グループ	Web アプリケーションに定義された URL グループ単位の同時実行数制御	• 同時実行スレッド数 • URL グループ単位の実行待ちリクエスト数 • URL グループ単位の実行待ちリクエスト数の上限からあふれたリクエスト数 • 受け付けリクエスト数 • 応答済みリクエスト数

ポイント

Web アプリケーション，Web コンテナおよび URL グループ単位のそれぞれの実行待ちリクエスト数の関係を次の表に示します。

対象	項目	説明
Web アプリケーション	Web アプリケーション単位の実行待ちリクエスト数	Web コンテナで実行するリクエストのうち，Web アプリケーション単位の同時実行スレッド数制御をしている Web アプリケーションに対する実行待ちリクエスト数です。 URL グループ単位の実行待ちリクエスト数は含まれません。
	Web アプリケーション単位の全体実行待ちリクエスト数	Web コンテナで実行するリクエストのうち，Web アプリケーション単位の同時実行スレッド数制御をしている Web アプリケーションに対する実行待ちリクエスト数です。 URL グループ単位の実行待ちリクエスト数も含まれます。
Web コンテナ	Web コンテナ単位またはデフォルトの実行待ちリクエスト数	Web コンテナで実行するリクエストのうち，Web アプリケーション単位の同時実行スレッド数制御をしていない Web アプリケーションに対する実行待ちリクエスト数です。
	Web コンテナ単位の全体実行待ちリクエスト数	Web コンテナ単位の実行待ちリクエスト数と，すべての Web アプリケーション単位の全体実行待ちリクエスト数の和です。
URL グループ	URL グループ単位の実行待ちリクエスト数	Web コンテナで実行するリクエストのうち，Web アプリケーション単位の同時実行スレッド数制御をしている Web アプリケーション内で URL グループ単位の同時実行スレッ

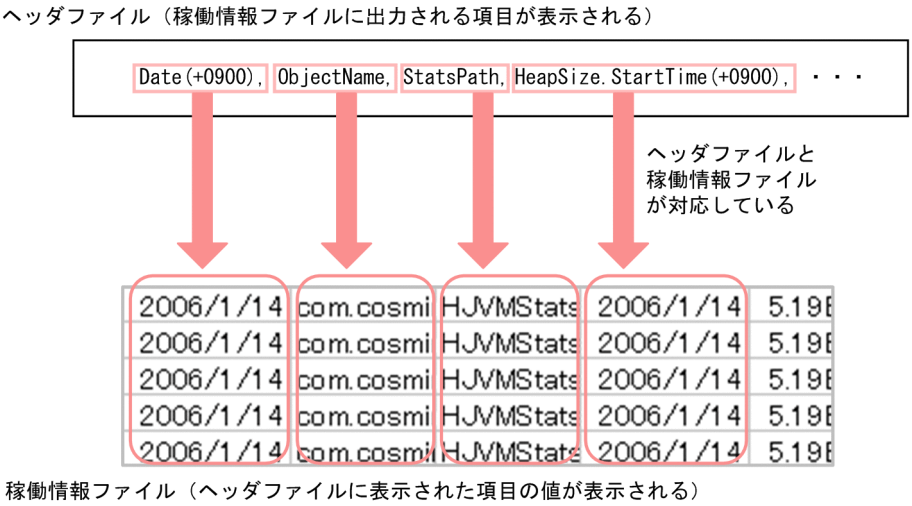
対象	項目	説明
		ド数制御している，特定の URL グループの実行待ちリクエスト数です。

3.3.2　稼働情報ファイルで収集できる情報

稼働情報ファイルを調査するためには，ヘッダファイルと組み合わせて参照する必要があります。ヘッダファイルは，稼働情報ファイルの対象となる J2EE サーバの機能と 1 対 1 の関係で出力されて，稼働情報ファイルに出力される情報に対応する項目名が出力されます。そのため，ヘッダファイルと稼働情報ファイルを対応させながら，稼働情報を調査する必要があります。

ヘッダファイルと稼働情報ファイルの対応の例を次に示します。

図 3-2　ヘッダファイルと稼働情報ファイルの対応



稼働情報ファイルを調査する場合，まずファイル名に表示される日付などを参考にして，調査したい稼働情報ファイルを選択してください。次に，選択した稼働情報ファイルと，それに対応するヘッダファイルを一つのファイルに編集して，Excel などのアプリケーションプログラムで参照してください。その場合，グラフや表などの形式に変換して参照することをお勧めします。また，時刻を表す項目を Excel などのアプリケーションプログラムで参照する場合，セルの表示設定を，時刻を表示する形式に設定することをお勧めします。例えば Excel を使用している場合は，セルの表示形式を「時刻」に設定してください。

ヘッダファイルは，J2EE サーバの起動時に次のフォルダに作成されます。

Windows の場合

<作業ディレクトリ>※¥ejb¥<J2EEサーバ名>¥stats

UNIX の場合

<作業ディレクトリ>※/ejb/<J2EEサーバ名>/stats

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，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」です。

稼働情報ファイルとヘッダファイルは，すべての稼働情報ファイルに共通の項目と，個別の項目で構成されます。個別の項目は，対象となる機能ごとに異なります。稼働情報ファイルの出力形式と出力内容については，「3.3.5　稼働情報ファイルの出力形式と出力内容」を参照してください。

3.3.3　稼働情報ファイルの出力先とファイル面数

ここでは，稼働情報ファイルの出力先，ファイル面数，面の切り替え間隔，およびファイル名について説明します。

(1)　稼働情報ファイルの出力先

デフォルトの設定の場合，稼働情報ファイルは，J2EE サーバの起動時に次のフォルダに作成されます。

Windows の場合

<作業ディレクトリ>※¥ejb¥<J2EEサーバ名>¥stats

UNIX の場合

<作業ディレクトリ>※/ejb/<J2EEサーバ名>/stats

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，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」です。

また，稼働情報ファイルは，出力先を変更することもできます。

稼働情報ファイルの出力先の設定については，「3.3.4　実行環境での設定（J2EE サーバの設定）」を参照してください。

(2)　稼働情報ファイルのファイル面数と面の切り替え間隔

稼働情報ファイルは，出力先に保存するファイルの面数とファイルの面を切り替える時間間隔を設定できます。例えば，保存するファイル面数を「7」，ファイルの面を切り替える時間間隔を「24（時間）」と設定した場合，稼働情報ファイルは 1 日に 1 回面が切り替わり，最新の 1 週間分（7 日分）のファイルを保存できることになります。

稼働情報ファイルの面数と面を切り替える時間間隔の設定については，「3.3.4　実行環境での設定（J2EE サーバの設定）」を参照してください。

(3) 稼働情報ファイルのファイル名

稼働情報ファイルのファイル名は，稼働情報を収集する機能ごとに異なります。機能ごとの稼働情報ファイルのファイル名を次の表に示します。

表 3-5 稼働情報ファイルのファイル名

機能	ファイル名
JavaVM	HJVMStats_<YYYYMMDDhhmm><TZ>.csv
プロセスリソース	HOSStats_<YYYYMMDDhhmm><TZ>.csv
Stateful Session Bean	HStatefulSessionBeanStats_<YYYYMMDDhhmm><TZ>.csv
Stateless Session Bean	HStatelessSessionBeanStats_<YYYYMMDDhhmm><TZ>.csv
Message-driven Bean	HMessageDrivenBeanStats_<YYYYMMDDhhmm><TZ>.csv
DB Connector	HDBConnectorStats_<YYYYMMDDhhmm><TZ>.csv
JCA リソース	HJCAConnectionPoolStats_<YYYYMMDDhhmm><TZ>.csv
トランザクションサービス	HJTAStats_<YYYYMMDDhhmm><TZ>.csv
Web アプリケーション	HWebModuleStats_<YYYYMMDDhhmm><TZ>.csv
Web コンテナ	HWebContainerStats_<YYYYMMDDhhmm><TZ>.csv
URL グループ	HWebURLGroupStats_<YYYYMMDDhhmm><TZ>.csv

注

- <YYYYMMDDhhmm>には，ファイルが出力された時刻を表す次の値が表示されます。
  YYYY：西暦年，MM：月，DD：日，hh：時，mm：分
- <TZ>には，タイムゾーンが，GMT（グリニッジ標準時）からの時差で表示されます。日本の場合，「+0900」が表示されます。

3.3.4 実行環境での設定（J2EE サーバの設定）

稼働情報ファイルを収集する場合，J2EE サーバの設定が必要です。J2EE サーバの設定は，簡易構築定義ファイルで実施します。稼働情報ファイルの収集の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に指定します。

稼働情報ファイルを出力する場合のデフォルトの出力先を次に示します。なお，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 パラメタで指定します。

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

<作業ディレクトリ>/ejb/<サーバ名称>/stats

デフォルトの設定では，J2EE サーバまたはバッチサーバが出力した稼働情報が収集されて，稼働情報ファイルに出力されます。また，稼働情報の監視対象がしきい値を超えたときにイベントが発行されます。稼働情報ファイルの出力先や面数を変更したい場合や，稼働情報のしきい値や監視間隔を変更したい場合には，簡易構築定義ファイルで設定を変更してください。

簡易構築定義ファイルでの稼働情報ファイルの収集の定義について次の表に示します。なお，バッチサーバの場合は，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。

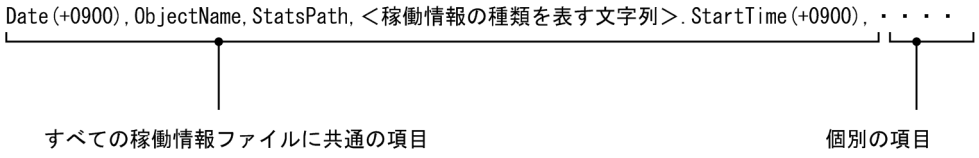
表 3-6　簡易構築定義ファイルでの稼働情報ファイルの収集に必要な定義

項目	指定するパラメタ	設定内容
稼働情報収集間隔	ejbserver.management.statistics.interval	稼働情報を取得する間隔（秒）を指定します。
稼働情報ファイルの出力	ejbserver.management.stats_file.enabled	稼働情報ファイルを出力するかどうかを指定します。デフォルトは true です。
	ejbserver.management.stats_file.dir	稼働情報ファイルの出力先ディレクトリを指定します。
	ejbserver.management.stats_file.num	稼働情報ファイルの面数を指定します。
	ejbserver.management.stats_file.period	稼働情報ファイルの面を切り替える時間間隔（時間）を指定します。
	ejbserver.management.stats_file.base_time	稼働情報ファイルの面を切り替える基点の時間を指定します。

簡易構築定義ファイル，および指定するパラメタの詳細は，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「4.3　簡易構築定義ファイル」を参照してください。

3.3.5　稼働情報ファイルの出力形式と出力内容

稼働情報ファイルとヘッダファイルは，すべての稼働情報ファイルに共通の項目と，個別の項目で構成されます。個別の項目は，対象となる機能ごとに異なります。ヘッダファイルの構成は，次のようになります。



すべての稼働情報ファイルに共通の項目の見方を次の表に示します。

表 3-7　すべての稼働情報ファイルに共通の項目の見方

項目（ヘッダファイルに出力される文字列）	説明
Date※	稼働情報ファイルを収集した時刻が，「YYYY/MM/DD hh:mm:ss.nnn」で表示されます。 YYYY：西暦年，MM：月，DD：日，hh：時，mm：分，ss：秒，nnn：ミリ秒
ObjectName	稼働情報ファイルごとに同じ形式の情報が表示されます。
StatsPath	稼働情報ファイルごとに固有の情報が出力されます。
＜稼働情報の種類を表す文字列＞.StartTime※	稼働情報の種類ごとに，稼働情報の収集対象が稼働中になった時刻が表示されます。時刻は，1970 年 1 月 1 日午前 0 時から経過した時間がミリ秒単位で表示されます。

注※　ヘッダファイルにはこの情報に加えて，タイムゾーンが GMT（グリニッジ標準時）からの時差で表示されます。
例（日本の場合）：「Date(+0900)」，「HeapSize.StartTime(+0900)」
ただし，リロードによる J2EE アプリケーションの入れ替えを実行した場合，リロードを実行した時刻が表示されます。

なお，「ObjectName」には，稼働情報ファイルごとに次の形式で情報が出力されます。

表 3-8　ObjectName の形式

稼働情報ファイル（ファイル名）	ObjectName の形式
JavaVM（HJVMStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:J2EEServer=<J2EE サーバ名>,j2eeType=JVM,name=jvm
プロセスリソース（HOSStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:J2EEServer=<J2EE サーバ名>,j2eeType=OSResource,name=os
Stateful Session Bean（HStatefulSessionBeanStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:EJBModule=<EJB アプリケーション表示名>,J2EEApplication=<J2EE アプリケーション表示名>,J2EEServer=<J2EE サーバ名>,j2eeType=StatefulSessionBean,mode=<アプリケーションの動作モード>※1,name=<Stateful Session Bean 名>
Stateless Session Bean（HStatelessSessionBeanStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:EJBModule=<EJB アプリケーション表示名>,J2EEApplication=<J2EE アプリケーション表示名>,J2EEServer=<J2EE サーバ名>,j2eeType=StatelessSessionBean,mode=<アプリケーションの動作モード>※1,name=<Stateless Session Bean 名>
Message-driven Bean（HMessageDrivenBeanStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:EJBModule=<EJB アプリケーション表示名>,J2EEApplication=<J2EE アプリケーション表示名>,J2EEServer=<J2EE サーバ名>,j2eeType=MessageDrivenBean,mode=<アプリケーションの動作モード>※1,name=<Message-driven Bean 名>
DB Connector（HDBConnectorStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:J2EEApplication=<J2EE アプリケーション表示名>※2,J2EEServer=<J2EE サーバ

稼働情報ファイル（ファイル名）	ObjectName の形式
	名>,ResourceAdapterModule=<リソースアダプタ表示名>,j2eeType=ResourceAdapter,mode=<アプリケーションの動作モード>※1,name=<リソースアダプタ表示名>
JCA リソース（HJCAConnectionPoolStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:J2EEServer=<J2EEサーバ名>,ResourceAdapter=<リソースアダプタ表示名>,j2eeType=JCAResource,mode=<アプリケーションの動作モード>※1,app=<J2EE アプリケーションの表示名>※3,name=<リソースアダプタ表示名>
トランザクションサービス（HJTAStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:J2EEServer=<J2EEサーバ名>,j2eeType=JTAResource,name=JTAResource
Web アプリケーション（HWebModuleStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:J2EEApplication=<J2EE アプリケーション表示名>,J2EEServer=<J2EE サーバ名>,j2eeType=WebModule,mode=<アプリケーションの動作モード>※1,name=<Web アプリケーション表示名>
Web コンテナ（HWebContainerStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:J2EEServer=<J2EEサーバ名>,j2eeType=WebContainer,name=WebContainer
URL グループ（HWebURLGroupStats_<YYYYMMDDhhmm><TZ>.csv）	com.cosminexus.management.j2ee:J2EEApplication=<J2EE アプリケーション表示名>,J2EEServer=<J2EE サーバ名>,WebModule=<Web アプリケーション名>,j2eeType=WebURLGroup,mode=<アプリケーションの動作モード>※1,name=<URL グループ単位の同時実行スレッド数制御の定義名>

注※1　<アプリケーションの動作モード>には，次の値が出力されます。

- test：アプリケーションの動作モードがテストモードの場合
- normal：アプリケーションの動作モードが通常モードの場合

注※2　<J2EE アプリケーション表示名>は，リソースアダプタが J2EE アプリケーションに含まれてデプロイされている場合にだけ出力されます。直接 J2EE サーバにデプロイされたリソースアダプタの場合は，「null」が出力されます。

注※3　「app=<J2EE アプリケーションの表示名>」は，リソースアダプタが J2EE アプリケーションに含まれてデプロイされている場合にだけ出力されます。

次に，それぞれの稼働情報ファイル個別の項目の見方について説明します。個別の項目は，ヘッダファイルに次の形式で表示されます。

＜稼働情報の種類を表す文字列＞.＜項目名＞

各項目は稼働情報の種類を表す文字列のあとに，「.」で区切られて出力されます。稼働情報の種類を表す文字列，および項目名について次の表に示します。

表 3-9　稼働情報の種類を表す文字列

稼働情報の種類	稼働情報の種類を表す文字列
JavaVM ヒープサイズ	HeapSize
CopyGC 回数	CopyGCCount
FullGC 回数	FullGCCount
ロードされているクラス数	LoadedClassCount
モニタロックのためにブロック状態であるスレッド数	ThreadBlockedCount
Explicit ヒープサイズ	EHeapSize
Explicit ヒープ領域の Explicit メモリブロックの個数	EMemoryBlockCount
Explicit メモリブロックの最大サイズ	EMemoryBlockMaxSize
HTTP セッションで取得した Explicit メモリブロックの最大サイズ	HTTPSessionEMemoryBlockMaxSize
HTTP セッションで取得した Explicit メモリブロックの個数	HTTPSessionEMemoryBlockCount
HTTP セッションで取得した Explicit ヒープ領域を除いたコンテナが管理している Explicit ヒープサイズ	ContainerEHeapSize
ユーザアプリケーションおよび JavaVM が管理している Explicit ヒープサイズ	ApplicationEHeapSize
スレッド数	ThreadCount
ファイルディスクリプタ数	FileDescriptorCount
接続セッション数	ActiveSessionCount
プールされたインスタンス数	PooledInstanceCount
プール内の使用中インスタンス数	ActivePooledInstanceCount
受け付けメッセージ数	MessageCount
実行待ちリクエスト数	WaitingRequestCount
プールされた PreparedStatement 数	PooledPreparedStatementCount
プールされた CallableStatement 数	PooledCallableStatementCount
PrepareStatement メソッドが呼び出された回数	InvokedPrepareStatementMethodCount
prepareCall メソッドが呼び出された回数	InvokedPrepareCallMethodCount
プール内 PreparedStatement ヒット回数	PooledPreparedStatementHitCount
プール内 CallableStatement ヒット回数	PooledCallableStatementHitCount
プールされたコネクション数	PoolSize
プール内の使用中コネクション数	ActivePoolSize
コネクション取得待ちスレッド数	WaitingThreadCount

稼働情報の種類	稼働情報の種類を表す文字列
コネクション取得失敗数	FailedRequestCount
決着済みトランザクション数	CompletionCount
トランザクションロールバック数	RolledbackCount
応答済みリクエスト数	ResponseCount
同時実行スレッド数	ActiveThreadCount
実行待ちリクエスト数の上限からあふれたリクエスト数	OverflowRequestCount
受け付けリクエスト数	RequestCount
セッション数	SessionCount

表 3-10　稼働統計情報ファイルの項目名

項目名（ヘッダファイルに出力される文字列）	説明
Count	オブジェクトの起動など稼働情報の取得を開始した時点からの累積回数，または累積個数
HighWaterMark	前回の稼働情報ファイルの出力時からの最大値
LowWaterMark	前回の稼働情報ファイルの出力時からの最小値
Current	現在値
UpperBound	上限値
LowerBound	下限値

次に，個別に出力される項目について，稼働情報ファイルごとに説明します。

(1)　JavaVM の稼働情報ファイルに出力される情報

JavaVM の稼働情報ファイルに出力される情報について説明します。JavaVM の稼働情報ファイルでは，J2EE サーバが使用する JavaVM の稼働情報を調査できます。

稼働情報ファイル名

HJVMStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HJVMStats.txt

出力される稼働情報

JavaVM の稼働情報ファイルに出力される内容を次に示します。

表 3-11　JavaVM の稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）
JavaVM ヒープサイズ	HeapSize.UpperBound
	HeapSize.LowerBound
	HeapSize.HighWaterMark
	HeapSize.LowWaterMark
	HeapSize.Current
CopyGC 回数※1※2	CopyGCCount.Count
FullGC 回数※2※3	FullGCCount.Count
ロードされているクラス数	LoadedClassCount.HighWaterMark
	LoadedClassCount.LowWaterMark
	LoadedClassCount.Current
稼働中のスレッド数	ThreadCount.HighWaterMark
	ThreadCount.LowWaterMark
	ThreadCount.Current
モニタロックのためにブロック状態であるスレッド数	ThreadBlockedCount.HighWaterMark
	ThreadBlockedCount.LowWaterMark
	ThreadBlockedCount.Current
Explicit ヒープサイズ	EHeapSize UpperBound
	EHeapSize LowerBound
	EHeapSize HighWaterMark
	EHeapSize LowWaterMark
	EHeapSize Current
Explicit ヒープ領域の Explicit メモリブロックの個数	EMemoryBlockCount HighWaterMark
	EMemoryBlockCount LowWaterMark
	EMemoryBlockCount Current
Explicit メモリブロックの最大サイズ	EMemoryBlockMaxSize HighWaterMark
	EMemoryBlockMaxSize LowWaterMark
	EMemoryBlockMaxSize Current
HTTP セッションで取得した Explicit メモリブロックの最大サイズ※4	HTTPSessionEMemoryBlockMaxSize.HighWaterMark
	HTTPSessionEMemoryBlockMaxSize.LowWaterMark

稼働情報の種類	項目（ヘッダファイルに出力される文字列）
	HTTPSessionEMemoryBlockMaxSize.Current
HTTP セッションで取得した Explicit メモリブロックの個数	HTTPSessionEMemoryBlockCount.HighWaterMark
	HTTPSessionEMemoryBlockCount.LowWaterMark
	HTTPSessionEMemoryBlockCount.Current
HTTP セッションで取得した Explicit ヒープ領域を除いたコンテナが管理している Explicit ヒープサイズ	ContainerEHeapSize.HighWaterMark
	ContainerEHeapSize.LowWaterMark
	ContainerEHeapSize.Current
ユーザアプリケーションおよび JavaVM が管理している Explicit ヒープサイズ※4	ApplicationEHeapSize.HighWaterMark
	ApplicationEHeapSize.LowWaterMark
	ApplicationEHeapSize.Current

注※1

SerialGC が有効な場合，CopyGC の回数が出力されます。

G1GC が有効な場合，Young GC と Mixed GC の合計回数が出力されます。

注※2

JavaVM の稼働情報である CopyGC 回数と FullGC 回数には，J2EE サーバプロセスが次のどれかのタイミングで明示的に発行する GC の発生回数も含まれます。

- J2EE アプリケーションのリロードで，Web アプリケーションを単体でリロードした場合
- J2EE アプリケーションのリロードで，EJB アプリケーションをリロードした場合
- J2EE アプリケーションを停止した場合
- J2EE アプリケーションの開始で，スタブを生成した場合
- J2EE アプリケーションの開始に失敗した場合
- J2EE アプリケーションを削除した場合

FullGC 回数のしきい値監視を有効にする場合，これらのタイミングでカウントされる回数を考慮する必要があります。

注※3

JavaVM の稼働情報である FullGC 回数には，バッチサーバプロセスが次のどちらかのタイミングで明示的に発行する GC の発生回数も含まれます。

- GC 制御機能で，メモリ使用量のしきい値を超えた場合
- バッチアプリケーションを停止した場合

FullGC の発生回数を監視する場合，これらのタイミングでカウントされる回数を考慮する必要があります。

注※4

HTTP セッションで利用する Explicit ヒープの省メモリ化機能を使用している場合，出力内容が異なります。詳細は，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「7.11.3　HTTP セッションで利用する Explicit ヒープの省メモリ化機能利用時の注意事項」を参照してください。

(2)　プロセスリソースの稼働情報ファイルに出力される情報

プロセスリソースの稼働情報ファイルに出力される情報について説明します。プロセスリソースの稼働情報ファイルでは，J2EE サーバプロセスが使用する OS リソースの稼働情報を調査できます。

稼働情報ファイル名

HOSStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HOSStats.txt

出力される稼働情報

プロセスリソースの稼働情報ファイルに出力される内容を次に示します。

表 3-12　プロセスリソースの稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）
J2EE サーバプロセスが生成したスレッド数	ThreadCount.UpperBound※1
	ThreadCount.LowerBound※2
	ThreadCount.HighWaterMark※2
	ThreadCount.LowWaterMark※2
	ThreadCount.Current※2
J2EE サーバプロセスが使用するファイルディスクリプタ数	FileDescriptorCount.UpperBound※3
	FileDescriptorCount.LowerBound※3
	FileDescriptorCount.HighWaterMark※4
	FileDescriptorCount.LowWaterMark※4
	FileDescriptorCount.Current※4

注※1　Windows，または Linux の場合は無効になり，「-1」が出力されます。

注※2　Linux の場合は無効になり，「-1」が出力されます。

注※3　Windows の場合は無効になり，「-1」が出力されます。

注※4　Windows，または AIX の場合は無効になり，「-1」が出力されます。

(3)　Stateful Session Bean の稼働情報ファイルに出力される情報

Stateful Session Bean の稼働情報ファイルに出力される情報について説明します。Stateful Session Bean の稼働情報ファイルでは，Stateful Session Bean の稼働情報を調査できます。

稼働情報ファイル名

HStatefulSessionBeanStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HStatefulSessionBeanStats.txt

出力される稼働情報

Stateful Session Bean の稼働情報ファイルに出力される内容を次に示します。

表 3-13　Stateful Session Bean の稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
接続中のセッション数	ActiveSessionCount.UpperBound	Stateful Session Bean のアクティブ・セッションの最大数が無制限のとき，「0」が出力されます。
	ActiveSessionCount.LowerBound	下限値を設定できないため，無効になり「-1」が出力されます。
	ActiveSessionCount.HighWaterMark	－
	ActiveSessionCount.LowWaterMark	－
	ActiveSessionCount.Current	－

（凡例）　－：該当なし

(4)　Stateless Session Bean の稼働情報ファイルに出力される情報

Stateless Session Bean の稼働情報ファイルに出力される情報について説明します。Stateless Session Bean の稼働情報ファイルでは，Stateless Session Bean の稼働情報を調査できます。

稼働情報ファイル名

HStatelessSessionBeanStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HStatelessSessionBeanStats.txt

出力される稼働情報

Stateless Session Bean の稼働情報ファイルに出力される内容を次に示します。

表 3-14　Stateless Session Bean の稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
プールされたインスタンス数	PooledInstanceCount.UpperBound	Stateless Session Bean のプール内インスタンスの最大数が無制限のとき，「0」が出力されます。
	PooledInstanceCount.LowerBound	－
	PooledInstanceCount.HighWaterMark	－

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
	PooledInstanceCount.LowWaterMark	−
	PooledInstanceCount.Current	−
プール内の使用中インスタンス数	ActivePooledInstanceCount.UpperBound	Stateless Session Bean のプール内インスタンスの最大数が無制限のとき，「0」が出力されます。
	ActivePooledInstanceCount.LowerBound	下限値を設定できないため，無効になり「-1」が出力されます。
	ActivePooledInstanceCount.HighWaterMark	−
	ActivePooledInstanceCount.LowWaterMark	−
	ActivePooledInstanceCount.Current	−
実行待ちリクエスト数	WaitingRequestCount.HighWaterMark	−
	WaitingRequestCount.LowWaterMark	−
	WaitingRequestCount.Current	−

（凡例） −：該当なし

(5) Message-driven Bean の稼働情報ファイルに出力される情報

Message-driven Bean の稼働情報ファイルに出力される情報について説明します。Message-driven Bean の稼働情報ファイルでは，Message-driven Bean の稼働情報を調査できます。

稼働情報ファイル名

HMessageDrivenBeanStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HMessageDrivenBeanStats.txt

出力される稼働情報

Message-driven Bean の稼働情報ファイルに出力される内容を次に示します。

表 3-15 Message-driven Bean の稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
プールされたインスタンス数	PooledInstanceCount.UpperBound	Message-driven Bean のプール内インスタンスの最大数が無制限のとき，「0」が出力されます。
	PooledInstanceCount.LowerBound	−
	PooledInstanceCount.HighWaterMark	−
	PooledInstanceCount.LowWaterMark	−
	PooledInstanceCount.Current	−

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
プール内の使用中インスタンス数	ActivePooledInstanceCount.UpperBound	Message-driven Bean のプール内インスタンスの最大数が無制限のとき，「0」が出力されます。
	ActivePooledInstanceCount.LowerBound	下限値を設定できないため，無効になり「-1」が出力されます。
	ActivePooledInstanceCount.HighWaterMark	－
	ActivePooledInstanceCount.LowWaterMark	－
	ActivePooledInstanceCount.Current	－
受け付けメッセージ数	MessageCount.Count	－

（凡例） －：該当なし

(6) DB Connector の稼働情報ファイルに出力される情報

DB Connector の稼働情報ファイルに出力される情報について説明します。DB Connector の稼働情報ファイルでは，DB Connector の稼働情報を調査できます。

ただし，コネクションプーリング機能を使用していない場合，prepareStatement メソッドが呼び出された回数（InvokedPrepareStatementMethodCount.Count），および PrepareCall メソッドが呼び出された回数（InvokedPrepareCallMethodCount.Count）以外の値は無効になり，「-1」が出力されます。

稼働情報ファイル名

HDBConnectorStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HDBConnectorStats.txt

出力される稼働情報

DB Connector の稼働情報ファイルに出力される内容を次に示します。

表 3-16 DB Connector の稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
ステートメントプーリング機能によってキャッシュされた PreparedStatement 数	PooledPreparedStatementCount.UpperBound※1	次の値が出力されます。 「出力値＝ DB Connector に設定したコネクションごとの PreparedStatement サイズ×コネクションプールの上限値」 コネクションプールの上限値を無制限にしたときは，無効になり「-1」が出力されます。
	PooledPreparedStatementCount.LowerBound	下限値を設定できないため，無効になり「-1」が出力されます。

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
	PooledPreparedStatementCount.HighWaterMark※1※2	－
	PooledPreparedStatementCount.LowWaterMark※1※2	－
	PooledPreparedStatementCount.Current※1※2	－
	PooledPreparedStatementCount.Current※1※2※4	－
ステートメントプーリング機能によってキャッシュされたCallableStatement数	PooledCallableStatementCount.UpperBound※3	次の値が出力されます。 「出力値＝ DB Connector に設定したコネクションごとのPooledCallableStatement サイズ×コネクションプールの上限値」 コネクションプールの上限値を無制限にしたときは，無効になり「-1」が出力されます。
	PooledCallableStatementCount.LowerBound	下限値を設定できないため，無効になり「-1」が出力されます。
	PooledCallableStatementCount.HighWaterMark※2※3	－
	PooledCallableStatementCount.LowWaterMark※2※3	－
	PooledCallableStatementCount.Current※2※3	－
	PooledCallableStatementCount.Current※2※3※4	－
PrepareStatement メソッドが呼び出された回数	InvokedPrepareStatementMethodCount.Count※2	－
PrepareCall メソッドが呼び出された回数	InvokedPrepareCallMethodCount.Count※2	－
ステートメントプーリング機能によってキャッシュされたPreparedStatementがヒットした回数	PooledPreparedStatementHitCount.Count※1※2	－
ステートメントプーリング機能によってキャッシュされたCallableStatementがヒットした回数	PooledCallableStatementHitCount.Count※2※3	－

（凡例） －：該当なし

注※1　DB Connector の設定であるコネクションごとの PreparedStatement のプールサイズに「0」を設定した場合，PreparedStatement がプールされなくなるため，無効になり「-1」が出力されます。

注※2　接続先データベースに Oracle を使用している場合，または接続先データベースに HiRDB 以外を使用してコネクションの障害検知を行った場合，ステートメントオブジェクトを作成していなくても，1 以上の値が出力されます。

注※3　DB Connector の設定であるコネクションごとの CallableStatement のプールサイズに 0 を設定した場合，CallableStatement がプールされなくなるため，無効になり「-1」が出力されます。

注※4　コネクション数調整機能またはコネクション障害検知と併用した場合，コネクションプールから取り除かれた未使用コネクションはコネクションプール内のコネクション数としてカウントされないため，プールされている PreparedStatement 数および CallableStatement 数は次の値を一時的に超える場合があります。

- PreparedStatement 数
  コネクションプールの最大値×PreparedStatementPoolSize
- CallableStatement 数
  コネクションプールの最大値×CallableStatementPoolSize

(7)　JCA リソースの稼働情報ファイルに出力される情報

JCA リソースの稼働情報ファイルに出力される情報について説明します。JCA リソースの稼働情報ファイルでは，JCA リソースの稼働情報を調査できます。ただし，コネクションプールが無効の場合，コネクション取得失敗数だけが出力されます。DB Connector for Reliable Messaging を使用している場合，稼働情報は Reliable Messaging 側に出力されます。DB Connector for Reliable Messaging 側には出力されません。

Connector 1.5 仕様に準拠したリソースアダプタを使用した場合は，一つ目のコネクション定義（ra.xml 内でいちばん先頭の定義）に対応する稼働情報が出力されます。Outbound のコネクション定義がない場合は，HJCAConnectionPoolStats の稼働情報は出力されません。

稼働情報ファイル名

HJCAConnectionPoolStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HJCAConnectionPoolStats.txt

出力される稼働情報

JCA リソースの稼働情報ファイルに出力される内容を次に示します。

表 3-17　JCA リソースの稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
プールされたコネクション数	PoolSize.UpperBound	−
	PoolSize.LowerBound	−
	PoolSize.HighWaterMark	−
	PoolSize.LowWaterMark	−
	PoolSize.Current	−
プール内の使用中コネクション数	ActivePoolSize.UpperBound	−
	ActivePoolSize.LowerBound	下限値を設定できないため，無効になり「-1」が出力されます。

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
	ActivePoolSize.HighWaterMark	コマンドの実行やコネクション数自動調節機能によって，値が一時的に変動する場合があります。
	ActivePoolSize.LowWaterMark	
	ActivePoolSize.Current	
コネクション取得待ちスレッド数	WaitingThreadCount.UpperBound	上限値を設定できないため，無効になり「-1」が出力されます。
	WaitingThreadCount.LowerBound	下限値を設定できないため，無効になり「-1」が出力されます。
	WaitingThreadCount.HighWaterMark	－
	WaitingThreadCount.LowWaterMark	－
	WaitingThreadCount.Current	－
コネクション取得失敗数	FailedRequestCount.Count	DB Connector for Reliable Messaging を使用している場合，Reliable Messaging 側に出力されます。DB Connector for Reliable Messaging 側には常に「0」が出力されます。

（凡例） －：該当なし

(8) トランザクションサービスの稼働情報ファイルに出力される情報

トランザクションサービスの稼働情報ファイルに出力される情報について説明します。トランザクションサービスの稼働情報ファイルでは，トランザクションサービスの稼働情報を調査できます。

稼働情報ファイル名

HJTAStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HJTAStats.txt

出力される稼働情報

トランザクションサービスの稼働情報ファイルに出力される内容を次に示します。

表 3-18 トランザクションサービスの稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）
決着したトランザクション数	CompletionCount.Count
ロールバックしたトランザクション数	RolledbackCount.Count

(9) Web アプリケーションの稼働情報ファイルに出力される情報

Web アプリケーションの稼働情報ファイルに出力される情報について説明します。Web アプリケーションの稼働情報ファイルでは，対象の Web アプリケーション全体の稼働情報を調査できます。

稼働情報ファイル名

HWebModuleStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HWebModuleStats.txt

出力される稼働情報

Web アプリケーションの稼働情報ファイルに出力される内容を次に示します。

表 3-19 Web アプリケーションの稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
最大同時実行スレッド数	ActiveThreadCount.UpperBound※1※2	－
	ActiveThreadCount.LowerBound※3	－
	ActiveThreadCount.HighWaterMark※1	－
	ActiveThreadCount.LowWaterMark※1	－
	ActiveThreadCount.Current※1	－
Web アプリケーション単位の実行待ちリクエスト数	WaitingRequestCount.UpperBound※1※2※4	－
	WaitingRequestCount.LowerBound※3※4	－
	WaitingRequestCount.HighWaterMark※1※4	－
	WaitingRequestCount.LowWaterMark※1※4	－
	WaitingRequestCount.Current※1※4	－
Web アプリケーション単位の全体実行待ちリクエスト数	WholeWaitingRequestCount.UpperBound※1※4	－
	WholeWaitingRequestCount.LowerBound※1※4	－
	WholeWaitingRequestCount.HighWaterMark※1※4	－
	WholeWaitingRequestCount.LowWaterMark※1※4	－
	WholeWaitingRequestCount.Current※1※4	－
Web アプリケーション単位の実行待ちリクエスト数の上限からあふれたリクエスト数	OverflowRequestCount.Count※1	－
受け付けリクエスト数	RequestCount.Count※4	Web コンテナが受け付けた Web アプリケーション単位のリクエスト数が出力されます。Web アプリ

稼働情報の種類	項目（ヘッダファイルに出力される文字列）	備考
		ケーション単位の実行待ちリクエスト数もカウントされます。Web サーバでは受信されていても，Web コンテナに転送されていないリクエスト数はカウントされません。
応答済みリクエスト数	ResponseCount.Count	－
セッション数	SessionCount.UpperBound	セッション数の上限を設定していない場合，無効になり「-1」が出力されます。
	SessionCount.LowerBound※3	－
	SessionCount.HighWaterMark	－
	SessionCount.LowWaterMark	－
	SessionCount.Current	－

（凡例） －：該当なし

注※1　Web アプリケーション単位の同時実行数制御機能を使用していない場合，または対象となる Web アプリケーションに，同時実行数制御機能を使用していない場合，無効になり，「-1」が出力されます。

注※2　対象となる Web アプリケーションに最大同時実行スレッド数制御が設定されている場合，Web アプリケーション単位の最大同時実行スレッド数が出力されます。最大同時実行スレッド数を動的に変更した場合，変更後の値が出力されます。

注※3　無効になり「-1」が出力されます。

注※4　リロード機能による入れ替え処理中の場合，Web アプリケーションが受け付けた実行待ちリクエスト数はカウントされません。リロード処理の完了後にカウントされます。

(10)　Web コンテナの稼働情報ファイルに出力される情報

Web コンテナの稼働情報ファイルに出力される情報について説明します。Web コンテナの稼働情報ファイルでは，開始している Web アプリケーションの稼働情報を調査できます。

稼働情報ファイル名

HWebContainerStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HWebContainerStats.txt

出力される稼働情報

Web コンテナの稼働情報ファイルに出力される内容を次に示します。

表 3-20　Web コンテナの稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）
最大同時実行スレッド数（同期，非同期，WebSocket すべての総数）	WorkerThreadCount.UpperBound
	WorkerThreadCount.LowerBound※1

稼働情報の種類	項目（ヘッダファイルに出力される文字列）
	WorkerThreadCount.HighWaterMark※2
	WorkerThreadCount.LowWaterMark※2
	WorkerThreadCount.Current※2
最大同時実行スレッド数（同期だけ）	ActiveThreadCount.UpperBound
	ActiveThreadCount.LowerBound※1
	ActiveThreadCount.HighWaterMark※2
	ActiveThreadCount.LowWaterMark※2
	ActiveThreadCount.Current※2
Web コンテナ単位またはデフォルトの実行待ちリクエスト数（同期，非同期，WebSocket すべての総数）	WaitingWorkerThreadCount.UpperBound※3
	WaitingWorkerThreadCount.LowerBound※1※3
	WaitingWorkerThreadCount.HighWaterMark※4
	WaitingWorkerThreadCount.LowWaterMark※4
	WaitingWorkerThreadCount.Current※4
Web コンテナ単位またはデフォルトの実行待ちリクエスト数（同期だけ）	WaitingRequestCount.UpperBound※3
	WaitingRequestCount.LowerBound※1※3
	WaitingRequestCount.HighWaterMark※4
	WaitingRequestCount.LowWaterMark※4
	WaitingRequestCount.Current※4
Web コンテナ単位の全体実行待ちリクエスト数	WholeWaitingRequestCount.UpperBound※5
	WholeWaitingRequestCount.LowerBound※5
	WholeWaitingRequestCount.HighWaterMark※5
	WholeWaitingRequestCount.LowWaterMark※5
	WholeWaitingRequestCount.Current※5
Web コンテナ単位の実行待ちリクエスト数の上限からあふれたリクエスト数	OverflowRequestCount.Count※3

注※1　無効になり「-1」が出力されます。

注※2　Web アプリケーション単位の最大同時実行スレッド数制御を無効に設定している場合，J2EE アプリケーションのリロード中に実行待ちになった新しいリクエストは，稼働中のスレッドとしてカウントされます。

注※3　Web アプリケーション単位の最大同時実行スレッド数制御を設定していない場合，無効になり「-1」が出力されます。Web アプリケーション単位の最大同時実行スレッド数制御を設定している場合，デフォルトの実行待ちキューの情報が出力されます。

注※4　Web アプリケーション単位の最大同時実行スレッド数制御を設定していない場合，Web コンテナ単位の実行待ちキューの情報が出力されます。Web アプリケーション単位の最大同時実行スレッド数制御を設定している場合，デフォルトの実行待ちキューの情報が出力されます。

注※5　リロード機能による入れ替え処理中の場合，Web コンテナが受け付けた実行待ちリクエスト数はカウントされません。リロード処理の完了後にカウントされます。

(11)　URL グループの稼働情報ファイルに出力される情報

URL グループの稼働情報ファイルに出力される情報について説明します。URL グループの稼働情報ファイルでは，URL グループの稼働情報を調査できます。

稼働情報ファイル名

HWebURLGroupStats_<YYYYMMDDhhmm><TZ>.csv

対応するヘッダファイル名

HWebURLGroupStats.txt

出力される稼働情報

URL グループの稼働情報ファイルに出力される内容を次に示します。

表 3-21　URL グループの稼働情報ファイルの出力内容

稼働情報の種類	項目（ヘッダファイルに出力される文字列）
最大同時実行スレッド数	ActiveThreadCount.UpperBound※1
	ActiveThreadCount.LowerBound※2
	ActiveThreadCount.HighWaterMark
	ActiveThreadCount.LowWaterMark
	ActiveThreadCount.Current
URL グループ単位の実行待ちリクエスト数	WaitingRequestCount.UpperBound※3
	WaitingRequestCount.LowerBound※2※3
	WaitingRequestCount.HighWaterMark※3
	WaitingRequestCount.LowWaterMark※3
	WaitingRequestCount.Current※3
URL グループ単位の実行待ちリクエスト数の上限からあふれたリクエスト数	OverflowRequestCount.Count
受け付けリクエスト数	RequestCount.Count※4
応答済みリクエスト数	ResponseCount.Count

注※1　Web アプリケーション単位の最大同時実行スレッド数を動的に変更した場合，設定値ではなく変更後の情報が出力されます。

注※2　無効になり「-1」が出力されます。

注※3　リロード機能による入れ替え処理中の場合，Web アプリケーションが受け付けた URL グループ単位の実行待ちリクエスト数はカウントされません。リロード処理の完了後にカウントされます。

注※4　Web コンテナが受け付けた URL グループ単位のリクエスト数が出力されます。URL グループ単位の実行待ちリクエスト数もカウントされます。Web サーバでは受信されていても，Web コンテナに転送されていないリクエスト数はカウントされません。

3.4　イベントの発行機能

この節では，監視対象にしきい値を設定し，監視対象がしきい値を超えた時にイベントを発行する機能について説明します。

この節の構成を次の表に示します。

表 3-22　この節の構成（イベントの発行機能）

分類	タイトル	参照先
解説	しきい値を設定できる監視対象	3.4.1
	イベントの発行方法	3.4.2
実装	cosminexus.xml での定義	3.4.3
設定	実行環境での設定（J2EE サーバの設定）	3.4.4

注　「運用」,「注意事項」について，この機能固有の説明はありません。

イベントの発行機能は，稼働情報の監視対象が，設定したしきい値を超えた時に，イベントを発行する機能です。イベントが発行されることで，稼働状態の異常を検知できます。

なお，イベント発行時にはメッセージが出力されます。このメッセージを利用して Management イベントを発行できます。Management イベントの詳細については，「9.　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」を参照してください。

3.4.1　しきい値を設定できる監視対象

しきい値を設定できる稼働情報の監視対象を次に示します。

- FullGC 回数

  FullGC の回数を監視します。FullGC が頻発する異常状態を検知できます。しきい値には，FullGC の回数を設定します。

  J2EE アプリケーションの実行環境およびバッチアプリケーションの実行環境で監視できます。

  Management イベントとして通知するメッセージ ID は，KDJE53850-W です。

- Web コンテナ単位の全体実行待ちリクエスト数

  Web コンテナ単位の全体実行待ちリクエスト数を監視します。Web コンテナ単位の全体実行待ちリクエスト数がリクエストキューサイズを超える前に，リクエストキューの空きが少なくなったことを検知できます。しきい値には，Web コンテナ単位の全体実行待ちリクエスト数の実行待ちリクエストキューサイズに対する Web コンテナ単位の全体実行待ちリクエスト数の割合を設定します。

  J2EE アプリケーションの実行環境で監視できます。

  Management イベントとして通知するメッセージ ID は，KDJE53862-W（上限しきい値），KDJE53863-I（下限しきい値）です。

- **Web コンテナ単位の実行待ちリクエスト数**

  Web コンテナ単位の実行待ちリクエスト数を監視します。Web コンテナ単位の実行待ちリクエスト数がリクエストキューサイズを超える前に，リクエストキューの空きが少なくなったことを検知できます。しきい値には，Web コンテナ単位の実行待ちリクエスト数の実行待ちリクエストキューサイズに対する Web コンテナ単位の実行待ちリクエスト数の割合を設定します。

  J2EE アプリケーションの実行環境で監視できます。

  Management イベントとして通知するメッセージ ID は，KDJE53864-W（上限しきい値），KDJE53865-I（下限しきい値）です。

- **Web アプリケーション単位の全体実行待ちリクエスト数**

  Web アプリケーション単位の全体実行待ちリクエスト数を監視します。Web アプリケーション単位の全体実行待ちリクエスト数がリクエストキューサイズを超える前に，リクエストキューの空きが少なくなったことを検知できます。しきい値には，Web アプリケーション単位の全体実行待ちリクエスト数の実行待ちリクエストキューサイズに対する Web アプリケーション単位の全体実行待ちリクエスト数の割合を設定します。

  J2EE アプリケーションの実行環境で監視できます。

  Management イベントとして通知するメッセージ ID は，KDJE53866-W（上限しきい値），KDJE53867-I（下限しきい値）です。

- **Web アプリケーション単位の実行待ちリクエスト数**

  Web アプリケーション単位の実行待ちリクエスト数を監視します。Web アプリケーション単位の実行待ちリクエスト数がリクエストキューサイズを超える前に，リクエストキューの空きが少なくなったことを検知できます。しきい値には，Web アプリケーション単位の実行待ちリクエスト数の実行待ちリクエストキューサイズに対する Web アプリケーション単位の実行待ちリクエスト数の割合を設定します。

  J2EE アプリケーションの実行環境で監視できます。

  Management イベントとして通知するメッセージ ID は，KDJE53868-W（上限しきい値），KDJE53869-I（下限しきい値）です。

- **URL グループ単位の実行待ちリクエスト数**

  URL グループ単位の実行待ちリクエストキューにあるリクエスト数を監視します。URL グループ単位の実行待ちリクエスト数がリクエストキューサイズを超える前に，リクエストキューの空きが少なくなったことを検知できます。しきい値には，URL グループ単位の実行待ちリクエストキューサイズに対する URL グループ単位の実行待ちリクエスト数の割合を設定します。

  J2EE アプリケーションの実行環境で監視できます。

  Management イベントとして通知するメッセージ ID は，KDJE53860-W（上限しきい値），KDJE53861-I（下限しきい値）です。

なお，Web コンテナのデフォルトの実行待ちキュー，および Web アプリケーション単位の実行待ちキューにあるリクエスト数に対して，しきい値を設定して監視する場合は，リソース枯渇監視機能を使用してください。リソース枯渇監視機能については，「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。

3.4.2　イベントの発行方法

イベントの発行方法は，しきい値を設定する監視対象によって，回数型と増減型の 2 種類があります。イベント発行方法と監視対象の対応を次の表に示します。

表 3-23　イベント発行方法と監視対象の対応

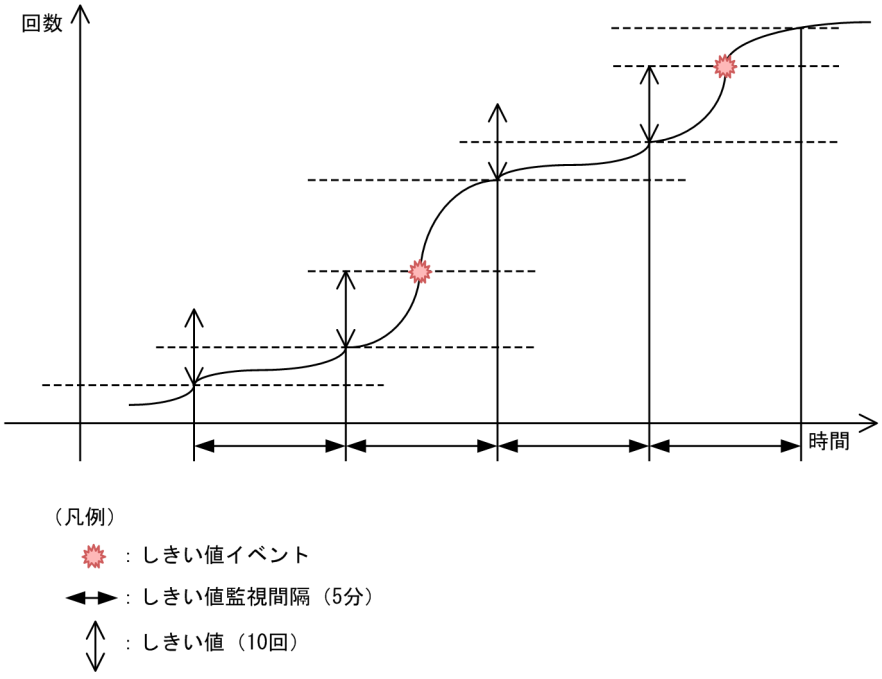
イベント発行方法	監視対象
回数型	FullGC 回数
増減型	• Web コンテナ単位の全体実行待ちリクエスト数 • Web コンテナ単位の実行待ちリクエスト数 • Web アプリケーション単位の全体実行待ちリクエスト数 • Web アプリケーション単位の実行待ちリクエスト数 • URL グループ単位の実行待ちリクエスト数

(1)　FullGC 回数の場合（回数型）

設定した監視時間（しきい値監視間隔）単位で発生した FullGC 回数をカウントし，しきい値に達した場合にイベントを発行します。

FullGC 回数の場合の，しきい値のイベント発行について次の図に示します。

図 3-3　しきい値のイベント発行（FullGC 回数の場合）



この図の場合，FullGC 回数のしきい値が 10，しきい値監視間隔が 5 分です。稼働情報の監視を開始した時点からしきい値監視間隔ごとに FullGC の回数をカウントし，一回のしきい値監視間隔で FullGC が 10 回を超えるとイベントを発行します。

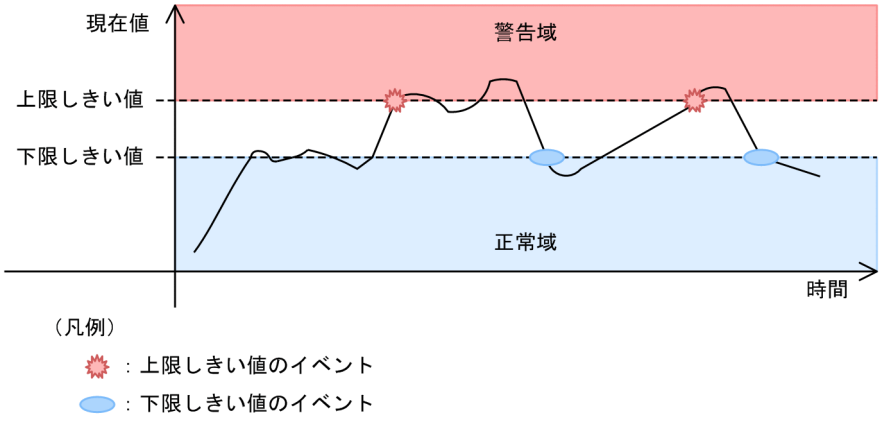
(2)　URL グループ単位の実行待ちリクエスト数の場合（増減型）

上限しきい値および下限しきい値を設定し，URL グループ単位の実行待ちリクエスト数が，それぞれのしきい値に達した時にイベントを発行します。上限しきい値のイベント，および下限しきい値のイベントは次のタイミングで発行されます。なお，上限しきい値は下限しきい値より大きく設定する必要があります。

- URL グループ単位の実行待ちリクエスト数が上限しきい値（警告域）に達した時に，上限しきい値のイベントが発行されます。
- 上限しきい値のイベントが発行されたあと，URL グループ単位の実行待ちリクエスト数が下限しきい値以下になった時に，正常域に戻ったことを示す下限しきい値のイベントが発行されます。
- 上限しきい値のイベントが発行されてから，URL グループ単位の実行待ちリクエスト数がいったん下限しきい値以下になってから再度上限しきい値に達した時に，上限しきい値のイベントが発行されます。上限しきい値のイベントが発行されたあと，下限しきい値に達しないで，再度上限しきい値に達した場合は，上限しきい値のイベントは発行されません。
- 下限しきい値のイベントが発行されてから，URL グループ単位の実行待ちリクエスト数がいったん上限しきい値以上になってから再度下限しきい値に達した時に，下限しきい値のイベントが発行されます。下限しきい値のイベントが発行されたあと，上限しきい値に達しないで，再度下限しきい値に達した場合は，下限しきい値のイベントは発行されません。

URL グループ単位の実行待ちリクエスト数の場合の，しきい値のイベント発行について次の図に示します。

図 3-4　しきい値のイベント発行（URL グループ単位の実行待ちリクエスト数の場合）



この図の場合，正常域から上限しきい値に達した時に，上限しきい値のイベントが発行されます。その後，上限しきい値以下になってから，再度上限しきい値に達しますが，この時には上限しきい値のイベントは発行されません。これ以降，下限しきい値に達した時に下限しきい値のイベントが，上限しきい値に達した時に上限しきい値のイベントが発行されます。

3.4.3　cosminexus.xml での定義

アプリケーションの開発環境で必要な cosminexus.xml での定義について説明します。

イベントの発行機能を使用するための定義は，cosminexus.xml の<war>タグ内に指定します。

cosminexus.xml でのイベントの発行機能の定義について次の表に示します。

表 3–24　cosminexus.xml でのイベントの発行機能の定義

監視対象	指定するタグ	設定内容
Web アプリケーション単位の全体実行待ちリクエスト数	<thread-control>-<thread-control-stats-monitor>-<whole-waiting-request-count>-<enabled>	Web アプリケーション単位の全体実行待ちリクエスト数の監視を有効にするかどうかを指定します。デフォルトは true です。
	<thread-control>-<thread-control-stats-monitor>-<whole-waiting-request-count>-<high-threshold>	Web アプリケーション単位の全体実行待ちリクエスト数の上限しきい値を指定します。値は，Web アプリケーション単位の全体実行待ちリクエストが格納された割合（%）で指定します。
	<thread-control>-<thread-control-stats-monitor>-<whole-waiting-request-count>-<low-threshold>	Web アプリケーション単位の全体実行待ちリクエスト数の下限しきい値を指定します。値は，Web アプリケーション単位の全体実行待ちリクエストが格納された割合（%）で指定します。
Web アプリケーション単位の実行待ちリクエスト数	<thread-control>-<thread-control-stats-monitor>-<waiting-request-count>-<enabled>	Web アプリケーション単位の実行待ちリクエスト数の監視を有効にするかどうかを指定します。デフォルトは true です。
	<thread-control>-<thread-control-stats-monitor>-<waiting-request-count>-<high-threshold>	Web アプリケーション単位の実行待ちリクエスト数の上限しきい値を指定します。値は，Web アプリケーション単位の実行待ちリクエストが格納された割合（%）で指定します。
	<thread-control>-<thread-control-stats-monitor>-<waiting-request-count>-<low-threshold>	Web アプリケーション単位の実行待ちリクエスト数の下限しきい値を指定します。値は，Web アプリケーション単位の実行待ちリクエストが格納された割合（%）で指定します。
URL グループ単位の実行待ちリクエスト数	<thread-control>-<urlgroup-thread-control>-<stats-monitor>-<waiting-request-count>-<enabled>	URL グループ単位の実行待ちリクエスト数の監視を有効にするかどうかを指定します。デフォルトは true です。
	<thread-control>-<urlgroup-thread-control>-<stats-monitor>-<waiting-request-count>-<high-threshold>	URL グループ単位の実行待ちリクエスト数の上限しきい値を指定します。値は，URL グループ単位の実行待ちリクエストが格納された割合（%）で指定します。
	<thread-control>-<urlgroup-thread-control>-<stats-monitor>-<waiting-request-count>-<low-threshold>	URL グループ単位の実行待ちリクエスト数の下限しきい値を指定します。値は，URL グループ単位の実行待ちリクエストが格納された割合（%）で指定します。

指定するタグの詳細は，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(アプリケーション／リソース定義)」の「2.2.6　War 属性の詳細」を参照してください。

3.4.4　実行環境での設定（J2EE サーバの設定）

稼働情報ファイルを使用したイベントの発行には，J2EE サーバおよび J2EE アプリケーションの設定が必要です。なお，J2EE アプリケーションの設定については，cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ参照してください。

(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。イベント発行の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に指定します。

簡易構築定義ファイルでのイベント発行の定義について次の表に示します。

表 3-25　簡易構築定義ファイルでのイベントの発行の定義

指定するパラメタ	設定内容
ejbserver.management.JVM.stats_monitor.FullGCCount.enabled	FullGC 回数の監視を有効にするかどうかを指定します。デフォルトは true です。
ejbserver.management.JVM.stats_monitor.FullGCCount.threshold	FullGC 回数の監視でのしきい値を指定します。
ejbserver.management.JVM.stats_monitor.FullGCCount.interval	FullGC 回数の監視でのしきい値監視の間隔を指定します。
ejbserver.management.webcontainer.stats_monitor.whole_waiting_request_count.enabled	Web コンテナ単位の全体実行待ちリクエスト数の監視を有効にするかどうかを指定します。デフォルトは true です。
ejbserver.management.webcontainer.stats_monitor.whole_waiting_request_count.high_threshold	Web コンテナ単位の全体実行待ちリクエスト数の上限しきい値を指定します。
ejbserver.management.webcontainer.stats_monitor.whole_waiting_request_count.low_threshold	Web コンテナ単位の全体実行待ちリクエスト数の下限しきい値を指定します。
ejbserver.management.webcontainer.stats_monitor.waiting_request_count.enabled	Web コンテナ単位の実行待ちリクエスト数の監視を有効にするかどうかを指定します。デフォルトは true です。
ejbserver.management.webcontainer.stats_monitor.waiting_request_count.high_threshold	Web コンテナ単位の実行待ちリクエスト数の上限しきい値を指定します。
ejbserver.management.webcontainer.stats_monitor.waiting_request_count.low_threshold	Web コンテナ単位の実行待ちリクエスト数の下限しきい値を指定します。

簡易構築定義ファイル，および指定するパラメタの詳細は，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「4.3　簡易構築定義ファイル」を参照してください。

(2)　J2EE アプリケーションの設定

実行環境での J2EE 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。イベント発行の定義には，WAR 属性ファイルを使用します。

WAR 属性ファイルで指定するタグは，cosminexus.xml と対応しています。cosminexus.xml での定義については，「3.4.3　cosminexus.xml での定義」を参照してください。

4 リソースの枯渇監視

この章では，アプリケーションサーバのシステムで提供するリソース枯渇監視機能について説明します。この機能は，Management イベントと組み合わせて利用することもできます。

4.1　この章の構成

リソース枯渇監視機能と参照先を次の表に示します。

表 4-1　リソース枯渇監視機能と参照先

機能	参照先
リソース枯渇監視機能の概要	4.2
リソース枯渇監視機能とリソース枯渇監視情報の出力	4.3

4.2　リソース枯渇監視機能の概要

この節では，J2EE サーバのリソースの使用率または使用数の推移を基に，リソース枯渇を監視する方法について説明します。

リソース枯渇監視機能で監視できるリソースは全部で 7 種類あります。すべてのリソースを監視することもできますが，監視するリソースを選択することもできます。リソース枯渇監視機能では，監視するリソースや，各リソースのしきい値などを設定します。なお，リソース枯渇監視で対象となるのは，J2EE サーバ，またはバッチサーバのリソースです。

J2EE サーバ，またはバッチサーバのリソース枯渇を監視する設定にしている場合，監視対象のリソースについての情報が，一定間隔でファイルに出力されます。この情報を**リソース枯渇監視情報**といいます。リソース枯渇監視情報は，監視するリソースの種類ごとに出力するかどうかを選択できます。なお，リソース枯渇監視情報を出力したファイルを，**リソース枯渇監視ログファイル**といいます。

リソース枯渇監視情報を利用すると，リソースの使用率または使用数の変化のしかたを確認できます。また，この情報は，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に，その原因を調査するためにも使用できます。

リソースの使用率または使用数が設定したしきい値を超えた場合，アラートが発生します。Management Server を使用している場合，アラートが発生すると，メッセージを出力し，Management Server に対してイベントを通知できます。このイベントを，**Management イベント**といいます。また，Management Server 側では，Management イベントが通知されたときの動作を定義しておくことで，Management イベントが発生すると自動的にアクションを実行できるようになります。このアクションを，**Management アクション**といいます。リソース枯渇監視機能と Management イベントを組み合わせて使用することによって，リソースを効率良く確実に監視できます。Management イベントと Management アクションの詳細については，「9.3　Management アクションの実行制御とは」を参照してください。

Management イベントによる処理の自動実行の設定の詳細については，「9.4　Management イベントによる処理の自動実行の設定」を参照してください。

リソース枯渇監視ログファイルの出力先については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「4.3.1　Component Container のログの取得」を参照してください。

リソース枯渇監視の設定については，「4.3.3　実行環境での設定」を参照してください。

> **参考**
>
> リソース枯渇監視では，スレッドダンプの出力先のファイルを確認します。スレッドダンプ出力先(デフォルトは，<作業ディレクトリ>/ejb/<J2EE サーバ名>)に多数のファイルがある場合，チェック処理にシステムリソース(CPU など)を多く消費します。

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

この節では，リソース枯渇監視機能で監視できるリソースの種類と，ログファイルに出力される情報について説明します。

この節の構成を次の表に示します。

表 4-2　この節の構成（ソース枯渇監視ログファイルの出力機能）

分類	タイトル	参照先
解説	監視できるリソースの種類	4.3.1
実装	cosminexus.xml での定義	4.3.2
設定	実行環境での設定	4.3.3
運用	リソース枯渇監視情報ファイルの出力先と出力情報の種類	4.3.4
	出力形式と出力内容	4.3.5

注　「注意事項」について，この機能固有の説明はありません。

4.3.1　監視できるリソースの種類

リソース枯渇監視機能では，7 種類のリソースを監視できます。7 種類すべてのリソースを監視することも，必要なリソースだけ監視することもできます。ただし，使用している OS によっては，監視できないリソースがあります。また，バッチサーバの場合，HTTP リクエスト実行待ちキュー監視およびセッション数監視はできません。

各リソースの詳細について次に説明します。

- **メモリ監視**

  メモリ監視では，JavaVM でのヒープ・メモリ領域を監視します。メモリの使用状況を監視することで，FullGC の予兆を検知できるので，監視結果をヒープサイズおよびメモリサイズのチューニングに利用できます。

  設定したしきい値を超えた場合は，KDJE34500-W が出力されます。

  ポイント

  この機能で監視しているメモリ使用率が 100%に近づいた場合に必ず FullGC が発生するわけではありません。監視した値は，FullGC が発生する可能性が高いという予兆を検知するために使用されます。

- **ファイルディスクリプタ監視**

  ファイルディスクリプタ監視では，サーバのプロセス内で開いているファイルディスクリプタを監視します。ファイルディスクリプタの数を監視することで，システムに割り当てられている，サーバのプロ

セスで使用できるファイルディスクリプタの枯渇を検知したり，見積もりで求めたファイルディスクリプタの数に到達する前に検知したりできます。

設定したしきい値に達した場合は，KDJE34520-W が出力されます。

なお，ファイルディスクリプタ監視は，Windows の場合，および AIX の場合は使用できません。

- **スレッド監視**

  スレッド監視では，サーバのプロセス内で生成するスレッド数を監視します。生成されたスレッド数を監視することで，システムに割り当てられている，サーバのプロセス内で生成できるスレッド数を超えたり，見積もりで求めたスレッド数に到達する前に検知したりできます。

  設定したしきい値に達した場合は，KDJE34540-W が出力されます。

  なお，スレッド監視は，Linux の場合は使用できません。

- **スレッドダンプ監視**

  スレッドダンプ監視では，cjdumpsv コマンドや J2EE アプリケーション実行時間の監視で出力されたスレッドダンプのファイル数を，サーバ単位で監視します。スレッドダンプのファイル数を監視することで，ファイル数が上限に達する前に検知できます。

  なお，環境変数 JAVACOREDIR を指定している場合，環境変数 JAVACOREDIR で指定したディレクトリとデフォルトの出力先ディレクトリ（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>）の両方のスレッドダンプのファイルの個数の合計を監視します。環境変数 JAVACOREDIR で指定したパスがディレクトリでないなど，ファイルのリストが取得できない場合は，デフォルトの出力先のファイル数だけを監視します。

  設定したしきい値に達した場合は，KDJE34580-W が出力されます。最大値に達した場合は，KDJE34581-E が出力されます。

  J2EE アプリケーション実行時間の監視については，「5.3　J2EE アプリケーションの実行時間の監視とキャンセル」を参照してください。

  注意事項

  スレッドダンプ監視の処理は，環境変数 JAVACOREDIR で指定されたディレクトリおよびデフォルトの出力先ディレクトリの，ファイルとディレクトリの数に比例して負荷が掛かります。そのため，それらのディレクトリに不要なファイルを配置しないか，スレッドダンプの監視間隔を長めに設定してください。

- **HTTP リクエスト実行待ちキュー監視**

  HTTP リクエスト実行待ちキュー監視では，Web アプリケーション単位の同時実行スレッド数の制御で実行待ちキューにあるリクエストの数を監視します。監視する実行待ちキューは，Web アプリケーションごとの実行待ちキューと，デフォルトの実行待ちキューです。監視は，J2EE サーバ単位，または Web アプリケーション単位でできます。HTTP リクエストの Web アプリケーション単位およびデフォルトの実行待ちキューを監視することで，リクエストがそれらの実行待ちキューからあふれる前の状態を検知できます。

  ただし，Web アプリケーション単位およびデフォルトの実行待ちキューサイズに「0」が指定されている場合は監視しません。

  なお，Web アプリケーション単位の実行待ちキューサイズが動的に変更された場合でも，しきい値は最初に設定した値から変更されることはありません。

設定したしきい値に達した場合は，KDJE34621-W が出力されます。

同時実行スレッド数については，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「2.15　Web アプリケーション単位での同時実行スレッド数の制御」を参照してください。

なお，URL グループ単位の実行待ちキューにあるリクエストの数を監視する場合は，稼働情報収集機能を使用してください。稼働情報収集機能を使用したしきい値の設定方法については，「3.4.1　しきい値を設定できる監視対象」を参照してください。

- **セッション数監視**

  セッション数監視では，Web アプリケーション上で生成されるセッション数を監視します。セッション数を監視することで，セッション数の増加を検知できます。

  ただし，セッション数の設定で作成できるセッション数に「0」が指定されている場合は，監視しません。

  設定したしきい値に達した場合は，KDJE34640-W が出力されます。

- **コネクションプール監視**

  コネクションプール監視では，コネクションプールの使用状態を監視します。コネクションプールの使用状態を監視することで，コネクションの枯渇を事前に検知できるので，コネクションプールのチューニングに利用できます。

  ただし，コネクションプールが無効となっている場合は，監視できません。また，コネクションプールの最大値が無制限の場合は，監視はしますが，Management イベントは通知されません。

  設定したしきい値を超えた場合は，KDJE34660-W が出力されます。最大数に達した場合は，KDJE34661-W が出力されます。

  コネクションプールについては，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「3.14.1　コネクションプーリング」を参照してください。

4.3.2　cosminexus.xml での定義

アプリケーションの開発環境で必要な cosminexus.xml での定義について説明します。

リソース枯渇監視機能を使用するための定義は，cosminexus.xml の<war>タグ内に指定します。

cosminexus.xml でのリソース枯渇監視機能の定義について次の表に示します。

表 4-3　cosminexus.xml でのリソース枯渇監視機能の定義

リソース種別	指定するタグ	設定内容
HTTP リクエスト実行待ちキュー	<war>－<thread-control>－<resource-watcher>タグ下次のタグ • <watcher-enabled>タグ • <watcher-threshold>タグ • <watcher-interval>タグ • <watcher-writefile-enabled>タグ	J2EE アプリケーション単位で，Web アプリケーションごとの実行待ちキューの場合の HTTP リクエスト実行待ちキュー監視の次の設定をします。 • HTTP リクエスト実行待ちキュー監視によるアラート出力の有無 • 監視のしきい値 • 監視の間隔

リソース種別	指定するタグ	設定内容
		• リソース枯渇監視ログファイルの出力の有無
セッション数	<http-session>－<resource-watcher>タグ下の次のタグ • <watcher-enabled>タグ • <watcher-threshold>タグ • <watcher-interval>タグ • <watcher-writefile-enabled>タグ	J2EE アプリケーション単位でセッション数監視の次の設定をします。 • セッション数監視によるアラート出力の有無 • 監視のしきい値 • 監視の間隔 • リソース枯渇監視ログファイルの出力の有無

指定するタグの詳細は，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(アプリケーション／リソース定義)」の「2.2.6　War 属性の詳細」を参照してください。

4.3.3　実行環境での設定

リソース枯渇監視機能を使用できるのは，J2EE サーバ，またはバッチサーバです。J2EE サーバの場合，監視対象にできるリソースは，メモリ，ファイルディスクリプタ，スレッド数，スレッドダンプ，HTTP リクエスト実行待ちキュー，セッション数，およびコネクションプールです。バッチサーバの場合，監視対象にできるリソースは，メモリ，ファイルディスクリプタ，スレッド数，スレッドダンプ，およびコネクションプールです。なお，Windows，AIX では，ファイルディスクリプタは監視できません。また，Linux ではスレッド数を監視できません。

リソース枯渇監視機能を使用する場合，次の設定が必要です。

- J2EE サーバ
- Connector 属性ファイル
- J2EE アプリケーション

  cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ参照してください。

(1)　J2EE サーバの設定

J2EE サーバの設定は，簡易構築ファイルで実施します。リソース枯渇監視機能の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に指定します。なお，バッチサーバ単位での設定の場合，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。

簡易構築定義ファイルでのリソース枯渇監視機能の定義には，次の二つがあります。

- リソース枯渇監視機能の有効化
- リソース種別ごとの監視の設定

(a) リソース枯渇監視機能の有効化

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.watch.enabled パラメタでリソース枯渇監視機能を使用するかどうかを指定します。デフォルトの設定では，リソース枯渇監視機能の使用が有効になっていて，すべてのリソースが監視されるようになっています。なお，リソース枯渇監視機能を無効にしたい場合には，ejbserver.watch.enabled パラメタに false（リソース枯渇監視機能を無効にする）を指定してください。

(b) リソース種別ごとの監視の設定

簡易構築定義ファイルでのリソース種別ごとの監視の定義について次の表に示します。

表 4-4 簡易構築定義ファイルでのリソース枯渇監視機能の定義

リソース種別	指定するパラメタ	設定内容
メモリ	• ejbserver.watch.memory.enabled • ejbserver.watch.memory.threshold • ejbserver.watch.memory.interval • ejbserver.watch.memory.writefile.enabled	J2EE サーバ単位，またはバッチサーバ単位で，メモリ監視の設定をします。 メモリ監視によるアラート出力の有無，監視のしきい値，監視の間隔，リソース枯渇監視ログファイルの出力の有無を指定します。 また，リソース枯渇監視ログファイルの面数，サイズを指定します。※1※2
ファイルディスクリプタ	• ejbserver.watch.fileDescriptor.enabled • ejbserver.watch.fileDescriptor.threshold • ejbserver.watch.fileDescriptor.interval • ejbserver.watch.fileDescriptor.writefile.enabled	J2EE サーバ単位，またはバッチサーバ単位で，ファイルディスクリプタ監視の設定をします。 ファイルディスクリプタ監視によるアラート出力の有無，監視のしきい値，監視の間隔，リソース枯渇監視ログファイルの出力の有無を指定します。 また，リソース枯渇監視ログファイルの面数，サイズを指定します。※2
スレッド数	• ejbserver.watch.thread.enabled • ejbserver.watch.thread.threshold • ejbserver.watch.thread.interval • ejbserver.watch.thread.writefile.enabled	J2EE サーバ単位で，スレッド数監視の設定をします。 スレッド数監視によるアラート出力の有無，監視のしきい値，監視の間隔，リソース枯渇監視ログファイルの出力の有無を指定します。 また，リソース枯渇監視ログファイルの面数，サイズを指定します。※2
スレッドダンプ	• ejbserver.watch.threaddump.enabled • ejbserver.watch.threaddump.threshold • ejbserver.watch.threaddump.interval • ejbserver.watch.threaddump.writefile.enabled • ejbserver.server.threaddump.filenum	J2EE サーバ単位，またはバッチサーバ単位で，スレッドダンプ監視の設定をします。 スレッドダンプ監視によるアラート出力の有無，監視のしきい値，監視の間隔，リソース枯渇監視ログファイルの出力の有無を指定します。 また，リソース枯渇監視ログファイルの面数，サイズを指定します。※2
HTTP リクエスト実行待ちキュー	• ejbserver.watch.defaultRequestQueue.enabled	J2EE サーバ単位で，デフォルトの実行待ちキューの場合の HTTP リクエスト実行待ちキュー監視の設定をします。

リソース種別	指定するパラメタ	設定内容
	• ejbserver.watch.defaultRequestQueue.threshold • ejbserver.watch.defaultRequestQueue.interval • ejbserver.watch.defaultRequestQueue.writefile.enabled	HTTP リクエスト実行待ちキュー監視によるアラート出力の有無，監視のしきい値，監視の間隔，リソース枯渇監視ログファイルの出力の有無を指定します。 また，リソース枯渇監視ログファイルの面数，サイズを指定します。※2
セッション数	−	J2EE サーバ単位で，リソース枯渇監視ログファイルの面数，サイズを指定します。※2
コネクションプール	−	J2EE サーバ単位，またはバッチサーバ単位で，リソース枯渇監視ログファイルの面数，サイズを指定します。※2

（凡例） −：なし

注※1　メモリ監視する場合は，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内で，JavaVM 起動パラメタを定義します。JavaVM 起動パラメタには，-XX:MetaspaceSize と-XX:MaxMetaspaceSize を指定し，メモリサイズは同じ値にしてください。メモリサイズに異なる値を指定した場合，アラートが出力されずに FullGC が発生することがあります。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:MetaspaceSiz=（設定値）
　-XX:MaxMetaspaceSize=（設定値）

また，FullGC の予兆を検知するためには，JavaVM の Old 領域，New 領域のサイズをチューニングする必要があります。

注※2　リソース枯渇監視ログファイルのファイルの面数，サイズは，簡易構築定義ファイルの次に示すパラメタで指定します。

- ejbserver.logger.channels.define.<監視リソース名>WatchLogFile.filenum
  リソース種別ごとにリソース枯渇監視ログファイルのファイル数を指定します。
- ejbserver.logger.channels.define.<監視リソース名>WatchLogFile.filesize
  リソース種別ごとにリソース枯渇監視ログファイルのファイルサイズを指定します。

これらのパラメタの監視リソース名には，リソース種別ごとのチャネル名を指定します。

- メモリの場合：MemoryWatchLogFile
- ファイルディスクリプタの場合：FileDescriptorWatchLogFile
- スレッド数の場合：ThreadWatchLogFile
- スレッドダンプの場合：ThreaddumpWatchLogFile
- HTTP リクエスト実行待ちキューの場合：RequestQueueWatchLogFile
- セッション数の場合：HttpSessionWatchLogFile
- コネクションプールの場合：ConnectionPoolWatchLogFile

簡易構築定義ファイル，および指定するパラメタの詳細は，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「4.3　簡易構築定義ファイル」を参照してください。

(2) Connector 属性ファイルの設定

リソースアダプタ単位でコネクションプール監視の設定をします。リソースアダプタのプロパティ設定は，サーバ管理コマンドおよび Connector 属性ファイルで実施します。

Connector 属性ファイルの<property-name>タグとして指定する<WatchEnabled>，<WatchThreshold>，<WatchInterval>，<WatchWriteFileEnabled>タグで，コネクションプール監視によるアラート出力の有無，監視のしきい値，監視の間隔，リソース枯渇監視ログファイルの出力の有無を指定します。

指定するタグの詳細は，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(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。

サーバ管理コマンドおよび Connector 属性ファイルでのリソースアダプタのプロパティの設定については，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「3.5.1 J2EE リソースのプロパティの設定手順」を参照してください。

(3) J2EE アプリケーションの設定

実行環境での J2EE 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。リソース枯渇監視機能の定義には，WAR 属性ファイルを使用します。

WAR 属性ファイルで指定するタグは，cosminexus.xml と対応しています。cosminexus.xml での定義については，「4.3.2 cosminexus.xml での定義」を参照してください。

4.3.4 リソース枯渇監視情報ファイルの出力先と出力情報の種類

(1) リソース枯渇監視情報の出力先

デフォルトの設定の場合，リソース枯渇監視情報ファイルは，J2EE サーバの起動時に次のフォルダに作成されます。

Windows の場合

<ejb.server.log.directory>¥watch¥以下に出力されます。

UNIX の場合

<ejb.server.log.directory>/watch/以下に出力されます。

注意事項

<ejb.server.log.directory>は，J2EE サーバのユーザ定義（usrconf.cfg ファイル中の ejb.server.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，Windows の場合，「<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」，UNIX の場合，「/opt/Cosminexus/CC/server/public/ejb/<サーバ名称>/logs」です。

<ejb.server.log.directory>を変更することで，リソース枯渇監視情報の出力先も変更されます。

(2) 出力情報の種類

リソース枯渇監視情報には，次の表に示す情報が出力されます。なお，Windows，または AIX の場合，ファイルディスクリプタの数は監視できません。また，Linux の場合，スレッド監視は使用できません。

表 4-5 リソース枯渇監視情報に出力される情報

情報の種類	ファイル名	情報の概要
メモリ枯渇監視情報	cjmemorywatch	JavaVM のヒープおよびメモリ領域の使用状況の監視結果が出力されます。 この情報は，usrconf.properties の ejbserver.watch.memory.writefile.enabled キーに true が設定されている場合に出力されます。
ファイルディスクリプタ枯渇監視情報	cjfiledescriptorwatch	J2EE サーバプロセスが使用しているファイルディスクリプタの数の監視結果が出力されます。 この情報は，usrconf.properties の ejbserver.watch.fileDescriptor.writefile.enabled キーに true が設定されている場合に出力されます。
スレッド枯渇監視情報	cjthreadwatch	J2EE サーバプロセスが生成したスレッド数の監視結果が出力されます。 この情報は，usrconf.properties の ejbserver.watch.thread.writefile.enabled キーに true が設定されている場合に出力されます。
スレッドダンプ枯渇監視情報	cjthreaddumpwatch	出力されたスレッドダンプのファイル数の監視結果が出力されます。 この情報は，usrconf.properties の ejbserver.watch.threaddump.writefile.enabled キーに true が設定されている場合に出力されます。
HTTP リクエスト実行待ちキュー枯渇監視情報	cjrequestqueuewatch	HTTP リクエスト実行待ちキューの監視結果として，次の 2 種類の実行待ちキューが監視され，結果が出力されます。 • Web アプリケーション単位の最大同時実行スレッド数が設定されていない Web アプリケーションの実行待ちキューが監視されて，リクエスト格納数の監視結果が出力されます。 この情報は，usrconf.properties の ejbserver.watch.defaultRequestQueue.writefile.enabled キーに true が設定されている場合に出力されます。 • Web アプリケーション単位の最大同時実行スレッド数が設定されている Web アプリケーションの実行待ちキューが監視されて，リクエスト格納数の監視結果が出力されます。 この情報は，アプリケーションの属性として，<war><hitachi-war-property><thread-control><resource-watcher>タグ下

情報の種類	ファイル名	情報の概要
		の<watcher-writefile-enabled>タグに true が設定されている場合に出力されます。 なお，Web アプリケーション単位の最大同時実行スレッド数の制御が有効な場合に，これらの監視が実行されます。 URL グループ単位の実行待ちキューの情報については，HTTP リクエスト実行待ちキュー枯渇監視情報には出力されません。URL グループ単位の実行待ちキューの稼働情報については，稼働情報収集機能を使用して監視してください。稼働情報収集機能については，「3. 稼働情報の監視（稼働情報収集機能）」を参照してください。
セッション数枯渇監視情報	cjhttpsessionwatch	Web アプリケーション単位のセッション数の監視結果が出力されます。 この情報は，アプリケーションの属性として，<war><hitachi-war-property><http-session><resource-watcher>タグ下の<watcher-writefile-enabled>タグに true が設定されている場合に出力されます。
コネクションプール枯渇監視情報	cjconnectionpoolwatch	リソースアダプタ単位のコネクションプール数の監視結果が出力されます。 この情報は，リソースアダプタのプロパティとして，WatchWriteFileEnabled プロパティに true が設定されている場合に出力されます。

4.3.5 出力形式と出力内容

リソース枯渇監視情報は，出力レコードが CSV 形式ファイルで出力されます。

出力形式を次の図に示します。

図 4-1 リソース枯渇監視情報の出力形式

```
     yyyy/mm/dd hh:mm:ss.sss       pid      tid      message-id message(LANG=ja)
0000 2004/99/99 99:99:99.999 HEJB 00000000 00000000
 ,"Title1","Title2","Title3",…
0000 2004/99/99 99:99:99.999 HEJB 00000000 00000000             ,"Data1","Data2","Data3",…
0000 2004/99/99 99:99:99.999 HEJB 00000000 00000000             ,"Data1","Data2","Data3",…
```

出力形式について説明します。

- それぞれのカラムの出力内容がわかるように，タイトル行が，100 行ごとに出力されます。
- タイトルには，リソース共通の情報として，次の項目が出力されます。これらの項目は，図中の，"Title1"，"Title2"，"Title3"…の個所に出力されます。
  - Rate：割合（使用率）
  - Current：現在値
  - Max：最大値

- Threshold：しきい値

このほか，リソースによって，必要な情報が出力されます。詳細は，それぞれのリソースの出力内容を参照してください。

- しきい値判定の計算結果，現在値情報，設定したしきい値の順で出力されます。しきい値判定の結果が複数ある場合は，複数出力されます。
- しきい値が%で設定されている場合，しきい値の計算結果も%で出力されます。例えば，計算結果が0.752 であった場合，しきい値が%で指定されているときには，75.2 と出力されます。
- しきい値が絶対値の場合は，しきい値の計算結果は出力されません。
- しきい値として上限が設定されていない場合，しきい値の計算結果は，「-」と出力されます。
- 値が小数になる場合は，小数点以下第 2 位が四捨五入され，小数点以下第 1 位までが出力されます。なお，しきい値判定にも，小数点以下第 2 位が四捨五入され，小数点以下第 1 位までにした値が利用されます。

なお，ファイルに出力される内容については，usrconf.properties，アプリケーションの属性，またはリソースアダプタの属性の設定内容に従います。

それぞれの出力情報について説明します。

(1) メモリ枯渇監視情報

メモリ枯渇監視情報には，次の表に示す情報が出力されます。

表 4-6 メモリ枯渇監視情報の出力内容

出力タイトル文字列	出力内容
Rate1	SerialGC が有効な場合： Old 領域のメモリ使用率が出力されます。 単位は%です。 Old 領域のメモリ使用率は，次の計算式で算出されます。 • Old 領域のメモリ使用率 ＝ Old 領域消費サイズ / Old 領域合計サイズ×100 （Rate1 ＝ (Total [Old] - Free [Old]) / Total [Old]×100） G1GC が有効な場合： Java ヒープ領域メモリ使用率が出力されます。 単位は%です。 Java ヒープ領域のメモリ使用率は，次の計算式で算出されます。 • Java ヒープ領域メモリ使用率[%] ＝ Java ヒープ領域消費サイズ／Java ヒープ領域サイズ ×100
Rate2	SerialGC が有効な場合： Old 領域最大空きメモリに対する New 領域の合計メモリ率が出力されます。 単位は%です。 Old 領域最大空きメモリに対する New 領域の合計メモリ率は，次の計算式で算出されます。

出力タイトル文字列	出力内容
	• Old 領域最大空きメモリに対する New 領域の合計メモリ率 = New 領域合計サイズ / Old 領域最大空きサイズ×100 (Rate2 = Total [New] / (Max [Old] - (Total [Old] - Free [Old]))×100) なお，この値は 100 を超える場合があります。 G1GC が有効な場合： この情報は使用しません。常に-1 が出力されます。
Rate3	Metaspace 領域のメモリ使用率が出力されます。 単位は%です。 Metaspace 領域のメモリ使用率は，次の計算式で算出されます。 • Metaspace 領域のメモリ使用率 = Metaspace 領域消費サイズ / Metaspace 領域最大サイズ×100 (Rate3 = (Total [Permanent] - Free [Permanent]) / Total [Permanent]×100)
Free [New]	New 領域の空きメモリサイズが出力されます。 単位はバイトです。
Total [New]	New 領域の合計メモリサイズが出力されます。 単位はバイトです。
Max [New]	SerialGC が有効な場合： New 領域の最大メモリサイズが出力されます。 単位はバイトです。 G1GC が有効な場合： この情報は使用しません。常に-1 が出力されます。
Free [Old]	Old 領域の空きメモリサイズが出力されます。 単位はバイトです。
Total [Old]	Old 領域の合計メモリサイズが出力されます。 単位はバイトです。
Max [Old]	SerialGC が有効な場合： Old 領域の最大メモリサイズが出力されます。 単位はバイトです。 G1GC が有効な場合： Java ヒープ領域の最大メモリサイズが出力されます。
Free [Permanent]	Metaspace 領域の空きメモリサイズが出力されます。 単位はバイトです。
Total [Permanent]	Metaspace 領域の合計メモリサイズが出力されます。 単位はバイトです。
Max [Permanent]	Metaspace 領域の最大メモリサイズが出力されます。 単位はバイトです。
Threshold	しきい値が出力されます。 単位は%です。

注意事項

- メモリ枯渇監視機能は，十分にメモリ設計をして，FullGC が起こりにくいようにチューニングされたシステムを前提としています。メモリ枯渇監視機能は，このようにチューニングされたシステムであっても，FullGC が発生してアプリケーションが一時的に停止する予兆を検知することを目的としています。したがって，メモリチューニングが十分でない場合は，不要なアラートが出力されることがありますので，ご注意ください。

  メモリ枯渇監視する場合は，Metaspace 領域の最大サイズと初期サイズに同じ値を設定する必要があります。具体的には usrconf.cfg の-XX:MaxMetaspaceSize と-XX:MetaspaceSize を同じ値にします。異なる値を設定した場合，Metaspace 領域の領域拡張でもアラートが出力される場合があります。そのほかのチューニングパラメタについては，Oracle の Web サイトなどの情報を参照してください。

  また，この機能で監視しているメモリ使用率が 100%に近づいた場合に必ず FullGC が発生するわけではありません。監視した値は，FullGC が発生する可能性が高いという予兆を検知するために使用されます。

- アプリケーションサーバ 09-00 から FullGC が起こりにくいように Java ヒープを有効活用するようになりました。これによって，Java ヒープの使用率が高いままシステムが稼働し続け，不要なアラートが出力されることがあります。Rate1 のアラート出力を抑止したい場合，しきい値を調整してください。Rate2 のアラート出力を抑止したい場合，usrconf.properties に，ejbserver.watch.memory.rate2alert.enabled=false を指定してください。

参考

JavaVM でのヒープおよびメモリには，次の 3 種類の領域があります。

- New 領域

  Eden と Survivor に該当する領域です。新しいオブジェクトが格納されます。

- Old 領域

  Tenured に該当する領域です。長時間存在するオブジェクトが格納されます。

- Metaspace 領域

  JavaVM にロードされたクラスが格納される領域です。

  アプリケーションサーバ 09-70 以降，Metaspace 領域となりました。

  アプリケーションサーバ 09-70 より前は Permanent 領域であったため，旧バージョンからの互換性を重視し，Metaspace 領域のメモリサイズを出力するメモリ監視出力の出力タイトル文字列内の表記は Permanent となっています。

メモリ監視結果情報には，これらの領域ごとに，その時点での空き領域，消費されている領域，および最大領域のサイズが出力されます。

メモリ監視を設定した場合，次の場合にメッセージが出力されます。

- Old 領域の合計サイズに対する Old 領域の消費サイズがしきい値以上の場合（SerialGC が有効な場合）

  「Old 領域消費サイズ／Old 領域合計サイズ×100≧しきい値（%）」の場合に出力されます。
- Old 領域の最大空きサイズに対する New 領域の合計サイズがしきい値以上の場合（SerialGC が有効な場合）

  「New 領域合計サイズ／Old 領域最大空きサイズ×100≧しきい値（%）」の場合に出力されます。
- Java ヒープ領域の合計サイズに対する Java ヒープ領域の消費サイズがしきい値以上の場合（G1GC が有効な場合）

  「Java ヒープ領域消費サイズ／Java ヒープ領域合計サイズ×100≧しきい値（%）」の場合に出力されます。
- Metaspace 領域の最大サイズに対する Metaspace 領域の消費サイズがしきい値以上の場合

  「Metaspace 領域消費サイズ／Metaspace 領域最大サイズ×100≧しきい値（%）」の場合に出力されます。

(2) ファイルディスクリプタ枯渇監視情報

ファイルディスクリプタ監視結果情報には，次の表に示す情報が出力されます。なお，Windows，または AIX の場合，ファイルディスクリプタの数は監視できません。

表 4-7 ファイルディスクリプタ枯渇監視情報の出力内容

出力タイトル文字列	出力内容
Current	J2EE サーバプロセスが使用しているファイルディスクリプタ数が出力されます。
Max	プロセスに割り当て可能なファイルディスクリプタの数が出力されます。
Threshold	しきい値が出力されます。

(3) スレッド枯渇監視情報

スレッド枯渇監視情報には，次の表に示す情報が出力されます。

表 4-8 スレッド枯渇監視情報の出力内容

出力タイトル文字列	出力内容
Current	自プロセスが使用しているスレッド数が出力されます。
Max	• Windows の場合 常に"-"が出力されます。 • UNIX の場合 プロセスに生成できるスレッド数が出力されます。

出力タイトル文字列	出力内容
Threshold	しきい値が出力されます。

(4) スレッドダンプ枯渇監視情報

スレッドダンプ枯渇監視情報には，次の表に示す情報が出力されます。

表 4-9 スレッドダンプ枯渇監視情報の出力内容

出力タイトル文字列	出力内容
Rate	現在のスレッドダンプのファイル数が，最大値に占める割合が出力されます。 単位は%です。
Current	スレッドダンプのファイル数の現在値が出力されます。
Max	スレッドダンプのファイル数の上限値が出力されます。
Threshold	しきい値が出力されます。 単位は%です。

(5) HTTP リクエスト実行待ちキュー枯渇監視情報

HTTP リクエスト実行待ちキューとは，Web アプリケーションの同時実行スレッドに対する，Web アプリケーション単位およびデフォルトの実行待ちキューのことです。

HTTP リクエスト実行待ちキュー枯渇監視情報には，次の表に示す情報が出力されます。なお，この枯渇監視情報には，次の 2 種類の情報が含まれます。

- J2EE サーバ単位でデフォルトのリクエスト実行待ちキュー格納数の監視結果
- Web アプリケーション単位のリクエスト実行待ちキュー格納数の監視結果

表 4-10 HTTP リクエスト実行待ちキュー枯渇監視情報の出力内容

出力タイトル文字列	出力内容
J2eeApplicationName	J2EE アプリケーション名が出力されます。 デフォルトのリクエスト実行待ちキューの監視結果の場合は，"-"が出力されます。
ContextRootName	コンテキストルート名が出力されます。 デフォルトのリクエスト実行待ちキューの監視結果の場合は，"-"が出力されます。
Rate	現在のキュー格納数の最大値に占める割合が出力されます。 単位は%です。
Current	キュー格納数の現在値が出力されます。
Max	Web アプリケーション単位またはデフォルトの実行待ちキューのリクエスト格納数の最大値が出力されます。
Threshold	しきい値が出力されます。

出力タイトル文字列	出力内容
	単位は%です。

(6) セッション数枯渇監視情報

セッション数枯渇監視情報には，次の表に示す情報が出力されます。

表 4-11 セッション数枯渇監視情報の出力内容

出力タイトル文字列	出力内容
J2eeApplicationName	J2EE アプリケーション名が出力されます。
ContextRootName	コンテキストルート名が出力されます。
Rate	現在のセッション作成数の最大値に占める割合が出力されます。 単位は%です。 最大値が設定されていない場合は，"-"が出力されます。
Current	セッション数の現在値が出力されます。
Max	セッション数の最大値が出力されます。 設定されていない場合は"-"が出力されます。
Threshold	しきい値が出力されます。 単位は%です。

(7) コネクションプール枯渇監視情報

コネクションプール枯渇監視情報には，次の表に示す情報が出力されます。

表 4-12 コネクションプール枯渇監視情報の出力内容

出力タイトル文字列	出力内容
ResourceName	ユーザが付けたリソースアダプタ名が次の形式で出力されます。 **リソースアダプタを直接 J2EE サーバにデプロイした場合** 　<リソースアダプタ名> **リソースアダプタを J2EE アプリケーションに含めてデプロイした場合** 　<J2EE アプリケーション名>：<リソースアダプタ名> **リソースアダプタを J2EE アプリケーションに含めてデプロイして，テストモードで開始した場合** 　TEST#<J2EE アプリケーション名>：<リソースアダプタ名>
Rate	コネクションプールの使用率が出力されます。 単位は%です。 コネクション数の最大値が無制限の場合，"-"が出力されます。
Active	使用中のコネクション数が出力されます。

出力タイトル文字列	出力内容
Free	未使用のコネクション数が出力されます。
Current	コネクション数の現在値が出力されます※。
Min	コネクション数の最小値が出力されます。
Max	コネクション数の最大値が出力されます。 コネクションプールが無制限の場合，"-1"が出力されます。
Threshold	しきい値が出力されます。 単位は%です。
All	コネクションの総数（コネクションプール管理内のコネクションとコネクションプール管理外のコネクションの総数）が出力されます※。

注※　コネクションを接続するタイミングによっては，一時的に All よりも多い値が Current に出力されることがあります。

5 J2EE アプリケーションの運用

この章では，J2EE アプリケーションの運用について説明します。

J2EE アプリケーションの運用では，J2EE アプリケーションの実行時間の監視，サービスの閉塞，停止，入れ替え，および J2EE アプリケーションからのネットワークリソースへのアクセスができます。

5.1　この章の構成

J2EE アプリケーションの運用と参照先を次の表に示します。

表 5-1　J2EE アプリケーションの運用と参照先

機能	参照先
J2EE アプリケーションの運用の概要	5.2
J2EE アプリケーションの実行時間の監視とキャンセル	5.3
J2EE アプリケーションのサービスの閉塞	5.4
J2EE アプリケーションの停止	5.5
J2EE アプリケーションの入れ替え	5.6
J2EE アプリケーションからのネットワークリソースへのアクセス	5.7

5.2　J2EE アプリケーションの運用の概要

J2EE アプリケーションの運用とは，J2EE サーバにデプロイした J2EE アプリケーションに対して，日常業務の中で定型的な運用を実施したり，業務状況の変化に応じて適切に設定を変更してチューニングを実施したりすることです。

ここでは，J2EE アプリケーションを運用する作業の概要を説明します。

5.2.1　タイムアウトしたリクエストをキャンセルする

J2EE アプリケーションで実行しているリクエストが無限ループなどの原因によって制御できなくなった場合，リクエストを強制的にキャンセルして，リソースを解放する必要があります。あらかじめ J2EE アプリケーション内のメソッドの実行状態を監視しておくことで，予定した時間内に完了しないリクエストに対して，キャンセルを実行できます。

5.2.2　J2EE アプリケーションを停止する

一日のうち特定の時間だけクライアントにサービスを提供する J2EE アプリケーションでは，毎日決まった時間に J2EE アプリケーションを停止して，次の日に再度開始する作業が必要です。また，24 時間連続稼働させているサービスをメンテナンスする場合は，一度にすべての J2EE アプリケーションを停止するのではなく，一部ずつ停止させて，サービスを提供し続けられるような運用を検討する必要があります。J2EE アプリケーションを停止するとき，まず，リクエストの受け付けを停止して，すでに受け付けたリクエストについては処理を実行してから J2EE アプリケーションを停止させる方法を，**サービス閉塞**といいます。日常運用の中で J2EE アプリケーションを停止する場合，サービス閉塞を適切に実行することで，安全にサービスを停止できます。実行できるサービス閉塞の方法は，J2EE アプリケーションの構成やシステムの構成によって異なります。

5.2.3　J2EE アプリケーションを強制停止する

日常運用で J2EE アプリケーションを停止する場合や，トラブルが発生して即座に J2EE アプリケーションを停止したい場合などに，J2EE アプリケーション内のリクエストが完了していないことが原因で停止処理が完了しないことがあります。この場合，J2EE アプリケーションを強制的に停止する必要があります。

J2EE アプリケーションの強制停止では，完了しないリクエストを強制的に終了させます。これによって，J2EE アプリケーションを停止できる状態にできます。

5.2.4　J2EE アプリケーションを入れ替える

運用中の J2EE アプリケーションをバージョンアップしたりメンテナンスしたりする場合，J2EE アプリケーションの入れ替え作業を実施します。

J2EE アプリケーションを入れ替える場合にサービスを停止したくないときは，適切なサービス閉塞をする必要があります。CTM を利用して J2EE サーバへのクライアントからのリクエストを制御して，システム全体を停止させないで J2EE アプリケーションを入れ替えられます。これによって，クライアントからのリクエストを受け付けながら，オンライン状態で J2EE アプリケーションを入れ替えられます。なお，CTM で制御できるのは，RMI-IIOP 通信を使用する，Stateless Session Bean に対するリモートインタフェース呼び出しのリクエストだけです。

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「2.2.1　製品と構成ソフトウェアの対応」を参照してください。

5.2.5　J2EE アプリケーションからネットワークリソースにアクセスする

J2EE アプリケーションから，ネットワークリソースのパスを UNC またはネットワークドライブを使用してアクセスできます。なお，この機能は Windows の場合に使用します。UNIX の場合，特に設定をしなくても，J2EE アプリケーションからネットワークリソースにアクセスできます。

5.2.6　J2EE アプリケーションの運用の作業と参照先

J2EE アプリケーションの運用の作業と参照先を，表 5-2 から表 5-5 に示します。なお，J2EE アプリケーションからのネットワークリソースまたはネットワークドライブへのアクセスについては，事前に設定するだけで使用できるため，運用時の作業はありません。

表 5-2　J2EE アプリケーションの実行時間の監視とキャンセル

作業内容	手段	作業概要	参照先マニュアル	参照先
J2EE アプリケーションの実行状態の確認	サーバ管理コマンド（cjlistthread）	実行中の J2EE アプリケーションの実行状態を確認します。	このマニュアル	5.3.11
タイムアウトが発生したリクエストのキャンセル	サーバ管理コマンド（cjstopthread）	J2EE アプリケーション内で実行されているリクエストのうち，終了しないリクエストをキャンセルします。		5.3.12

表 5-3　J2EE アプリケーションの閉塞

作業内容	手段	作業概要	参照先マニュアル	参照個所
負荷分散機を使用したサービス閉塞	負荷分散機の機能	Web フロントに負荷分散機を使用している場合に，次の方法でサービスを閉塞します。 • 負荷分散機のリクエストの振り分け先を変更する • 負荷分散機を停止する • サービスを構成する一部の Web アプリケーションだけを停止する	このマニュアル	5.4.3
CTM を利用したサービス閉塞	運用管理コマンド（mngsvrutil）	CTM を使用しているバックシステムで，ホスト内の J2EE アプリケーションを一度に停止したり，キューを共有する J2EE アプリケーションを一度に停止したりする場合，J2EE アプリケーションのスケジュールキューに対して直接閉塞を実行します。	機能解説 拡張編	3.7.4

表 5-4　J2EE アプリケーションの停止

作業内容	手段	作業概要	参照先マニュアル	参照個所
J2EE アプリケーションの停止	サーバ管理コマンド（cjstopapp）	J2EE アプリケーションを通常停止，または強制停止します。通常停止にタイムアウトを設定して，強制停止を自動で実行することもできます。	このマニュアル	5.5.7

表 5-5　J2EE アプリケーションの入れ替えと保守

作業内容	手段	作業概要	参照先マニュアル	参照個所
J2EE アプリケーションの通常入れ替え	サーバ管理コマンド（cjstopapp，cjimportapp※）	J2EE アプリケーションを停止してから新しいアプリケーションに入れ替えます。入れ替え後，J2EE アプリケーションを再開始します。	このマニュアル	5.6.3(1)
リデプロイによる J2EE アプリケーションの入れ替え	サーバ管理コマンド（cjreplaceapp）	リデプロイによって J2EE アプリケーションを入れ替えます。アーカイブ形式の J2EE アプリケーションを入れ替える場合に，少ない手順で入れ替えることができます。		5.6.3(2)

作業内容	手段	作業概要	参照先マニュアル	参照個所
リロードによる J2EE アプリケーションの入れ替え	サーバ管理コマンド（cjreloadapp）	リロードによって J2EE アプリケーションを入れ替えます。展開ディレクトリ形式の J2EE アプリケーションを入れ替える場合に，少ない手順で入れ替えることができます。		5.6.3(3)
JSP を事前にコンパイルしてから実行する J2EE アプリケーションの入れ替え	サーバ管理コマンド（cjstartapp）／cjjspc コマンド	JSP 事前コンパイル機能を使用して，JSP を事前にコンパイルしてから J2EE アプリケーションを入れ替えます。		5.6.3(4)
J2EE アプリケーションの名称変更	サーバ管理コマンド（cjrenameapp）	名称を変更して，入れ替え前や入れ替え後の J2EE アプリケーションを管理しやすくします。		5.6.3(5)
CTM を利用したオンライン状態での J2EE アプリケーションの入れ替え	運用管理コマンド（mngsvrutil）	スケジュールキューの出口を閉じたあと，J2EE アプリケーションを入れ替えます。	機能解説 拡張編	3.7.2

注※　WAR アプリケーションの場合，cjimportwar コマンドを使用します。

5.3　J2EE アプリケーションの実行時間の監視とキャンセル

この節では，J2EE アプリケーション実行時間の監視機能と，J2EE アプリケーション実行時間の監視機能で提供するメソッドタイムアウト機能，およびメソッドキャンセル機能について説明します。

注意事項

Java の仕様や JavaVM 実装上の制限によって，メソッドキャンセルができないことがあります。この場合，メソッドキャンセルができない領域は，クラス単位ではない特殊な保護区としています。Java の仕様や JavaVM 実装上の制限によって，メソッドキャンセルができなかった場合は，メッセージ KDJE52718-W の詳細情報で次に示すクラス名が表示されます。

JP.co.Hitachi.soft.jvm.CriticalClass.dummy(Native Method)

なお，保護区については，「5.3.4(1)　保護区と非保護区」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 5-6　この節の構成（J2EE アプリケーションの実行時間の監視とキャンセル）

分類	タイトル	参照先
解説	J2EE アプリケーションの実行時間の監視とキャンセルの概要	5.3.1
	J2EE アプリケーション実行時間の監視とは	5.3.2
	メソッドタイムアウトとは	5.3.3
	メソッドキャンセルとは	5.3.4
	タイムアウト値の設定例と設定値の有効範囲	5.3.5
	使用するスレッド数	5.3.6
実装	cosminexus.xml での定義	5.3.7
設定	実行環境での設定	5.3.9
運用	J2EE アプリケーションの実行時間の監視とキャンセルの流れ	5.3.10
	J2EE アプリケーションの実行状態の確認	5.3.11
	タイムアウトが発生したリクエストのキャンセル	5.3.12
	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	5.3.13
注意事項	実装時の注意事項	5.3.8

5.3.1　J2EE アプリケーションの実行時間の監視とキャンセルの概要

J2EE アプリケーションの内部で無限ループなどが発生すると，J2EE アプリケーションの動作を制御できなくなります。このようなトラブルへの対策として，J2EE アプリケーションの実行時間にタイムアウト値を設定しておき，一定時間を過ぎたら処理が完了しなくても制御が戻るように設定する方法があります。

このような設定をするには，次の 2 種類の機能を使用できます。

- メソッドタイムアウト機能
- メソッドキャンセル機能

これらの機能では，J2EE アプリケーション内のメソッド単位にタイムアウト時間を設定して監視し，一定時間内に終了しなかったリクエストのタイムアウトを検知できます（メソッドタイムアウト機能）。また，タイムアウトが発生した場合，強制的にメソッドをキャンセルできます（メソッドキャンセル機能）。このとき，トランザクションも強制決着されます。メソッドキャンセルは，自動的に実行されるように設定できます。

5.3.2　J2EE アプリケーション実行時間の監視とは

J2EE アプリケーション実行時間の監視とは，EJB および Web アプリケーションのリクエスト実行時間を監視する機能です。J2EE アプリケーション実行時間の監視機能によって，J2EE アプリケーションで無限ループなどの障害発生を検知すると，一定時間内に終了しなかったメソッド処理をタイムアウトしてユーザに通知し，さらにメソッドキャンセル機能によって使用中のトランザクションの強制決着とメソッドのキャンセルを実施できます。

J2EE アプリケーション実行時間の監視機能には，メソッドタイムアウト機能と，メソッドキャンセル機能があります。

メソッドタイムアウト機能

監視基盤にあるリクエストのうち，一定時間内に終了しなかったメソッド処理を，タイムアウトとしてユーザに通知する機能です。**監視基盤**とは，J2EE アプリケーション実行時間の監視機能を実現するために必要なインタフェースを提供し，監視情報を管理する領域です。

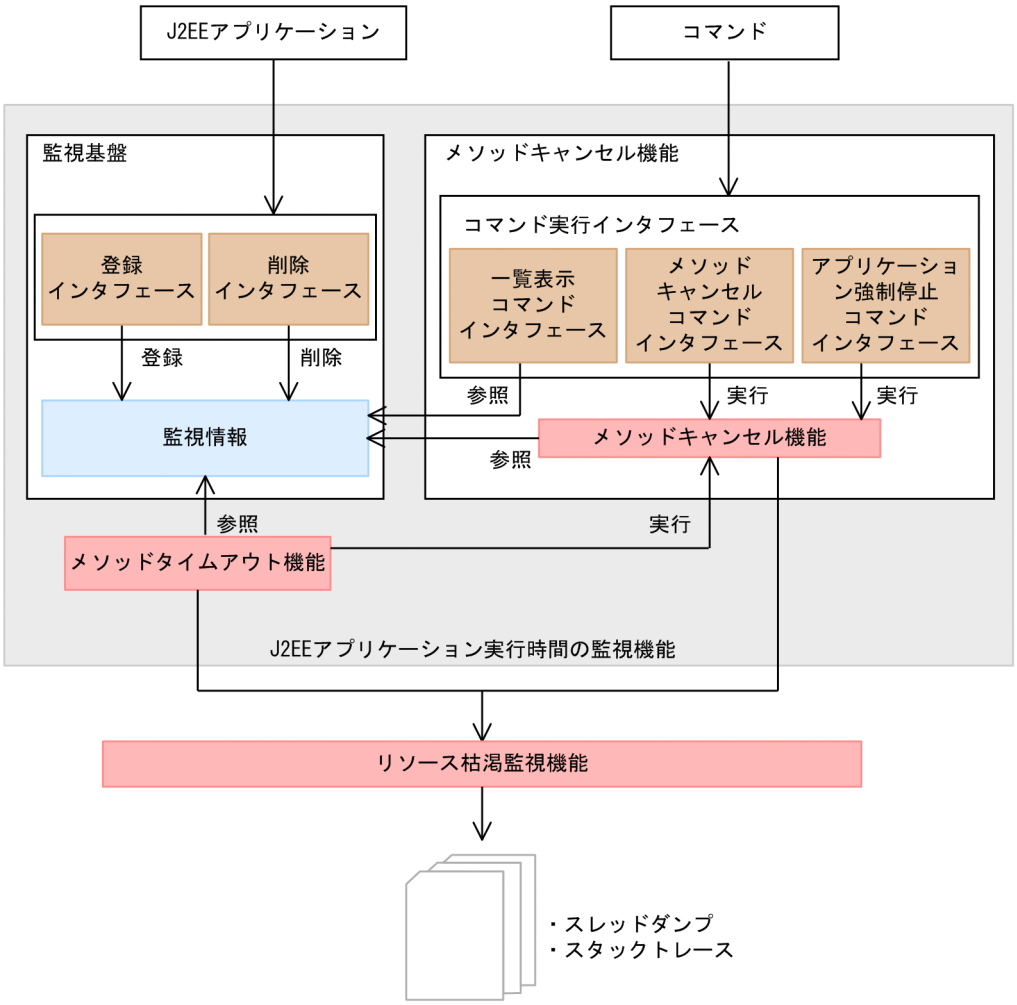
メソッドキャンセル機能

メソッドタイムアウト機能でタイムアウトが通知されたあと，メソッドのキャンセルを実施する機能です。キャンセル時にデータベースにアクセスしている場合は，トランザクションの強制決着もします。

なお，メソッドタイムアウト機能およびメソッドキャンセル機能では，ログ情報としてスレッドダンプを出力します。出力されたスレッドダンプのファイル数は，リソース枯渇監視機能で監視できます。リソース枯渇監視機能については，「4.　リソースの枯渇監視」を参照してください。

J2EE アプリケーション実行時間の監視機能について次の図に示します。

図 5-1　J2EE アプリケーション実行時間の監視機能の概要



図中のコマンドについて説明します。

- **一覧表示コマンド**

  メソッドキャンセルコマンドを実施する前に，メソッドキャンセルコマンドを実行できるかどうかを確認するときに使用します。一覧表示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「cjlistthread（スレッド情報の表示)」を参照してください。

- **メソッドキャンセルコマンド**

  タイムアウトが検知された場合，またはスレッドの状態が不正な場合に使用します。メソッドキャンセル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「cjstopthread（スレッドの削除)」を参照してください。

- **アプリケーション強制停止コマンド**

  アプリケーションを強制停止するときに使用します。アプリケーションの強制停止については，「5.5 J2EE アプリケーションの停止」を参照してください。アプリケーション強制停止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「cjstopapp（J2EE アプリケーションの停止)」を参照してください。

5.3.3　メソッドタイムアウトとは

メソッドタイムアウトとは，一定時間内に終了しなかったメソッド処理を，タイムアウトとしてユーザに通知する機能です。

(1)　タイムアウトの判定とタイムアウト後の動作

タイムアウトは，監視基盤で監視されるメソッドの呼び出しが，次の式を満たす場合に発生します。

タイムアウト判定時点の時刻－メソッド開始時刻 ＞ タイムアウトの時間

また，タイムアウト発生後は，次のどちらかの動作（**メソッドキャンセルのモード**）を選択できます。

- KDJE52703-W のメッセージを出力する
- KDJE52703-W のメッセージを出力して，メソッドキャンセル機能を実行する

どちらの動作を選択した場合も，タイムアウトした呼び出しが終了したときに，KDJE52716-I のメッセージが出力されます。

(2)　メソッドタイムアウト機能の対象となる処理

メソッドタイムアウトの対象となる処理に，タイムアウトを設定します。設定する個所は次のとおりです。

- Web アプリケーションのリクエスト処理
- EJB のメソッド呼び出し処理

なお，メソッドタイムアウトの適用対象として，次の処理は含まれません。

- J2EE アプリケーションのデプロイ時やアンデプロイ時に動作する処理
- EJB コンテナの状態管理機能，およびアクティブセッションのタイムアウト機能で動作する Enterprise Bean の処理

  Enterprise Bean のメソッド呼び出し処理とは別に，EJB コンテナが設定によって自動的に行う処理は，メソッドタイムアウトの適用対象に含まれません。

また，J2EE アプリケーションのデータ構造によっては，メソッドタイムアウトができない場合があります。詳細については，「5.3.8　実装時の注意事項」を参照してください。

(3)　タイムアウト検知間隔とメソッドタイムアウト時間

メソッドタイムアウト機能を使用する場合，次の設定が必要です。

- **タイムアウトを検知する時間の間隔**

  タイムアウトを検知するための時間間隔を設定します。

  メソッドタイムアウトが発生しているかどうかの判定が，指定した間隔ごとに J2EE サーバによって判定されます。これは，J2EE サーバ単位に設定します。

また，タイムアウトの検知は一定間隔で行われるため，検知までに最大で次の時間が掛かります。

タイムアウトの検知に掛かる時間の算出式

タイムアウトの検知に掛かる時間＝タイムアウトの時間＋タイムアウトを検知する時間の間隔

- **J2EE アプリケーションごとのメソッドタイムアウト時間**

  メソッドタイムアウトにする時間を，J2EE アプリケーションに設定します。

  設定は，J2EE アプリケーション内の Enterprise Bean 単位，または Web アプリケーション単位に設定します。

J2EE アプリケーションの実行時間監視の設定については，「5.3.9　実行環境での設定」を参照してください。

5.3.4　メソッドキャンセルとは

メソッドキャンセルとは，タイムアウトの原因となっている実行中の処理をキャンセルする処理です。メソッドキャンセル時には，現在実行されている処理がキャンセルできるかどうかを判断します。

ポイント

Connector 1.5 仕様に準拠したリソースアダプタを使用している場合，次の処理はメソッドキャンセルの対象にできません。

- リソースアダプタ独自のメソッド
- ワーク管理で実行されている Work

ただし，Work によって呼び出された Message-driven Bean 内の処理については，メソッドキャンセルの対象にできます。

(1)　保護区と非保護区

メソッドをキャンセルできる領域のことを**非保護区**，メソッドをキャンセルできない領域を**保護区**といいます。

保護区とは，メソッドキャンセルのできない領域です。保護区では，J2EE サーバの動作で共有されるデータや領域を保持したり，JavaVM 内で行われる処理を保証したりするため，メソッドキャンセルができません。J2EE サービス，Web コンテナ，EJB コンテナが保護区に該当します。保護区として定義されている内容については，「付録 C　保護区リストの内容」を参照してください。ここに記載されている保護区に該当するクラス以外で，保護区として扱いたいクラスがある場合は，保護区リストファイルに記述してください。

一方，非保護区は，メソッドキャンセルができる領域です。J2EE アプリケーションが非保護区になります。

保護区，非保護区の判定は，クラス単位で行われます。ただし，非保護区の場合は実行時の条件によって保護区として扱われることがあります。J2EE アプリケーションでも，ネイティブメソッドを呼び出している場合や，スタティックイニシャライザを実行している場合は，保護区と判定されます。また，java.lang.Object クラスの wait メソッドは，スタティックイニシャライザの延長での呼び出しではない場合だけ，例外的に非保護区と判定されます。※

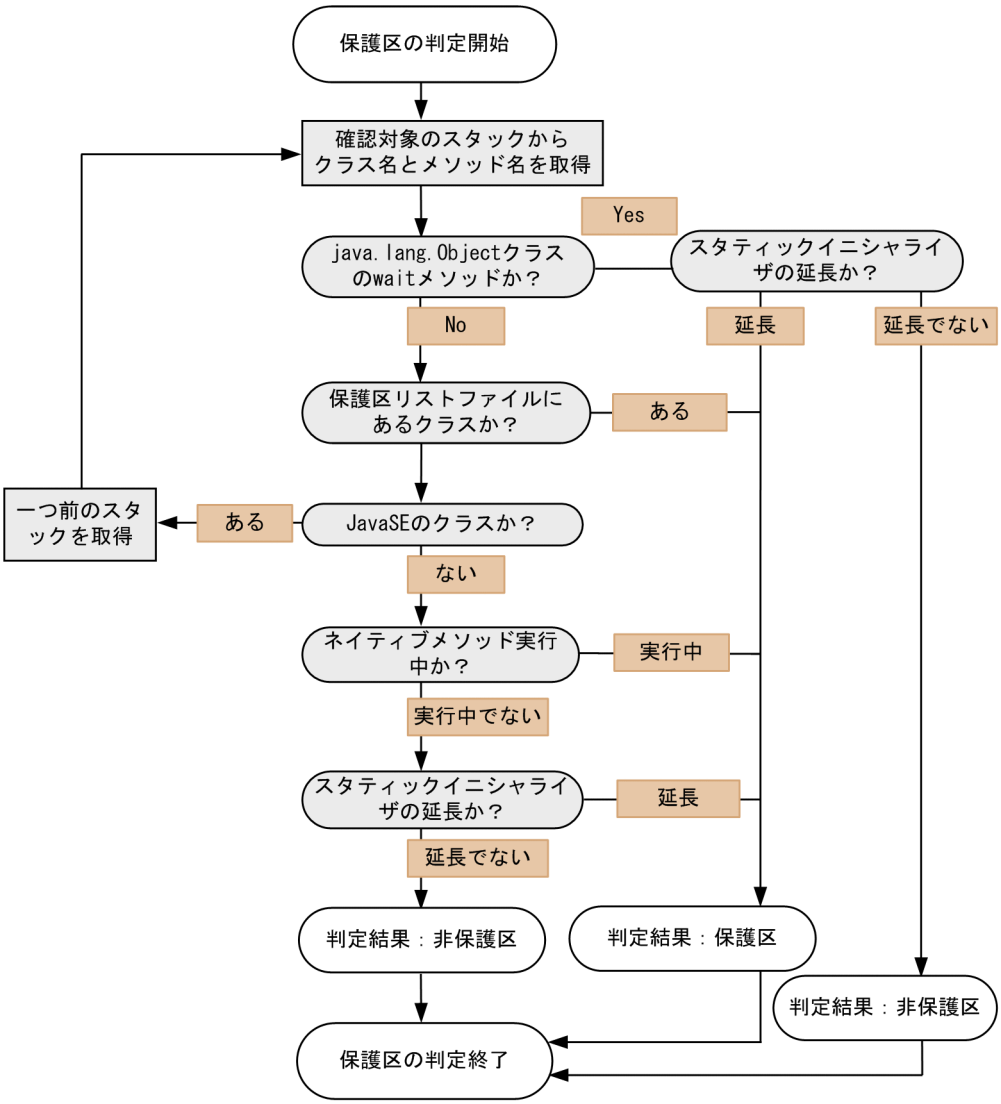
注※　java.lang.Object クラスの wait メソッドは，指定時間の経過や notify/notifyAll によって処理が再開される時に，スレッドのモニタ（ロック）を取得します。このモニタ取得待ちの間に保護区の判定を行った場合は，非保護区と判定されますが，実際にキャンセルが実行されるのはスレッドのモニタを取得した直後となります。

(2)　メソッドキャンセルの処理

メソッドキャンセルは，現在実行中の処理が非保護区の場合だけ実行されます。メソッドキャンセルができるかどうかを確認するため，保護区の判定では，キャンセル対象となるメソッドが非保護区で実行中かどうかを判定します。実行している処理が非保護区の場合は，メソッドキャンセルが行われます。保護区の場合は，一定間隔で保護区の判定をリトライします。判定処理で保護区と判定されるたびに，KDJE52718-W のメッセージが出力されます。一定時間内に非保護区に制御が移らない場合は，メソッドキャンセルが失敗したと見なされ，メソッドキャンセルの処理が終了します。

保護区の判定処理の流れを次の図に示します。

図 5-2　保護区の判定処理の流れ



なお，メソッドキャンセル時には，次の処理が実施されます。

- トランザクションを開始している場合は，トランザクションが強制的にタイムアウトされ，ロールバックにマークされます。
- SQL を実行中でステートメントキャンセル機能が有効な場合は，SQL がキャンセルされます。

上記の処理は，保護区を実行中のためにメソッドキャンセルが行われない場合も実施されます。

トランザクションタイムアウトの詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「3.15.8　トランザクションタイムアウトとステートメントキャンセル」を参照してください。

(3) Web コンテナおよび EJB コンテナでのメソッドキャンセル時の動作

メソッドキャンセルを実行すると，キャンセル対象となるメソッドを実行中のスレッドで，ThreadDeath が発生します。Web コンテナおよび EJB コンテナでは，この ThreadDeath をキャッチして必要な処理を実施します。

Web コンテナおよび EJB コンテナでのメソッドキャンセル時の動作について説明します。

(a) Web コンテナでの動作

ThreadDeath がスローされるタイミングによって動作が異なります。

Web アプリケーションのフィルタ/サーブレット/JSP でのリクエスト処理中に ThreadDeath がスローされた場合

呼び出し元となるフィルタ/サーブレット/JSP に javax.servlet.ServletException がスローされます。

例えば，フィルタから javax.servlet.FilterChain の doFilter メソッド呼び出しの延長でサーブレットが実行され，そのサーブレットの実行中に ThreadDeath が発生した場合，doFilter メソッドの呼び出しで，javax.servlet.ServletException がスローされます。

同様に，javax.servlet.RequestDispatcher の forward メソッド，または include メソッドを呼び出して，リクエストをサーブレット/JSP に転送した場合，javax.servlet.ServletException の getRootCause メソッドでは，ThreadDeath オブジェクトを返します。

Web アプリケーションのリスナの処理中に ThreadDeath がスローされた場合

Web コンテナは ThreadDeath をキャッチしますが，リスナが呼び出される契機となったイベントの発生元となるユーザプログラム処理に対しては，例外はスローされません。

例えば，javax.servlet.http.HttpServletRequest の getSession メソッドの呼び出しによって HttpSession を作成した場合，javax.servlet.http.HttpSessionListener の sessionCreated メソッドが呼び出されます。この sessionCreated メソッドの実行中に ThreadDeath がスローされた場合，Web コンテナはスローされた ThreadDeath をキャッチしますが，HttpSession 生成のイベントを発生させた getSession メソッド呼び出しには例外をスローしません。

(b) EJB コンテナでの動作

EJB のメソッド呼び出し中に ThreadDeath がスローされた場合は，EJB 仕様で定められたシステム例外が発生した場合と同等の動作をします。呼び出し元に返る例外の getCause メソッドでは，ThreadDeath オブジェクトを返します。

(4) メソッドキャンセル実行のタイミング

メソッドキャンセル実行のタイミング，およびメソッドキャンセル実行までに掛かる最大の時間について説明します。

- タイムアウト発生時

メソッドタイムアウト時のメソッドキャンセルのモードとして，スレッドを停止することが設定されている場合，タイムアウト発生時にメソッドキャンセルが実行されます。メッセージを出力するだけの設定の場合は，タイムアウト時でもメソッドキャンセルは実行されません。

タイムアウトが発生したメソッドのキャンセル処理は，一定の間隔で実行されます。また，メソッドキャンセルの処理は，動作中のリクエスト処理がタイムアウトしていないかを調査する（タイムアウトを監視する）スレッドとは別のスレッドで，非同期で実行されます。そのため，タイムアウトの検知後にメソッドキャンセルが実行されるまでには，最大で，メソッドキャンセル処理の時間間隔分掛かります。

メソッドキャンセル処理の時間間隔は，タイムアウトを監視する時間間隔と同じです。メソッドキャンセル処理の時間間隔の設定については，「5.3.9　実行環境での設定」を参照してください。

- **メソッドキャンセルコマンドの実行時**

  運用中にメソッドキャンセルコマンドを実行すると，メソッドキャンセルが実行されます。

  メソッドキャンセルコマンドは，タイムアウトが発生したメソッドを実行中のスレッドの状態を確認して，メソッドキャンセルが実行できることを確認してから，必要に応じて実行します。

  メソッドキャンセルを実行するための手順については，「5.3.10　J2EE アプリケーションの実行時間の監視とキャンセルの流れ」を参照してください。

  メソッドキャンセル処理は，コマンド実行後に非同期に実行されます。また，メソッドキャンセル処理は，メソッドキャンセル処理の時間間隔には影響されません。

- **J2EE アプリケーション強制停止の実行時**

  J2EE アプリケーションを強制停止した場合に，停止対象となる J2EE アプリケーションで実行中のスレッドがあるときは，メソッドキャンセルが実施されます。

  アプリケーションの強制停止については，「5.5.4　強制停止処理とは」を参照してください。

メソッドキャンセル処理は，コマンド実行後に非同期で実行されます。また，メソッドキャンセル処理は，メソッドキャンセル処理の時間間隔には影響されません。

(5)　メソッドキャンセル時の注意事項

メソッドキャンセル時，デフォルトではローカル変数情報がスタックトレースに出力されます。

性能への影響が出るおそれがあるため，メソッドキャンセル時にローカル変数情報をスタックトレースに出力しないように，JavaVM 拡張オプションの「-XX:-HitachiLocalsInStackTrace」の設定を推奨します。

5.3.5　タイムアウト値の設定例と設定値の有効範囲

J2EE アプリケーション実行時間の監視機能を使用するときの，タイムアウト値の設定例と設定値の有効範囲について，Web アプリケーションの場合と EJB の場合に分けて説明します。

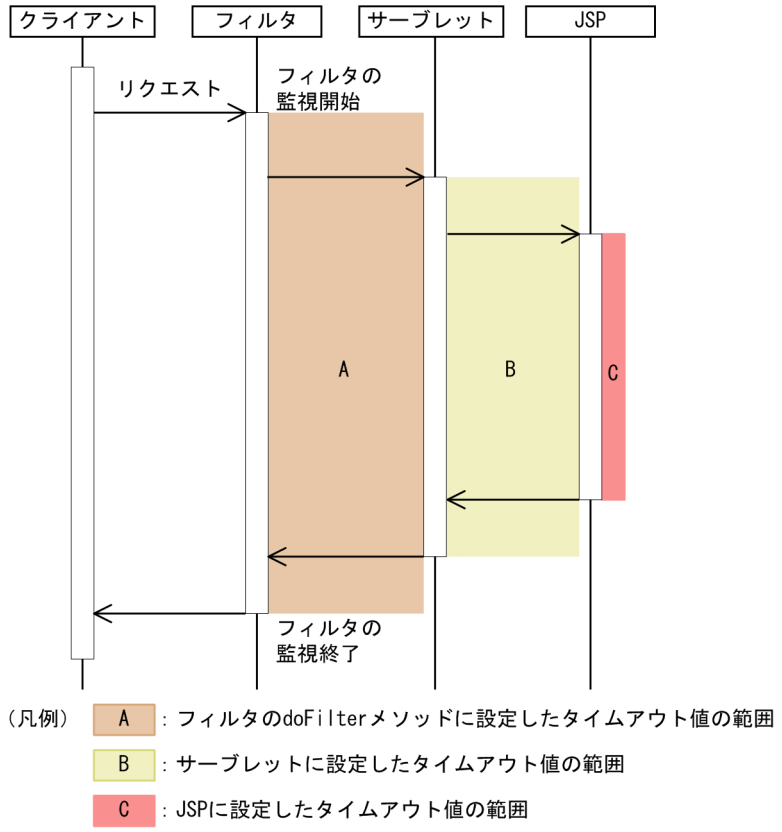
(1)　Web アプリケーションの場合

Web アプリケーションの場合のタイムアウト値の設定について説明します。

(a)　フィルタ/サーブレット/JSP への呼び出し

フィルタ/サーブレット/JSP 呼び出しのタイムアウト設定について，次の図に示します。

図 5-3　フィルタ/サーブレット/JSP 呼び出しのタイムアウト設定



各場所のタイムアウト値は，呼び出しの順序を考慮して適切に設定する必要があります。例えば，図中 A の範囲の doFilter メソッドのタイムアウト値は，メソッドの開始から終了までの処理時間の監視に適用されます。このため，B または C の処理時間が A のタイムアウト値に設定した時間に達してしまうと，その時点でタイムアウトされてしまいます。例を次に示します。

処理中にタイムアウトが発生する例

図中の A，B，C のそれぞれのタイムアウト値として次の表に示す時間が設定されていることとします。

表 5-7　タイムアウト値の設定例 1

場所	設定したタイムアウト値
図中 A（フィルタの doFilter メソッド）	240 秒
図中 B（サーブレット）	180 秒
図中 C（JSP）	120 秒

このような設定の場合に，A から B への処理，および B から C の処理で次の時間が掛かったとします。

- A から B への処理：120 秒
- B から C への処理：60 秒

このとき，C での処理が 60 秒以上掛かると，図中 A のタイムアウト値の 240 秒に達してしまうため，処理が完了する前にタイムアウトが発生します。

- **設定時の注意事項**

  アップロード，ダウンロードを実施するサーブレット/JSP 内の処理の延長では，クライアントとの通信が発生する場合があります。この場合，アップロード中，ダウンロード中はサーブレット/JSP の処理中として扱われるため，メソッドの実行時間が監視されます。

  このため，アップロード，ダウンロードを実施するサーブレット/JSP は，クライアントとの通信遅延などを考慮してタイムアウトを設定する必要があります。特に，ダウンロードについては，ブラウザでダウンロード確認用のダイアログを表示して，クライアントに，ダウンロードの確認を求める場合があります。この場合，ダウンロードするデータ量によっては，クライアント上でユーザがダウンロード確認用のダイアログで操作をするまで，ダウンロードが完了しない（サーブレット/JSP の処理が完了しない）ことがあるので注意してください。

  なお，フィルタを経由する場合は，該当するフィルタに関しても同様の注意が必要です。

(b) サーブレットへの初回リクエスト

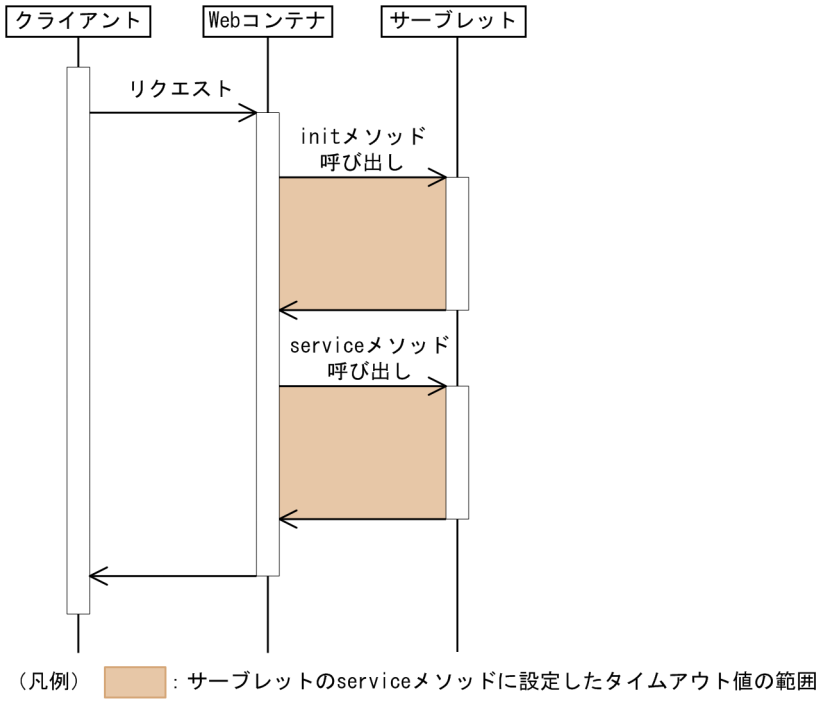
サーブレットへの初回アクセス時には，サーブレットの service メソッドに設定したタイムアウト値が有効になります。

ただし，DD（web.xml）の<load-on-startup>タグを設定していないサーブレット/JSP の場合，初回リクエスト時には，init メソッドが実行されます。このとき，init メソッドには，該当するサーブレット/JSP の service メソッドに設定したタイムアウト値が有効になります。

また，サーブレット/JSP が service メソッドの処理で利用できないことを示す，javax.servlet.UnavailableException をスローした場合に実行される destroy メソッドにも，該当するサーブレット/JSP の service メソッドに設定したタイムアウト値が適用されます。

サーブレットへの初回リクエストのタイムアウト設定について，次の図に示します。

図 5-4　サーブレットへの初回リクエストのタイムアウト設定



(c)　フィルタを経由したサーブレット/JSP の初回リクエスト

フィルタを経由したサーブレットおよび JSP の初回リクエストのタイムアウト設定について，次の図に示します。

図 5-5　フィルタを経由したサーブレットの初回リクエストのタイムアウト設定

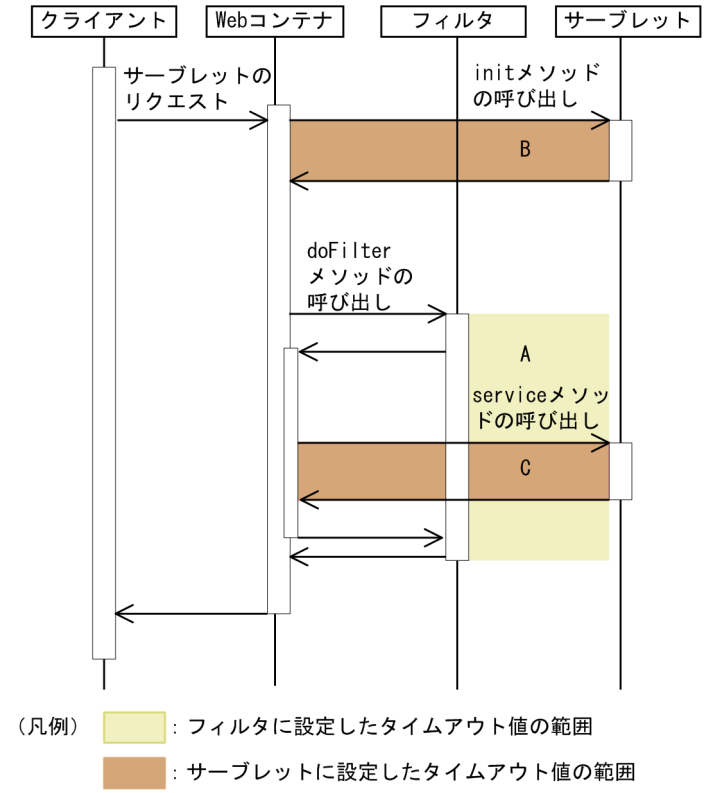
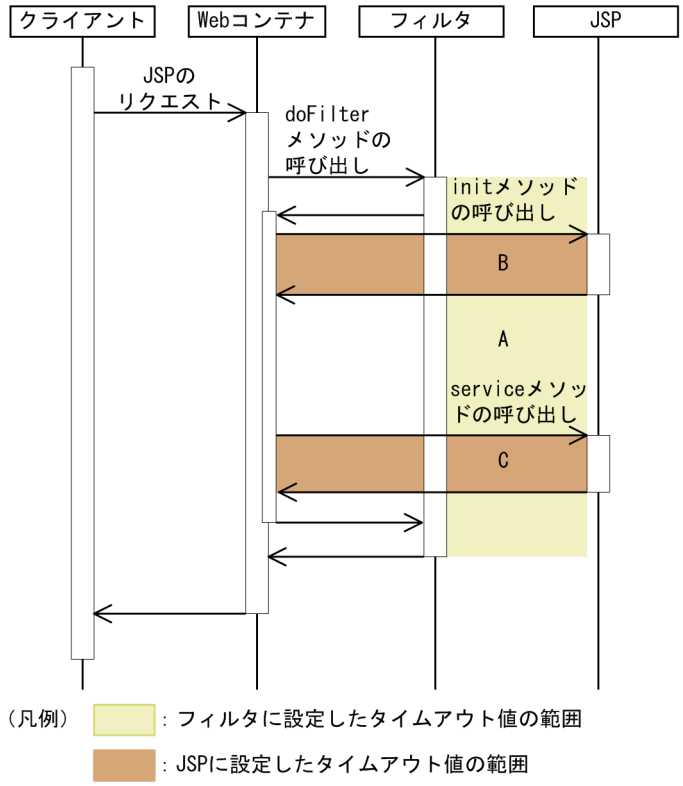


図 5-6　フィルタを経由した JSP の初回リクエストのタイムアウト設定



フィルタの処理の延長で，サーブレット/JSP が呼び出される場合のフィルタのタイムアウト値は，フィルタの延長で呼び出されるサーブレット/JSP のタイムアウト値を考慮して設定する必要があります。

例えば，図 5-6 の場合，図中 A の範囲のフィルタのタイムアウト値は，フィルタの処理時間に適用されます。このため，B または C の処理時間が A のタイムアウト値に設定した時間に達してしまうと，その時点でタイムアウトされてしまいます。図 5-6 の場合の例を次に示します。

処理中にタイムアウトが発生する例

図中の A，B，C のそれぞれのタイムアウト値として次の表に示す時間が設定されていることとします。

表 5-8　タイムアウト値の設定例 2

場所	設定したタイムアウト値
図中 A（フィルタ）	240 秒
図中 B および C（サーブレット）	180 秒

このような設定の場合に，フィルタでの処理，およびサーブレットでの処理で次の時間が掛かったとします。

- フィルタでの処理：60 秒
- サーブレットの init メソッドで処理：120 秒

このとき，サーブレットの service メソッドでの処理が 60 秒以上掛かると，図中 A のタイムアウト値の 240 秒に達してしまうため，サーブレットの service メソッドの処理が完了する前にタイムアウトが発生します。

また，DD（web.xml）の<servlet>タグを定義していて，<load-on-startup>タグを設定していないサーブレット/JSP の場合で，初回リクエストをフィルタの doFilter メソッドの延長で実行するとき，サーブレット/JSP の init メソッドが実行されます。init メソッドが実行されるタイミングを次の表に示します。

表 5-9　init メソッドが実行されるタイミング

対象	<servlet>タグの定義	<load-on-startup>タグの定義	init メソッドの実行されるタイミング
サーブレット	あり	なし	doFilter メソッドが呼び出される前
	なし	なし	doFilter メソッドの延長
JSP	あり	なし	doFilter メソッドの延長
	なし	なし	doFilter メソッドの延長

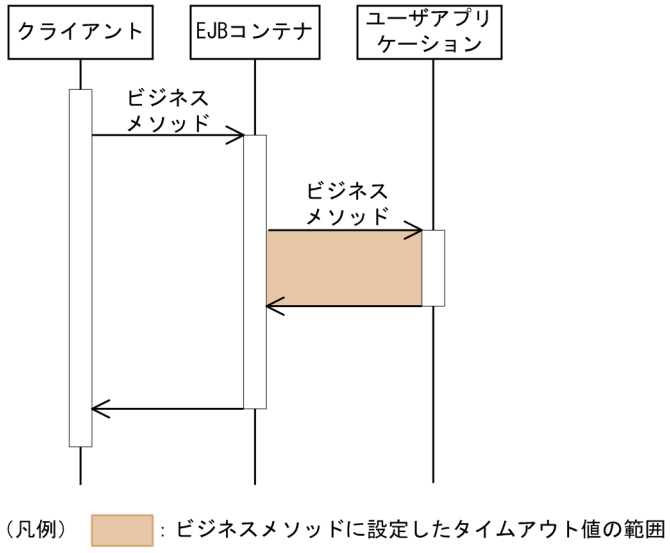
このとき，init メソッドのタイムアウト値には，次のどちらかの値が適用されます。

- フィルタのタイムアウト値≧Web アプリケーションのタイムアウト値の場合
  該当するサーブレット/JSP の service メソッドに設定したタイムアウト値が有効になります。
- フィルタのタイムアウト値＜ Web アプリケーションのタイムアウト値の場合
  フィルタに設定したタイムアウト値が有効になります。

(2)　EJB の場合

Stateless Session Bean のビジネスメソッド呼び出しのタイムアウト設定について，次の図に説明します。なお，ほかの Bean の種類（Stateful Session Bean，Entity Bean，Message-driven Bean）についても同様に設定できます。

図 5-7　Stateless Session Bean のビジネスメソッド呼び出しのタイムアウト設定



ビジネスメソッドのタイムアウト値に設定した時間を超えると，タイムアウトが発生します。

なお，EJB のメソッド呼び出しの延長で呼び出されるコールバックは，業務処理の一部であるため，タイムアウト値を設定するときには，コールバックメソッドの実行時間を考慮してください。各メソッドで呼び出されるコールバックについては，「5.3.8(2)　Enterprise Bean のメソッド呼び出し処理でのタイムアウト値の設定方法」を参照してください。

5.3.6　使用するスレッド数

J2EE アプリケーション実行時間の監視機能を使用する場合に，生成されるスレッドとスレッド数を次の表に示します。

表 5-10　生成されるスレッドとスレッド数

スレッド	スレッド数
メソッドタイムアウトの監視用スレッド※	1
タイムアウトによるメソッドキャンセル実行用スレッド	4
コマンドによるメソッドキャンセル実行用スレッド	4
現在時刻の取得用スレッド	1

注※　動作中のリクエスト処理がタイムアウトしていないかを調査するためのスレッドです。

なお，コマンドによるメソッドキャンセル実行用スレッドは，メソッドキャンセルコマンド，およびアプリケーション強制停止コマンドを実行した時に生成され，コマンドを終了した時に削除されます。そのほかのスレッドは，J2EE サーバを起動した時に生成され，J2EE サーバが終了するまで動作します。

5.3.7　cosminexus.xml での定義

J2EE アプリケーション実行時間の監視機能を使用するための定義は，cosminexus.xml の<war>タグ内に指定します。

cosminexus.xml での J2EE アプリケーション実行時間の監視機能の定義について次の表に示します。

表 5-11　cosminexus.xml での J2EE アプリケーション実行時間の監視機能の定義

項目		指定するタグ	設定内容
メソッドタイムアウトの時間の設定	EJB 単位での設定	<session>，<entity>または<message>タグ下の<ejb-method-observation-timeout>－<method-observation-timeout>タグ	メソッドタイムアウトの時間を指定します。
	Web アプリケーション単位での設定	<war>－<servlet>－<method-observation-timeout>タグ，また	サーブレット内のメソッド共通のメソッドタイムアウトの時間を指定します。

項目		指定するタグ	設定内容
		は<war>－<filter>－<method-observation-timeout>タグ	
メソッドキャンセルのモードの設定		<cosminexus-app>－<method-observation-recovery-mode>タグ	メソッドキャンセルのモードを指定します。

指定するタグの詳細は，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(アプリケーション／リソース定義)」の「2.2　アプリケーション属性ファイル（cosminexus.xml）で指定する各属性の詳細」を参照してください。

5.3.8　実装時の注意事項

ここでは，メソッドタイムアウト機能，およびメソッドキャンセル機能を実装する場合の設定や注意事項について説明します。

(1)　Web アプリケーションのリクエスト処理でのタイムアウト値の設定方法

Web アプリケーションのリクエスト処理では，次の表に示すメソッドにタイムアウト値を設定できます。各メソッドのタイムアウト値の有効範囲については，「5.3.5　タイムアウト値の設定例と設定値の有効範囲」を参照してください。

なお，タイムアウトを利用するサーブレットおよび JSP は，DD（web.xml）の<servlet>タグで定義します。

表 5-12　メソッドタイムアウトの適用対象（Web アプリケーションのリクエスト処理）

ケース	メソッド	条件
サーブレット，JSP	service	条件はありません。
	init	DD（web.xml）に<load-on-startup>タグを定義していないサーブレット（JSP 利用時は，JSP）に対して，初回リクエストで実行される場合だけ，メソッドタイムアウトの適用対象となります。
	destroy	service メソッドを呼び出した時に，永久に利用できないことを示す javax.servlet.UnavailableException 例外をスローしたことで実行される場合だけ，メソッドタイムアウトの適用対象となります。
フィルタ	doFilter	条件はありません。

javax.servlet.http.HttpServlet クラスのサブクラスとしてサーブレットを実装し，javax.servlet.http.HttpServlet クラスの service メソッドから呼び出される doXXX メソッド（doGet メソッド,doPost メソッドなど）をオーバーライドする場合，オーバーライドした doXXX メソッドがタイムアウトの適用対象となります。

JSP の service メソッドとは，JSP から生成された javax.servlet.jsp.JspPage インタフェースを実装したクラスの service メソッドを指します。

JSP 内に定義する_jspService メソッドは，JspPage インタフェースの実装クラスから，JspPage の service メソッドの延長で実行されます。JSP で JspPage を指定しない場合，Web コンテナが提供する JspPage インタフェースの実装クラスが使用されます。

(2) Enterprise Bean のメソッド呼び出し処理でのタイムアウト値の設定方法

Enterprise Bean のメソッド呼び出し処理では，メソッド単位にタイムアウト値を設定できます。Enterprise Bean 単位にタイムアウト値を設定できません。

J2EE アプリケーションのメソッドを呼び出す側でのメソッドタイムアウト適用対象メソッドを表 5-13 に，J2EE アプリケーションに実装されるコールバックメソッド側でのメソッドタイムアウトの適用対象メソッドを表 5-14 にそれぞれ示します。

CMP2.0 の find メソッド／select メソッド，および CMP1.0 の find メソッドのように，Java プログラムに実装されていないメソッドは，メソッドタイムアウトの適用対象外となります。また，メソッドが呼び出しされたときの状態によって，呼び出されるかどうかが決定するコールバックメソッド（ejbActivate, ejbLoad など）についても，適用対象外となります。

表 5-13　メソッドタイムアウトの適用対象（Enterprise Bean 呼び出し側）

インタフェース	メソッド	Stateless Session Bean	Stateful Session Bean	Singleton Session Bean	Entity Bean (BMP)	Entity Bean (CMP2.0)	Entity Bean (CMP1.1)	Message -driven Bean
ホームインタフェース	create	×	○	−	○	○	○	×
	finder	−	−	−	○	×	×	−
	select	−	−	−	−	×	−	−
	home	−	−	−	○	○	−	−
コンポーネントインタフェース	remove	×	○	−	○	○	○	×
	ビジネスメソッド	○	○	−	○	○	○	−
ビジネスインタフェース	ビジネスメソッド	○	○	△	−	−	−	−
	ビジネスメソッド（非同期）	△	×	△	−	−	−	−
javax.jms.MessageListener	onMessage	−	−	−	−	−	−	○

インタフェース	メソッド	Stateless Session Bean	Stateful Session Bean	Singleton Session Bean	Entity Bean (BMP)	Entity Bean (CMP2.0)	Entity Bean (CMP1.1)	Message -driven Bean
任意のメッセージリスナインタフェース（EJB 2.1 以降）	任意のメッセージリスナメソッド	−	−	−	−	−	−	○

（凡例）

○：適用されます。

×：適用されません。

△：メソッドタイムアウトは適用されますが，メソッドキャンセルは適用されません。

−：該当しません。

表 5-14　メソッドタイムアウトの適用対象（コールバックメソッド側）

メソッド	Stateless Session Bean	Stateful Session Bean	Singleton Session Bean	Entity Bean (BMP)	Entity Bean (CMP2.0)	Entity Bean (CMP1.1)	Message -driven Bean
コンストラクタ	×	×	×	×	×	×	×
setSessionContext／setEntityContext／setMessageDrivenContext	×	×	×	×	×	×	×
unsetSessionContext／unsetEntityContext	×	×	×	×	×	×	−
ejbCreate<Method>	×	○	−	○	○	○	×
ejbPostCreate<Method>	−	−	−	×	×	×	−
ejbRemove※	×	○	−	○	○	○	×
ejbActivate	×	×	−	×	×	×	−
ejbPassivate	×	×	−	×	×	×	−
ejbLoad	−	−	−	×	×	×	−
ejbStore	−	−	−	×	×	×	−
ejbfind<Method>	−	−	−	○	−	−	−
ejbSelect<Method>	−	−	−	−	−	−	−
@PostConstruct	×	○	△	−	−	−	−
@PreDestroy	×	○	△	−	−	−	−
home メソッド	−	−	−	○	○	−	−

メソッド	Stateless Session Bean	Stateful Session Bean	Singleton Session Bean	Entity Bean (BMP)	Entity Bean (CMP2.0)	Entity Bean (CMP1.1)	Message -driven Bean
ビジネスメソッド	○	○	△	○	○	○	－
onMessage	－	－	－	－	－	－	○
任意のメッセージリスナメソッド（EJB 2.1 以降）	－	－	－	－	－	－	○

（凡例）

○：適用されます。

×：適用されません。

△：メソッドタイムアウトは適用されますが，メソッドキャンセルは適用されません。

－：該当しません。

注※

タイムアウト機能などのコンテナの処理で呼び出された場合は対象外です。

(3) メソッドタイムアウト発生時のオプションメッセージの設定

メソッドタイムアウトが発生した場合，オプションを設定しておくことによって，タイムアウトを通知するメッセージの詳細部分に任意の文字列を追加することができます。実装例を次に示します。

```
    // ユーザの処理
....
    // メッセージの追加
RequestMonitorMessage.setMessage(”メッセージを追加します。”);
....
// ユーザの処理
```

次のような監視対象ではないアプリケーションでは，このメソッドは使用できません。

- EJB クライアントアプリケーション
- J2EE アプリケーションが生成，起動したスレッドで動作するアプリケーション

(4) メソッドキャンセルを利用するアプリケーション開発時の注意事項

メソッドキャンセルを使用する場合，アプリケーションを開発するときに，アプリケーションのデータ構造について注意する必要があります。

メソッドキャンセルを実行すると，スレッドの実行が予期しない場所で中断されます。そのため，メソッドキャンセルの対象スレッドとほかのスレッドで，更新または削除を実行する共有データがある場合，メソッドキャンセルを使用しないでください。共有データが破壊され，ほかのリクエストの処理に影響を及ぼすことがあります。

例えば，ある Enterprise Bean に共有の領域（変数 A=1，B=2）を設けて，複数スレッド（スレッド 1 とスレッド 2）でアクセスした場合に，スレッド 1 が共有の領域を更新中（A=10，B=20）に

ThreadDeath 例外が発生すると，更新が中途半端のまま（A=10，B=2）スレッド 1 が終了します。このとき，スレッド 1 が更新途中のデータをスレッド 2 で参照して，A*B を計算すると，演算結果が不正になります（正：200，誤：20）。そのため，このような例では，メソッドキャンセルは使用できません。

また，ローカル呼び出し最適化機能を使用している場合は，アプリケーション内で，またはアプリケーションをわたって，複数スレッドで更新や削除を実行するようなデータ構造がないか確認する必要もあります。

5.3.9　実行環境での設定

J2EE アプリケーション実行時間の監視機能を使用する場合，J2EE サーバおよび J2EE アプリケーションの設定が必要です。

なお，J2EE アプリケーションの設定については，cosminexus.xml を含まない J2EE アプリケーションのプロパティを設定または変更する場合にだけ参照してください。

(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。J2EE アプリケーション実行時間の監視機能の定義は，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，ejbserver.ext.method_observation.interval パラメタで，J2EE アプリケーション実行時間の監視機能を使用するかどうかを指定します。0 を指定すると，J2EE アプリケーションの実行時間の監視機能が無効になります。1 以上の有効値を指定すると，J2EE アプリケーションの実行時間の監視機能が有効になり，その値が次の二つの時間間隔として使用されます。

- 動作中のリクエスト処理がタイムアウトしていないかを調査する時間間隔
- タイムアウトしたリクエスト（メソッド）をキャンセルする時間間隔

簡易構築定義ファイル，および指定するパラメタの詳細は，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「4.3　簡易構築定義ファイル」を参照してください。

(2)　J2EE アプリケーションの設定

実行環境での J2EE 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。J2EE アプリケーション実行時間の監視機能の定義では，次の属性ファイルを使用します。

メソッドタイムアウトの時間の設定

- EJB 単位での設定

  SessionBean 属性ファイル，EntityBean 属性ファイル，または MessageDrivenBean 属性ファイル
- Web アプリケーション単位での設定

  サーブレット属性ファイル，またはフィルタ属性ファイル

メソッドキャンセルモードの設定

アプリケーション属性ファイル

各属性ファイルで指定するタグは，cosminexus.xml と対応しています。cosminexus.xml での定義については，「5.3.7　cosminexus.xml での定義」を参照してください。

5.3.10　J2EE アプリケーションの実行時間の監視とキャンセルの流れ

J2EE アプリケーションの実行時間を監視してタイムアウトが発生したリクエストを自動的にキャンセルするための設定は，サーバ管理コマンドを使用して実行します。また，J2EE アプリケーションの実行時間を監視するかどうか，および監視時間間隔は，J2EE サーバの動作設定のカスタマイズで設定します。J2EE アプリケーション実行時間の監視の設定については，「5.3.9　実行環境での設定」を参照してください。

また，J2EE アプリケーションでメソッドタイムアウトが発生した場合や，メソッドキャンセルに失敗したスレッドが存在する場合は，サーバ管理コマンドを使用して，手動でメソッドキャンセルを実行（再実行）する必要があります。

サーバ管理コマンドによってメソッドキャンセルを実行する手順を次に示します。

手順

タイムアウトが発生したことを示すメッセージ，またはスレッドの状態が不正なことを示すメッセージを受け取ったあとで，次の手順で実行します。

1. J2EE アプリケーションの実行状態を確認する（5.3.11 参照）

   サーバ管理コマンドを使用して実行します。

2. タイムアウトが発生している場合，J2EE アプリケーションで実行しているリクエストをキャンセルする（5.3.12 参照）

   サーバ管理コマンドを使用して実行します。

なお，J2EE アプリケーションを強制的に停止した場合も，メソッドはキャンセルされます。J2EE アプリケーションの強制停止についての詳細は，「5.5.7　J2EE アプリケーションの停止」を参照してください。

5.3.11　J2EE アプリケーションの実行状態の確認

J2EE アプリケーション内のメソッドでタイムアウトが発生したことを示すメッセージが出力された場合，またはメソッドキャンセルに失敗したことによってスレッドの状態が不正になったことを示すメッセージが出力された場合に，実行中の J2EE アプリケーションの実行状態を確認します。確認結果によって，メソッドキャンセルを実行できるかどうかを判断します。

実行状態の確認には，サーバ管理コマンド（cjlistthread）を使用します。

なお，このコマンドは，スレッドの状態が遷移した時に取得したスタックトレースを確認する場合にも使用できます。ただし，スレッドの状態が「running」の場合は，スタックトレースは出力されません。また，スレッドの状態が「stopping」の場合は，複数回スタックトレースが出力されるため，遷移した時点のスタックトレースではなく，最新のスタックトレースが出力されます。

また，Web コンテナで設定している最大同時実行数を上回る数のスレッドが一覧に表示される事があります。

実行形式と実行例を次に示します。なお，cjlistthread 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「cjlistthread（スレッド情報の表示)」を参照してください。

実行形式

```
cjlistthread <J2EEサーバ名>
```

-detail オプションを指定すると，動作中のスレッド情報の詳細として，スタックトレースも出力できます。

```
cjlistthread <J2EEサーバ名> -detail
```

実行例

```
cjlistthread MyServer -detail
```

-detail オプションを指定した場合の実行結果の形式を次に示します。

実行結果

```
Current Time=HH:MM:SS
ThreadID=11111,RootApInfo=RootAP1,Status=timeout,AppName=AP1,StartTime=HH:MM:SS,TimeOut=6
0
    com.hitachi.XXXX
     at com.hitachi.YYYY
     at user.code.UserClass1
     at com.hitachi.ZZZZ
              .
              .
              .
ThreadID=22222,RootApInfo=RootAP2,Status=stopping,AppName=AP2,StartTime=HH:MM:SS,TimeOut=
60
    com.hitachi.xxxx
     at com.hitachi.yyyy
     at user.code.UserClass2
     at com.hitachi.zzzz
```

実行結果の，「Status=」の後ろに出力されているのが，スレッドの状態です。

スレッドの状態とメソッドキャンセルの実行可否を，次の表に示します。

表 5-15　スレッドの状態とメソッドキャンセルの実行可否

スレッドの状態	意味	メソッドキャンセルの実行可否
running	実行時間が監視されている状態です。正常に動作しています。	実行できます。
timeout	メソッドタイムアウトによって，タイムアウトが検知された状態です。	実行できます。
stopping	メソッドキャンセル処理が実行されている状態です。	実行できません。
stopped	メソッドキャンセル処理の完了待ちの状態です。	実行できません。
failed	メソッドキャンセル処理が失敗した状態です。	実行できます。

ただし，スレッドの状態がメソッドキャンセルを実行できる状態の場合でも，そのメソッドが**保護区**で実行されている場合は，メソッドはキャンセルできません。保護区については，「5.3.4　メソッドキャンセルとは」を参照してください。

ポイント

このコマンドで出力されるスレッドの情報は，cjlistthread コマンドを実行した時点でのスナップショットです。このため，刻々と変化します。正確なスレッドの状態を取得するためには，複数回コマンドを実行して確認することをお勧めします。

また，-detail オプションを指定すると，実行結果の出力量が多くなります。-datail オプションを指定する場合は，出力先をファイルにすることをお勧めします。

5.3.12　タイムアウトが発生したリクエストのキャンセル

J2EE アプリケーションでタイムアウトが発生したリクエストをキャンセルします。この処理をメソッドキャンセルといいます。メソッドキャンセルは，「5.3.11　J2EE アプリケーションの実行状態の確認」の内容に従って cjlistthread コマンドを実行してスレッドの実行状態を確認した結果，メソッドキャンセルが可能な状態（running，timeout または failed）だった場合に実行してください。

メソッドキャンセルには，サーバ管理コマンド（cjstopthread）を使用します。一度のコマンド実行で，複数のスレッドのメソッドをキャンセルできます。

実行形式と実行例を次に示します。なお，cjstopthread 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「cjstopthread（スレッドの削除）」を参照してください。

実行形式

```
cjstopthread <J2EEサーバ名> -tid <スレッドID> [-tid <スレッドID>…]
```

実行例

```
cjstopthread MyServer -tid 11111
```

-tid オプションのオプション引数の<スレッド ID>には，タイムアウトが発生したことを示すメッセージ，またはスレッドの状態が不正なことを示すメッセージに出力されていたスレッド ID を指定してください。

なお，このコマンドを実行したときに，すでにメソッドキャンセルが実行中の場合，またはメソッドキャンセルの対象に指定したスレッドが存在しない場合は，メソッドキャンセルは実行されないで，コマンドは正常終了します。コマンド実行後のスレッドの状態は，cjlistthread コマンドで確認できます。「5.3.11 J2EE アプリケーションの実行状態の確認」に示した手順で確認してください。

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

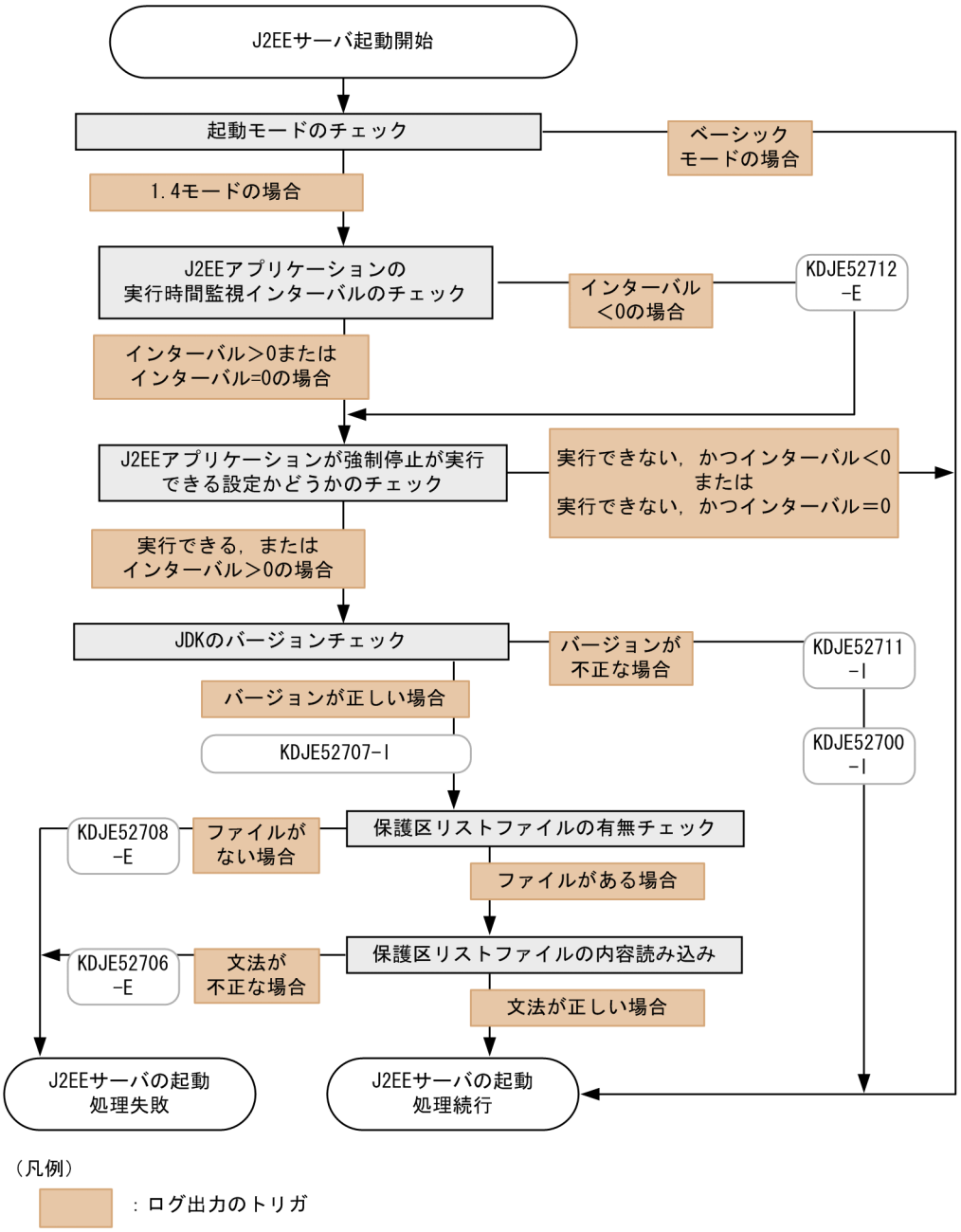
ここでは，J2EE アプリケーションの実行時間監視機能で出力されるログ情報について説明します。

(1)　J2EE サーバ起動時に出力されるログ情報

J2EE サーバ起動時には，プロパティの読み込み，JavaVM のバージョンチェック，保護区リストファイルの読み込みが実行されます。

このときに出力されるログ情報を次の図に示します。

図 5-8　J2EE サーバ起動時に出力されるログ情報



(2)　タイムアウトの延長でメソッドキャンセルを実行した場合に出力されるログ情報

タイムアウトが発生して，その処理の延長でメソッドキャンセルが実行された場合に出力されるログ情報を次の図に示します。どのメッセージが出力された場合も，処理は続行されます。なお，メソッドキャンセルモードが有効か無効かによって，スレッドダンプの出力回数が異なります。

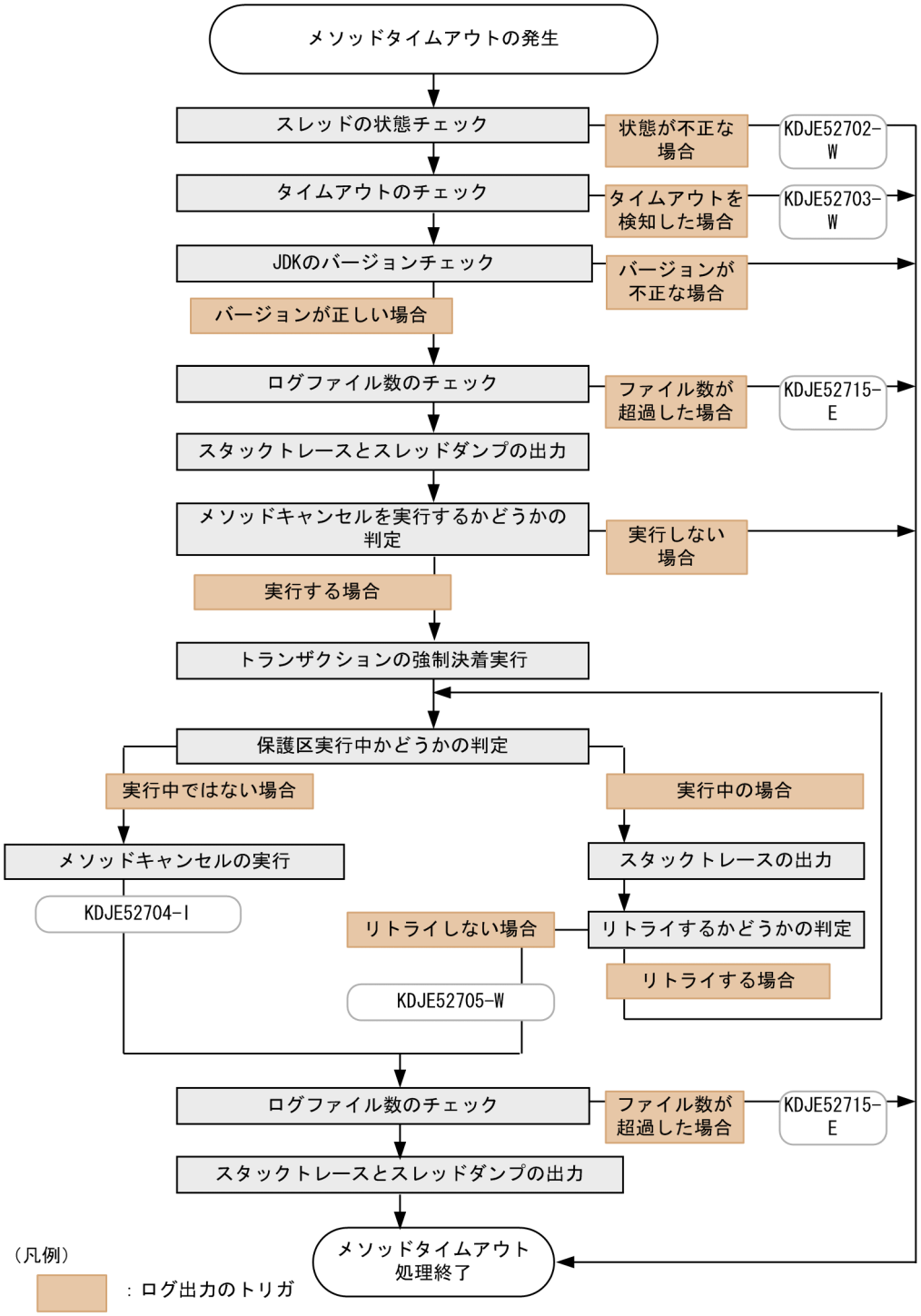
- メソッドキャンセルモードが有効の場合

  タイムアウト発生時と，メソッドキャンセルの成功時または失敗時に，2 回出力されます。
- メソッドキャンセルモードが無効の場合

タイムアウト発生時に 1 回だけ出力されます。

また，タイムアウトを検知したメソッドが終了した場合には，KDJE52716-I が出力されます。これは，メソッドキャンセルの実行が成功したかどうかとは関係ありません。また，実行しないで終了した場合にも出力されます。つまり，一つのメソッドの実行について KDJE52703-W と KDJE52716-I は，1 対 1 で出力されます。

図 5-9　タイムアウトの延長でメソッドキャンセルを実行した場合に出力されるログ情報

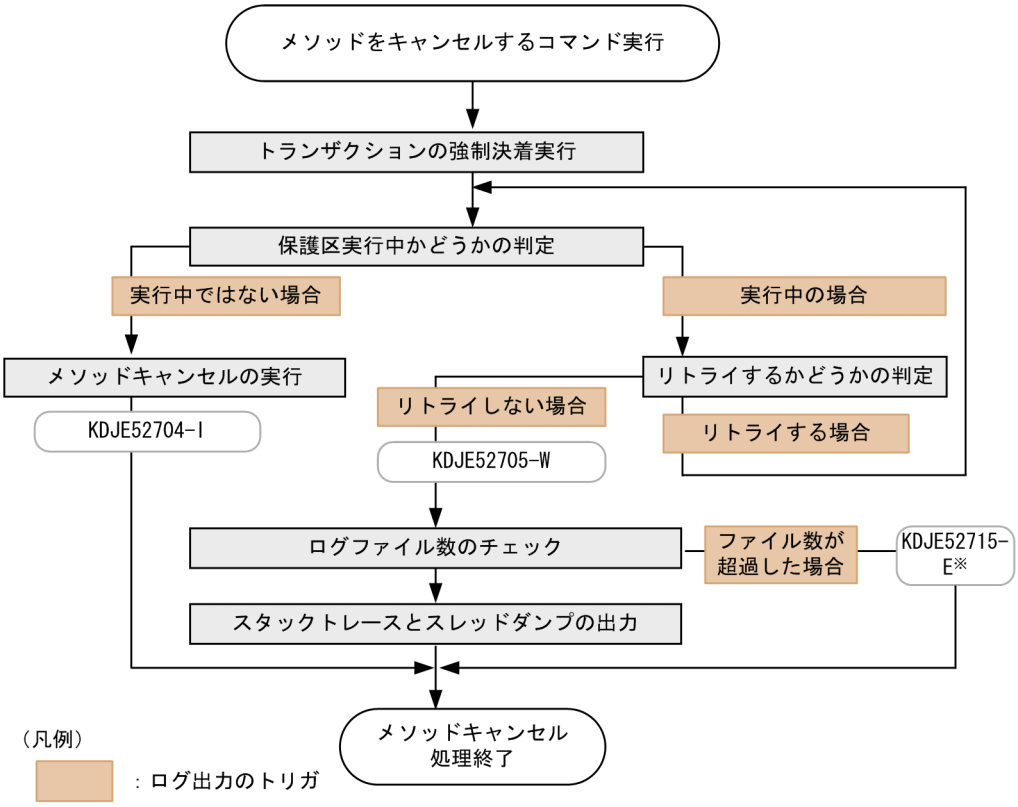


(3) コマンドによってメソッドキャンセルを実行した場合に出力されるログ情報

タイムアウトが発生したメソッドを，J2EE アプリケーションの強制停止またはメソッドキャンセルコマンド（cjstopthread）を実行してキャンセルする場合に出力されるログ情報を次の図に示します。どのメッセージが出力された場合も，処理は続行されます。

なお，スレッドダンプはメソッドキャンセルが失敗した場合に 1 回だけ出力されます。また，メソッドキャンセルコマンドの実行に失敗した場合は，メッセージ KDJE52713-E が出力されます。

図 5-10　コマンドによってメソッドキャンセルを実行した場合に出力されるログ情報



注※
usrconf.propertiesでejbserver.server.threaddump.filenumに指定した値よりも多くのログファイルを出力しようとした場合，メッセージ「KDJE52715-E」が出力されて，スタックトレースとスレッドダンプは出力されません。

(4) 注意事項

J2EE サーバでのスレッドダンプファイル数の上限チェック後，スレッドダンプ出力要求と JavaVM でのスレッドダンプ出力処理は非同期に実行されるため，タイムラグが発生することがあります。このため，スレッドダンプファイルが上限値を超えて出力されることがあります。

スレッドダンプの出力数の上限値は，「設定したスレッドダンプの上限値+9」になります。

また，JavaVM のスレッドダンプ出力処理では，同時に複数の出力要求があった場合，スレッドダンプを出力できないことがあります。この場合には，スレッドダンプの数が，実際に出力しようとした数よりも少なくなります。

5.4　J2EE アプリケーションのサービスの閉塞

この節では，Web アプリケーションのサービスの閉塞と，CTM のスケジュールキューの閉塞について説明します。

この節の構成を次の表に示します。

表 5-16　この節の構成（J2EE アプリケーションのサービスの閉塞）

分類	タイトル	参照先
解説	Web アプリケーションのサービスの閉塞と閉塞解除とは	5.4.1
運用	システムごとに実行できるサービスの閉塞方法および J2EE アプリケーションの停止方法	5.4.2
	負荷分散機を利用したサービス閉塞	5.4.3

注　「実装」,「設定」,「注意事項」について，この機能固有の説明はありません。

サービスとは，J2EE アプリケーションで提供される業務を指します。新しいリクエストについてはエラーを返したり，実行中のリクエストについては処理を続行してエラーを返さないようにしたりして，サービスを閉塞できます。

5.4.1　Web アプリケーションのサービスの閉塞と閉塞解除とは

Web アプリケーションのサービスの閉塞とは，Web アプリケーションをフロントとする J2EE アプリケーションのサービスを閉塞することを指します。Web アプリケーションのサービスの閉塞は，例えば，業務時間外やシステムのメンテナンス時に，サービスを停止するときに実施します。

(1)　サービス閉塞の方法

Web アプリケーションのサービスを閉塞する方法には，次の五つがあります。

- 負荷分散機でリクエストの振り分け先を変更する方法※1
- 負荷分散機を停止する方法※2
- Web サーバを停止する方法※3
- J2EE サーバを停止する方法
- アプリケーションを停止する方法

注※1

負荷分散機でリクエストの振り分け先を変更する方法の場合，実行中のリクエストの処理を，エラーを返さないで続行させるため，負荷分散機の次の機能を使用します。

- リクエスト振り分け先の変更時に，変更前の負荷分散機と Web サーバ間の接続を，リクエスト処理が完了するまで保持する機能

注※2

負荷分散機を停止する方法の場合，負荷分散機の停止中に，実行中のリクエストの処理を，エラーを返さないで続行させるため，負荷分散機の次の機能を使用します。

- 負荷分散機の停止時に，負荷分散機と Web サーバ間の接続を，リクエスト処理が完了するまで保持する機能

注※3

Web サーバを停止する方法は，Web サーバに HTTP Server を使用している場合だけできます。Microsoft IIS を使用している場合は，実行中のリクエストにエラーを返すため，使用できません。

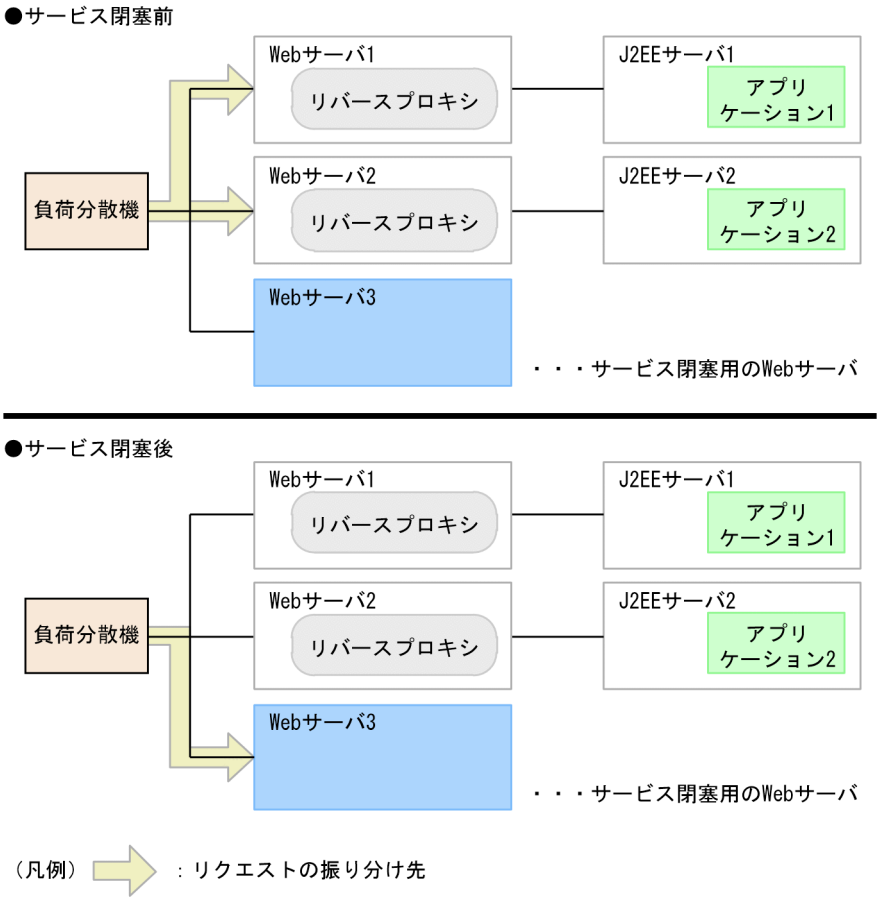
(2) 負荷分散機を使用したシステム構成でのサービス閉塞

サービス閉塞の方法には，負荷分散機を使用する場合と使用しない場合で手順が異なります。ここでは，負荷分散機でリクエストの振り分け先を変更することによって，サービスを閉塞する場合を例に，サービスの閉塞について説明します。

負荷分散機を使用してリクエストの振り分け先を変更する場合，サービス閉塞用の Web サーバを配置する必要があります。稼働中の Web アプリケーションの閉塞後は，サービス閉塞用の Web サーバにリクエストを転送するよう，負荷分散機を設定します。

サービス閉塞前とサービス閉塞後のリクエストの振り分けについて，次の図に示します。

図 5-11　サービス閉塞前とサービス閉塞後のリクエストの振り分け



なお，サービス閉塞用の Web サーバでは，Web アプリケーションのサービス閉塞後，リクエストが来たときには，エラーページを返すようにします。

表示させるエラーページはカスタマイズすることもできます。表示されるエラーページについては，「5.4.1(3)　エラーページの表示」を参照してください。

(3)　エラーページの表示

閉塞した Web アプリケーションのサービスにアクセスした場合，クライアントにはエラーページが表示されます。表示されるエラーページは，次のどれかとなります。

- Web サーバのデフォルトのエラーページ，またはユーザが作成したエラーページ
- 接続エラーのページ※
- エラーステータスコードの表示されたエラーページ

注※

接続先の HTTP ポートが閉じられている場合に表示されます。

(a) サービス閉塞の方法と表示されるエラーページ

表示されるエラーページは，Web アプリケーションのサービス閉塞の方法によって異なります。サービス閉塞の方法と，表示されるエラーページを次の表に示します。

表 5-17 サービス閉塞の方法と表示されるエラーページ

サービス閉塞の方法	表示されるエラーページ
負荷分散機でリクエストの振り分け先を変更する方法	サービス閉塞用の Web サーバで設定したエラーページ
負荷分散機を停止する方法	接続エラーのページ
Web サーバを停止する方法	接続エラーのページ※1
J2EE サーバを停止する方法	次のエラーステータスコードが表示されたページ • 404 エラー※2 • 500 エラー※3 • 503 エラー※4
アプリケーションを停止する方法	次のエラーステータスコードが表示されたページ • 404 エラー※5 • 503 エラー※4

注※1

ただし，負荷分散機を使用している場合は，使用している負荷分散機の仕様に依存します。

注※2

J2EE サーバを停止したときの 404 エラーは，アプリケーションの停止時に，J2EE サーバにはリクエストが到達していて，Web アプリケーションへの実行待ちキューには到達していないリクエストに対して返されます。

注※3

500 エラーは，アプリケーションの停止時に，J2EE サーバに到達していないリクエストに対して返されます。

注※4

503 エラーは，アプリケーションの停止時に，Web アプリケーションの実行待ちキューにあるリクエストに対して返されます。

注※5

アプリケーションを停止したときの 404 エラーは，アプリケーションの停止時に，Web アプリケーションへの実行待ちキューに到達していないリクエストに対して返されます。

(b) ユーザ作成のエラーページの使用

表示されるエラーページは，ユーザ作成のエラーページにカスタマイズできます。ただし，カスタマイズはサービス閉塞の方法によって，できる場合とできない場合があります。サービス閉塞の方法と，エラーページのカスタマイズについて次の表に示します。

表 5-18 サービス閉塞の方法とエラーページのカスタマイズ

サービス閉塞の方法	エラーページのカスタマイズ
負荷分散機でリクエストの振り分け先を変更する方法	○
負荷分散機を停止する方法	×

サービス閉塞の方法	エラーページのカスタマイズ
Web サーバを停止する方法	×
J2EE サーバを停止する方法	○※
アプリケーションを停止する方法	○※

（凡例） ○：できる　　×：できない

注※

エラーページのカスタマイズをするときの Web サーバの設定で，httpsd.conf ファイルの ErrorDocument ディレクティブでフル URL（http://で始まる URL を記述し，他サイトのコンテンツを指定した URL）を指定しているときには，302 エラーが返ります。

エラーページのカスタマイズの設定については，マニュアル「HTTP Server」を参照してください。エラーページのカスタマイズは，Web サーバに HTTP Server を使用している場合だけできます。

(4)　サービス閉塞の解除

Web アプリケーションのサービス閉塞の解除は，サービス閉塞の方法によって異なります。サービス閉塞の方法ごとに，サービス閉塞の解除の方法を示します。

- 負荷分散機でリクエストの振り分け先を変更する方法の場合
  J2EE サーバへリクエストが転送されるように，負荷分散機のリクエスト振り分け先を変更します。
- 負荷分散機を停止する方法の場合
  負荷分散機を起動します。
- Web サーバを停止する方法の場合
  Web サーバを起動します。
- J2EE サーバを停止する方法の場合
  J2EE サーバを起動します。
- アプリケーションを停止する方法の場合
  アプリケーションを開始します。

5.4.2　システムごとに実行できるサービスの閉塞方法および J2EE アプリケーションの停止方法

サービス閉塞は，クライアントから見てリクエストの受け口になる要素（**フロント**）に対して実行します。フロントを閉塞すると，クライアントから新しいリクエストを受け付けることなく，すでに受け付けた J2EE アプリケーション内のリクエストを継続実行できます。

Web フロントシステムの場合や CTM を使用したバックシステムの場合，J2EE アプリケーションのフロントに，負荷分散機，Web サーバまたは CTM などがあります。J2EE アプリケーションを直接停止する前にこれのフロントに当たる要素を閉塞すると，確実なサービス閉塞が実行できます。

なお，サービス閉塞後に実行する J2EE アプリケーションは，運用管理コマンド（mngsvrutil）で停止してください。

Web フロントシステム，バックシステムなどのシステムの種類の意味については，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「3.1.1　システムの目的と構成」を参照してください。

(1)　Web フロントシステムでのサービス閉塞

Web フロントシステムとは，フロントエンドである Web ブラウザから送信されるリクエストを受け付けて，そのリクエストを処理するシステムです。

サービス閉塞の対象は，J2EE アプリケーション内の Web アプリケーションです。

このシステムでは，次の表に示す方法でサービス閉塞を実行できます。J2EE アプリケーションの停止を実行する前にほかの方法でサービス閉塞を実行することで，サービスを確実に閉塞できます。

表 5-19　Web フロントシステムで実行できるサービス閉塞

サービスの閉塞方法	説明	手段	参照先
負荷分散機のリクエストの振り分け先を変更する※	負荷分散機の設定を変えて，J2EE アプリケーションを停止する J2EE サーバにリクエストを振り分けないようにします。これによって，J2EE サーバはリクエストを受け付けなくなります。	負荷分散機の機能	5.4.3
負荷分散機を停止する※	負荷分散機を停止して，J2EE サーバへのリクエストの転送を停止します。これによって，J2EE サーバはリクエストを受け付けなくなります。	負荷分散機の機能	5.4.3
J2EE アプリケーションを通常停止する	J2EE アプリケーションを通常停止すると，その J2EE アプリケーションは新たなリクエストを受け付けなくなります。ただし，実行中の処理が終了するまで，停止は完了しません。	サーバ管理コマンド	5.5.7

注※

負荷分散機を使用したシステムの場合に実行できます。

(2)　バックシステムでのサービス閉塞（CTM を使用しているシステムの場合）

バックシステムとは，Web フロントシステムの背後などで動作する，複数の業務システムに共通な業務サービスを実行するためのシステムです。

CTM を使用しているシステムの場合，CTM のスケジュールキューを対象にしたサービス閉塞を実行できます。

このシステムでは，次の表に示す方法でサービス閉塞を実行できます。J2EE アプリケーションの停止を実行する前に，ほかの方法でサービス閉塞を実行することで，サービスを確実に閉塞できます。

表 5-20　バックシステム（CTM を使用している場合）で実行できるサービス閉塞

サービスの閉塞方法	説明	手段	参照先マニュアル	参照個所
CTM の機能を使用してサービス閉塞を実行する	CTM の機能を使用してスケジュールキューを閉塞して，J2EE サーバへのリクエストの転送を停止します。これによって，J2EE サーバはリクエストを受け付けなくなります。	運用管理コマンド	機能解説 拡張編	3.7.4
J2EE アプリケーションを通常停止する	J2EE アプリケーションを通常停止すると，その J2EE アプリケーションは新たなリクエストを受け付けなくなります。ただし，実行中の処理が終了するまで，停止は完了しません。	サーバ管理コマンド	このマニュアル	5.5.7

(3)　バックシステムでのサービス閉塞（CTM を使用していないシステムの場合）

CTM を使用していないバックシステムの場合，サービス閉塞の対象は，J2EE アプリケーション内のフロントの Enterprise Bean（**フロント EJB**）です。

このシステムでは，次の方法でサービス閉塞を実行できます。

表 5-21　バックシステム（CTM を使用していない場合）で実行できるサービス閉塞

サービスの閉塞方法	説明	手段	参照先
J2EE アプリケーションを通常停止する	J2EE アプリケーションを通常停止すると，その J2EE アプリケーションは新たなリクエストを受け付けなくなります。ただし，実行中の処理が終了するまで，停止は完了しません。	サーバ管理コマンド	5.5.7

なお，システム構築時にあらかじめフロント EJB を定義しておく必要があります。

5.4.3　負荷分散機を利用したサービス閉塞

ここでは，負荷分散機を利用したサービス閉塞の実行方法について説明します。負荷分散機を利用したサービス閉塞は，Web フロントシステムの場合に，負荷分散機を利用しているときに実行できます。負荷分散機を利用したサービスの閉塞には，次の二つの方法があります。

- 通常のサービス閉塞
- サービス部分閉塞

ぞれぞれの実行方法について説明します。

ポイント

負荷分散機を利用したサービス閉塞は，Smart Composer 機能で負荷分散機を管理していない場合に使用します。Smart Composer 機能で負荷分散機を管理している場合のサービス閉塞は，cmx_stop_target コマンドを使用します。cmx_stop_target 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_stop_target（Web システムまたはサービスユニットの停止)」を参照してください。

(1) 通常のサービス閉塞

負荷分散機を利用した通常のサービス閉塞は，次のどちらかの方法で実行します。

- 負荷分散機のリクエストの振り分け先を変更する
  通常使用する Web サーバのほかにサービス閉塞用の Web サーバを用意しておいて，サービス閉塞するタイミングでリクエストの振り分け先をサービス閉塞用の Web サーバに切り替える方法です。サービス閉塞用の Web サーバには，エラーページ表示の設定などをしておきます。
- 負荷分散機を停止する
  負荷分散機を停止して，リクエストの受け付けを停止する方法です。すでに Web サーバに送られたリクエストはそのまま実行されます。

なお，サービス閉塞を解除する場合は，リクエストの振り分け先を再度変更するか，または負荷分散機を起動します。

手順については，ご使用の負荷分散機の使用方法に従ってください。

(2) 負荷分散機を利用したサービス部分閉塞

リクエストを受け付ける Web サーバ，およびリクエストを実行する J2EE サーバを負荷分散機を使用して冗長化している構成の場合，サービス全体を停止させないで，サービスを構成する一部の Web アプリケーションだけを停止できます。この閉塞方法を，**サービス部分閉塞**といいます。

なお，Web アプリケーションのサービス部分閉塞ができるのは，次のような構成のシステムです。

- 負荷分散機を使用したシステム
- Web サーバと J2EE サーバが冗長化されていて，Web サーバと J2EE サーバが 1:1 または 1:N の構成であること（N は 1 以上の整数）

Web アプリケーションのサービス部分閉塞は，負荷分散機によるリクエストの振り分け先を変更して，サービスを閉塞する系にリクエストを振り分けないことで実現できます。実行方法については，負荷分散機の使用方法を確認してください。

ポイント

負荷分散機によってサービス部分閉塞を実行する場合，停止する Web サーバに対して作成されたセッションを使用したリクエストについては，セッションが切断されます。この場合，稼働中の Web サーバに対してセッションを接続し直す必要があります。ただし，セッションフェイルオーバ機能を使用している場合は，セッションの一部の情報を引き継げます。セッションフェイルオーバ機能については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「5. J2EE サーバ間のセッション情報の引き継ぎ」を参照してください。

5.5　J2EE アプリケーションの停止

この節では，通常停止と強制停止の概要や，処理の流れについて説明します。

この節の構成を次の表に示します。

表 5-22　この節の構成（J2EE アプリケーションの停止）

分類	タイトル	参照先
解説	J2EE アプリケーションの停止とは	5.5.1
	閉塞処理とは	5.5.2
	停止処理とは	5.5.3
	強制停止処理とは	5.5.4
実装	cosminexus.xml での定義	5.5.5
設定	実行環境での設定	5.5.6
運用	J2EE アプリケーションの停止	5.5.7

注　「注意事項」について，この機能固有の説明はありません。

5.5.1　J2EE アプリケーションの停止とは

J2EE アプリケーションの停止とは，J2EE アプリケーションを構成する Web アプリケーションや Enterprise Bean の終了処理を実行して，J2EE アプリケーションをクライアントからのリクエストを受け付けない状態にすることです。この状態を，停止状態といいます。そのあとで，J2EE アプリケーション自体を停止します。

ここでは，J2EE アプリケーション停止の種類，停止のしかたについて説明します。

(1)　J2EE アプリケーション停止の種類

J2EE アプリケーション停止の種類には，次の二つがあります。

通常停止

日常運用の中で，J2EE アプリケーションを停止するときの停止方法です。J2EE アプリケーション内の Web アプリケーションや Enterprise Bean を順番に停止して，サービスを安全に停止させます。なお，J2EE アプリケーション内でリクエストが実行されている場合，リクエストが終了するまで J2EE アプリケーションは停止しません。ただし，通常停止を実行するときには，タイムアウトを設定できます。タイムアウトを設定しておくと，J2EE アプリケーションが停止していない場合でも，一定時間が経過したらクライアントに制御が戻ります。

通常停止では，まず，サービスを閉塞してリクエストの受け付けを中止してから，J2EE アプリケーションを停止します。**サービス閉塞**とは，新しいリクエストを受け付けない状態で，すでに受け付けたリクエストを実行することです。

例えば，次のような場合に計画的に J2EE アプリケーションを停止するときには，まず，サービス閉塞を実行してから，J2EE アプリケーションを停止します。

- 一日のうち特定の時間内にだけサービスを提供する場合
- システムをメンテナンスする場合

強制停止

J2EE アプリケーションで処理中のリクエストが完了するのを待たないで，リクエストを強制的に停止する方法です。強制停止によって，使用されていたリソースは解放されます。

J2EE アプリケーションでトラブルが発生した場合など，クライアントに制御が戻らないときには，強制停止によってサービスが迅速に停止できるようになります。

ポイント

次の場合には J2EE アプリケーションの強制停止を実行できません。

- **ステートメントキャンセルが実行できない場合**

  データベースを操作している場合に実行中の SQL の処理が戻らなくなったとき，ステートメントキャンセルが実行されます。**ステートメントキャンセル**は，実行中の SQL の処理を取り消す処理のことです。

  ただし，リソースアダプタの設定，システム構成，使用しているトランザクションの種類などによっては，ステートメントキャンセルができない場合があります。ステートメントキャンセルができない場合，SQL を実行中のトランザクションのタイムアウト，および SQL の処理が戻らない J2EE アプリケーションの強制停止は実行できません。

  ステートメントキャンセルについては，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「3.15.8　トランザクションタイムアウトとステートメントキャンセル」を参照してください。

- **メソッドキャンセルが実行できない場合**

  強制停止を実行すると，実行中のスレッドは，メソッドキャンセルによって予期しない場所で中断されます。このため，メソッドキャンセルの対象スレッドとほかのスレッドで更新または削除を実行する共有データがある J2EE アプリケーションの場合，メソッドキャンセルを使用できません。

  メソッドキャンセルが実行できない J2EE アプリケーションの構成については，「5.3.4　メソッドキャンセルとは」を参照してください。

- **CTM を使用する環境で，J2EE アプリケーションのアンデプロイ時のスケジュールキューを非活性にするための設定に，タイムアウト時間として 0（ゼロ）を設定した場合**

  CTM のスケジュールキューを非活性にするための処理のタイムアウト時間として 0 を設定した場合，リクエストが終了するまで CTM による閉塞が完了しません。このため，J2EE アプリケーションを停止できなくなります。

詳細は，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「3.7.4　スケジュールキューの閉塞制御」を参照してください。

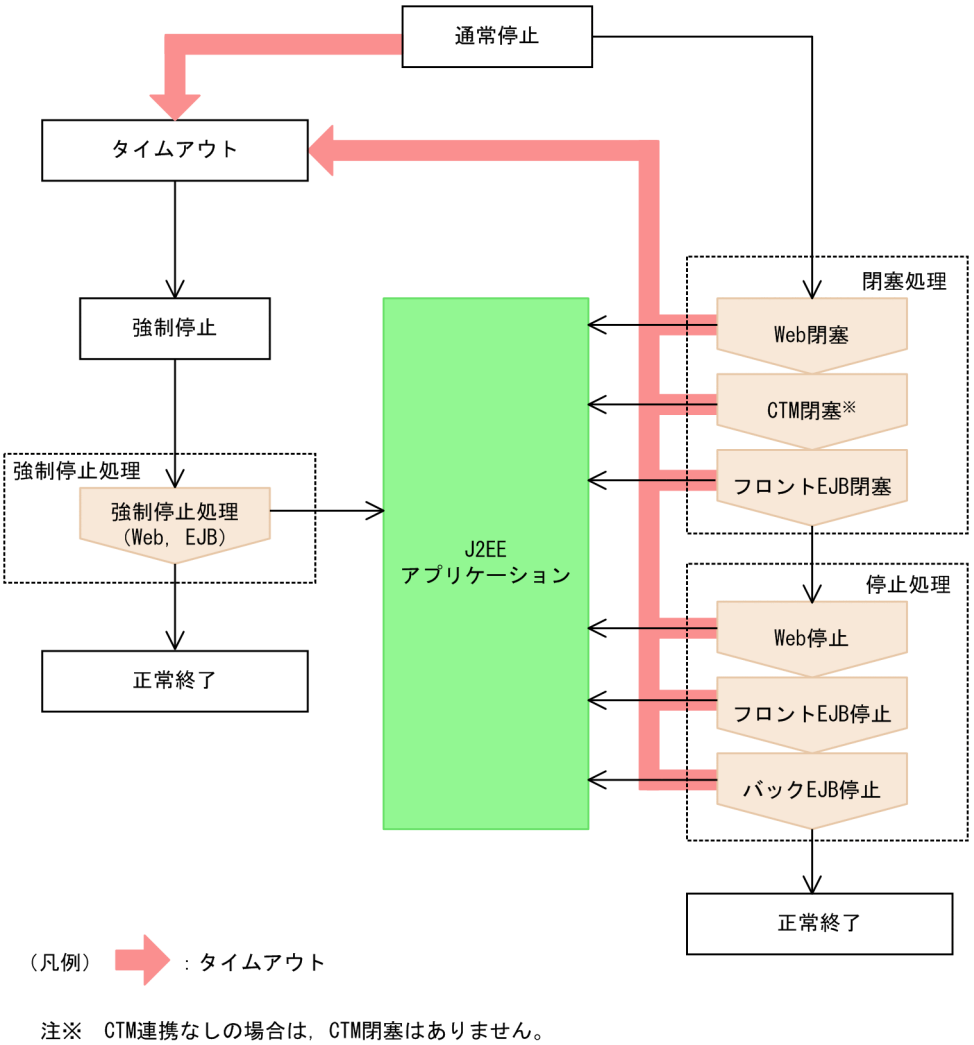
(2)　停止のしかた

通常停止，および強制停止は，サーバ管理コマンドで実行します。J2EE アプリケーションの停止については，「5.5　J2EE アプリケーションの停止」を参照してください。

(3)　J2EE アプリケーション停止処理の流れ

ここでは，J2EE アプリケーション停止処理の流れについて説明します。通常停止と強制停止の処理の流れを次の図に示します。

図 5-12　J2EE アプリケーション停止処理の流れ



注※　CTM連携なしの場合は，CTM閉塞はありません。

通常停止処理では，閉塞処理と停止処理が実施されます。

閉塞処理，停止処理，および強制停止処理について，次項以降で説明します。

5.5.2　閉塞処理とは

閉塞処理では，閉塞対象の J2EE アプリケーションを構成する要素のうち，リクエストの受け口となるフロント部分を閉塞し，新規リクエストの受け付けを終了します。

なお，処理中のリクエストは，引き続き処理されます。

(1)　閉塞順序と処理内容

閉塞順序を次に示します。

1. Web アプリケーションの閉塞処理
2. CTM のスケジュールキューの閉塞処理※
3. フロント EJB の閉塞処理

注※

CTM 連携なしの場合は，CTM のスケジュールキューの閉塞処理はありません。

それぞれの閉塞処理での処理内容について説明します。

(a)　Web アプリケーションの閉塞処理

次の処理が実行されます。

- 新規リクエストの受け付けを終了します。
- 処理中のリクエストは引き続き処理します。
- Web アプリケーションの同時実行数制御のキューにあるリクエストのうち，Web コンテナで処理を開始していないリクエストについては，処理しません。すべて 503 エラーをクライアントに返します。

(b)　CTM のスケジュールキューの閉塞処理

次の処理が実行されます。

- 新規リクエストの受け付けを終了します。
- CTM のスケジュールキューに投入されたリクエストのうち，J2EE サーバに振り分けられていないリクエストについては，処理しません。java.rmi.RemoteException をクライアントに返します。
- CTM のスケジュールキューに投入されたリクエストのうち，すでに J2EE サーバで処理が開始しているリクエストについては，引き続き処理します。

(c)　フロント EJB の閉塞処理

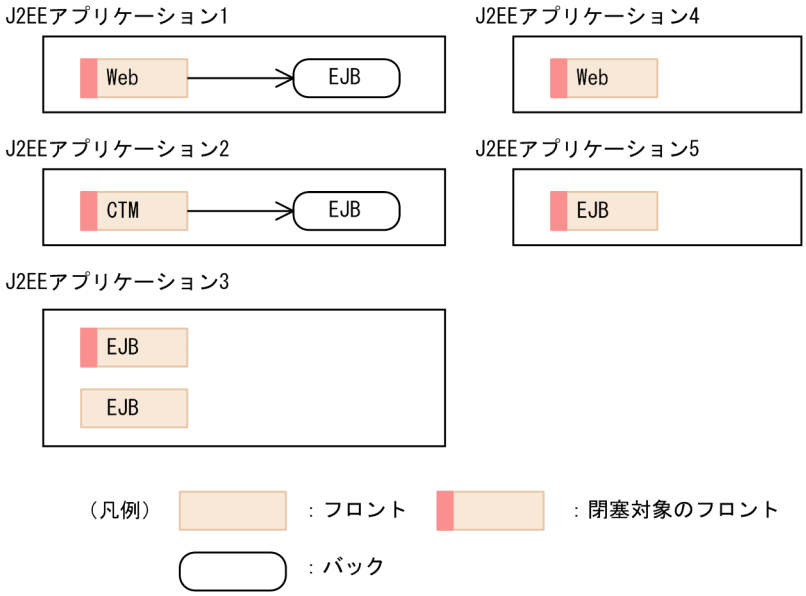
次の処理が実行されます。

- 新規リクエストの受け付けを終了します。
- 処理中のリクエストは，引き続き処理します。

(2)　J2EE アプリケーションの構成パターンと閉塞方法

J2EE アプリケーションの構成パターン別の閉塞方法について，J2EE アプリケーションの形態を図に示し，それぞれの閉塞方法を説明します。

図 5-13　J2EE アプリケーションの構成パターン



- **J2EE アプリケーション 1 の閉塞方法**

  J2EE アプリケーション 1 は，Web アプリケーションを含みます。

  この場合，フロントは通常，Web アプリケーションとなります。J2EE アプリケーション 1 では，Web アプリケーションを閉塞します。

- **J2EE アプリケーション 2 の閉塞方法**

  J2EE アプリケーション 2 は，EJB を CTM 連携で呼び出します。

  この場合，クライアントからの呼び出しが必ず CTM になるので，CTM がフロントとなります。J2EE アプリケーション 2 では，CTM を閉塞します。

- **J2EE アプリケーション 3 の閉塞方法**

  J2EE アプリケーション 3 は，CTM を使用しないで EJB を呼び出します。

  この場合，どの EJB がフロントとなるのか J2EE アプリケーション構成からは判断できません。フロントとなる EJB（フロント EJB）は，ユーザが指定します。J2EE アプリケーション 3 では，ユーザが指定したフロント EJB を閉塞します。

- **J2EE アプリケーション 4 の閉塞方法**

  J2EE アプリケーション 4 は，Web アプリケーションが単独である J2EE アプリケーションです。

  この場合，Web アプリケーションがフロントとなります。J2EE アプリケーション 4 では，Web アプリケーションを閉塞します。

- **J2EE アプリケーション 5 の閉塞方法**

  J2EE アプリケーション 5 は，EJB が単独である J2EE アプリケーションです。

この場合，EJB がフロントとなります。フロントとなる EJB（フロント EJB）は，ユーザが指定します。J2EE アプリケーション 5 では，ユーザが指定したフロント EJB を閉塞します。

なお，フロント EJB の指定については，「5.5.6　実行環境での設定」を参照してください。

(3)　閉塞処理を実行する場合の注意事項

閉塞処理を実行する場合は，次の点に注意する必要があります。

(a)　CTM 使用時の注意

CTM 使用時に閉塞処理を実行する場合の注意事項を次に示します。

- **Web アプリケーションまたは EJB が，CTM を使用して同一 J2EE アプリケーション内にある EJB にアクセスする場合**

  閉塞処理が終了した時点で，処理中のリクエストもクライアントに例外を返します。このような形態の場合，該当する Web アプリケーションまたは EJB を呼び出すクライアントの処理を停止したあとに，J2EE アプリケーションを停止してください。

- **CTM から呼び出されている EJB に対して，閉塞処理後に CTM を使用しないでリクエストが投入された場合**

  EJB の閉塞処理が実施されていないため，新規リクエストが受け付けられます。このような形態の場合，該当する EJB を呼び出すクライアントの処理を停止したあとに，J2EE アプリケーションを停止してください。

- **usrconf.properties の ejbserver.ctm.DeactivateTimeOut キーに 0 を指定している場合**

  usrconf.properties では，ejbserver.ctm.DeactivateTimeOut キーには 0 を指定しないでください。0 を指定している場合，CTM 経由で仕掛り中のリクエストがある状態でアプリケーションを停止しようとすると，リクエストが終了するまで閉塞が完了しません。このため，リクエストが障害などの理由で終了しない場合，強制停止ができなくなります。

(b)　J2EE アプリケーションをわたった Web アプリケーション呼び出しの注意

Web アプリケーションでは，javax.servlet.ServletContext の getContext を使用して，同じ Web コンテナ上の Web アプリケーションに対してアクセスできます。アクセスする Web アプリケーションが，別の J2EE アプリケーションに配置されている場合，呼び出し元になるフロントの Web アプリケーションを含む J2EE アプリケーションを最初に停止してください。

5.5.3　停止処理とは

停止処理では，J2EE アプリケーションのストップ処理，アンデプロイ処理の順に実施されます。それぞれの処理について説明します。

(1) ストップ処理

ストップ処理は，次の順で実施されます。なお，処理完了後は，J2EE アプリケーションは実行できなくなります。

1. Web アプリケーションのストップ処理
2. EJB のストップ処理
3. J2EE アプリケーションのストップ処理

それぞれの処理について説明します。

(a) Web アプリケーションのストップ処理

Web アプリケーションのストップ処理は，次の順序で実施されます。

1. **サーブレット/JSP の終了処理**
   サーブレットの destroy，JSP の jspDestroy が実施されます。
2. **HTTP セッションの破棄**
   HTTP セッションの属性の削除および無効化が実施されます。
3. **フィルタの終了処理**
   フィルタの destroy が実施されます。
4. **サーブレットコンテキストの破棄**
   サーブレットコンテキストの属性の破棄および無効化が実施されます。
5. **名前空間から Web アプリケーションの実行環境の削除**
   名前空間から，ejb-ref，ejb-local-ref，resource-ref，env-entity などが削除されます。

(b) EJB のストップ処理

EJB のストップ処理は，次の順序で実施されます。

1. **名前空間から EJB の実行環境の削除**
   名前空間から，ejb-ref，ejb-local-ref，resource-ref，env-entity などが削除されます。
2. **名前空間からの EJB ホームオブジェクトまたは EJB ローカルホームオブジェクトの削除**
   名前空間から，EJB ホームオブジェクトまたは EJB ローカルホームオブジェクトが削除されます。
3. **プールの削除**
   Session Bean，および Message-driven Bean の場合は，method-ready プールが削除されます。
   Entity Bean の場合は，pool プールと ready プールが削除されます。

(c) J2EE アプリケーションのストップ処理

J2EE アプリケーションのストップ処理では，名前空間から J2EE アプリケーションのエントリが削除されます。

(2) アンデプロイ処理

アンデプロイ処理は，次の順で実施されます。なお，処理完了後は，J2EE アプリケーションは J2EE サーバから削除されます。

1. Web アプリケーションのアンデプロイ処理
2. EJB のアンデプロイ処理
3. J2EE アプリケーションのアンデプロイ処理

それぞれの処理について説明します。

(a) Web アプリケーションのアンデプロイ処理

JSP のコンパイル結果である，Java ソースファイルおよびクラスファイルを削除します。なお，JSP のコンパイル結果については，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「2.5.6　JSP 事前コンパイルを使用しない場合の JSP コンパイル結果」を参照してください。

(b) EJB のアンデプロイ処理

EJB のアンデプロイ処理は，次の順序で実施されます。

1. リモートアダプタの削除（リモートインタフェースサポート時だけ）
2. CMR 使用時に作成したテーブルの削除

(c) J2EE アプリケーションのアンデプロイ処理

J2EE アプリケーションのアンデプロイ処理では，ユーザが手動でダウンロードしたファイルを削除します。

5.5.4 強制停止処理とは

強制停止処理では，通常停止のときの閉塞処理や停止処理は実行されません。J2EE アプリケーションで処理中のリクエストがある場合でも，強制的に処理が中止されます。なお，実行中のメソッドについてはメソッドキャンセルが実行されます。

J2EE アプリケーションを強制停止したときのトランザクションの扱い

J2EE アプリケーションの強制停止を実行すると，仕掛かり中のトランザクションは強制的にタイムアウトされます。このため，J2EE アプリケーションを強制停止したあとにトランザクションを開始しようとすると，例外が発生してトランザクションを開始できません。

ただし，インプロセス OTS が複数の JavaVM にわたって動作しているときには，J2EE アプリケーションを強制停止しても，トランザクションは強制的にタイムアウトされません。

また，J2EE アプリケーションの強制停止時に，Statement，CallableStatement，または PreparedStatement で SQL が実行されている場合，実行は取り消され，制御を J2EE アプリケーションに戻そうとします。J2EE アプリケーションで使用していたコネクションは破棄され，プールには戻りません。

トランザクションタイムアウトの詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「3.15.8　トランザクションタイムアウトとステートメントキャンセル」を参照してください。

5.5.5　cosminexus.xml での定義

アプリケーションの開発環境で必要な cosminexus.xml での定義について説明します。

J2EE アプリケーションの強制停止の機能を使用するための定義は，cosminexus.xml の<session>，<entity>，または<message>タグ下の<front-ejb>タグ内に指定します。

指定するタグの詳細は，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(アプリケーション／リソース定義)」の「2.2　アプリケーション属性ファイル（cosminexus.xml）で指定する各属性の詳細」を参照してください。

5.5.6　実行環境での設定

J2EE アプリケーションの運用機能を使用する場合，J2EE サーバおよび J2EE アプリケーションの設定が必要です。

(1)　J2EE サーバの設定

J2EE アプリケーションの運用機能を使用する場合，J2EE サーバの設定が必要です。J2EE サーバの設定は，簡易構築定義ファイルで実施します。J2EE アプリケーションの運用機能の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に指定します。

表 5-23　J2EE アプリケーションの運用機能に必要な定義

項目	指定するパラメタ	設定内容
J2EE アプリケーションの強制停止	ejbserver.deploy.app.stopforcibly.disabled※1	J2EE アプリケーションの強制停止機能を使用するかどうかを指定します。 なお，J2EE アプリケーションのフロント（クライアントから見てリクエストの受け口になる要素）が Enterprise Bean の場合には，サーバ管理コマンドを使用してフロントとなる Enterprise Bean（フロント EJB）を指定する必要があります。サーバ管理コマンドを使用した J2EE アプリケーションの

項目	指定するパラメタ	設定内容
		強制停止の設定については，「5.5.6(2)　J2EE アプリケーションの設定」を参照してください。
J2EE アプリケーション実行時間の監視	ejbserver.ext.method_observation.interval※2	J2EE アプリケーション実行時間の監視機能を使用するかどうか，動作中のリクエスト処理がタイムアウトしていないかを調査する時間間隔，およびタイムアウトしたリクエスト（メソッド）をキャンセルする時間間隔を指定します。 なお，J2EE アプリケーション実行時間の監視機能で使用するメソッドタイムアウトの時間，メソッドキャンセルのモードは，サーバ管理コマンドで指定します。 設定方法については，「5.3.9(2)　J2EE アプリケーションの設定」を参照してください。

注※1　簡易構築定義ファイルの，J2EE サーバの JavaVM のシステムプロパティに設定します。

注※2　簡易構築定義ファイルの，J2EE サーバ用のユーザプロパティに設定します。

簡易構築定義ファイル，および指定するパラメタの詳細は，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「4.3　簡易構築定義ファイル」を参照してください。

(2)　J2EE アプリケーションの設定

実行環境での J2EE 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。J2EE アプリケーションの強制停止の定義には，SessionBean 属性ファイル，EntityBean 属性ファイル，または MessageDrivenBean 属性ファイルを使用します。

各属性ファイルで指定するタグは，cosminexus.xml と対応しています。cosminexus.xml での定義については，「5.5.5　cosminexus.xml での定義」を参照してください。

5.5.7　J2EE アプリケーションの停止

ここでは，サーバ管理コマンドを使用して J2EE アプリケーションを停止する方法について説明します。

サーバ管理コマンドを使用した J2EE アプリケーションの停止には，次の五つの方法があります。

1. デフォルトのタイムアウト時間で通常停止を実行する
2. 任意のタイムアウト時間を設定して通常停止を実行する
3. デフォルトのタイムアウト時間で通常停止を実行して，停止しなかった場合に強制停止を実行する
4. 任意のタイムアウト時間を設定して通常停止を実行して，停止しなかった場合に強制停止を実行する
5. 任意のタイムアウト時間を設定して通常停止を実行して，停止しなかった場合に自動的に強制停止を実行する

これらの方法以外の順序でコマンドを実行した場合，コマンドが異常終了します。例えば，通常停止をしない状態では，強制停止の形式でコマンドを実行できません。

また，1.，2.，または 5.の処理は，一つの J2EE アプリケーションに対してどれか 1 回だけ実行できます。

ただし，強制停止は，J2EE サーバの動作設定をカスタマイズして，強制停止が有効になるように設定されているシステムだけで実行できます。J2EE サーバの強制停止の設定については，「5.5.6　実行環境での設定」を参照してください。

J2EE アプリケーションの停止処理では，次の処理が実行されます。

J2EE アプリケーションのフロントが Web アプリケーションの場合

- 新規リクエストを受け付けなくなります。
- 処理中のリクエストは引き続き処理されます。
- Web アプリケーションの同時実行数制御のキューに格納されているリクエストのうち，Web コンテナでの処理が開始されていないものに対して，HTTP503 エラーが返却されます。

J2EE アプリケーションのフロントが Enterprise Bean の場合

- 新規リクエストを受け付けなくなります。
- 処理中のリクエストは引き続き処理されます。

停止後のアプリケーションに含まれている Enterprise Bean のメソッドの呼び出しはできません。停止後のアプリケーションに含まれている Enterprise Bean のメソッドを呼び出そうとすると，「stop could not lock <文字列>」というメッセージが J2EE サーバの標準エラー出力に出力されることがありますが，問題ありません。

次に，それぞれのパターンについて，サーバ管理コマンドの実行手順を説明します。

(1)　デフォルトのタイムアウト時間で通常停止を実行する

コマンドの実行手順，実行形式および実行例を次に示します。

1. **J2EE アプリケーションを通常停止します。**

   cjstopapp コマンドを実行します。

   実行形式

   ```
   cjstopapp <J2EEサーバ名> -name <J2EEアプリケーション名>
   ```

   実行例

   ```
   cjstopapp Myserver -name App1
   ```

   なお，デフォルトのタイムアウト時間でコマンドを実行した場合，停止処理が完了しなくても 60 秒後にコマンドの制御が戻ります。

2. **コマンドの制御がタイムアウトによって戻った場合は，J2EE アプリケーションの停止状態を確認します。**

   タイムアウトによってコマンドの制御が戻った場合，J2EE アプリケーションの停止処理は完了していないことがあります。このため，タイムアウトが発生した場合は，J2EE アプリケーションの状態を確認してください。

J2EE アプリケーションの状態は，サーバ管理コマンドの cjlistapp コマンドを実行して確認します。

実行形式

```
cjlistapp 〈サーバ名称〉
```

実行例

```
cjlistapp MyServer
```

J2EE アプリケーションの状態として，次のどれかの状態が出力されます。

表 5-24　J2EE アプリケーションの状態

出力される文字列	状態の意味
running	開始状態
stopped	停止状態
stopFailure	通常停止失敗状態
forceStopFailure	強制停止失敗状態
blockadeFailure	閉塞失敗状態
blockading	閉塞中
blockaded	閉塞状態
stopping	通常停止中
forceStopping	強制停止中

なお，手順の実行後，J2EE サーバの状態を確認してください。次の状態の場合は，J2EE サーバを再起動する必要があります。

- 閉塞失敗
- 通常停止失敗

(2)　任意のタイムアウト時間を設定して通常停止を実行する

コマンドの実行手順，実行形式および実行例を次に示します。

1. **J2EE アプリケーションを通常停止します。このとき，コマンドのオプションに，コマンドの制御を戻すためのタイムアウト時間を設定します。**

   cjstopapp コマンドに，-t オプションを指定して実行します。

   実行形式

   ```
   cjstopapp 〈J2EEサーバ名〉 -name 〈J2EEアプリケーション名〉 -t 〈タイムアウト時間〉
   ```

   実行例

   ```
   cjstopapp MyServer -name App1 -t 120
   ```

この実行例を実行した場合，停止処理が完了しなくても 120 秒後にコマンドの制御が戻ります。

2. タイムアウトによってコマンドの制御が戻った場合は，J2EE アプリケーションの停止状態を確認します。

コマンドの制御がタイムアウトによって戻った場合，J2EE アプリケーションの停止処理は完了していないことがあります。このため，タイムアウトが発生した場合は，J2EE アプリケーションの状態を確認してください。

J2EE アプリケーションの状態は，cjlistapp コマンドを実行して確認します。

実行形式

```
cjlistapp <J2EEサーバ名>
```

実行例

```
cjlistapp MyServer
```

J2EE アプリケーションの状態として出力される内容については，「5.5.7(1)　デフォルトのタイムアウト時間で通常停止を実行する」の手順 2.を参照してください。

なお，手順の実行後，J2EE サーバの状態を確認してください。次の状態の場合は，J2EE サーバを再起動する必要があります。

- 閉塞失敗
- 通常停止失敗

(3)　デフォルトのタイムアウト時間で通常停止を実行して，停止しなかった場合に強制停止を実行する

コマンドの実行手順，実行形式および実行例を次に示します。

1. J2EE アプリケーションを通常停止します。

cjstopapp コマンドを実行します。

実行形式

```
cjstopapp <J2EEサーバ名> -name <J2EEアプリケーション名>
```

実行例

```
cjstopapp MyServer -name App1
```

なお，デフォルトのタイムアウト時間でコマンドを実行した場合，停止処理が完了しなくても 60 秒後にコマンドの制御が戻ります。

2. タイムアウトによってコマンドの制御が戻った場合は，J2EE アプリケーションの停止状態を確認します。

コマンドの制御がタイムアウトによって戻った場合，J2EE アプリケーションの停止処理は完了していないことがあります。このため，タイムアウトが発生した場合は，J2EE アプリケーションの状態を確認してください。

J2EE アプリケーションの状態は，cjlistapp コマンドを実行して確認します。

実行形式

```
cjlistapp <J2EEサーバ名>
```

実行例

```
cjlistapp MyServer
```

J2EE アプリケーションの状態として出力される内容については，「5.5.7(1)　デフォルトのタイムアウト時間で通常停止を実行する」の手順 2.を参照してください。

3. J2EE アプリケーションの強制停止を実行します。

J2EE アプリケーションが通常停止で停止しなかった場合，強制停止を実行します。

cjstopapp コマンドに，-cancel オプションを指定して実行します。

実行形式

```
cjstopapp <J2EEサーバ名> -name <J2EEアプリケーション名> -cancel
```

実行例

```
cjstopapp MyServer -name App1 -cancel
```

強制停止が成功すると，J2EE アプリケーションで処理中のリクエストが中止されて，先に実行していた通常停止が終了します。

なお，手順の実行後，J2EE サーバの状態を確認してください。次の状態の場合は，J2EE サーバを再起動する必要があります。

- 強制停止失敗
- 閉塞失敗
- 通常停止失敗

(4)　任意のタイムアウト時間を設定して通常停止を実行して，停止しなかった場合に強制停止を実行する

コマンドの実行手順，実行形式および実行例を次に示します。

1. J2EE アプリケーションを通常停止します。このとき，コマンドのオプションに，コマンドの制御を戻すためのタイムアウト時間を設定します。

cjstopapp コマンドに，-t オプションを指定して実行します。

実行形式

```
cjstopapp <J2EEサーバ名> -name <J2EEアプリケーション名> -t <タイムアウト時間>
```

実行例

```
cjstopapp MyServer -name App1 -t 120
```

この実行例を実行した場合，停止処理が完了しなくても 120 秒後にコマンドの制御が戻ります。

2. タイムアウトによってコマンドの制御が戻った場合は，J2EE アプリケーションの停止状態を確認します。

コマンドの制御がタイムアウトによって戻った場合，J2EE アプリケーションの停止処理は完了していないことがあります。このため，タイムアウトが発生した場合は，J2EE アプリケーションの状態を確認してください。

J2EE アプリケーションの状態は，cjlistapp コマンドを実行して，確認します。

実行形式

```
cjlistapp <J2EEサーバ名>
```

実行例

```
cjlistapp MyServer
```

J2EE アプリケーションの状態として出力される内容については，「5.5.7(1)　デフォルトのタイムアウト時間で通常停止を実行する」の手順 2.を参照してください。

3. J2EE アプリケーションの強制停止を実行します。

J2EE アプリケーションが通常停止で停止しなかった場合，強制停止を実行します。

cjstopapp コマンドに，-cancel オプションを指定して実行します。

実行形式

```
cjstopapp <J2EEサーバ名> -name <J2EEアプリケーション名> -cancel
```

実行例

```
cjstopapp Myserver -name App1 -cancel
```

強制停止が成功すると，J2EE アプリケーションで処理中のリクエストが中止されて，先に実行していた通常停止が終了します。

なお，手順の実行後，J2EE サーバの状態を確認してください。次の状態の場合は，J2EE サーバを再起動する必要があります。

- 強制停止失敗
- 閉塞失敗
- 通常停止失敗

(5)　任意のタイムアウト時間を設定して通常停止を実行して，停止しなかった場合に自動的に強制停止を実行する

コマンドの実行手順，実行形式および実行例を次に示します。

1. タイムアウト時間後に強制停止を実行する形式で，J2EE アプリケーションを停止します。

この形式では，J2EE アプリケーションをまず通常停止で停止して，指定したタイムアウト時間内に停止しなかった場合に，強制停止を実行します。

cjstopapp コマンドに，-t オプションと-force オプションを指定して実行します。

実行形式

```
cjstopapp <J2EEサーバ名> -name <J2EEアプリケーション名> -t <タイムアウト時間> -force
```

実行例

```
cjstopapp MyServer -name App1 -t 120 -force
```

なお，手順の実行後，J2EE サーバの状態を確認してください。次の状態の場合は，J2EE サーバを再起動する必要があります。

- 強制停止失敗
- 閉塞失敗
- 通常停止失敗

5.6　J2EE アプリケーションの入れ替え

この節では，J2EE アプリケーションの入れ替えについて，入れ替えの概要や入れ替え方法について説明します。

この節の構成を次の表に示します。

表 5-25　この節の構成（J2EE アプリケーションの入れ替え）

分類	タイトル	参照先
解説	J2EE アプリケーションの入れ替えとは	5.6.1
	Web アプリケーションのサービスの部分閉塞による入れ替え	5.6.2
運用	J2EE アプリケーションの入れ替えと保守	5.6.3

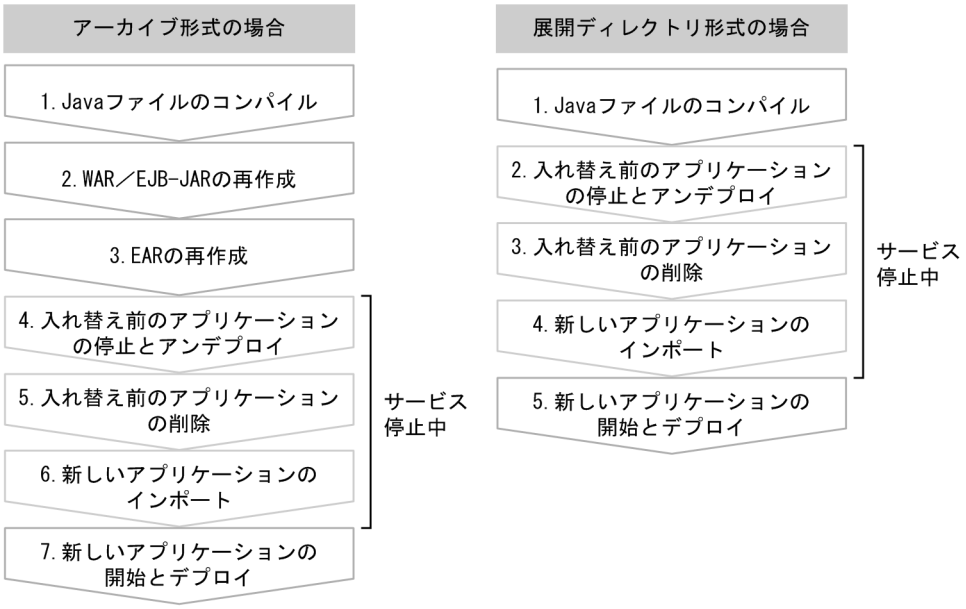
注　「実装」,「設定」,「注意事項」について，この機能固有の説明はありません。

5.6.1　J2EE アプリケーションの入れ替えとは

システムの運用を開始したあとで，J2EE アプリケーションのバージョンアップやメンテナンスを実施するために，J2EE アプリケーションを入れ替えることがあります。

通常，J2EE アプリケーションを入れ替える場合には，J2EE サーバ上で動作している J2EE アプリケーションを停止したあと削除し，新しい J2EE アプリケーションをインポート，デプロイする必要があります。通常の J2EE アプリケーションの入れ替えの手順を次の図に示します。

図 5-14　通常の J2EE アプリケーションの入れ替えの手順



通常の J2EE アプリケーションの入れ替えでは，J2EE アプリケーションを入れ替える間，サービスを停止する必要がありますが，サービスの部分閉塞を使用すると，サービスを停止することなく J2EE アプリケー

ションを入れ替えることができます。また，リデプロイ機能やリロード機能を使用すると，通常の J2EE アプリケーションの入れ替えに比べて，少ない手順で入れ替えができるようになります。

J2EE アプリケーションの入れ替え方法を次の表に示します。

表 5-26　J2EE アプリケーションの入れ替え方法

入れ替え方法	対象の J2EE アプリケーション	
	アーカイブ形式	展開ディレクトリ形式
サービスの部分閉塞による入れ替え	○	○
リデプロイ機能による入れ替え	○	－
リロード機能による入れ替え	－	○

（凡例）　○：入れ替えできる　－：入れ替えできない

それぞれの入れ替え方法について説明します。なお，入れ替え方法の詳細については，「5.6.3　J2EE アプリケーションの入れ替えと保守」を参照してください。

また，J2EE アプリケーションを入れ替えるときに，既存の J2EE アプリケーションの名前を変更することによって，J2EE アプリケーションの世代も管理できます。

24 時間サービス提供が必要な J2EE アプリケーションを入れ替える場合には，Web アプリケーションのサービスの部分閉塞や CTM の使用によって，サービスを停止しないで J2EE アプリケーションを入れ替えることもできます。

(1)　サービスの部分閉塞による入れ替え

部分的なサービスの閉塞です。サービスを停止することなく，システムをメンテナンスしたいときに使用します。24 時間サービス提供が必要な J2EE アプリケーションを入れ替える場合などには，サービスの部分閉塞によって，サービスを停止することなく，システムをメンテナンスできるようになります。

サービス部分閉塞の実行方法は，J2EE アプリケーションの形態によって異なります。例えば，Web アプリケーションの場合には，入れ替え対象の J2EE アプリケーションが動作する J2EE サーバに対して負荷分散機からのリクエストの振り分けを停止し，J2EE アプリケーションを入れ替えます。入れ替えている間は，ほかの J2EE サーバでリクエストを処理することで，サービスを停止することなく J2EE アプリケーションを入れ替えられます。

J2EE アプリケーションが Web アプリケーションの場合に，サービス部分閉塞を使用して J2EE アプリケーションを入れ替える方法については，「5.6.2　Web アプリケーションのサービスの部分閉塞による入れ替え」を参照してください。

参考

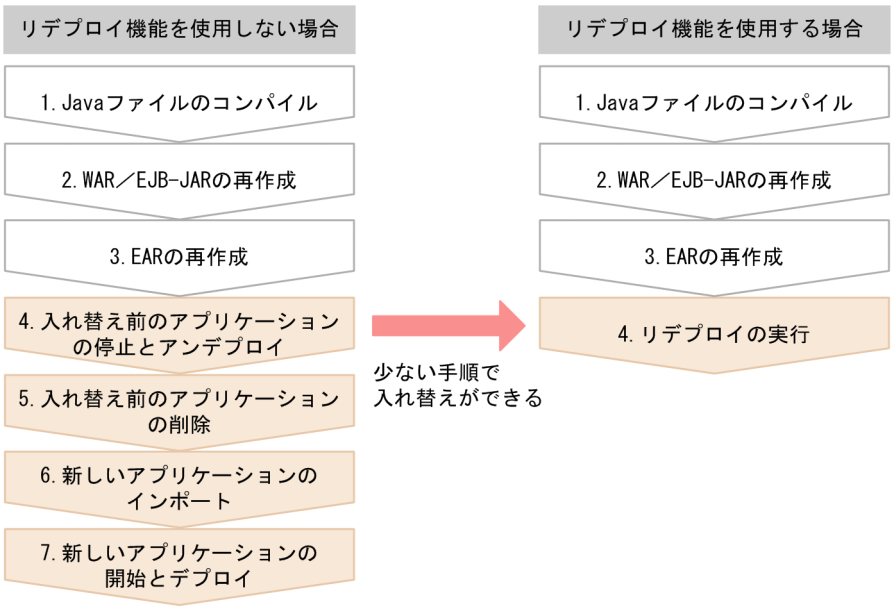
サービス部分閉塞は，CTM を利用して実行することもできます。CTM のスケジュールキューの閉塞を利用して J2EE アプリケーションを入れ替える方法については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「3.7　リクエストの閉塞制御」を参照してください。

(2)　リデプロイ機能による入れ替え

アプリケーション開発でのテストやシステムの運用中に，修正した J2EE アプリケーションと動作中の J2EE アプリケーションを入れ替えたい場合，リデプロイ機能を使用した入れ替えができます。**リデプロイ**とは，少ない手順で高速に J2EE アプリケーションを入れ替えられるデプロイ方法です。ロジックだけを変更した J2EE アプリケーションを入れ替えたい場合などに利用できます。ただし，リデプロイ機能を実行できる条件に合わない場合は，通常の手順で入れ替える必要があります。リデプロイ機能を実行できる条件については，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「18.7 J2EE アプリケーションのリデプロイ」を参照してください。

リデプロイ機能は，アーカイブ形式の J2EE アプリケーションを入れ替える場合に使用できます。リデプロイ機能による入れ替え手順を次に示します。

図 5-15　リデプロイ機能による入れ替え手順



リデプロイ機能を使用する場合には，通常のアーカイブ形式の J2EE アプリケーションの入れ替えと比べて，少ない手順で入れ替えができます。

参考

リデプロイを実行する前に，JSP 事前コンパイル機能を実行しておくことをお勧めします。JSP 事前コンパイル機能は，Web アプリケーションに含まれる JSP ファイルをデプロイ前にコンパイル

し，クラスファイルを生成する機能です。あらかじめ，クラスファイルの生成までを実施しておくので，JSP に最初にリクエストが到着したときのレスポンスタイムおよび Web アプリケーションの開始時間を短縮できます。JSP 事前コンパイル機能については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「2.5　JSP 事前コンパイル機能とコンパイル結果の保持」を参照してください。

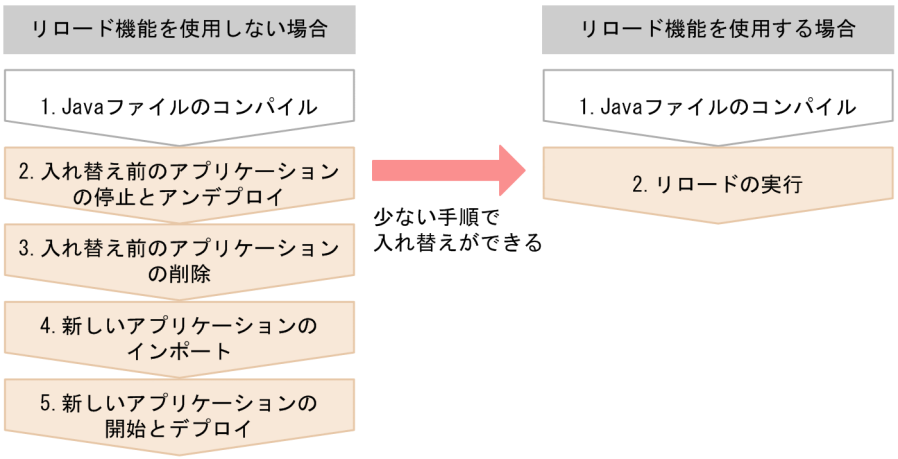
リデプロイ機能については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.7　J2EE アプリケーションのリデプロイ」を参照してください。

(3)　リロード機能による入れ替え

アプリケーション開発でのテストやシステムの運用中に，修正した J2EE アプリケーションと動作中の J2EE アプリケーションを入れ替えたい場合，リロード機能を使用した入れ替えができます。展開ディレクトリ形式の J2EE アプリケーションを構成するファイルを更新した場合に，更新検知やコマンド実行によって，更新した J2EE アプリケーションをリロードできます。リロード機能を使用することで，少ない手順で J2EE アプリケーションを動的に入れ替えられるようになります。

リロード機能による入れ替えは，展開ディレクトリ形式の J2EE アプリケーションを入れ替える場合に使用できます。リロード機能による入れ替え手順を次に示します。

図 5-16　リロード機能による入れ替え手順



リロード機能を使用する場合には，通常の展開ディレクトリ形式の J2EE アプリケーションの入れ替えと比べて，少ない手順で入れ替えができます。

リロード機能については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.8　J2EE アプリケーションの更新検知とリロード」を参照してください。

J2EE アプリケーションの入れ替えと保守については，「5.6.3　J2EE アプリケーションの入れ替えと保守」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。

5.6.2　Web アプリケーションのサービスの部分閉塞による入れ替え

Web アプリケーションのサービス閉塞では，部分的にサービスを閉塞することもできます。Web アプリケーションのサービスの部分閉塞によって，サービスを停止することなく，J2EE アプリケーションの入れ替えができ，システムをメンテナンスできます。

(1)　サービスの部分閉塞ができるシステム構成

サービスの部分閉塞ができるシステム構成を次に示します。

- 負荷分散機を使用して，複数の Web サーバにリクエストを振り分けていること
- Web サーバと J2EE サーバは，1:1 または 1:n の構成になっていること

なお，セッションを使用するアプリケーションの場合，同一セッションのリクエストは同一の J2EE サーバで処理する必要がありますが，Web アプリケーションのサービスの部分閉塞をすると，リクエストは，閉塞された Web アプリケーションがある J2EE サーバとは別の J2EE サーバで処理されます。セッション情報を引き継ぐためには，セッションフェイルオーバ機能を使用してください。セッションフェイルオーバ機能については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「5.　J2EE サーバ間のセッション情報の引き継ぎ」を参照してください。

(2)　サービスの部分閉塞の方法

Web アプリケーションのサービスの部分閉塞をするには，負荷分散機のリクエスト振り分け先を変更して，閉塞したい Web アプリケーションにリクエストが転送されないようにします。

このとき，リクエストにエラーが返されないようにするため，J2EE サーバで実行されるリクエストがないことを確認します。使用している Web サーバによって，次のことを実施してください。

- Web サーバに HTTP Server を使用している場合

  HTTP Server を停止してください。HTTP Server を停止すると，J2EE サーバにリクエストは転送されません。
- Web サーバに Microsoft IIS を使用している場合

  該当する J2EE サーバで実行中のリクエストがないことを確認してください。J2EE サーバで実行中のリクエスト数を確認するには，Management Server の mngsvrutil コマンドによって稼働情報を取得してください。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「mngsvrutil (Management Server の運用管理コマンド)」を参照してください。

なお，負荷分散機には，リクエストの振り分け先を変更したときに，変更前にあった負荷分散機と Web サーバの接続を，リクエスト処理が完了するまで保持する機能が必要です。この機能のない負荷分散機を使用すると，実行中のリクエストにエラーを返すので注意してください。

5.6.3　J2EE アプリケーションの入れ替えと保守

ここでは，J2EE アプリケーションの入れ替えと保守について説明します。

J2EE アプリケーションのバージョンアップやメンテナンスを実施する場合，J2EE アプリケーションを入れ替えることがあります。

また，24 時間サービス提供が必要な J2EE アプリケーションを入れ替える場合，CTM を使用したオンライン状態での J2EE アプリケーションの入れ替えの方法を使用すると，サービスを停止しないで J2EE アプリケーションを入れ替えられます。

(1)　J2EE アプリケーションの入れ替え

ここでは，サーバ管理コマンドを使用して J2EE アプリケーションを入れ替える手順を説明します。J2EE アプリケーションは，停止して，削除してから新しいアプリケーションに入れ替えます。入れ替え後，J2EE アプリケーションを再開始します。

なお，ここでは，cosminexus.xml を含むアプリケーションを入れ替える手順について説明します。J2EE アプリケーションに必要な情報はすべて cosminexus.xml で定義されているものとします。

入れ替えは，次の手順で実行します。

1. 入れ替える J2EE アプリケーションを停止します。

   cjstopapp コマンドを実行します。実行形式と実行例を次に示します。

   実行形式

   ```
   cjstopapp 〈J2EEサーバ名〉 -name 〈J2EEアプリケーション名〉
   ```

   実行例

   ```
   cjstopapp MyServer -name App1
   ```

2. J2EE アプリケーションを削除します。

   cjdeleteapp コマンドを実行します。実行形式と実行例を次に示します。

   実行形式

   ```
   cjdeleteapp 〈J2EEサーバ名〉 -name 〈J2EEアプリケーション名〉
   ```

   実行例

   ```
   cjdeleteapp MyServer -name App1
   ```

3. 入れ替え後の J2EE アプリケーションをインポートします。

   サーバ管理コマンドを使用する場合，cjimportapp コマンドを実行します。実行形式と実行例を次に示します。

   実行形式

   ```
   cjimportapp 〈J2EEサーバ名〉 -f 〈EARファイルのパス〉
   ```

実行例

```
cjimportapp MyServer -f App1.ear
```

なお，WAR アプリケーションの場合，cjimportwar コマンドを実行します。

4. J2EE アプリケーションを開始します。

サーバ管理コマンドを使用する場合，cjstartapp コマンドを実行します。実行形式と実行例を次に示します。

実行形式

```
cjstartapp <J2EEサーバ名> -name <J2EEアプリケーション名>
```

実行例

```
cjstartapp MyServer -name App1
```

参考

必要な情報を定義した cosminexus.xml を含むアプリケーションの場合，アプリケーションをインポートしたあとの属性ファイルの取得および変更は必要ありません。サーバ管理コマンドによる属性ファイルの取得および変更の手順については，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　属性ファイルによるプロパティの設定」を参照してください。

(2)　リデプロイによる J2EE アプリケーションの入れ替え

ここでは，リデプロイによる J2EE アプリケーションの入れ替えについて説明します。

リデプロイとは，アーカイブ形式の J2EE アプリケーションを入れ替える場合に，少ない手順で高速に入れ替えられるデプロイ方法です。ロジックだけを変更した J2EE アプリケーションを入れ替えたい場合などに利用できます。リデプロイは，サーバ管理コマンドを使用して実行できます。

リデプロイを実行できる条件を次に示します。

リデプロイを実行できる条件

- 入れ替えられるのは，実行時情報を含まない J2EE アプリケーションだけです。実行時情報を含む J2EE アプリケーション（ZIP ファイル）はリデプロイできません。
- 入れ替え前と入れ替え後の J2EE アプリケーションの構成が同じである必要があります。J2EE アプリケーションに含まれる EJB-JAR，リソースアダプタおよび WAR の数が異なったり，それらのファイル名称が異なったりする場合は，リデプロイはできません。また，J2EE アプリケーションの名称も同じである必要があります。
- 入れ替え前と入れ替え後の J2EE アプリケーションに含まれる EJB-JAR 内のホームインタフェース（ローカル，リモート），コンポーネントインタフェース（ローカル，リモート），ビジネスインタフェース（ローカル，リモート）のメソッド定義，およびアノテーションの値が同じである必要があります。

- ランタイム属性だけを引き継ぐ設定をしている場合に，アプリケーション開発環境で設定済みの DD ファイル（application.xml，ejb-jar.xml，ra.xml および web.xml）の定義内容が同じである必要があります。

また，J2EE アプリケーションを入れ替える時に，入れ替え前の J2EE アプリケーションを別な名称に変更して退避しておくと，名称による J2EE アプリケーションの世代管理が実現できます。ここでは，J2EE アプリケーションの名称変更についてもあわせて説明します。

リデプロイでは，入れ替え前の J2EE アプリケーションの情報を，入れ替え後の J2EE アプリケーションに引き継げます。デフォルトの設定の場合，入れ替え後の J2EE アプリケーションには，入れ替え前の J2EE アプリケーションのすべての属性が引き継がれます。ランタイム属性※だけを引き継ぎたい場合は，オプションを指定して cjreplaceapp コマンドを実行する必要があります。コマンドの詳細は，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjreplaceapp（アプリケーションの入れ替え）」を参照してください。

注※

属性ファイルには，DD(application.xml，ejb-jar.xml，ra.xml，web.xml)の定義と属性ファイル独自の定義が設定できます。属性ファイル独自の定義のことを，ランタイム属性といいます。

リデプロイを実行するとき，J2EE アプリケーションは開始，停止どちらの状態でもかまいません。開始状態の J2EE アプリケーションを入れ替えた場合，J2EE アプリケーションは入れ替え後に自動的に開始されます。ただし，プールやキャッシュに格納されていた J2EE アプリケーション関連のオブジェクトは破棄されます。停止状態の J2EE アプリケーションを入れ替えた場合は，入れ替え後の J2EE アプリケーションも停止した状態になります。

ポイント

J2EE アプリケーションが開始されている状態でリデプロイを実行した場合，リデプロイ処理の中で J2EE アプリケーションは停止され，入れ替え後に再開始されます。このとき，停止処理の実行時間がサーバ管理コマンド（cjreplaceapp）で設定したタイムアウト時間を超過した場合，J2EE アプリケーションの強制停止が実行されます。タイムアウト時間を指定しなかった場合は，デフォルトのタイムアウト時間である 60 秒を超過すると，強制停止が実行されます。強制停止実行後にさらにタイムアウト時間を超過した場合は，コマンドが異常終了します。

また，入れ替え後に J2EE アプリケーションを再開始するとき，開始処理の実行時間がサーバ管理コマンド用の usrconf.properties の ejbserver.rmi.request.timeout キーに指定したタイムアウト時間を超過した場合も，コマンドが異常終了します。

入れ替え作業の実行形式と実行例を次に示します。

実行形式

```
cjreplaceapp 〈J2EEサーバ名〉 -name 〈J2EEアプリケーション名〉 -f 〈入れ替えるアプリケーションのファイルパス〉
```

実行例

```
cjreplaceapp MyServer -name App1 -f App1.ear
```

注意事項

- cosminexus.xml を含むアプリケーションをリデプロイする場合，cosminexus.xml に定義したアプリケーションサーバ独自の情報は入れ替え後の cosminexus.xml に定義した情報に上書きされます。そのほかの J2EE アプリケーションを構成する要素（EJB-JAR ファイル，リソースアダプタなど）や DD などの定義情報は，入れ替え前の情報を引き継ぎます。
- WAR アプリケーションをリデプロイで入れ替える場合，cosminexus.xml ファイルの再読み込みはできません。WAR アプリケーションのアプリケーション属性を変更する場合は，サーバ管理コマンド（cjgetappprop コマンドおよび cjsetappprop コマンド）を使用してください。

(3)　リロードによる J2EE アプリケーションの入れ替え

ここでは，リロードによる J2EE アプリケーションの入れ替え方法について説明します。

なお，cosminexus.xml を含むアプリケーションをリロードした場合，cosminexus.xml に定義されたアプリケーションサーバ独自の情報はリロードされません。cosminexus.xml を含むアプリケーションの形式が展開ディレクトリ形式の場合の入れ替えについては，「5.6.3(1)　J2EE アプリケーションの入れ替え」の手順を参照してください。

リロードとは，少ない手順で展開ディレクトリ形式の J2EE アプリケーションの入れ替えを実行できる機能です。リロードによる J2EE アプリケーションの入れ替えでは，既存の J2EE アプリケーションの停止，削除，入れ替え後の J2EE アプリケーションのアーカイブ，インポート，再開などの作業が不要です。クラスファイルを更新してリロードを実行するだけで J2EE アプリケーションを更新できるため，メンテナンスが頻繁に発生するシステムの運用などで特に有効な機能です。

なお，リロードによる J2EE アプリケーションの入れ替えを実行するには，事前に設定が必要です。設定方法の詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.8.12　J2EE アプリケーションの更新検知とリロードの設定」を参照してください。

リロードによる J2EE アプリケーションの入れ替えは，サーバ管理コマンド（cjreloadapp コマンド）を使用して実行できます。cjreloadapp 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「cjreloadapp（アプリケーションのリロード)」を参照してください。

リロードは，次の手順で実行します。

1. メンテナンスの内容に従って Java ソースファイルを編集，または作成して，クラスファイルにコンパイルします。

2. J2EE アプリケーションのリロードを実行します。

   cjreloadapp コマンドを実行します。実行形式と実行例を次に示します。

実行形式

```
cjreloadapp <J2EEサーバ名> -name <J2EEアプリケーション名>
```

実行例

```
cjreloadapp MyServer -name App1
```

注意事項

リロードに失敗したアプリケーションを削除するには，リロードの成功後，アプリケーションを停止，削除するか，J2EE サーバを再起動したあとにアプリケーションを削除してください。

(4) 入れ替える J2EE アプリケーション内の JSP の事前コンパイル

J2EE アプリケーションのメンテナンスで JSP を編集した場合，通常，入れ替え後の J2EE アプリケーションに最初のリクエストがあったときに，JSP のコンパイルが実行されます。 **JSP 事前コンパイル機能**を使用すると，デプロイ前の J2EE アプリケーションに対して JSP のコンパイルを実行しておけるため，最初のリクエストに対するレスポンスの時間を短縮することができます。

JSP 事前コンパイル機能を実行するには，サーバ管理コマンド，または **cjjspc コマンド**を使用します。ここでは，JSP 事前コンパイル機能の実行タイミングと実行方法について説明します。

システム運用時に JSP 事前コンパイル機能を実行するタイミングを次に示します。

- J2EE アプリケーション開始時（サーバ管理コマンド）
- リロードによる J2EE アプリケーションの入れ替え時（cjjspc コマンド）
- リデプロイによる J2EE アプリケーションの入れ替え時（cjjspc コマンド）

それぞれのタイミングでの，JSP 事前コンパイル機能の実行方法について説明します。

(a) J2EE アプリケーションを開始するときの JSP の事前コンパイル

J2EE アプリケーションを開始するときに JSP 事前コンパイル機能を実行します。この場合，cjstartapp コマンドを，-jspc オプションを指定して実行します。

実行形式と実行例を次に示します。

実行形式

```
cjstartapp ＜サーバ名称＞ -name ＜J2EEアプリケーション名＞ -jspc
```

実行例

```
cjstartapp MyServer -name account -jspc
```

cjstartapp 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「cjstartapp（J2EE アプリケーションの開始)」を参照してください。

(b) リロードによって J2EE アプリケーションを入れ替えるときの JSP の事前コンパイル

リロードによって J2EE アプリケーションを入れ替える場合，cjreloadapp コマンドを実行する前に JSP 事前コンパイル機能を実行します。この場合，cjjspc コマンドを実行します。

実行形式と実行例を次に示します。

実行形式

```
cjjspc -root <Webアプリケーションのルートディレクトリ>
```

実行例（Windows の場合）

```
cjjspc -root d:¥app¥webapp1
```

実行例（UNIX の場合）

```
cjjspc -root /tmp/app/webapp1
```

cjjspc 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「cjjspc（JSP の事前コンパイル）」を参照してください。

(c) リデプロイによって J2EE アプリケーションを入れ替えるときの JSP の事前コンパイル

リデプロイによって J2EE アプリケーションを入れ替える場合，cjreplaceapp コマンドを実行する前に JSP 事前コンパイル機能を実行します。この場合，cjjspc コマンドを実行します。

実行形式と実行例を次に示します。

実行形式

```
cjjspc -root <Webアプリケーションのルートディレクトリ>
```

実行例（Windows の場合）

```
cjjspc -root d:¥app¥webapp1
```

実行例（UNIX の場合）

```
cjjspc -root /tmp/app/webapp1
```

cjjspc 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「cjjspc（JSP の事前コンパイル）」を参照してください。

注意事項

- リデプロイによって入れ替える J2EE アプリケーションに事前コンパイル機能を使用する場合，入れ替える J2EE アプリケーションに JSP コンパイル結果が含まれていないと JSP 事前コンパイル機能は実行されません。入れ替え後も JSP 事前コンパイル機能を使用する場合は，Web ア

プリケーション単位の JSP 事前コンパイルを実行して，入れ替える J2EE アプリケーションに JSP コンパイル結果を含めるようにしてください。

- タグファイル，静的インクルードされたファイル，または TLD ファイルを更新した場合，更新したファイルを参照するすべての JSP ファイルをコンパイルしてください。
- JSP 事前コンパイル機能を使用した展開ディレクトリ形式のアプリケーションに，JSP ファイル，またはタグファイルを追加した場合，JSP 事前コンパイルを再度実施して，JSP ファイル，またはタグファイルを参照するすべての JSP ファイルをコンパイルしてください。
- 展開ディレクトリ形式のアプリケーションの JSP ワークディレクトリに含まれるクラスファイルを，開発環境からコピーして更新する場合は，開発環境で JSP 事前コンパイルを実施したすべてのクラスファイルをコピーしてください。
- cjjspc コマンドを使用して JSP の事前コンパイルを実施した場合，JSP ファイルまたはタグファイルのトランスレーション時にエラーが発生すると，エラーメッセージがコンソールに出力されます。また，cjstartapp コマンドを使用して JSP の事前コンパイルを実施した場合，JSP ファイルまたはタグファイルのトランスレーション時にエラーが発生すると，エラーメッセージがサーブレットログに出力されます。

(5) J2EE アプリケーションの名称変更

J2EE アプリケーションを入れ替える時，既存の J2EE アプリケーションの名称を変更して退避しておくと，J2EE アプリケーションの世代管理ができます。また，入れ替え前の J2EE アプリケーションに戻したい場合などにもスムーズに対処できます。

J2EE アプリケーションの名称変更は，サーバ管理コマンドで実行します。すでに変更後の名称と同じ名称の J2EE アプリケーションがある場合，その名称には変更できません。大文字，小文字の違いによる区別はできません。

なお，名称を変更すると，ルックアップ名称の変更も必要になります。J2EE アプリケーションの構成要素に，リモート呼び出しをするホームインタフェースまたはコンポーネントインタフェースを使用した Enterprise Bean が含まれている場合，RMI-IIOP スタブおよびインタフェースを取得し直してください。

RMI-IIOP スタブおよびインタフェースの取得は，サーバ管理コマンド（cjgetstubsjar）を使用して実行します。J2EE アプリケーションを一度も実行していない状態では取得できません。また，次の場合，RMI-IIOP スタブおよびインタフェースの取得はエラーになります。

- J2EE アプリケーションに WAR しか含まれない場合
- Enterprise Bean で使用されているホームインタフェースおよびコンポーネントインタフェースがすべてローカル呼び出しの場合
- Enterprise Bean が Message-driven Bean だけの場合

名称変更は，次の手順で実行します。

1. 名称を変更する J2EE アプリケーションを停止します。

サーバ管理コマンド（cjstopapp）を実行します。

2. J2EE アプリケーションの名称を変更します。

サーバ管理コマンド（cjrenameapp）を実行します。

実行形式と実行例を次に示します。

実行形式

```
cjrenameapp <J2EEサーバ名> -name <変更前のJ2EEアプリケーションの名称> -newname <変更後のJ2EEアプリケーションの名称>
```

実行例

```
cjrenameapp MyServer -name App1 -newname App1bak
```

3. J2EE アプリケーションを開始します。

サーバ管理コマンド（cjstartapp）を実行します。

4. J2EE アプリケーションに含まれる Enterprise Bean 内のホームインタフェースまたはコンポーネントインタフェースがリモート呼び出しを実行するように定義されている場合，RMI-IIOP スタブおよびインタフェースを取得します。

サーバ管理コマンド（cjgetstubsjar）を実行します。

実行形式と実行例を次に示します。

実行形式

```
cjgetstubsjar <J2EEサーバ名> -name <変更後のJ2EEアプリケーションの名称> -d <RMI-IIOPスタブおよびインタフェースを格納するディレクトリのパス>
```

実行例

```
cjgetstubsjar MyServer -name App1bak -d temp
```

5.7　J2EE アプリケーションからのネットワークリソースへのアクセス

Windows の場合で，ネットワークリソースのパスを UNC で指定したり，ネットワークドライブで指定したりした J2EE アプリケーションから他ホストにアクセスするには設定が必要になります。この節では，J2EE アプリケーションからのネットワークリソースへのアクセスについて説明します。なお，UNIX の場合，この節で説明する機能を使用しなくても，ネットワークリソースにアクセスできます。

この節の構成を次の表に示します。

表 5-27　この節の構成（J2EE アプリケーションからのネットワークリソースへのアクセス）

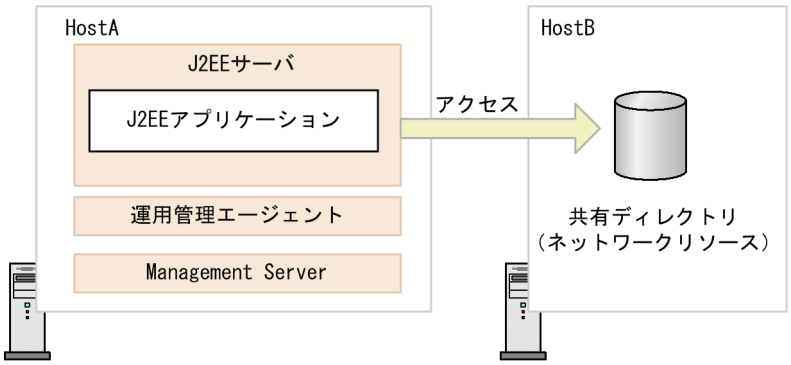
分類	タイトル	参照先
解説	ネットワークリソースへのアクセスの概要	5.7.1
設定	ネットワークリソースにアクセスするための設定	5.7.2
注意事項	運用時のトラブルについて	5.7.3

注　「実装」,「運用」について，この機能固有の説明はありません。

5.7.1　ネットワークリソースへのアクセスの概要

ネットワークリソースへのアクセス機能とは，Windows で動作する J2EE サーバ上の J2EE アプリケーションから，他ホストのリソースにアクセスするための機能です。この機能を使用することで，J2EE アプリケーションから他ホストへのリソースのパスを UNC またはネットワークドライブを指定してアクセスできるようになります。ネットワークリソースへのアクセスの概要を次の図に示します。

図 5-17　ネットワークリソースへのアクセスの概要



この図では，HostA の X ドライブに HostB の共有ディレクトリが割り当てられているとします。ネットワークリソースへのアクセス機能を使用することで，HostA の J2EE アプリケーションから，X ドライブに割り当てられた HostB の共有ディレクトリにアクセスできます。なお，この機能を使用するためには，J2EE アプリケーションが動作する HostA 上の運用管理エージェントを Administrators グループに所属するアカウントで起動する必要があります。

5.7.2　ネットワークリソースにアクセスするための設定

ネットワークリソースにアクセスするためには，次に示す手順で設定します。なお，ここでは，Windows Server 2019 の場合を例に説明します。

1. 「コントロールパネル」から［管理ツール］－［サービス］を開く。

2. 運用管理エージェントのログオンアカウントを Administrators グループに所属するアカウントに変更する。

   [Cosminexus Management Server - Administration Agent］のログオンアカウントをローカルホストの Administrators グループに所属するアカウントに変更します。

3. adminagentuser.cfg に，J2EE アプリケーションで使用するネットワークドライブを設定する。

   <Application Server のインストールディレクトリ>¥manager¥config¥adminagentuser.cfg に，次の記述を追加してください。この例は，X ドライブに¥¥host¥dir を割り当てる場合です。

   add.network.drive=X=¥¥host¥dir

   なお，この設定は，ネットワークドライブを使用してネットワークリソースにアクセスする場合だけ実施します。UNC でアクセスする場合は設定不要です。

ポイント

手順 3.のネットワークドライブの割り当ての設定について

この設定は，運用管理エージェントを起動する前に実施します。設定したネットワークドライブは，運用管理エージェント起動時に割り当てられます。ただし，運用管理エージェントのログオンアカウントがローカルシステムアカウントの場合は割り当てられません。手順 3.で設定する内容について詳細を説明します。

- 使用するファイル

  <Application Server のインストールディレクトリ>¥manager¥config¥adminagentuser.cfg

- 設定内容

  add.network.drive キーに，ネットワークドライブとして割り当てるドライブの名前と割り当て先のディレクトリパスを指定します。

  指定例：X=¥¥host¥dir

  なお，この設定を UNIX の環境で設定した場合は，<Manager のログ出力ディレクトリ>/adminagent.err に KEOS21401-E メッセージが出力され，終了コード 1 で運用管理エージェントが終了されます。

  また，ネットワークドライブの割り当ては複数，指定できます。

add.network.drive キーの設定例と動作について次の表に示します。

表 5-28　add.network.drive キーの設定例と動作内容

指定値	設定例	動作内容
キーなし	(add.network.drive の定義なし)	ネットワークドライブの割り当てをしない。
値なし	add.network.drive=	
一つのネットワークドライブを指定	add.network.drive=X=¥¥host¥dir	KEOS21304-I メッセージを出力して，X ドライブに¥¥host¥dir を割り当てる。
複数のネットワークドライブを指定	add.network.drive=X=¥¥host¥dir add.network.drive=Y=¥¥host¥dir2	KEOS21304-I メッセージを出力して，X ドライブに¥¥host¥dir を，Y ドライブに¥¥host¥dir2 を割り当てる※1。
複数のネットワークドライブを指定（ドライブ名が同じ場合）	add.network.drive=X=¥¥host¥dir add.network.drive=X=¥¥host¥dir2	KEOS21304-I メッセージを出力して，X ドライブに¥¥host¥dir を割り当てる※1。 KEOS21305-W メッセージを出力して，二つ目の¥¥host¥dir2 の割り当てには失敗する※2。

注　運用管理エージェントのログオンアカウントがローカルシステムアカウントの場合は，<Manager のログ出力ディレクトリ>¥adminagent.err に KEOS21307-W メッセージを出力して，ネットワークドライブを割り当てないで処理を続行します。

注※1　指定したネットワークドライブの割り当てに成功した場合は，<Manager のログ出力ディレクトリ>¥adminagent.err に KEOS21304-I メッセージを出力して，処理を続行します。

注※2　指定したネットワークドライブの割り当てに失敗した場合は，<Manager のログ出力ディレクトリ>¥adminagent.err に KEOS21305-W メッセージを出力して，処理を続行します。

4. 「コントロールパネル」から，［ユーザアカウント］－［ネットワークパスワードの管理］を開く。

ログオンアカウントがドメインユーザの場合は「パスワードの管理」を開いてください。ここで，ネットワークリソースへアクセスする際にアクセス認証処理を省略するための設定をします。この設定は運用管理エージェントを起動するログオンアカウントのデスクトップ上から実施してください。

5. ［ユーザ名およびパスワードの保存］ダイアログでアクセス先のホスト名，ユーザ名，パスワードを設定する。

ただし，設定するユーザは共有ディレクトリへのアクセスが許可されている必要があります。

ポイント

手順 5.のアクセス先ホスト名の指定について

手順 5.で設定するアクセス先のホスト名には，IP アドレスではなくホスト名を指定してください。手順 5.のネットワークドライブの設定でホスト名に<IP アドレス>を指定している場合で，手順 3.で割り当てるホストが<ホスト名>のときは，Windows の仕様によってアクセス先のホストにログインができないため，割り当てに失敗します。

ホスト名「host」(IP=10.10.10.10)を割り当てる場合を例に，手順 3.で指定するホスト名と手順 5.で指定するホスト名の組み合わせごとに，割り当てができるかどうかを次の表に示します。

表 5-29　ネットワークの割り当ての可否

手順 5.で指定するホスト名	手順 3.で指定するホスト名	割り当て
「host」	add.network.drive=X=¥¥host¥dir	○
	add.network.drive=X=¥¥10.10.10.10¥dir	○
「10.10.10.10」	add.network.drive=X=¥¥host¥dir	×
	add.network.drive=X=¥¥10.10.10.10¥dir	○

（凡例）○：割り当てできる　×：割り当てできない

5.7.3　運用時のトラブルについて

Administrators グループに所属していないアカウントで運用管理エージェントを起動すると，次のような現象が起こります。

- 発生するタイミング

  Windows のサービスから［Cosminexus Management Server - Administration Agent］を起動したとき，または adminagentctl start コマンドを実行したとき
- 現象

  運用管理エージェントの起動に失敗します。また，イベントログに，「KEOS21108-E The administration agent could not be started.(detail = The logon user is not using a local system account, or does not belong to the Administrators group)」が出力されます。

対処方法は，Windows のサービスから［Cosminexus Management Server - Administration Agent］にログオンするアカウントをローカルシステムアカウントにするか，または Administrators グループに所属するアカウントに変更してください。

5.8 J2EE アプリケーション運用時の注意事項

AIX で，J2EE アプリケーションのデプロイ中に，「exec error: パラメータ・リストまたは環境リストが長すぎます。」（日本語環境以外の場合：「exec error: Arg list too long」）というエラーが発生した場合，java2iiop コマンドの引数の長さが，OS のカーネルパラメタである ARG/ENV リストの設定値を超えています。デフォルトでは，デプロイ対象のアプリケーションが持つ，EnterpriseBean のホームインタフェースおよびコンポーネントインタフェースの数が，およそ 570 個以上のとき（パッケージ名を含めたインタフェース名の平均を 40 文字と仮定）に発生します。このエラーが発生した場合は，次の対処を実施してください。

1. 次のコマンドを実行して，ARG/ENV リストの値を確認します。

```
lsattr -E -l sys0 -a ncargs
```

なお，デフォルト値は，6（単位：4 キロバイトブロック）です。

2. 次のコマンドを実行して，ARG/ENV リストの値を算出した値以上に変更します。

```
chdev -l sys0 -a ncargs=<変更後のARG/ENVリストの値>
```

<変更後の ARG/ENV リストの値>は，6～128 の範囲で指定します。次の式で算出した値以上を指定してください。

変更後の ARG/ENV リストの値≧（(A ＋（B×C)）※＋ 4,095）/4,096

- A：約 1,600 バイト（vbj コマンドに渡すパラメタを除いたコマンドのバイト数）
- B：パッケージ名を含めたクラス名の長さの平均（単位：バイト）
- C：ホームインタフェースおよびコンポーネントインタフェースの数

注※ 「A ＋（B×C)」では，実行するコマンドの長さがバイト単位で算出できます。

3. J2EE アプリケーションを再デプロイします。

6 監査ログ出力機能

この章では，システムの監査を行うための監査ログ出力機能について説明します。

6.1　この章の構成

この章の構成を次の表に示します。

表 6-1　この章の構成（監査ログ出力機能）

分類	機能名	参照先
解説	監査ログ出力機能の概要	6.2
	監査ログとは	6.3
	監査ログの出力	6.4
	監査ログの保存	6.5
	監査ログを出力するコマンド・操作一覧	6.6
	監査ログの出力ポイント	6.7
実装	アプリケーションの監査ログを出力するための実装	6.8
設定	監査ログ出力の設定	6.9

注　「運用」,「注意事項」について，この機能固有の説明はありません。

6.2　監査ログ出力機能の概要

近年，組織の健全な運営の保障や，複雑化・多様化する IT システムの安全な構築・運用の観点から，組織の内部統制の重要性が高まっています。内部統制の目的は，いつ，だれが，どんな業務を実行したかを把握することで，業務が各種法規制に準拠して遂行されているかを検証することです。そのために，インフラの面では，システムに対して正当な権限を持つ者が正当な操作を行ったことを監査するための仕組みが必要です。

アプリケーションサーバでは，システムの監査を支援する機能として，システムやアプリケーションに対する操作履歴など，監査の観点で必要な情報を記録する次の機能を提供しています。

- **監査ログ出力機能**
- **データベース監査証跡連携機能**

なお，システムの監査を支援する機能は，J2EE アプリケーションの実行環境，およびバッチアプリケーションの実行環境で使用できます。

ここでは，監査ログ出力機能を使用してシステムを監査する作業について説明します。データベース監査証跡連携機能については，「7.　データベース監査証跡連携機能」を参照してください。

監査ログ出力機能は，システムの構築者や運用者がアプリケーションサーバのプログラムに対して実行した操作，およびその操作に伴うプログラムの動作の履歴を監査ログとして出力します。また，アプリケーションサーバが提供している監査ログ出力用の API を使用して，システムで動作する J2EE アプリケーションやバッチアプリケーションの監査ログを出力する機能を実装できます。

6.3　監査ログとは

この節では，監査ログを使用したシステムの監査の概要について説明します。

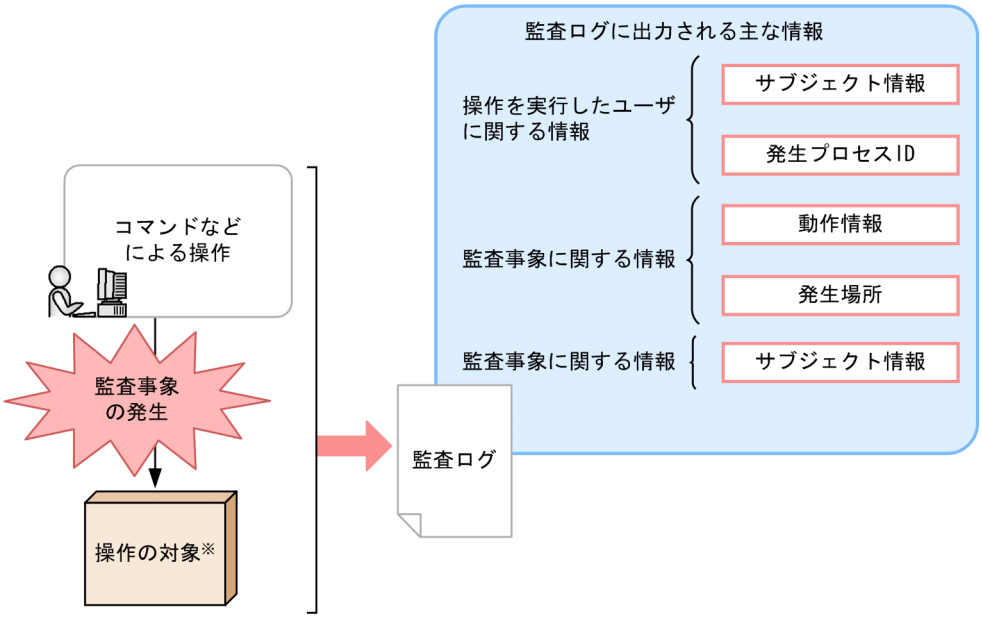
監査ログとは，システム構築者やシステム運用者がアプリケーションサーバのプログラムに対して実行した操作，およびその操作に伴うプログラムの動作の履歴が出力されるファイルです。監査者が監査ログを調査することで，「いつ」「だれが」「何をしたか」を知ることができて，システムの運用が法規制，セキュリティ評価基準，または業界ごとの各種の基準に準拠していることを証明できます。

監査ログには，コマンドなどによる操作を実行したユーザに関する情報や，その操作に伴う処理が成功したか失敗したかなどの監査事象に関する情報，操作や処理の対象に関する情報などが出力されます。これらの情報をシステムの監査に役立てることができます。監査ログに出力される情報の詳細については，「6.4 監査ログの出力」を参照してください。

6.3.1　監査ログが出力される流れと主な出力情報

監査ログが出力される流れと主な出力情報を次の図に示します。

図 6-1　監査ログが出力される流れと主な出力情報



注※
コマンドなどによる操作の対象を示します。
例えば，サーバを起動するためのコマンドを実行した場合，起動する対象となるサーバを示します。

6.3.2　監査事象の定義

監査ログが出力される契機になるのは，コマンドの実行など，アプリケーションサーバのプログラムに対する操作です。この操作は，システム管理者，システム運用者など，その作業に応じたユーザが実行しま

す。また，監査ログは，アプリケーションサーバの各プロセスで**監査事象**が発生したタイミングで出力されます。監査事象とは，アプリケーションサーバのプログラムに対して実行した操作，およびその操作に伴うプログラムの処理のうち，システムの構築や運用の正当性を証明するために記録する必要がある事象のことです。アプリケーションサーバでは，監査事象を次のとおり分類して定義しています。

表 6-2　監査事象の定義

監査事象	説明
StartStop	ソフトウェアの起動と終了を示す事象です。次の操作が該当します。 • 各プロセスの起動・停止 • J2EE アプリケーションの開始・停止 • J2EE リソースの開始・停止
Authentication	システム管理者やシステム運用者が実行した，Management Server の管理者 ID と管理者パスワードによる認証を伴う操作が，成功または失敗したことを示す事象です。
ConfigurationAccess	システム管理者が実行した操作が正常終了または失敗したことを示す事象です。次の操作が該当します。 • 設定変更 • 構成変更 • 構成情報の参照

アプリケーションサーバで定義している監査事象の例を次に示します。

- Smart Composer 機能のコマンドの実行，および Smart Composer 機能のコマンドの実行に伴う処理
- サーバ管理コマンドの実行，およびサーバ管理コマンドの実行に伴う処理
- Management Server を仲介した，管理者 ID やパスワードの作成・変更
- 運用管理エージェントを仲介した，定義ファイルの編集

監査事象は，コマンドごとに定義されています。監査事象となるコマンドの一覧については，「6.6.1　J2EE サーバで使用するコマンド一覧」を参照してください。コマンドごとの監査ログの出力ポイントについては，「6.7　監査ログの出力ポイント」を参照してください。

また，アプリケーションサーバでは，J2EE アプリケーションまたはバッチアプリケーションに使用する監査ログ出力用の API を提供します。監査ログ出力用の API を使用すると，監査事象のタイミング以外にも，アプリケーションへの操作の実行や，アプリケーションによる処理のタイミングで監査ログを出力することもできます。監査ログ出力用の API を使用した J2EE アプリケーションまたはバッチアプリケーションの実装については，「6.8　アプリケーションの監査ログを出力するための実装」を参照してください。

6.3.3　監査ログ出力機能を使用するシステムで必要な作業と参照先

監査ログを使用したシステムの監査で必要な作業と参照先を次の表に示します。

表 6-3　システムの監査で必要な作業と参照先

作業	参照先マニュアル	参照個所	記載内容
監査を実施するセキュアなシステムの構成を検討する	機能解説 セキュリティ管理機能編	4 章	• 監査ログを使用してセキュアなシステムを構成するための考え方
監査ログを取得するための設定をする	このマニュアル	6.9	• 監査ログ取得の設定方法
監査ログを収集する		6.4.1	• 監査ログの出力方式 • 監査ログの収集方法
監査ログを調査する		6.3	• 監査ログの調査方法
		6.6	• 監査ログが出力されるコマンド一覧
		6.7	• 監査ログの出力ポイント
	アプリケーションサーバ メッセージ(監査者用)	全体	• 監査ログに出力されるメッセージの見方
アプリケーションに監査ログ出力用の API を実装する	このマニュアル	6.8	• 監査ログ出力用の API の実装例 • 監査ログ出力用の API を実装するときの注意事項
	リファレンス API 編	8 章	• 監査ログ出力用の API の詳細

6.4　監査ログの出力

この節では，システムが出力する監査ログの出力先，出力形式，および出力項目の詳細について説明します。

6.4.1　監査ログの出力方式

監査ログは，監査ログを出力したサーバのローカルファイルに，ラップアラウンド方式で出力されます。監査ログが出力されるファイルの面数が切り替わるタイミングで，面数が切り替わったことを通知するメッセージが出力されます。また，障害が発生して，監査ログの出力に失敗すると，監査ログの出力でエラーが発生したことを通知するメッセージが出力されます。

6.4.2　監査ログの出力先

デフォルトの場合の監査ログの出力先を次に示します。

- Windows の場合

  <Application Server のインストールディレクトリ>¥auditlog¥audit[n]※.log
- UNIX の場合

  /opt/Cosminexus/auditlog/audit[n]※.log

注※　[n]の部分には，面の番号が付きます。

デフォルトでは，監査ログの最大ファイルサイズは「32MB」，ファイル面数は「4」に設定されています。最大ファイルサイズ，ファイル面数，および出力先は，監査ログの定義ファイル（auditlog.properties）で変更できます。監査ログの設定については，「6.9　監査ログ出力の設定」を参照してください。

6.4.3　監査ログの出力形式

監査ログは，メッセージとして出力されます。監査ログのメッセージは，次の形式で出力されます。

```
CALFHM 1.0,出力項目1=値1,出力項目2=値2,出力項目3=値3,・・・出力項目n=値n
```

先頭の「CALFHM 1.0」は，ヘッダ情報です。監査ログに，共通で出力されます。

システムの監査で使用するメッセージの出力例を次に示します。

出力例

```
CALFHM 1.0,seqnum=1,msgid=KDJE54400-I,date= 2007-01-22T16:09:59.884+09:00,
progid=Cosminexus,compid=CCC,pid=00EB7859,ocp:host=host01,
ctgry=ConfigurationAccess,result=Occurrence,subj:euid=account01,
```

```
obj="Server01",op="Add",to:host=host02,to:port=28080,
msg="account01 executes the request (cjimportapp Server01 -f App1.ear)."
```

注　実際に出力されるメッセージは，1 行で表示されます。この例のように改行されません。

このメッセージは，「2007 年 1 月 22 日 16 時 9 分 59.884 秒」に，「ホスト 01」というホストで，cjimportapp コマンドが実行されたことを示しています。また，コマンドを実行したアカウントは「account01」，「Server01」というサーバに対してコマンド実行されたことを示しています。メッセージ内容の詳細については，「6.4.4　監査ログの出力項目」を参照してください。

6.4.4　監査ログの出力項目

監査で使用するメッセージの出力項目には，すべてのメッセージで共通の項目，およびメッセージごとに固有の項目があります。それぞれの項目の意味を次に示します。

すべてのメッセージで共通の項目

すべてのメッセージで同じ意味の可変値が出力されたり，共通の文字列が出力されたりする項目が該当します。

メッセージごとに固有の項目

メッセージごとに固有の意味を持つ可変値が出力されたり，メッセージごとに出力される文字列が決まっていたりする項目が該当します。

監査で使用するメッセージの出力項目の詳細について，次の表に示します。

表 6-4　監査で使用するメッセージの出力項目

出力項目名	出力項目の意味	詳細	共通／固有
seqnum	通番	監査ログの通番が出力されます。 出力される値は，1 から 9999999999 までの整数です。	共通
msgid	メッセージ ID	メッセージ ID が出力されます。	固有
date	日付・時刻	メッセージが出力された日時が，次の形式で出力されます。 *YYYY-MM-DD*T*hh:mm:ss.sss*TZD • *YYYY*：西暦年 • *MM*：月 • *DD*：日 • T：日付と時間の区切り • *hh*：時 • *mm*：分 • *ss*：秒 • *sss*：マイクロ秒 • *TZD*：タイムゾーン※1	共通

出力項目名	出力項目の意味	詳細	共通／固有
progid	発生プログラム名	監査事象が発生したプログラムの名称が出力されます。アプリケーションサーバの場合，「Cosminexus」という文字列が出力されます。	共通
compid	発生構成ソフトウェア名	監査事象が発生した構成ソフトウェアの略称が，次の形式で出力されます。 • CCC：Component Container • CTM：Component Transaction Monitor • HWS：HTTP Server • MNG：Management Server • PRF：Performance Tracer • UAP_<任意の名称>※2：ユーザが J2EE アプリケーション，またはバッチアプリケーション実装時に指定した任意の名称	固有
pid	発生プロセス ID	監査事象が発生したプロセスのプロセス ID が出力されます。	共通
ocp:host	発生場所	監査事象が発生したホストの名称が出力されます。ただし，発生場所の情報が取得できなかった場合，「(null)」が出力されます。	共通
ctgry	監査事象の種別	監査事象の種別が，次のとおり分類されて出力されます。 StartStop サーバ，プロセス，サービスなどの起動・終了を示す事象です。 Authentication ログイン，ログアウトなど，Management Server への管理者 ID と管理者パスワードによる認証が実行されたことを示す事象です。 ConfigurationAccess 設定および構成の変更，構成情報の参照を示す事象です。 AccessControl※2 管理者またはエンドユーザが管理リソース，およびセキュリティリソースへのアクセスを試みて，成功または失敗したことを示す事象です。 Failure ソフトウェアの異常を示す事象です。 LinkStatus※2 機器間のリング状態を示す事象です。 ExternalService※2 外部サービスとの通信結果を示す事象です。	固有

出力項目名	出力項目の意味	詳細	共通／固有
		ContentAccess※2 ユーザの重要なデータへのアクセスを試みて，成功または失敗したことを示す事象です。 Maintenance※2 保守作業を実行して，操作が正常終了または失敗したことを示す事象です。 AnomalyEvent※2 しきい値のオーバなどの異常が発生したこと，または異常な通信の発生を示す事象です。 ManagementAction※2 プログラムの重要なアクションが実行されたことを示す事象，またはほかの監査事象を契機として実行されるアクションを示す事象です。	
result	監査事象の結果	監査事象の結果（成功・失敗・発生）が，次の形式で出力されます。 Success 監査事象の成功を示します。 Failure 監査事象の失敗を示します。 Occurrence 成功および失敗の区別がない事象の発生を示します。	固有
subj:uid	サブジェクト識別情報（アカウント識別子の場合）	監査事象を発生させたものがアカウント情報に割り付けられている場合，アカウント識別子（ユーザ ID など）が出力されます。	固有
subj:euid	サブジェクト識別情報（OS のアカウントの場合）	監査事象を発生させたものが OS が提供するアカウント情報に割り付けられている場合，OS のアカウント※3 が出力されます。ただし，サブジェクト識別情報が取得できなかった場合，「(null)」が出力されます。	固有
obj	オブジェクト情報	監査事象となった操作の対象の情報が，「"」で囲まれた形式で出力されます。	固有
op	動作情報	監査事象となった操作の種別が，「"」で囲まれた次の形式で出力されます。 Start 起動を示します。 Stop 停止を示します。 Login ログインを示します。	固有

出力項目名	出力項目の意味	詳細	共通／固有
		Logout ログアウトを示します。 Logon ログオンを示します。 Logoff ログオフを示します。 Refer 設定情報の参照を示します。 Add 設定情報の追加を示します。 Update 設定情報の更新を示します。 Delete 設定情報の削除を示します。 Occur 障害などの発生を示します。 Enforce※2 処理の実施を示します。 Up※2 リンクの活性を示します。 Down※2 リンクの非活性を示します。 Request※2 要求を示します。 Response※2 応答を示します。 Send※2 発信を示します。 Receive※2 受信を示します。 Install※2 インストールを示します。 Uninstall※2 アンインストールを示します。 Backup※2 バックアップを示します。 Maintain※2 保守作業を示します。	

出力項目名	出力項目の意味	詳細	共通／固有
		Invoke※2 管理者などの呼び出しを示します。 Notify※2 管理者などへの通知を示します。	
objloc	オブジェクトロケーション情報	オブジェクトロケーション情報が出力されます。	固有
to:host	リクエスト送信先ホスト	監査事象が複数のプログラム間で連携して動作するリクエストに関連する場合に，リクエストの送信先のホストの情報が出力されます。	固有
to:port	リクエスト送信先ポート番号	監査事象が複数のプログラム間で連携して動作するリクエストに関連する場合に，リクエストの送信先のポート番号が出力されます。	固有
msg	自由記述	監査事象の内容を示す文章が出力されます。	固有
before※2	変更前情報	変更前の情報が出力されます。	固有
after※2	変更後情報	変更後の情報が出力されます。	固有
auth※2	権限情報	監査事象を発生させたときのサブジェクトの権限が出力されます。	固有
sins※2	サービスインスタンス名	サービスの利用者の識別子が出力されます。	固有
haid※2	冗長化識別情報	監査事象の発生場所が冗長化構成の場合，発生場所の系を表す情報が出力されます。	固有
from:host※2	リクエスト送信元ホスト	監査事象が複数のプログラム間で連携して動作するリクエストに関連する場合に，リクエストの送信元の場所情報が出力されます。	固有
from:port※2	リクエスト送信元ポート番号	監査事象が複数のプログラム間で連携して動作するリクエストに関連する場合に，リクエストの送信元のポート番号が出力されます。	固有
outp:host※2（ホスト名で出力する場合）	出力元の場所	出力元が動作している場所情報が出力されます。	固有
subjp:host※2（ホスト名で出力する場合）	指示元の場所	サブジェクトが指示を出した場所情報が出力されます。	固有
dtp:host※2（ホスト名で出力する場合）	検出場所	検出エンティティが動作している場所情報が出力されます。	固有
loc※2	ロケーション情報	ユーザが設定したロケーション識別情報が出力されます。ユーザが設定しなかった場合は，ルートアプリケーション情報が，「"」で囲まれた形式で出力されます。ただし，ルートアプリケーション情報が取得できなかった場合，「null」が「"」で囲まれた形式で出力されます。	固有

（凡例）

共通：すべてのメッセージで共通の項目であることを示します。

固有：メッセージごとに固有の項目であることを示します。メッセージごとに固有の項目の詳細については，マニュアル「アプリケーションサーバ メッセージ(監査者用)」のそれぞれのメッセージの説明を参照してください。

注※1　タイムゾーンは，UTC からの時差で表示されます。表示形式を次に示します。

- *+hh:mm*：UTC から *hh* 時間 *mm* 分進んでいる
- *-hh:mm*：UTC から *hh* 時間 *mm* 分遅れている
- *Z*：UTC と同じ

（例）日本の場合：+09:00

注※2　監査ログ出力用の API を使用して，J2EE アプリケーション，またはバッチアプリケーションから出力するように実装した場合にだけ出力される情報です。

注※3　OS のアカウントには，Windows の場合はユーザ名が，UNIX の場合は実行ユーザ ID が出力されます。

6.5　監査ログの保存

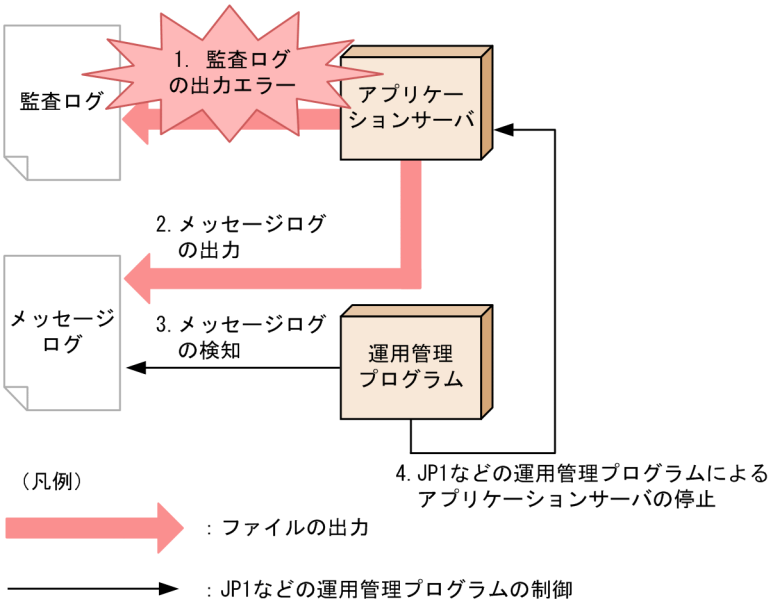
監査ログを使用して監査を実施するシステムでは，監査ログをすべて保存しておくことが求められるため，監査ログの出力に失敗した場合は，システムを停止する必要があります。また，ラップアラウンドによって削除される監査ログを，アーカイブする必要があります。これらの処理は，JP1 などの運用管理プログラムを使用して，自動実行することをお勧めします。JP1 などの運用管理プログラムを使用して，監査ログの出力に失敗した場合にシステムを自動で停止する例と，監査ログを自動でアーカイブする例について説明します。

6.5.1　監査ログの出力に失敗した場合にシステムを自動で停止する流れ

監査ログの出力に失敗すると，監査ログの出力に失敗したことを通知するメッセージが，メッセージログに出力されます。JP1 などの運用管理プログラムを使用すると，監査ログの出力に失敗したことを通知するメッセージログを監視して，システムを自動で停止することができます。

JP1 などの運用管理プログラムを使用して，監査ログの出力に失敗した場合にシステムを自動で停止する流れを次に示します。

図 6-2　JP1 などの運用管理プログラムを使用して監査ログの出力に失敗した場合にシステムを自動で停止する流れ



1. 障害によってアプリケーションサーバが監査ログの出力に失敗します。
2. アプリケーションサーバが，監査ログの出力に失敗したことを通知するメッセージログを出力します。
3. JP1 などの運用管理プログラムが，メッセージログが出力されたことを検知します。
4. JP1 などの運用管理プログラムがアプリケーションサーバを停止します。

6.5.2　監査ログを自動でアーカイブする流れ

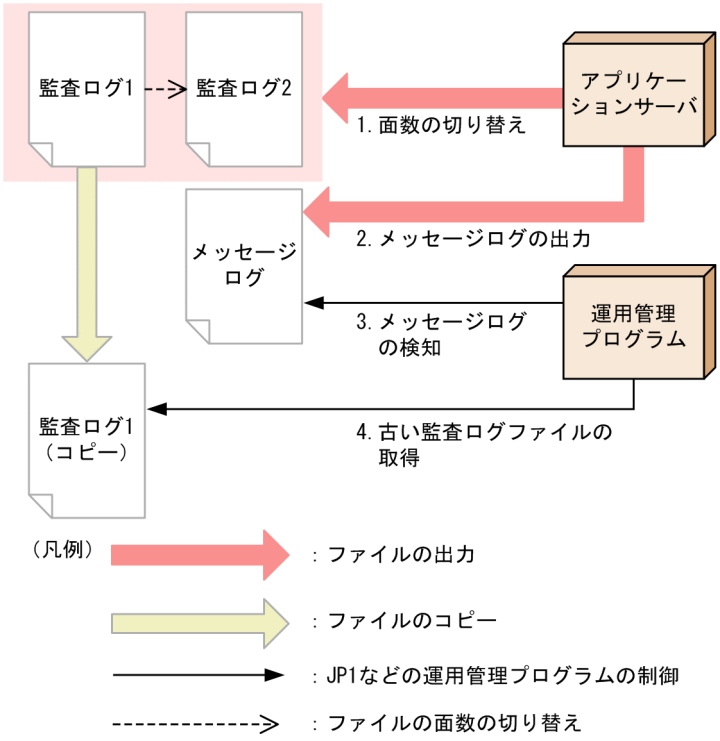
監査ログの面数が切り替わると，面数が切り替わったことを通知するメッセージが，メッセージログに出力されます。JP1 などの運用管理プログラムを使用すると，面数が切り替わったことを通知するメッセージログを監視して，自動でアーカイブすることができます。アーカイブする必要があるファイルを次に示します。

表 6-5　アーカイブする必要があるファイル

アーカイブに含める必要があるファイル	パス
監査ログ	<監査ログの出力ディレクトリ>audit[n].log
監査ログのメッセージログ	<監査ログの出力ディレクトリ>¥rasmessage[n].log

JP1 などの運用管理プログラムを使用して，監査ログを自動でアーカイブする流れを次に示します。

図 6-3　JP1 などの運用管理プログラムを使用して監査ログを自動でアーカイブする流れ



1. 監査ログを出力するファイルの面数が切り替わります。
2. ファイルの面数が切り替わったことを通知するメッセージログを，アプリケーションサーバが出力します。
3. JP1 などの運用管理プログラムが，メッセージログが出力されたことを検知します。
4. JP1 などの運用管理プログラムが，古い監査ログファイルを取得します。

6.6　監査ログを出力するコマンド・操作一覧

この節では，監査ログを出力するコマンドおよび操作について説明します。監査ログは，ここで説明するコマンドと操作を実行したタイミング，およびそのコマンドと操作による処理が実行されたタイミングで出力されます。

ここでは，コマンドと操作を次の分類に分けて，それぞれの分類で監査ログを出力するコマンドおよび操作の一覧を示します。

- J2EE サーバで使用するコマンド
- 性能解析トレース・CTM で使用するコマンド
- Management Server で使用するコマンド
- EJB クライアントアプリケーションで使用するコマンド
- HTTP Server で使用するコマンド・操作

6.6.1　J2EE サーバで使用するコマンド一覧

J2EE サーバで使用するコマンドのうち，監査ログを出力するコマンドの一覧を次の表に示します。

表 6-6　監査ログを出力するコマンドの一覧（J2EE サーバで使用するコマンド）

コマンド名	監査ログ出力ポイントの参照先※
cjaddapp	6.7.1(3)
cjaddsec	6.7.1(3)
cjchmodapp	6.7.1(3)
cjclearpool	6.7.1(3)
cjclosecn	6.7.1(3)
cjcommittrn	6.7.1(3)
cjcopyres	6.7.1(3)
cjdeleteapp	6.7.1(3)
cjdeletejb	6.7.1(3)
cjdeletelibjar	6.7.1(3)
cjdeleteres	6.7.1(3)
cjdeletesec	6.7.1(3)
cjdeployrar	6.7.1(3)
cjdumpsv	6.7.1(2)

コマンド名	監査ログ出力ポイントの参照先※
cjenvsetup	6.7.1(2)
cjenvupdate	6.7.1(2)
cjexportapp	6.7.1(3)
cjexportrar	6.7.1(3)
cjforgettrn	6.7.1(3)
cjgencmpsql	6.7.1(3)
cjgetappprop	6.7.1(3)
cjgetjbprop	6.7.1(3)
cjgetrarprop	6.7.1(3)
cjgetresprop	6.7.1(3)
cjgetstubsjar	6.7.1(3)
cjimportapp	6.7.1(3)
cjimportjb	6.7.1(3)
cjimportlibjar	6.7.1(3)
cjimportres	6.7.1(3)
cjimportwar	6.7.1(3)
cjlistapp	6.7.1(3)
cjlistjb	6.7.1(3)
cjlistlibjar	6.7.1(3)
cjlistpool	6.7.1(3)
cjlistrar	6.7.1(3)
cjlistres	6.7.1(3)
cjlistsec	6.7.1(3)
cjlistthread	6.7.1(3)
cjlisttrn	6.7.1(3)
cjlisttrnfile	6.7.1(3)
cjmapsec	6.7.1(3)
cjrarupdate	6.7.1(2)
cjreloadapp	6.7.1(3)
cjrenameapp	6.7.1(3)
cjreplaceapp	6.7.1(3)

コマンド名	監査ログ出力ポイントの参照先※
cjresetsv	6.7.1(4)
cjresumepool	6.7.1(3)
cjrollbacktrn	6.7.1(3)
cjsetappprop	6.7.1(3)
cjsetjbprop	6.7.1(3)
cjsetrarprop	6.7.1(3)
cjsetresprop	6.7.1(3)
cjsetup	6.7.1(2)
cjstartapp	6.7.1(3)
cjstartjb	6.7.1(3)
cjstartrar	6.7.1(3)
cjstartrecover	6.7.1(2)
cjstartsv	6.7.1(1)
cjstopapp	6.7.1(3)
cjstopjb	6.7.1(3)
cjstoprar	6.7.1(3)
cjstopsv	6.7.1(1)
cjstopthread	6.7.1(3)
cjsuspendpool	6.7.1(3)
cjtestres	6.7.1(3)
cjundeployrar	6.7.1(3)
cjunmapsec	6.7.1(3)

注※　監査ログが出力されるタイミングについての情報の参照先を示します。

6.6.2　バッチサーバで使用するコマンド一覧

バッチサーバで使用するコマンドのうち，監査ログを出力するコマンドの一覧を次の表に示します。

表 6–7　監査ログを出力するコマンドの一覧（バッチサーバで使用するコマンド）

コマンド名	監査ログ出力ポイントの参照先※
cjclearpool	6.7.1(3)

コマンド名	監査ログ出力ポイントの参照先※
cjcopyres	6.7.1(3)
cjdeleteres	6.7.1(3)
cjdeployrar	6.7.1(3)
cjdumpsv	6.7.1(2)
cjenvsetup	6.7.1(2)
cjenvupdate	6.7.1(2)
cjexecjob	6.7.2
cjexportrar	6.7.1(3)
cjgetrarprop	6.7.1(3)
cjgetresprop	6.7.1(3)
cjimportres	6.7.1(3)
cjkilljob	6.7.2
cjlistjob	6.7.2
cjlistpool	6.7.1(3)
cjlistrar	6.7.1(3)
cjlistres	6.7.1(3)
cjlistthread	6.7.1(3)
cjrarupdate	6.7.1(2)
cjresetsv	6.7.1(4)
cjresumepool	6.7.1(3)
cjsetrarprop	6.7.1(3)
cjsetresprop	6.7.1(3)
cjsetup	6.7.1(2)
cjstartrar	6.7.1(3)
cjstartrecover	6.7.1(2)
cjstartsv	6.7.1(1)
cjstoprar	6.7.1(3)
cjstopsv	6.7.1(1)
cjstopthread	6.7.1(3)
cjsuspendpool	6.7.1(3)
cjtestres	6.7.1(3)

コマンド名	監査ログ出力ポイントの参照先※
cjundeployrar	6.7.1(3)

注※　監査ログが出力されるタイミングについての情報の参照先を示します。

6.6.3　性能解析トレース・CTM で使用するコマンド一覧

性能解析トレース，および CTM で使用するコマンドのうち，監査ログを出力するコマンドの一覧を次の表に示します。

表 6-8　監査ログを出力するコマンドの一覧（性能解析トレース・CTM で使用するコマンド）

コマンド名	監査ログ出力ポイントの参照先※
cprfgetpid	6.7.3(3)
cprfstart	6.7.3(1)
cprfstop	6.7.3(1)
ctmdmstart	6.7.3(1)
ctmdmstop	6.7.3(1)
ctmgetpid	6.7.3(3)
ctmstart	6.7.3(1)
ctmstop	6.7.3(1)
ctmstsstart	6.7.3(2)
ctmstsstop	6.7.3(2)

注※　監査ログが出力されるタイミングについての情報の参照先を示します。

6.6.4　Management Server で使用するコマンド一覧

Management Server で使用するコマンドうち，監査ログを出力するコマンドの一覧を次の表に示します。

表 6-9　監査ログを出力するコマンドの一覧（Management Server で使用するコマンド）

コマンド名（サブコマンド）	監査ログ出力ポイントの参照先※1
adminagentctl（start）	6.7.4(3)
adminagentctl（stop）	6.7.4(3)
cmx_admin_passwd	6.7.4(5)
cmx_build_model	6.7.4(5)

コマンド名（サブコマンド）	監査ログ出力ポイントの参照先※1
cmx_build_system※2	6.7.4(6)
cmx_change_model	6.7.4(5)
cmx_delete_system※2	6.7.4(5)
cmx_export_model	6.7.4(4)
cmx_list_model	6.7.4(4)
cmx_list_status	6.7.4(4)
cmx_resume_lb	6.7.4(5)
cmx_scaleout_host	6.7.4(5)
cmx_start_target※2	6.7.4(5)
cmx_stop_target※2	6.7.4(5)
cmx_test_lb※2	6.7.4(4)
cmx_undefined_application	6.7.4(5)
mngsvrctl（setup）	6.7.4(1)，6.7.4(5)
mngsvrctl（start）※2	6.7.4(2)
mngsvrctl（stop）	6.7.4(2)
HTTP Server を操作するコマンド	6.7.4(7)

注※1　監査ログが出力されるタイミングについての情報の参照先を示します。

注※2　これらのコマンドを実行すると，内部処理のために複数のコマンドが自動で実行される場合があります。監査ログを調査して監査を実施するには，内部処理のために実行されるそれぞれのコマンドが出力する監査ログも調査してください。内部処理のために複数のコマンドが自動で実行されるコマンドと，実行されるコマンドごとの監査ログの出力の可否を次の表に示します。

表 6-10　内部処理のために複数のコマンドが自動で実行されるコマンドと実行されるコマンドごとの監査ログの出力の可否

実行するコマンド（サブコマンドまたは引数）	内部処理で実行されるコマンド（引数）	監査ログの出力可否
cmx_build_system	mstartupwebsetup	×
	mngsvrssh	×
	mngsvrsetsid※	×
	cjsetup	○
	hwsserveredit	○
	hwsconfigedit	○
cmx_delete_system	mngsvrssh	×

実行するコマンド（サブコマンドまたは引数）	内部処理で実行されるコマンド（引数）	監査ログの出力可否
	mngsvrsetsid※	×
cmx_start_target	prfinit	×
	mstartupwebinit	×
	mstartupwebstart	×
	ctdminit	×
	mngsvrssh	×
	mngsvrsetsid※	×
	cprfstart	○
	cprfgetpid	○
	cjstartsv	○
	httpsd	○
	osagent	×
	nameserv	×
	ctmdmstart	○
	ctmgetpid	○
	ctmstart	○
cmx_stop_target	mstartupwebstop	×
	mngsvrssh	×
	mngsvrsetsid※	×
	cprfstop	○
	cjstopsv	○
	httpsd	○
	ctmdmstop	○
	ctmstop	○
	ctmstsstop	○
cmx_test_lb	mngsvrssh	×
	mngsvrsetsid※	×
mngsvrctl（start）	mngsvrctl（setup）	○

（凡例） ○：出力される。 ×：出力されない。

注※ UNIX の場合にだけ，内部処理のために実行されるコマンドです。

6.6.5　EJB クライアントアプリケーションで使用するコマンド一覧

EJB クライアントアプリケーションで使用するコマンドのうち，監査ログを出力するコマンドの一覧を次の表に示します。

表 6–11　監査ログを出力するコマンドの一覧（EJB クライアントアプリケーションで使用するコマンド）

コマンド名	監査ログ出力ポイントの参照先※
cjcldellog	6.7.5
cjcldumpap	6.7.5
cjclstartap	6.7.5

注※　監査ログが出力されるタイミングについての情報の参照先を示します。

6.6.6　HTTP Server で使用するコマンド・操作一覧（Windows の場合）

Windows の場合の HTTP Server で使用するコマンドおよび操作のうち，監査ログを出力するコマンドおよび操作の一覧を次の表に示します。

表 6–12　監査ログを出力するコマンド・操作の一覧（Windows の場合の HTTP Server で使用するコマンド・操作）

コマンド名（オプション），操作	監査ログ出力ポイントの参照先※
httpsd（-k gracefulstop）	6.7.6(1)
httpsd（-k restart）	6.7.6(1)
httpsd（-k start）	6.7.6(1)
httpsd（-k stop）	6.7.6(1)
コントロールパネルによる操作	6.7.6(2)

注※　監査ログが出力されるタイミングについての情報の参照先を示します。

6.6.7　HTTP Server で使用するコマンド・操作一覧（UNIX の場合）

UNIX の場合の HTTP Server で使用するコマンドおよび操作のうち，監査ログを出力するコマンドおよび操作の一覧を次の表に示します。

表 6–13　監査ログを出力するコマンド・操作の一覧（UNIX の場合の HTTP Server で使用するコマンド・操作）

コマンド名，操作	監査ログ出力ポイントの参照先※1
httpsd※2	6.7.7(2)
httpsdctl による操作	6.7.7(1)
シグナル送信による操作	6.7.7(3)

注※1　監査ログが出力されるタイミングについての情報の参照先を示します。

注※2　コマンドを直接実行した場合です。

6.7　監査ログの出力ポイント

この節では，監査ログを出力するコマンド・操作を実行したとき，およびそのコマンド・操作による処理が実行されたときの，監査ログが出力されるポイントについて説明します。

ここでは，コマンドおよび操作を次の分類に分けて，それぞれのコマンドで監査ログを出力するポイントについて説明します。

- J2EE サーバで使用するコマンド
- バッチサーバで使用するコマンド
- 性能解析トレース・CTM で使用するコマンド
- Management Server で使用するコマンド
- EJB クライアントアプリケーションで使用するコマンド
- HTTP Server で使用するコマンド

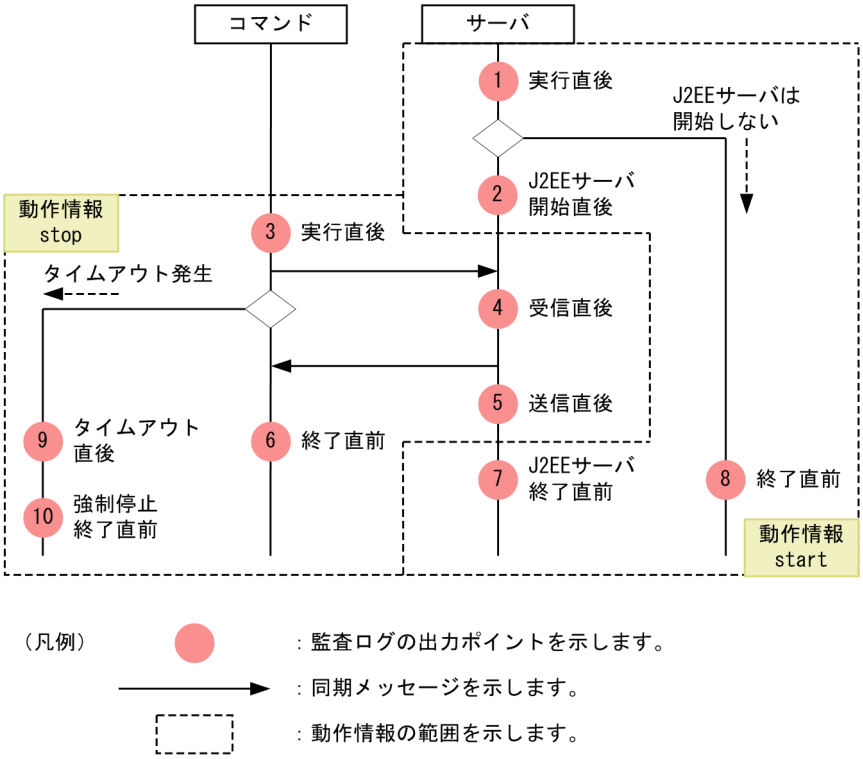
6.7.1　J2EE サーバで使用するコマンドの監査ログの出力ポイント

J2EE サーバで使用するコマンドの監査ログの出力ポイントについて説明します。

(1)　サーバプロセスを起動・停止するコマンド（サーバ管理コマンド以外）

サーバプロセスを起動・停止する，サーバ管理コマンド以外のコマンドの監査ログの出力ポイントを次の図に示します。

図 6-4　監査ログの出力ポイント（サーバプロセスを起動・停止する，サーバ管理コマンド以外のコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-14　監査ログの出力ポイントとメッセージ ID の対応（サーバプロセスを起動・停止する，サーバ管理コマンド以外のコマンド）

コマンド（オプション）	図中の番号	出力される監査ログのメッセージ ID
cjstartsv	1	KDJE54112-I
	2	KDJE54113-I
	7	KDJE54126-I
		KDJE54127-E
	8	KDJE54114-I
		KDJE54115-E
cjstopsv（-wait）	3	KDJE54116-I
	6	KDJE54117-I
		KDJE54118-E
	9	KDJE54119-W
cjstopsv（-f，または-fd）	3	KDJE54116-I
	6	KDJE54117-I

コマンド（オプション）	図中の番号	出力される監査ログのメッセージ ID
		KDJE54118-E
cjstopsv（-wait -f，または-wait -fd）	3	KDJE54116-I
	6	KDJE54117-I
		KDJE54118-E
	9	KDJE54119-W
	10	KDJE54117-I
		KDJE54118-E
cjstopsv（-wait，-f，-fd，-wait -f，および-wait -fd 以外のオプション）	3	KDJE54116-I
	6	KDJE54117-I
		KDJE54118-E
―※1	4	KDJE54138-I
	5	KDJE54139-I
		KDJE54140-E
	―※2	KDJE54155-E

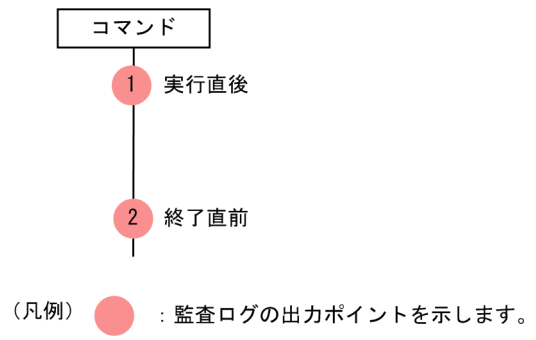
注※1　この表で示しているすべてのコマンドに共通で出力されるメッセージです。ただし，-f，および-fd 以外のオプションを指定してコマンドを実行した場合に限ります。

注※2　この表で示しているコマンドを実行して，メモリ不足が発生した場合に出力されることがあります。

(2)　サーバプロセスを起動・停止するコマンド以外のコマンド（サーバ管理コマンド以外）

サーバプロセスを起動・停止する，サーバ管理コマンド以外のコマンドの監査ログの出力ポイントを次の図に示します。

図 6-5　監査ログの出力ポイント（サーバプロセスを起動・停止する，サーバ管理コマンド以外のコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-15　監査ログの出力ポイントとメッセージ ID の対応（サーバプロセスを起動・停止する，サーバ管理コマンド以外のコマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cjsetup	1	KDJE54100-I
	2	KDJE54101-I
		KDJE54102-E
cjenvsetup	1	KDJE54103-I
	2	KDJE54104-I
		KDJE54105-E
cjenvupdate	1	KDJE54106-I
	2	KDJE54107-I
		KDJE54108-E
cjrarupdate	1	KDJE54109-I
	2	KDJE54110-I
		KDJE54111-E
cjstartrecover※1	1	KDJE54120-I
	2	KDJE54121-I
		KDJE54122-E
cjdumpsv	1	KDJE54123-I
	2	KDJE54124-I
		KDJE54125-E
−	−※2	KDJE54155-E

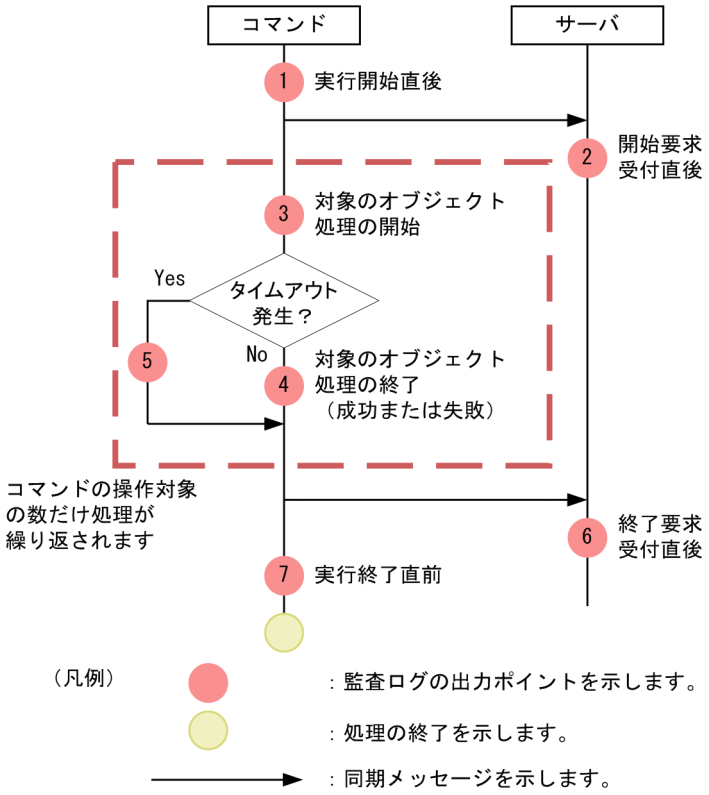
注※1　cjstartrecover コマンドは，内部処理で cjstartsv コマンドを実行します。そのため，cjstartrecover コマンド実行時には cjstartsv コマンドの監査ログも出力されます。

注※2　この表で示しているコマンドを実行して，メモリ不足が発生した場合に出力されることがあります。

(3)　サーバ管理コマンド（cjresetsv 以外のサーバ管理コマンド）

サーバ管理コマンド（cjresetsv 以外のサーバ管理コマンド）の監査ログの出力ポイントを次の図に示します。

図 6-6　監査ログの出力ポイント（cjresetsv 以外のサーバ管理コマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-16　監査ログの出力ポイントとメッセージ ID の対応（cjresetsv 以外のサーバ管理コマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
cjaddapp	3	KDJE54424-I	Add
	4	KDJE54425-I	
		KDJE54426-E	
	5	KDJE54427-E	
cjaddsec	3	KDJE54576-I	
	4	KDJE54577-I	
		KDJE54578-E	
	5	KDJE54579-E	
cjchmodapp	3	KDJE54608-I	Update
	4	KDJE54609-I	
		KDJE54610-E	
	5	KDJE54611-E	
cjclearpool	3	KDJE54600-I	Delete

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
	4	KDJE54601-I	
		KDJE54602-E	
	5	KDJE54603-E	
cjclosecn	3	KDJE54604-I	
	4	KDJE54605-I	
		KDJE54606-E	
	5	KDJE54607-E	
cjcommittrn	3	KDJE54645-I	Update
	4	KDJE54646-I	
		KDJE54647-E	
	5	KDJE54648-E	
cjcopyres	3	KDJE54476-I	Add
	4	KDJE54477-I	
		KDJE54478-E	
	5	KDJE54479-E	
cjdeleteapp	3	KDJE54428-I	Delete
	4	KDJE54429-I	
		KDJE54430-E	
	5	KDJE54431-E	
cjdeletejb	3	KDJE54440-I	
	4	KDJE54441-I	
		KDJE54442-E	
	5	KDJE54443-E	
cjdeletelibjar	3	KDJE54436-I	
	4	KDJE54437-I	
		KDJE54438-E	
	5	KDJE54439-E	
cjdeleteres	3	KDJE54432-I	
	4	KDJE54433-I	
		KDJE54434-E	

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
	5	KDJE54435-E	
cjdeletesec	3	KDJE54580-I	
	4	KDJE54581-I	
		KDJE54582-E	
	5	KDJE54583-E	
cjdeployrar	3	KDJE54568-I	Update
	4	KDJE54569-I	
		KDJE54570-E	
	5	KDJE54571-E	
cjexportapp	3	KDJE54416-I	Refer
	4	KDJE54417-I	
		KDJE54418-E	
	5	KDJE54419-E	
cjexportrar	3	KDJE54420-I	
	4	KDJE54421-I	
		KDJE54422-E	
	5	KDJE54423-E	
cjforgettrn	3	KDJE54653-I	Delete
	4	KDJE54654-I	
		KDJE54655-E	
	5	KDJE54656-E	
cjgencmpsql	3	KDJE54620-I	Update
	4	KDJE54621-I	
		KDJE54622-E	
	5	KDJE54623-E	
cjgetappprop	3	KDJE54444-I	Refer
	4	KDJE54445-I	
		KDJE54446-E	
	5	KDJE54447-E	
cjgetjbprop	3	KDJE54456-I	

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
	4	KDJE54457-I	
		KDJE54458-E	
	5	KDJE54459-E	
cjgetrarprop	3	KDJE54452-I	
	4	KDJE54453-I	
		KDJE54454-E	
	5	KDJE54455-E	
cjgetresprop	3	KDJE54448-I	
	4	KDJE54449-I	
		KDJE54450-E	
	5	KDJE54451-E	
cjgetstubsjar	3	KDJE54612-I	
	4	KDJE54613-I	
		KDJE54614-E	
	5	KDJE54615-E	
cjimportapp	3	KDJE54400-I	Add
	4	KDJE54401-I	
		KDJE54402-E	
	5	KDJE54403-E	
cjimportjb	3	KDJE54412-I	
	4	KDJE54413-I	
		KDJE54414-E	
	5	KDJE54415-E	
cjimportlibjar	3	KDJE54408-I	
	4	KDJE54409-I	
		KDJE54410-E	
	5	KDJE54411-E	
cjimportres	3	KDJE54404-I	
	4	KDJE54405-I	
		KDJE54406-E	

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
	5	KDJE54407-E	
cjimportwar	3	KDJE54661-I	
	4	KDJE54662-I	
		KDJE54663-E	
	5	KDJE54664-E	
cjlistapp	3	KDJE54480-I	Refer
	4	KDJE54481-I	
		KDJE54482-E	
	5	KDJE54483-E	
cjlistjb	3	KDJE54512-I	
	4	KDJE54513-I	
		KDJE54514-E	
	5	KDJE54515-E	
cjlistlibjar	3	KDJE54492-I	
	4	KDJE54493-I	
		KDJE54494-E	
	5	KDJE54495-E	
cjlistpool	3	KDJE54508-I	
	4	KDJE54509-I	
		KDJE54510-E	
	5	KDJE54511-E	
cjlistrar	3	KDJE54488-I	
	4	KDJE54489-I	
		KDJE54490-E	
	5	KDJE54491-E	
cjlistres	3	KDJE54484-I	
	4	KDJE54485-I	
		KDJE54486-E	
	5	KDJE54487-E	
cjlistsec	3	KDJE54584-I	

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
	4	KDJE54585-I	
		KDJE54586-E	
	5	KDJE54587-E	
cjlistthread	3	KDJE54496-I	
	4	KDJE54497-I	
		KDJE54498-E	
	5	KDJE54499-E	
cjlisttrn	3	KDJE54500-I	
	4	KDJE54501-I	
		KDJE54502-E	
	5	KDJE54503-E	
cjlisttrnfile	3	KDJE54504-I	
	4	KDJE54505-I	
		KDJE54506-E	
	5	KDJE54507-E	
cjmapsec	3	KDJE54588-I	Update
	4	KDJE54589-I	
		KDJE54590-E	
	5	KDJE54591-E	
cjreloadapp	3	KDJE54556-I	
	4	KDJE54557-I	
		KDJE54558-E	
	5	KDJE54559-E	
cjrenameapp	3	KDJE54616-I	
	4	KDJE54617-I	
		KDJE54618-E	
	5	KDJE54619-E	
cjreplaceapp	3	KDJE54536-I	
	4	KDJE54537-I	
		KDJE54538-E	

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
	5	KDJE54539-E	
cjresumepool	3	KDJE54564-I	Start
	4	KDJE54565-I	
		KDJE54566-E	
	5	KDJE54567-E	
cjrollbacktrn	3	KDJE54649-I	Update
	4	KDJE54650-I	
		KDJE54651-E	
	5	KDJE54652-E	
cjsetappprop	3	KDJE54460-I	
	4	KDJE54461-I	
		KDJE54462-E	
	5	KDJE54463-E	
cjsetjbprop	3	KDJE54472-I	
	4	KDJE54473-I	
		KDJE54474-E	
	5	KDJE54475-E	
cjsetrarprop	3	KDJE54468-I	
	4	KDJE54469-I	
		KDJE54470-E	
	5	KDJE54471-E	
cjsetresprop	3	KDJE54464-I	
	4	KDJE54465-I	
		KDJE54466-E	
	5	KDJE54467-E	
cjstartapp	3	KDJE54520-I	Start
	4	KDJE54521-I	
		KDJE54522-E	
	5	KDJE54523-E	
cjstartjb	3	KDJE54548-I	

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
	4	KDJE54549-I	
		KDJE54550-E	
	5	KDJE54551-E	
cjstartrar	3	KDJE54528-I	
	4	KDJE54529-I	
		KDJE54530-E	
	5	KDJE54531-E	
cjstopapp	3	KDJE54524-I	Stop
	4	KDJE54525-I	
		KDJE54526-E	
	5	KDJE54527-E	
cjstopjb	3	KDJE54552-I	
	4	KDJE54553-I	
		KDJE54554-E	
	5	KDJE54555-E	
cjstoprar	3	KDJE54532-I	
	4	KDJE54533-I	
		KDJE54534-E	
	5	KDJE54535-E	
cjstopthread	3	KDJE54544-I	
	4	KDJE54545-I	
		KDJE54546-E	
	5	KDJE54547-E	
cjsuspendpool	3	KDJE54560-I	
	4	KDJE54561-I	
		KDJE54562-E	
	5	KDJE54563-E	
cjtestres	3	KDJE54516-I	Refer
	4	KDJE54517-I	
		KDJE54518-E	

コマンド	図中の番号	出力される監査ログのメッセージ ID	動作情報
	5	KDJE54519-E	
cjundeployrar	3	KDJE54572-I	Update
	4	KDJE54573-I	
		KDJE54574-E	
	5	KDJE54575-E	
cjunmapsec	3	KDJE54592-I	
	4	KDJE54593-I	
		KDJE54594-E	
	5	KDJE54595-E	

なお，それぞれのコマンドが出力する監査ログの動作情報ごとに，図中の「1」，「2」，「6」および「7」で出力されるメッセージが異なります。動作情報と図中の「1」，「2」，「6」および「7」で出力されるメッセージ ID の対応を次の表に示します。

表 6–17　動作情報とメッセージ ID の対応（cjresetsv 以外のサーバ管理コマンド）

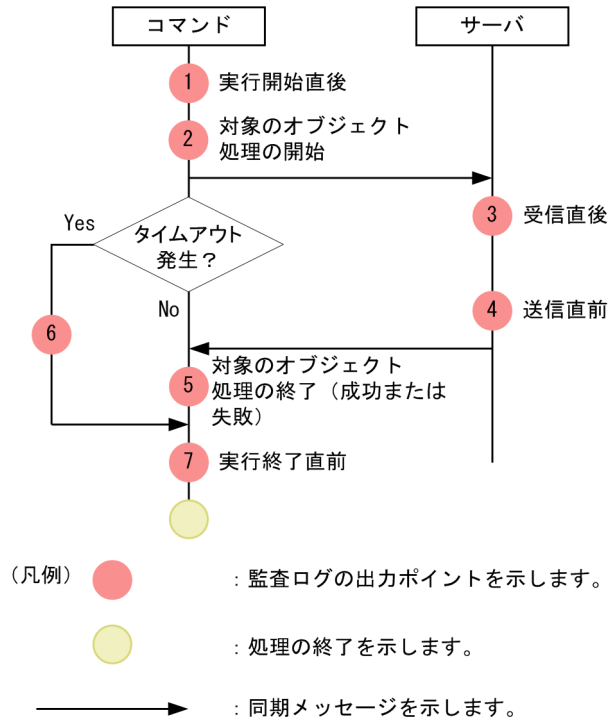
動作情報	図中の番号	メッセージ ID
Start	1	KDJE54624-I
	2	KDJE54141-I
	6	KDJE54142-I
	7	KDJE54625-I
		KDJE54626-E
Stop	1	KDJE54627-I
	2	KDJE54143-I
	6	KDJE54144-I
	7	KDJE54628-I
		KDJE54629-E
Refer	1	KDJE54630-I
	2	KDJE54145-I
	6	KDJE54146-I
	7	KDJE54631-I
		KDJE54632-E
Add	1	KDJE54633-I

動作情報	図中の番号	メッセージ ID
	2	KDJE54147-I
	6	KDJE54148-I
	7	KDJE54634-I
		KDJE54635-E
Update	1	KDJE54636-I
	2	KDJE54149-I
	6	KDJE54150-I
	7	KDJE54637-I
		KDJE54638-E
Delete	1	KDJE54639-I
	2	KDJE54151-I
	6	KDJE54152-I
	7	KDJE54640-I
		KDJE54641-E

(4)　サーバ管理コマンド（cjresetsv）

サーバ管理コマンド（cjresetsv）の監査ログの出力ポイントを次の図に示します。

図 6-7　監査ログの出力ポイント（cjresetsv）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-18　監査ログの出力ポイントとメッセージ ID の対応（cjresetsv）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cjresetsv	1	KDJE54642-I
	2	KDJE54596-I
	3	KDJE54153-I
	4	KDJE54154-I
	5	KDJE54597-I
		KDJE54598-E
	6	KDJE54599-E
	7	KDJE54643-I
		KDJE54644-E

6.7.2　バッチサーバで使用するコマンドの監査ログの出力ポイント

バッチサーバで使用するコマンドの監査ログの出力ポイントについて説明します。

ここでは，バッチサーバで使用するコマンドのうち，cjexecjob コマンド，cjkilljob コマンド，および cjlistjob コマンドの監査ログの出力ポイントについて説明します。

バッチサーバで使用するそのほかのコマンドについては，「6.7.1　J2EE サーバで使用するコマンドの監査ログの出力ポイント」の次の表に示した個所を参照してください。

表 6-19　バッチサーバで使用するそのほかのコマンドの参照先

バッチサーバで使用するコマンド	参照先
サーバプロセスを起動・停止するコマンド（サーバ管理コマンド以外）	6.7.1(1)
サーバプロセスを起動・停止するコマンド以外のコマンド（サーバ管理コマンド以外）	6.7.1(2)
サーバ管理コマンド（cjresetsv 以外のサーバ管理コマンド）※	6.7.1(3)
サーバ管理コマンド（cjresetsv）	6.7.1(4)

注※　「6.7.1(3)　サーバ管理コマンド（cjresetsv 以外のサーバ管理コマンド）」で説明しているコマンドのうち，バッチサーバで使用できるのは次のコマンドだけです。

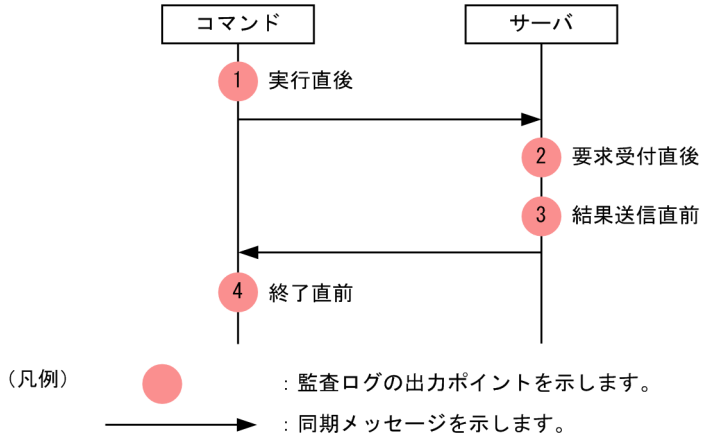
cjclearpool，cjcopyres，cjdeleteres，cjdeployrar，cjexportapp，cjexportrar，cjgetrarprop，cjgetresprop，cjimportres，cjlistpool，cjlistrar，cjlistres，cjlistthread，cjresumepool，cjsetrarprop，cjsetresprop，cjstartrar，cjstoprar，cjstopthread，cjsuspendpool，cjtestres，cjundeployrar，cjunmapsec

(1) バッチアプリケーションのスケジューリング機能を使用しない場合

バッチアプリケーションのスケジューリング機能を使用しない場合の監査ログの出力ポイントについて説明します。

cjexecjob コマンド，cjkilljob コマンド，および cjlistjob コマンドの監査ログの出力ポイントを次の図に示します。

図 6-8　監査ログの出力ポイント（cjexecjob コマンド，cjkilljob コマンド，および cjlistjob コマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-20　監査ログの出力ポイントとメッセージ ID の対応（cjexecjob コマンド，cjkilljob コマンド，および cjlistjob コマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cjexecjob	1	KDJE54156-I
	4	KDJE54157-I
		KDJE54158-E※1
cjkilljob	1	KDJE54162-I
	4	KDJE54163-I
		KDJE54164-W
cjlistjob	1	KDJE54170-I
	4	KDJE54171-I
		KDJE54172-E※2
cjstartsv（cjexecjob コマンド実行時）	2	KDJE54159-I
	3	KDJE54160-I
		KDJE54161-E※3

コマンド	図中の番号	出力される監査ログのメッセージ ID
cjstartsv（cjkilljob コマンド実行時）	2	KDJE54165-I
	3	KDJE54166-I
		KDJE54167-E
cjstartsv（cjlistjob コマンド実行時）	2	KDJE54174-I
	3	KDJE54175-I
		KDJE54176-E

注※1　KDJE54158-E は次の場合に出力されます。

- 指定したサーバが見つからないなどの原因でバッチの実行に失敗した場合
- バッチの実行時，0 以外の終了コード，または JavaVM 終了メソッドに指定した引数以外が出力された場合

注※2　KDJE54172-E は次の場合に出力されます。

- main メソッドが見つからないなどの原因でバッチの実行に失敗した場合

注※3　KDJE54161-E は次の場合に出力されます。

- main メソッドが見つからないなどの原因でバッチの実行に失敗した場合
- バッチの実行時，0 以外の終了コード，または JavaVM 終了メソッドに指定した引数以外が出力された場合

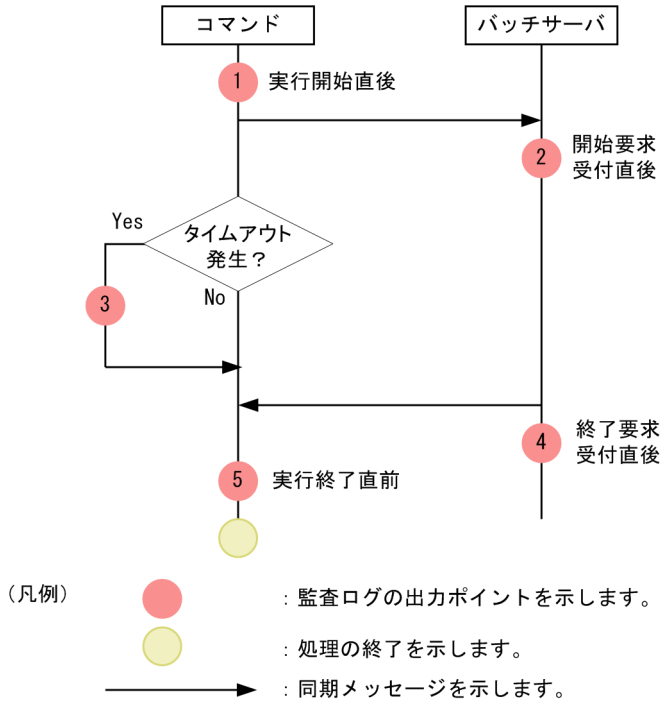
(2)　バッチアプリケーションのスケジューリング機能を使用する場合

バッチアプリケーションのスケジューリング機能を使用する場合の監査ログの出力ポイントについて説明します。

(a)　cjexecjob コマンドの監査ログの出力ポイント

cjexecjob コマンドの監査ログの出力ポイントを次の図に示します。

図 6-9　監査ログの出力ポイント（cjexecjob コマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-21　監査ログの出力ポイントとメッセージ ID の対応（cjexecjob コマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cjexecjob	1	KDJE54156-I
	3	KDJE54168-W
	5	KDJE54157-I
		KDJE54158-E※1
cjstartsv（cjexecjob コマンド実行時）	2	KDJE54159-I
	4	KDJE54160-I
		KDJE54161-E※2

注※1　KDJE54158-E は次の場合に出力されます。

- 指定したサーバが見つからないなどの原因でバッチの実行に失敗した場合
- バッチの実行時，0 以外の終了コード，または JavaVM 終了メソッドに指定した引数以外が出力された場合

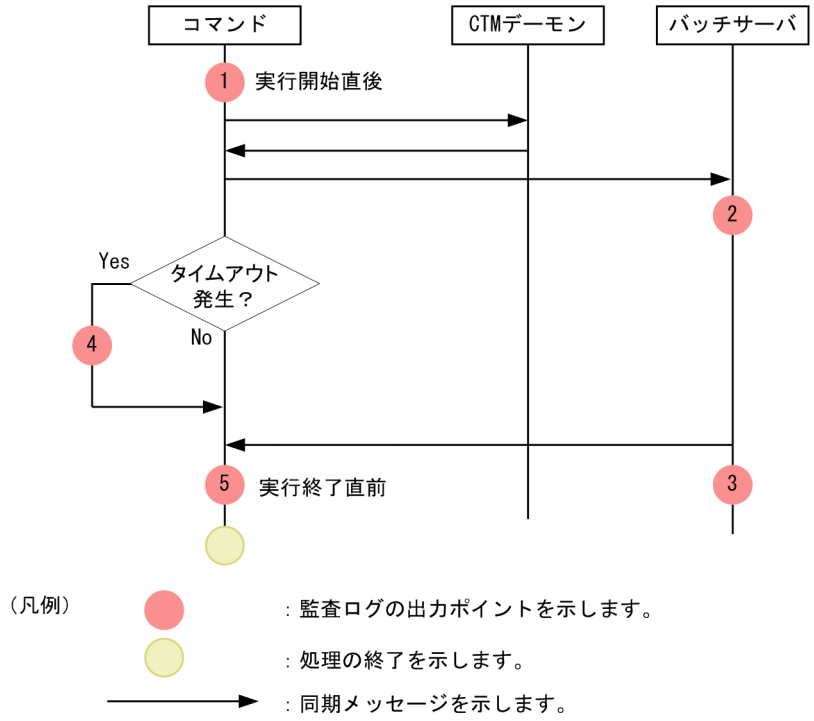
注※2　KDJE54161-E は次の場合に出力されます。

- main メソッドが見つからないなどの原因でバッチの実行に失敗した場合
- バッチの実行時，0 以外の終了コード，または JavaVM 終了メソッドに指定した引数以外が出力された場合

(b)　cjkilljob コマンドの監査ログの出力ポイント

cjkilljob コマンドの監査ログの出力ポイントを次の図に示します。

図 6-10　監査ログの出力ポイント（cjkilljob コマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

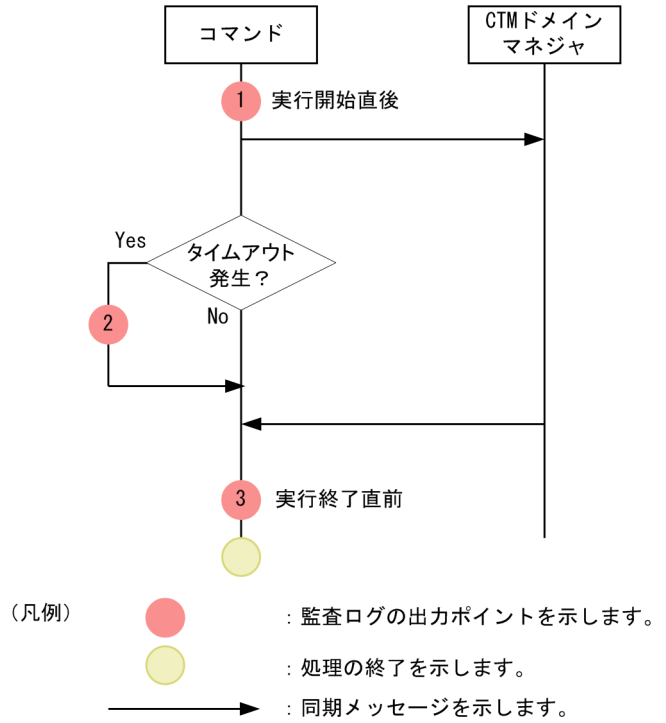
表 6-22　監査ログの出力ポイントとメッセージ ID の対応（cjkilljob コマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cjkilljob	1	KDJE54162-I
	4	KDJE54169-W
	5	KDJE54163-I
		KDJE54164-W
cjstartsv（cjkilljob コマンド実行時）	2	KDJE54165-I
	3	KDJE54166-I
		KDJE54167-E

(c)　cjlistjob コマンドの監査ログの出力ポイント

cjlistjob コマンドの監査ログの出力ポイントを次の図に示します。

図 6-11　監査ログの出力ポイント（cjlistjob コマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-23　監査ログの出力ポイントとメッセージ ID の対応（cjlistjob コマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cjlistjob	1	KDJE54170-I
	2	KDJE54173-W
	3	KDJE54171-I
		KDJE54172-E※

注※　KDJE54172-E は次の場合に出力されます。

- main メソッドが見つからないなどの原因でバッチの実行に失敗した場合

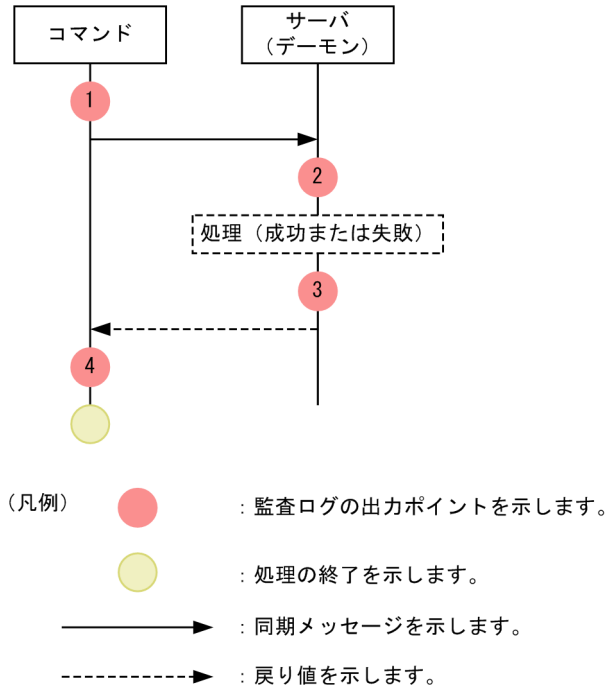
6.7.3　性能解析トレース・CTM で使用するコマンドの監査ログの出力ポイント

性能解析トレース・CTM で使用するコマンドの監査ログの出力ポイントについて説明します。

(1)　サーバプロセスを起動・停止するコマンド

サーバプロセス(デーモン)を起動・停止するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-12　監査ログの出力ポイント（サーバプロセスを起動・停止するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-24　監査ログの出力ポイントとメッセージ ID の対応（サーバプロセスを起動・停止するコマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cprfstart	1	KFCT81000-I
	4	KFCT81001-I
		KFCT81002-E
cprfstop	1	KFCT81020-I
	4	KFCT81021-I
		KFCT81022-E
ctmdmstart	1	KFCT91000-I
	4	KFCT91001-I
		KFCT91002-E
ctmdmstop	1	KFCT91020-I
	4	KFCT91021-I
		KFCT91022-E
ctmstart	1	KFCT91040-I
	4	KFCT91041-I

コマンド	図中の番号	出力される監査ログのメッセージ ID
		KFCT91042-E
ctmstop	1	KFCT91060-I
	4	KFCT91061-I
		KFCT91062-E

なお，それぞれのコマンドがリクエストを送信するサーバプロセス（デーモン）ごとに，図中の「2」および「3」で出力されるメッセージが異なります。サーバプロセスの種類と図中の「2」および「3」で出力されるメッセージ ID の対応を次の表に示します。

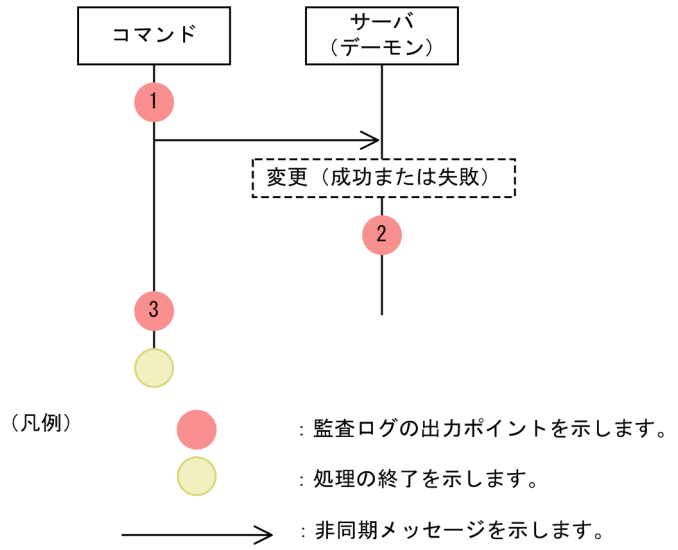
表 6-25　サーバプロセス（デーモン）の種類とメッセージ ID の対応（サーバプロセスを起動・停止するコマンド）

サーバプロセス（デーモン）	図中の番号	メッセージ ID
PRF デーモン	2	KFCT81010-I
		KFCT81030-I
	3	KFCT81011-I
		KFCT81012-E
		KFCT81031-I
		KFCT81032-E
CTM ドメインマネジャ	2	KFCT91010-I
		KFCT91030-I
	3	KFCT91011-I
		KFCT91012-E
		KFCT91031-I
		KFCT91032-E
CTM デーモン	2	KFCT91050-I
		KFCT91070-I
	3	KFCT91051-I
		KFCT91052-E
		KFCT91071-I
		KFCT91072-E

(2)　プロセスへの通信・共有メモリへのアクセスを実行するコマンド

プロセスへの通信・共有メモリへのアクセスを実行するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-13　監査ログの出力ポイント（プロセスへの通信・共有メモリへのアクセスを実行するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-26　監査ログの出力ポイントとメッセージ ID の対応（プロセスへの通信・共有メモリへのアクセスを実行するコマンド）

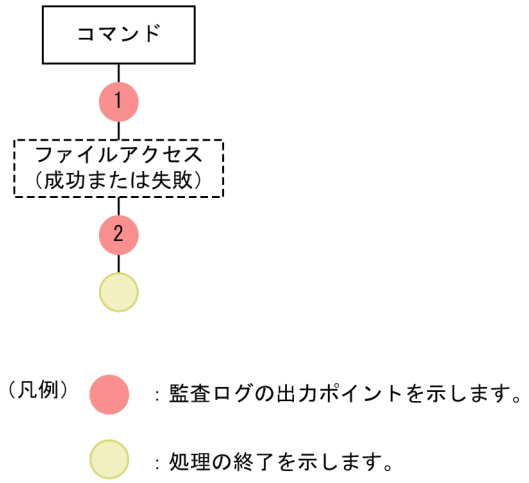
コマンド	図中の番号	出力される監査ログのメッセージ ID
ctmstsstart	1	KFCT91080-I
	3	KFCT91081-I
		KFCT91082-E
ctmstsstop	1	KFCT91100-I
	3	KFCT91101-I
		KFCT91102-E
_※	2	KFCT91091-I
		KFCT91111-I

注※　この表で示しているすべてのコマンドに共通で出力されるメッセージです。

(3)　情報を参照するコマンド

情報を参照するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-14　監査ログの出力ポイント（情報を参照するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-27　監査ログの出力ポイントとメッセージ ID の対応（情報を参照するコマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cprfgetpid	1	KFCT84000-I
	2	KFCT84001-I
		KFCT84002-E
ctmgetpid	1	KFCT94000-I
	2	KFCT94001-I
		KFCT94002-E

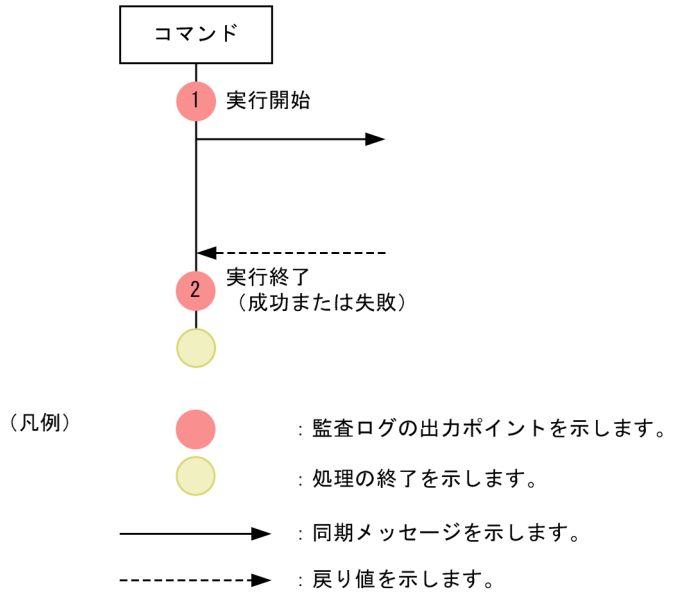
6.7.4　Management Server で使用するコマンドの監査ログの出力ポイント

Management Server で使用するコマンドの監査ログの出力ポイントについて説明します。

(1)　Management Server のセットアップコマンド

Management Server のセットアップコマンドの監査ログの出力ポイントを次の図に示します。

図 6–15　監査ログの出力ポイント（Management Server のセットアップコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

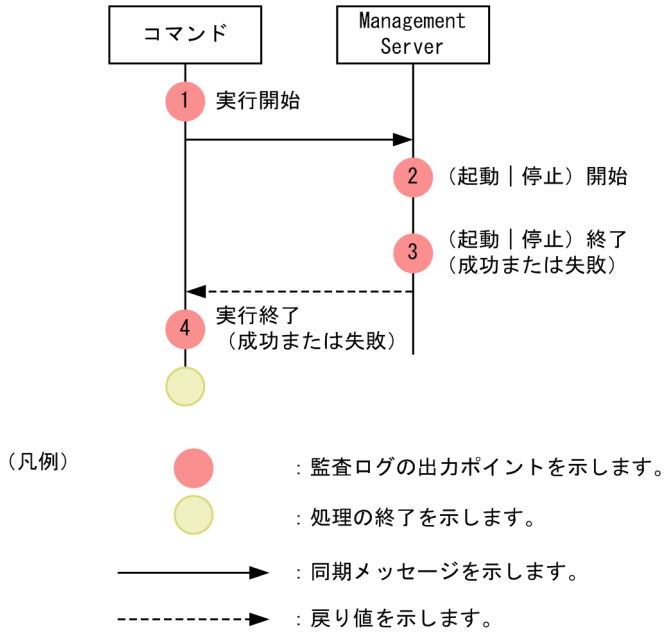
表 6–28　監査ログの出力ポイントとメッセージ ID の対応（Management Server のセットアップコマンド）

コマンド（オプション）	図中の番号	出力される監査ログのメッセージ ID
mngsvrctl（setup）	1	KEOS80006-I
	2	KEOS80007-I
		KEOS80008-E

(2)　Management Server を起動・停止するコマンド

Management Server を起動・停止するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-16　監査ログの出力ポイント（Management Server を起動・停止するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

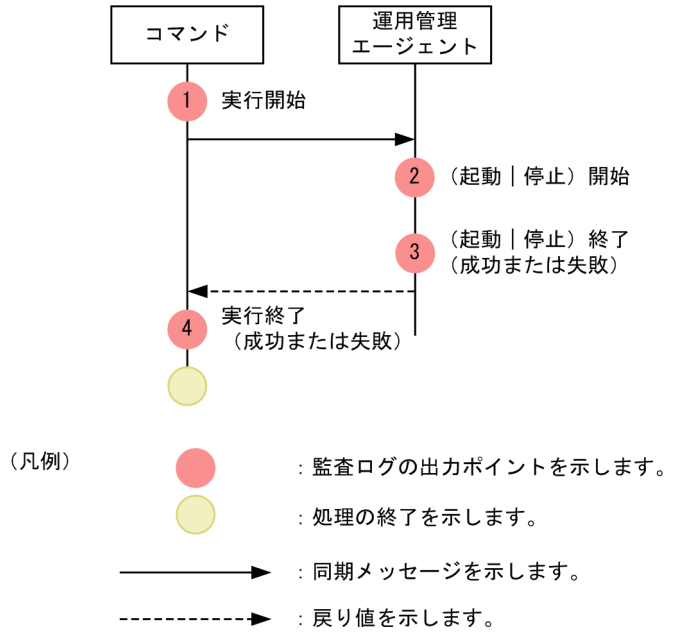
表 6-29　監査ログの出力ポイントとメッセージ ID の対応（Management Server を起動・停止するコマンド）

コマンド（サブコマンド）	図中の番号	出力される監査ログのメッセージ ID
mngsvrctl（start）	1	KEOS80000-I
	2	KEOS80108-I
	3	KEOS80109-I
		KEOS80110-E
	4	KEOS80001-I
		KEOS80002-E
mngsvrctl（stop）	1	KEOS80003-I
	2	KEOS80111-I
	3	KEOS80112-I
		KEOS80113-E
	4	KEOS80004-I
		KEOS80005-E

(3)　運用管理エージェントの起動・停止を実行するコマンド

運用管理エージェントの起動・停止の監査ログの出力ポイントを次の図に示します。

図 6-17　監査ログの出力ポイント（運用管理エージェントの起動・停止）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

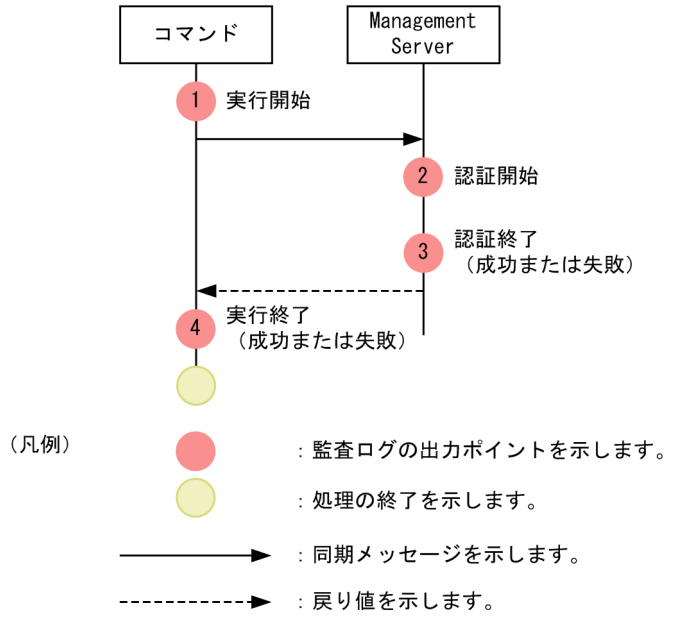
表 6-30　監査ログの出力ポイントとメッセージ ID の対応（運用管理エージェントの起動・停止）

コマンド（サブコマンド）	図中の番号	出力される監査ログのメッセージ ID
adminagentctl（start）	1	KEOS80009-I
	2	KEOS80114-I
	3	KEOS80115-I
		KEOS80116-E
	4	KEOS80010-I
		KEOS80011-E
adminagentctl（stop）	1	KEOS80012-I
	2	KEOS80117-I
	3	KEOS80118-I
		KEOS80119-E
	4	KEOS80013-I
		KEOS80014-E

(4)　Management Server による承認処理を実行するコマンド

Management Server による承認処理を実行するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-18　監査ログの出力ポイント（Management Server による承認処理を実行するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-31　監査ログの出力ポイントとメッセージ ID の対応（Management Server による承認処理を実行するコマンド）

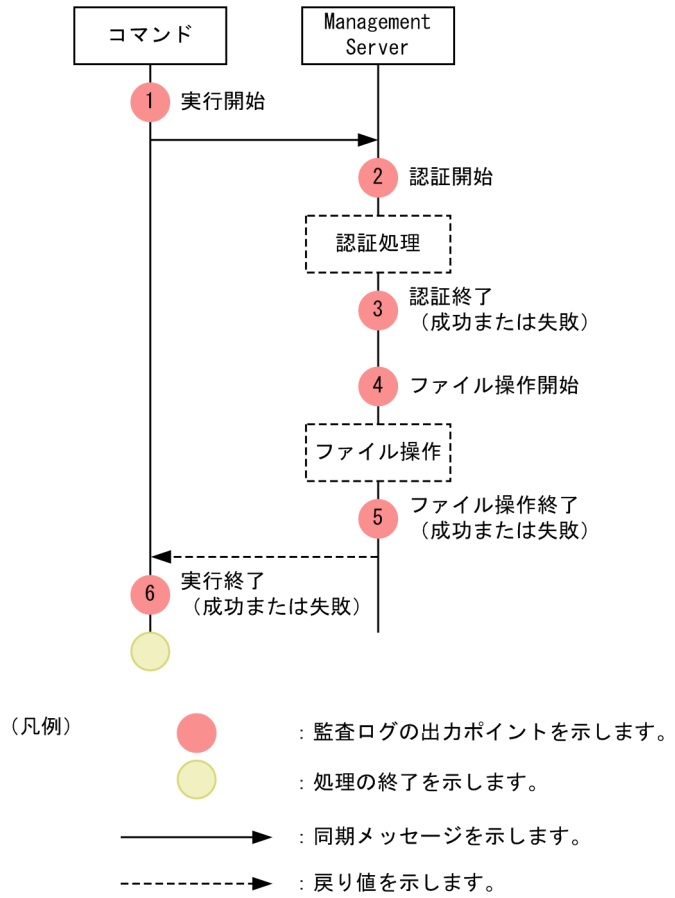
コマンド	図中の番号	出力される監査ログのメッセージ ID
cmx_test_lb	1	KEOS80036-I
	4	KEOS80037-I
		KEOS80038-E
cmx_list_status	1	KEOS80045-I
	4	KEOS80046-I
		KEOS80047-E
cmx_list_model	1	KEOS80102-I
	4	KEOS80103-I
		KEOS80104-E
cmx_export_model	1	KEOS80105-I
	4	KEOS80106-I
		KEOS80107-E
―※	2	KEOS80120-I
	3	KEOS80121-I
		KEOS80122-E

注※　この表で示しているすべてのコマンドに共通で出力されるメッセージです。

(5) Management Server によるファイル操作を実行するコマンド

Management Server によるファイル操作を実行するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-19 監査ログの出力ポイント（Management Server によるファイル操作を実行するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-32 監査ログの出力ポイントとメッセージ ID の対応（Management Server によるファイル操作を実行するコマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
mngsvrctl setup	1	KEOS80006-I
	6	KEOS80007-I
		KEOS80008-E
cmx_admin_passwd	1	KEOS80015-I
	6	KEOS80016-I
		KEOS80017-E

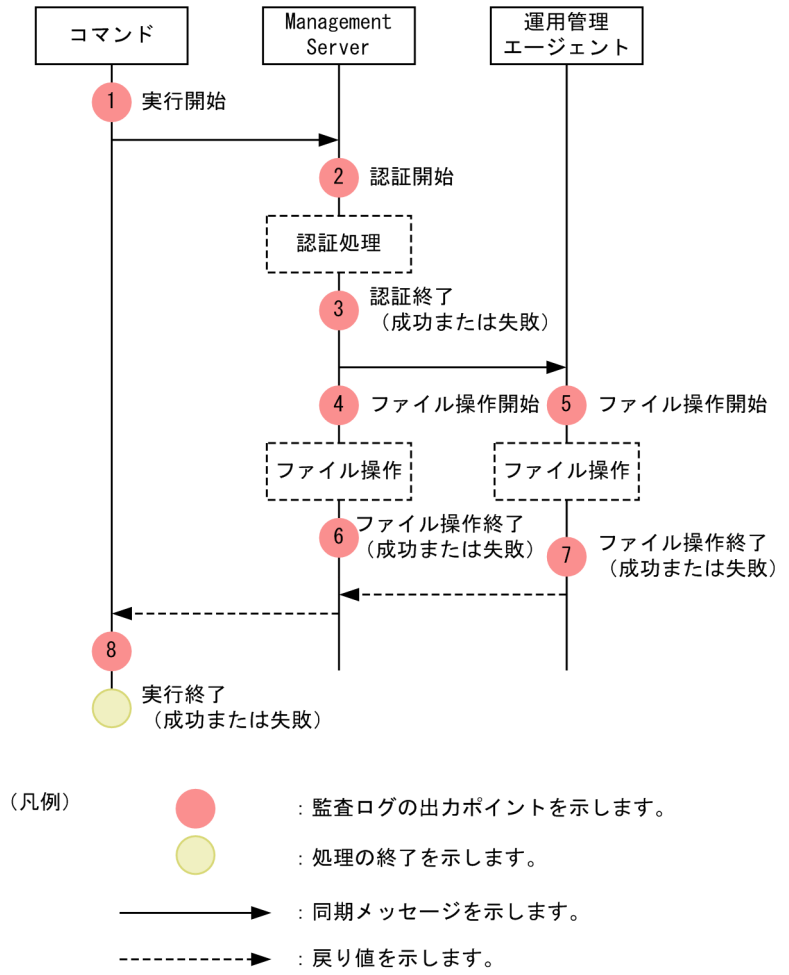
コマンド	図中の番号	出力される監査ログのメッセージ ID
cmx_build_model	1	KEOS80018-I
	6	KEOS80019-I
		KEOS80020-E
cmx_change_model	1	KEOS80024-I
	6	KEOS80025-I
		KEOS80026-E
cmx_delete_system	1	KEOS80027-I
	6	KEOS80028-I
		KEOS80029-E
cmx_resume_lb	1	KEOS80030-I
	6	KEOS80031-I
		KEOS80032-E
cmx_scaleout_host	1	KEOS80033-I
	6	KEOS80034-I
		KEOS80035-E
cmx_start_target	1	KEOS80048-I
	6	KEOS80049-I
		KEOS80050-E
cmx_stop_target	1	KEOS80051-I
	6	KEOS80052-I
		KEOS80053-E
cmx_undefined_application	1	KEOS80093-I
	6	KEOS80094-I
		KEOS80095-E
_※	2	KEOS80120-I
	3	KEOS80121-I
		KEOS80122-E
	4	KEOS80123-I
	5	KEOS80124-I
		KEOS80125-E

注※　この表で示しているすべてのコマンドに共通で出力されるメッセージです。

(6) Management Server・運用管理エージェントによるファイル操作を実行するコマンド

Management Server・運用管理エージェントによるファイル操作を実行するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-20 監査ログの出力ポイント（Management Server・運用管理エージェントによるファイル操作を実行するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-33 監査ログの出力ポイントとメッセージ ID の対応（Management Server・運用管理エージェントによるファイル操作を実行するコマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cmx_build_system	1	KEOS80021-I
	2	KEOS80120-I
	3	KEOS80121-I
		KEOS80122-E

コマンド	図中の番号	出力される監査ログのメッセージ ID
	4	KEOS80123-I
	6	KEOS80124-I
		KEOS80125-E
	8	KEOS80022-I
		KEOS80023-E

なお，それぞれのコマンドが操作を実行するファイルごとに，図中の「5」および「7」で出力されるメッセージが異なります。操作を実行するファイルと図中の「5」および「7」で出力されるメッセージ ID の対応を次の表に示します。

表 6-34　操作を実行するファイルと出力されるメッセージ ID の対応

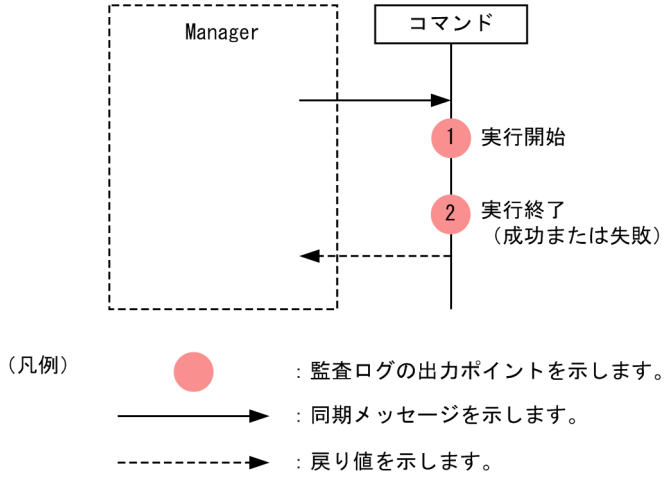
操作を実行するファイル	図中の番号	出力される監査ログのメッセージ ID
運用監視エージェントの設定ファイル	5	KEOS80126-I
	7	KEOS80127-I
		KEOS80128-E
J2EE サーバ用 JP1 イベント発行用プロパティファイル	5	KEOS80129-I
	7	KEOS80130-I
		KEOS80131-E
Management イベント発行用プロパティファイル	5	KEOS80132-I
	7	KEOS80133-I
		KEOS80134-E
J2EE サーバ用ユーザプロパティファイル	5	KEOS80135-I
	7	KEOS80136-I
		KEOS80137-E
J2EE サーバ用オプション定義ファイル	5	KEOS80138-I
	7	KEOS80139-I
		KEOS80140-E
HTTP Server 用リダイレクタ動作定義ファイル	5	KEOS80144-I
	7	KEOS80145-I
		KEOS80146-E
ワーカ定義ファイル	5	KEOS80147-I
	7	KEOS80148-I

操作を実行するファイル	図中の番号	出力される監査ログのメッセージ ID
		KEOS80149-E

(7) HTTP Server を操作するコマンド

HTTP Server を操作するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-21 監査ログの出力ポイント（HTTP Server を操作するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-35 監査ログの出力ポイントとメッセージ ID の対応（HTTP Server を操作するコマンド）

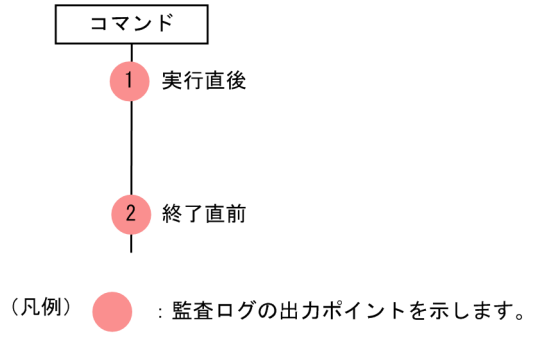
コマンド	図中の番号	出力される監査ログのメッセージ ID
設定ファイル編集コマンド	1	KAWS10501-I
	2	KAWS10502-I
		KAWS10503-E
	1	KAWS10504-I
	2	KAWS10505-I
		KAWS10506-E
hwsserveredit	1	KAWS10601-I
	2	KAWS10602-I
		KAWS10603-E
	1	KAWS10604-I
	2	KAWS10605-I
		KAWS10606-E
	1	KAWS10607-I

コマンド	図中の番号	出力される監査ログのメッセージ ID
	2	KAWS10608-I
		KAWS10609-E

6.7.5 EJB クライアントアプリケーションで使用するコマンドの監査ログの出力ポイント

EJB クライアントアプリケーションで使用するコマンドの監査ログの出力ポイントについて説明します。

図 6-22 監査ログの出力ポイント（EJB クライアントアプリケーションで使用するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-36 監査ログの出力ポイントとメッセージ ID の対応（EJB クライアントアプリケーションで使用するコマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
cjclstartap	1	KDJE54129-I
	2	KDJE54130-I
		KDJE54131-E※
cjcldumpap	1	KDJE54132-I
	2	KDJE54133-I
		KDJE54134-E
cjcldellog	1	KDJE54135-I
	2	KDJE54136-I
		KDJE54137-E

注※ KDJE54131-E は次の場合に出力されます。

- JavaVM の初期化など，コマンドの内部処理が失敗した場合
- Java アプリケーションの実行後に終了コードが 0 以外の場合

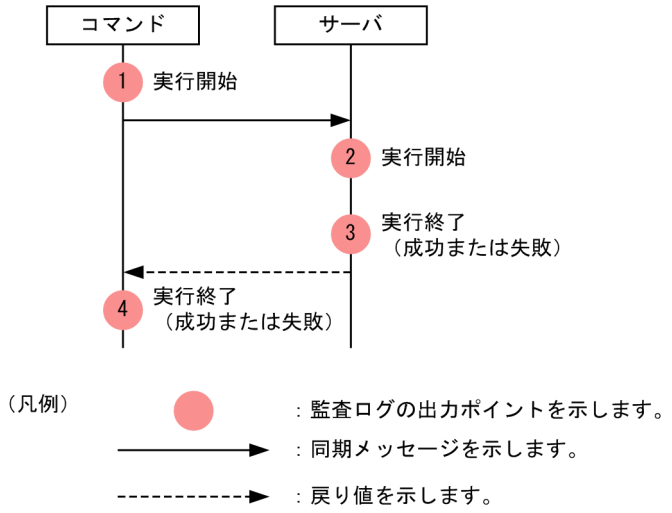
6.7.6　HTTP Server で使用するコマンド，および HTTP Server に対する操作の監査ログの出力ポイント（Windows の場合）

Windows の場合の HTTP Server で使用するコマンドおよび HTTP Server に対する操作の監査ログの出力ポイントについて説明します。

(1)　HTTP Server の起動・停止を実行するコマンド

HTTP Server の起動・停止を実行するコマンドの監査ログの出力ポイントを次の図に示します。

図 6–23　監査ログの出力ポイント（HTTP Server の起動・停止を実行するコマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6–37　監査ログの出力ポイントとメッセージ ID の対応（HTTP Server の起動・停止を実行するコマンド）

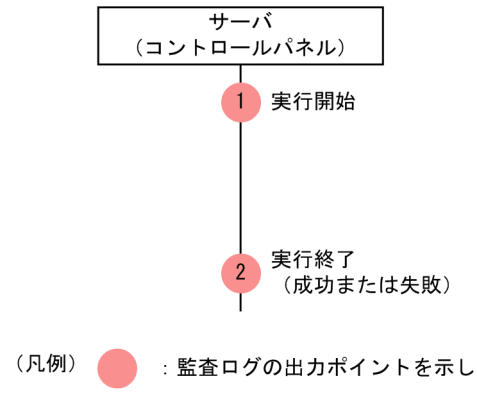
コマンド（オプション）	図中の番号	出力される監査ログのメッセージ ID
httpsd（-k start）	1	KAWS10001-I
	2	KAWS10004-I
	3	KAWS10005-I
		KAWS10006-E
	4	KAWS10002-I
		KAWS10003-E
httpsd（-k stop）	1	KAWS10010-I
	2	KAWS10013-I
	3	KAWS10014-I
		KAWS10015-E

コマンド（オプション）	図中の番号	出力される監査ログのメッセージ ID
	4	KAWS10011-I
		KAWS10012-E
httpsd（-k restart）	1	KAWS10019-I
	2	KAWS10022-I
	3	KAWS10023-I
		KAWS10024-E
	4	KAWS10020-I
		KAWS10021-E
httpsd（-k gracefulstop）	1	KAWS10025-I
	2	KAWS10028-I
	3	KAWS10029-I
		KAWS10030-E
	4	KAWS10026-I
		KAWS10027-E

(2)　HTTP Server の起動・停止を実行する操作（コントロールパネルから）

コントロールパネルから HTTP Server の起動・停止を実行する操作の監査ログの出力ポイントを次の図に示します。

図 6–24　監査ログの出力ポイント（コントロールパネルから HTTP Server の起動・停止を実行する操作）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-38　監査ログの出力ポイントとメッセージ ID の対応（コントロールパネルから HTTP Server の起動・停止を実行する操作）

図中の番号	出力される監査ログのメッセージ ID
1	KAWS10004-I
	KAWS10007-I
	KAWS10013-I
	KAWS10016-I
2	KAWS10005-I
	KAWS10006-E
	KAWS10008-I
	KAWS10009-E
	KAWS10014-I
	KAWS10015-E
	KAWS10017-I
	KAWS10018-E

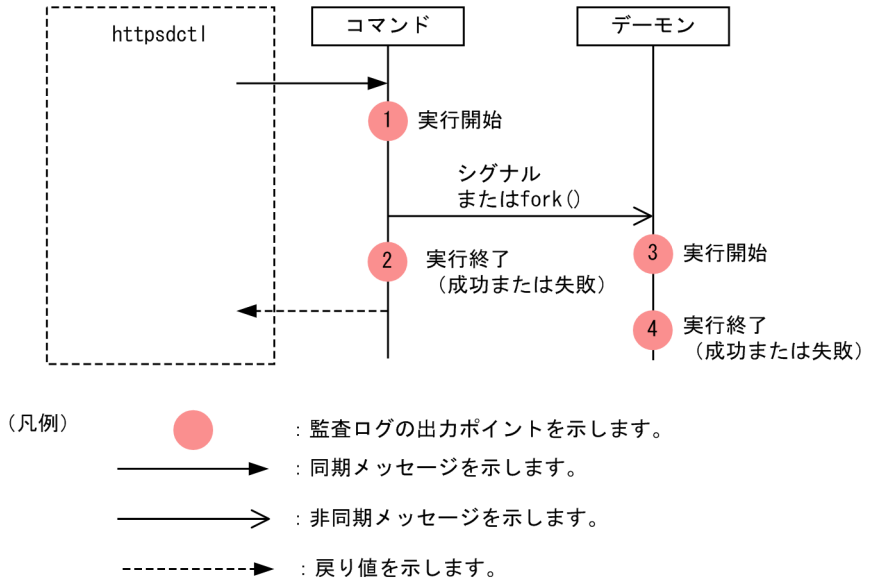
6.7.7　HTTP Server で使用するコマンド，および HTTP Server に対する操作の監査ログの出力ポイント（UNIX の場合）

UNIX の場合の HTTP Server で使用するコマンドおよび HTTP Server に対する操作の監査ログの出力ポイントについて説明します。

(1)　HTTP Server の起動・停止を実行するコマンド（httpsdctl ユティリティから）

httpsdctl ユティリティから HTTP Server の起動・停止を実行するコマンドの監査ログの出力ポイントを次の図に示します。

図 6-25　監査ログの出力ポイント（httpsdctl ユティリティから HTTP Server の起動・停止を実行するコマンド）



（凡例）
：監査ログの出力ポイントを示します。
：同期メッセージを示します。
：非同期メッセージを示します。
：戻り値を示します。

この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-39　監査ログの出力ポイントとメッセージ ID の対応（httpsdctl ユティリティから HTTP Server の起動・停止を実行するコマンド）

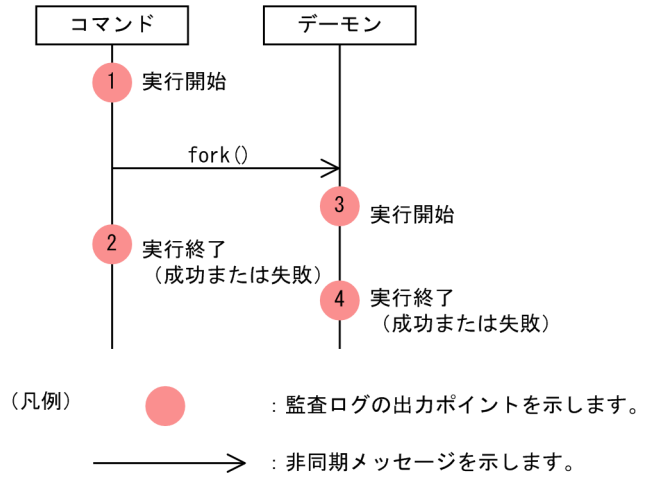
コマンド（httpsdctl ユティリティの操作）	図中の番号	出力される監査ログのメッセージ ID
httpsd（httpsdctl start）	1	KAWS10101-I
	2	KAWS10102-I
		KAWS10103-E
	3	KAWS10151-I
	4	KAWS10152-I
		KAWS10153-E
httpsd（httpsdctl restart）	1	KAWS10104-I
	2	KAWS10105-I
		KAWS10106-E
		KAWS10116-I
		KAWS10117-E
	3	KAWS10154-I
	4	KAWS10155-I
		KAWS10156-E
httpsd（httpsdctl graceful）	1	KAWS10107-I

コマンド（httpsdctl ユティリティの操作）	図中の番号	出力される監査ログのメッセージ ID
	2	KAWS10108-I
		KAWS10109-E
		KAWS10118-I
		KAWS10119-E
	3	KAWS10157-I
	4	KAWS10158-I
		KAWS10159-E
httpsd（httpsdctl stop）	1	KAWS10110-I
	2	KAWS10111-I
		KAWS10112-E
	3	KAWS10160-I
	4	KAWS10161-I
		KAWS10162-E
httpsd（httpsdctl gracefulstop）	1	KAWS10113-I
	2	KAWS10114-I
		KAWS10115-E
	3	KAWS10163-I
	4	KAWS10164-I
		KAWS10165-E

(2) httpsd コマンド（コマンドの直接実行）

httpsd コマンドを直接実行する場合の監査ログの出力ポイントを次の図に示します。

図 6-26　監査ログの出力ポイント（直接実行する httpsd コマンド）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

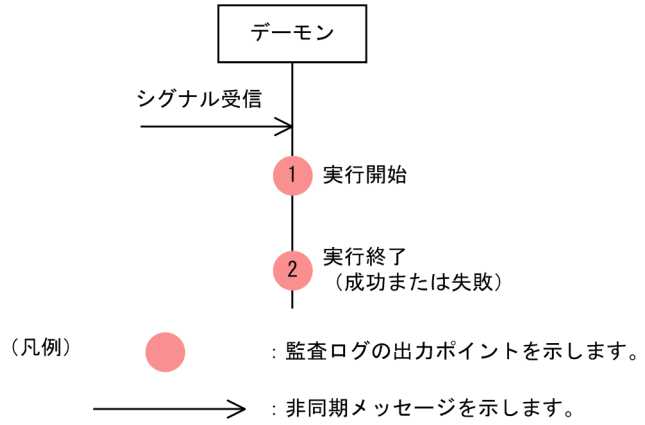
表 6-40　監査ログの出力ポイントとメッセージ ID の対応（直接実行する httpsd コマンド）

コマンド	図中の番号	出力される監査ログのメッセージ ID
httpsd	1	KAWS10101-I
	2	KAWS10102-I
		KAWS10103-E
	3	KAWS10151-I
	4	KAWS10152-I
		KAWS10153-E

(3)　シグナル送信によるデーモンプロセスの操作

シグナル送信によってデーモンプロセスを操作するときの監査ログの出力ポイントを次の図に示します。

図 6-27　監査ログの出力ポイント（シグナル送信によるデーモンプロセスの操作）



この図に示した監査ログの出力ポイントと出力される監査ログのメッセージ ID の対応を次の表に示します。

表 6-41　監査ログの出力ポイントとメッセージ ID の対応（シグナル送信によるデーモンプロセスの操作）

シグナル送信による操作	図中の番号	出力される監査ログのメッセージ ID
再起動（SIGHUP）	1	KAWS10154-I
	2	KAWS10155-I
		KAWS10156-E
再起動（SIGUSR1）	1	KAWS10157-I
	2	KAWS10158-I
		KAWS10159-E
停止（SIGTERM）	1	KAWS10160-I
	2	KAWS10161-I
		KAWS10162-E
計画停止（SIGUSR2）	1	KAWS10163-I
	2	KAWS10164-I
		KAWS10165-E

6.8　アプリケーションの監査ログを出力するための実装

アプリケーションサーバでは，J2EE アプリケーションまたはバッチアプリケーションに実装できる監査ログ出力用の API を提供しています。監査ログ出力用の API を使用することで，アプリケーションに対する操作，およびアプリケーションによる処理が実行されたタイミングで，操作および処理の履歴を監査ログに出力させることができます。ここでは，監査ログ出力用の API の実装例と実装時の注意事項について説明します。

なお，監査ログ出力用の 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」の「8.　監査ログ出力で使用する API」を参照してください。

6.8.1　監査ログ出力用の API の実装例

監査ログ出力用の API の実装例を次に示します。

```
if (UserAuditLogger.isEnabled())
{
    try
    {
        UserAuditLogger logger = UserAuditLogger.getLogger(”UserComponent”);
        if (logger.isLoggable(”Message1”))
        {
            AuditLogRecord record = new AuditLogRecord();
            record.setMessageId(”Message1”);
            record.setCategory(AuditLogRecord.CATEGORY_CONFIGURATION_ACCESS);
            record.setResult(AuditLogRecord.RESULT_SUCCESS);
            record.setObjectInfo(”Object”);
            record.setOperation(AuditLogRecord.OPERATION_REFER);
            record.setMessage(”Message”);
            logger.log(record);
        }
    }
    catch (AuditLogException e)
    {
        // 監査ログの出力に失敗したときの処理
    }
}
```

このように実装した場合の監査ログの出力例を次に示します。なお，この例は Windows の場合です。

```
CALFHM 1.0, seqnum=2, msgid=Message1, date=2007-05-31T19:12:53.788+09:00, progid=Cosminexus,
 compid=UAP_UserComponent, pid=3984, ocp:host=hostname, ctgry=ConfigurationAccess, result=Su
ccess, subj:euid=username, obj=”Object”, op=”Refer”, loc=”10.209.15.130/1234/0x0000000000000
001”, msg=”Message”
```

6.8.2　監査ログ出力用の API 実装時の注意事項

監査ログ出力用の API を実装するときの注意事項について説明します。

システムの管理・運用に関する情報について

システム管理や運用に関する情報のうち，監査を実行するユーザなどに参照させてはいけない情報を監査ログに出力しないようにしてください。監査ログに出力してはいけない情報の例を次に示します。

- データベースにアクセスするユーザのユーザ名，およびパスワード
- 監査ログファイルや監査ログの設定ファイルに対して書き込み権限を所有するユーザのユーザ名，およびパスワード

複数マシン間での時刻の設定について

複数のマシンにわたって監査ログを出力する場合は，それぞれのマシンに同じ時刻を設定しておいてください。

監査ログに出力する項目について

J2EE アプリケーションまたはバッチアプリケーションが出力する監査ログには，次の項目が必ず出力されるように実装してください。

- メッセージ ID
- 発生コンポーネント名
- 監査事象の種別
- 監査事象の結果

6.9　監査ログ出力の設定

この節では，監査ログを出力するために必要な設定について説明します。監査ログを出力する機能については，「6.3　監査ログとは」を参照してください。

また，ここでは，監査ログの動作に関する情報が出力される，監査ログのメッセージログ，および監査ログの例外情報の出力に関する設定についても説明します。監査ログのメッセージログ，および監査ログの例外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 機能解説 保守／移行編」の「4.3.4 監査ログで出力するログの取得」を参照してください。

監査ログ，監査ログのメッセージログ，および監査ログの例外情報の出力についての設定は，製品のインストール単位に設定します。

なお，監査ログの格納場所を変更したい場合は，それぞれのログの設定をする前に，次の環境変数をシステム環境変数に設定しておいてください。

```
COSMINEXUS_AUDITLOG_CONF=”<監査ログ定義ファイルの格納場所のパス>”
```

また，監査ログを出力する場合は，この節で説明している手順を必ず実施してください。この節で説明している手順を実施しなかった場合，監査ログ定義ファイルに設定した，出力ディレクトリや監査ログファイルに対するアクセス権が適用されません。監査ログ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「11.2.1　監査ログ定義ファイル」を参照してください。それぞれのログの設定手順を次に示します。

1. **監査ログ定義ファイルを編集します。**

   監査ログ，監査ログのメッセージログ，および監査ログの例外情報について，出力の有無やそれぞれのログの出力先などを，監査ログ定義ファイルのプロパティに設定します。

   監査ログ定義ファイルに設定できるプロパティの詳細は，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「11.2.1　監査ログ定義ファイル」を参照してください。

2. **監査ログセットアップコマンド（auditsetup コマンド）を実行します。**

   監査ログセットアップコマンドを実行すると，監査ログ，監査ログのメッセージログ，および監査ログの例外情報に必要なアクセス権限が自動で設定されます。監査ログセットアップ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「12.2　監査ログの設定で使用するコマンドの詳細」を参照してください。

   監査ログセットアップコマンドの実行例を次に示します。

   Windows の場合

   C:¥Program Files¥Hitachi¥Cosminexus¥common¥bin¥auditsetup.exe

   UNIX の場合

   /opt/Cosminexus/common/bin/auditsetup

7 データベース監査証跡連携機能

この章では，データベースが提供するデータベース監査証跡の機能と連携する機能，およびアプリケーションサーバのシステムでできることについて説明します。

7.1　この章の構成

この章の構成を次の表に示します。

表 7-1　この章の構成（データベース監査証跡連携機能）

分類	タイトル	参照先
解説	データベース監査証跡連携機能の概要	7.2
	データベース監査証跡との連携	7.3
	データベース監査証跡への情報の出力	7.4
設定	データベース監査証跡との連携機能の設定	7.5
運用	データベース監査証跡と連携した運用例	7.6
注意事項	ルートアプリケーション情報をデータベースに設定する場合の注意事項	7.7

注「実装」について，この機能固有の説明はありません。

7.2　データベース監査証跡連携機能の概要

ここでは，データベース監査証跡と連携した運用の作業について説明します。

データベース監査証跡連携機能は，データベースが提供するデータベース監査証跡の機能と連携する機能です。アプリケーションサーバ上のアプリケーション※1 またはサーバ内※2 からデータベースにアクセスした際に，アプリケーションサーバのシステムが持つ情報をデータベース監査証跡に出力できます。

注※1　J2EE アプリケーションまたはバッチアプリケーションが対象となります。

注※2　J2EE サーバ内またはバッチサーバ内が対象となります。

なお，データベース監査証跡と連携した運用では，データベース監査証跡，性能解析トレースおよびアプリケーションのユーザログを利用して，データベースの監査対象のテーブルにアクセスしたリクエストを特定できます。

7.3　データベース監査証跡との連携

この節では，データベースが提供するデータベース監査証跡の機能と連携する機能，およびアプリケーションサーバのシステムでできることについて説明します。

7.3.1　データベース監査証跡とは

データベース監査証跡とは，データベースに対する操作を記録したものです。監査者がデータベース監査証跡を調査することによって，監査対象のテーブルに決められた操作権限どおりに運用が行われているか，不正なアクセスがないかをチェックできます。データベース監査証跡の出力，取得，参照などをするための機能は，データベースが提供しています。データベース監査証跡の詳細については，データベースのマニュアルを参照してください。

7.3.2　データベース監査証跡と連携して取得できる情報

アプリケーションサーバのシステムでは，データベースが提供するデータベース監査証跡の機能と連携できます。この機能を**データベース監査証跡連携機能**といいます。この機能によって，次に示すアプリケーションまたはサーバ内からデータベースにアクセスした際に，アプリケーションサーバのシステムが持つ情報をデータベース監査証跡の情報の一部として出力させることができます。

- J2EE アプリケーションまたは J2EE サーバ内からデータベースにアクセスした際
- バッチアプリケーションまたはバッチサーバ内からデータベースにアクセスした際

データベース監査証跡に出力できるアプリケーションサーバのシステムの情報は，クライアントからのリクエストごとに割り振られるルートアプリケーション情報です。

データベース監査証跡，アプリケーションのユーザログ，および性能解析トレースを組み合わせて，データベースの監査対象のテーブルにアクセスしたリクエストのセッション ID，ログイン名を特定できます。

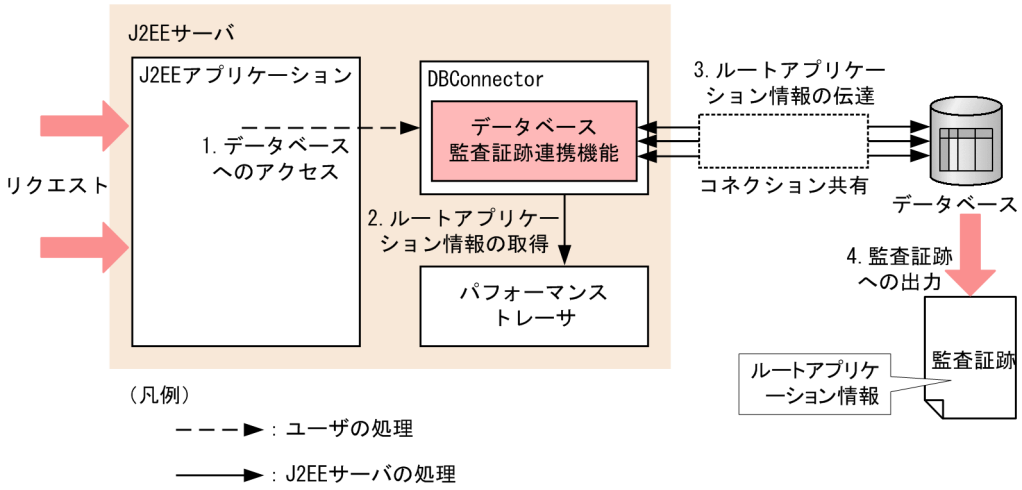
7.4　データベース監査証跡への情報の出力

この節では，データベース監査証跡連携機能を使用して，データベース監査証跡に出力する情報について説明します。

7.4.1　データベース監査証跡への出力の流れ

データベース監査証跡連携機能を使用して，データベース監査証跡にアプリケーションサーバのシステムの情報を出力する処理の流れを次の図に示します。この図では，J2EE アプリケーションからデータベースにアクセスした場合について示します。

図 7-1　データベース監査証跡への情報の出力の流れ



1. J2EE アプリケーションから DBConnector を経由してデータベースにアクセスする処理を実行します。
2. データベース監査証跡連携機能がルートアプリケーション情報をパフォーマンストレーサから取得します。
3. 取得したルートアプリケーション情報を，コネクションを通じてデータベースに渡します。
4. データベースで情報を処理し，データベース監査証跡の情報としてルートアプリケーション情報を出力します。

7.4.2　前提条件

データベース監査証跡連携機能を使用してデータベース監査証跡にアプリケーションサーバのシステムの情報を出力するには，次の表に示すデータベース，および DBConnector を使用している必要があります。

表 7-2　データベース監査証跡連携機能を使用できるデータベースと DBConnector

データベース	使用する DBConnector
HiRDB（バージョン 08-02 以降）	HiRDB Type4 JDBC Driver に対応した次の DB Connector • DBConnector_HiRDB_Type4_CP.rar • DBConnector_HiRDB_Type4_XA.rar • DBConnector_HiRDB_Type4_CP_Cosminexus_RM.rar • DBConnector_HiRDB_Type4_XA_Cosminexus_RM.rar

J2EE アプリケーションで，DBConnector からデータベースへのアクセスを実装している場合に，そのアクセスが Java EE およびアプリケーションサーバの仕様で DBConnector からのアクセスが許可されているメソッドまたはその延長の処理内であれば，データベース監査証跡連携機能が動作します。データベース監査証跡連携機能が動作することで，アプリケーションサーバのシステムの情報をデータベース監査証跡に出力できます。

次の場合はデータベース監査証跡連携機能が動作しないため，データベース監査証跡にアプリケーションサーバのシステムの情報は出力されません。

- 表 7-2 以外の DBConnector を使用した場合
- データベースが提供している JDBC ドライバから直接コネクションを取得してデータベースにアクセスした場合
- データベースが提供しているデータベース監査証跡機能の API を使用した場合

7.4.3　データベース監査証跡に出力する情報

データベース監査証跡連携機能を使用して，データベース監査証跡に出力できるアプリケーションサーバのシステムの情報は，ルートアプリケーション情報です。ルートアプリケーション情報以外にデータベース監査証跡に出力されている情報は，データベースが出力した情報です。

データベース監査証跡に出力されるルートアプリケーション情報は，データベースにアクセスしたリクエストのルートアプリケーション情報です。

また，リクエストの延長で J2EE サーバがデータベースにアクセスする次のような場合には，リクエストのルートアプリケーション情報が出力されます。

- Entity Bean の CMP を使用し，EJB コンテナ内からデータベースにアクセスした場合
- コネクションの自動クローズが実行された場合

ただし，リクエストと異なるルートアプリケーション情報が出力される場合，およびルートアプリケーション情報が出力されない場合もあります。それぞれの場合について説明します。

リクエストとは異なるルートアプリケーション情報が出力される場合

リクエストの延長ではないところでデータベースにアクセスした次のような場合，リクエストとは異なるルートアプリケーション情報が出力されます。

- Message-driven Bean のメッセージ受信側でデータベースにアクセスした場合
  ただし，Reliable Messaging の場合，メッセージ送信元のルートアプリケーション情報と関連づけることもできます。詳細については，マニュアル「Reliable Messaging」の「2.6.3　Message-driven Bean との連携」を参照してください。
- Timer Service のコールバックメソッド内でデータベースにアクセスした場合

ルートアプリケーション情報が出力されない場合

次の場合，ルートアプリケーション情報は出力されません。

- コネクション取得前にデータベースにアクセスした場合
- コネクションプールのウォーミングアップのコネクションを取得した場合
- リソースの接続テストをした場合

7.4.4　データベース監査証跡の出力個所

HiRDB のデータベース監査証跡表の次の列に情報が出力されます。

出力する列名称

PRODUCT_INFO_1

列の型

VARCHAR(255)

出力される情報

ルートアプリケーション情報

7.5　データベース監査証跡との連携機能の設定

データベース監査証跡連携機能を使用するには，次の設定が必要です。

- 実行環境での設定
- データベースサーバの設定

それぞれの設定について説明します。

7.5.1　実行環境での設定

データベース監査証跡との連携機能を使用する場合，J2EE サーバの設定が必要です。

J2EE サーバの設定は，簡易構築定義ファイルで実施します。データベース監査証跡との連携機能の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に指定します。

簡易構築定義ファイルでのデータベース監査証跡との連携機能の定義について次の表に示します。

表 7-3　簡易構築定義ファイルでのデータベース監査証跡との連携機能の定義

項目	指定するパラメタ	設定内容
データベース監査証跡機能に対応した JDBC ドライバの JAR ファイルのパスの設定	add.class.path	JDBC ドライバの JAR ファイル※1 を J2EE サーバのユーザクラスパスに追加します。 パラメタの設定値には，JAR ファイルのパスを指定します。add.class.path は，簡易構築定義ファイルの，J2EE サーバの拡張パラメタに設定します。
データベース監査証跡連携機能の設定※2	ejbserver.container.audit_trail.enabled	データベース監査証跡連携機能を使用するかどうかを指定します。プロパティを省略した場合，データベース監査証跡連携機能は無効になります。 パラメタは，J2EE サーバの JavaVM のシステムプロパティで定義します。

注※1

HiRDB と連携する場合に使用する JDBC ドライバは，HiRDB Type4 JDBC Driver です。

なお，データベース監査証跡連携機能の設定のパラメタで機能を使用する設定をした場合でも，データベース監査証跡連携機能に対応した JDBC ドライバの JAR ファイルが J2EE サーバのユーザクラスパスに追加されていないときは，KDJE54001-W のメッセージが出力され，データベース監査証跡連携機能は無効になります。この場合，J2EE サーバ起動時に KDJE54000-I は出力されません。

注※2

データベース監査証跡連携機能が有効かどうかは，J2EE サーバ起動時に確認できます。データベース監査証跡連携機能が有効な場合，J2EE サーバ起動時に KDJE54000-I のメッセージが出力され，無効な場合，メッセージは出力されません。このデータベース監査証跡連携機能の有効・無効を通知するメッセージは，J2EE サーバ起動後には出力されません。そのため，データベース監査証跡連携機能を使用する場合は，J2EE サーバ起動時に KDJE54000-I のメッセージが出力されることを確認してください。

簡易構築定義ファイル，および指定するパラメタの詳細は，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「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

データベース監査証跡連携機能を使用して，監査証跡にアプリケーションサーバのシステムの情報を出力できるデータベース以外のデータベースと接続する場合は，この機能を使用する設定にしないでください。アプリケーションサーバから JDBC ドライバに監査証跡を出力しようとする不要な処理が行われる場合があります。この場合，KDJE54000-I のメッセージが出力されても，データベース監査証跡連携機能は使用できません。

7.5.2　データベースサーバでの設定

データベース監査証跡連携機能を使用するには，データベースサーバでデータベース監査証跡の機能を有効にする必要があります。データベース監査証跡の機能を設定する際には，監査対象とするテーブルや操作を検討してください。なお，アプリケーションサーバのシステムでは，Component Container やそのほかの構成ソフトウェアで使用する独自のテーブルを生成しています。これらのテーブルを監査対象にするかどうかも検討してください。

データベースサーバの監査証跡の機能の設定方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

7.6　データベース監査証跡と連携した運用例

この節では，データベース監査証跡と連携した運用例について説明します。この運用例では，アプリケーションのユーザログ，データベース監査証跡，および性能解析トレースに出力されたルートアプリケーション情報を利用して，データベースの監査対象のテーブルにアクセスしたアプリケーションのログイン名やセッション ID を特定します。この場合，次の作業を実行します。

1. アプリケーションのユーザログを取得する

2. データベースの監査証跡情報を取得する

3. 性能解析トレースを取得する

4. 1.～3.で取得した情報を突き合わせて，ログイン名，セッション ID を特定する

それぞれの出力例や作業について説明します。

7.6.1　アプリケーションのユーザログの取得

あらかじめ J2EE アプリケーションやバッチアプリケーションでユーザログを取得するための実装をしておくと，次の情報がユーザログに出力されます。

- アプリケーションのログイン名
- ルートアプリケーション情報
- セッション ID

ここでは，J2EE アプリケーションのユーザログの実装例および出力例を示します。

実装例

ログイン名，ルートアプリケーション情報，およびセッション ID を J2EE アプリケーションのユーザログに出力する場合の実装例を示します。なお，ログイン名は Web コンテナの Basic 認証または Form 認証でログインしたユーザ ID を取得して出力します。

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import com.hitachi.software.ejb.application.prf.CprfTrace;

(中略)
public void doGet (HttpServletRequest req, HttpServletResponse res)
              throws ServletException, IOException {

  HttpSession ses = req.getSession();
  String sessionID = (ses != null) ? ses.getId() : null;  // セッションIDを取得
  String rootAPInfo = CprfTrace.getRootApInfo(); // ルートアプリケーション情報を取得
  String userID = req.getRemoteUser(); // J2EEアプリケーションのログイン名を取得
```

```
ユーザログ出力メソッド(sessionID,rootAPInfo,userID);
  :
```

出力例

J2EE アプリケーションのユーザログの出力例を示します。この例では，「セッション ID」，「ルートアプリケーション情報」，「ログイン名」の順で出力されています。

```
010D7AE670AEB55C022B6A4DA1C1E8B1CtEPmR9H,10.209.15.130/1234/0x0000000000000001,user1   :
```

7.6.2 データベースの監査証跡情報の取得

データベースが保持する監査証跡の情報を取得します。データベースの監査証跡表にデータ登録（監査証跡表へのデータロード）を行い，監査証跡表の「PRODUCT_INFO_1」列で監査対象のテーブルのルートアプリケーション情報を取得します。データベースの監査証跡の情報の取得方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。監査証跡表の出力例を次に示します。

出力例

HiRDB の監査証跡表の出力例を示します。

```
列    EVENT_TYPE EVENT_SUBTYPE OBJECT_NAME SQL_SOURCE PRODUCT_INFO_1
      ---------- ------------- ----------- ---------- ------------------------------------
行    ACS        INS           Table1      INSERT…    10.209.15.130/1234/0x0000000000000001
 :
```

この例で出力されている「SQL_SOURCE」列は，オプションを指定することで出力できます。オプションを指定して出力する列については，使用するシステムの要件に合わせて設定してください。オプションの設定方法，および監査証跡表の詳細については，マニュアル「HiRDB システム運用ガイド」を参照してください。

7.6.3 性能解析トレースの取得

性能解析トレースファイルを取得します。性能解析トレースの収集方法，および出力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 機能解説 保守／移行編」の「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。性能解析トレースの出力例を次に示します。

出力例

```
Event ,Date,Time ,Time(msec/usec/nsec)‥,RootAP IP     ,RootAP PID,RootAP CommonNo     .
0x8405,2007/xx/yy,12:00:02,000/000/000‥,10.209.15.130,1234       ,0x0000000000000001 .
0x8406,2007/xx/yy,12:00:02,000/000/000‥,10.209.15.130,1234       ,0x0000000000000001 .
   :
```

この例では，性能解析トレースのトレース取得レベル（PRF トレース取得レベル）を「詳細レベル」にしてセッション ID を出力しています。PRF トレース取得レベルについては，使用するシステムの要件に合わせて設定してください。また，PRF トレースレベルの変更方法については，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。

7.6.4　情報の特定

アプリケーションのユーザログ，データベースの監査証跡表，および性能解析トレースの情報を突き合わせて，監査対象のテーブルにアクセスしたアプリケーションのログイン名，およびセッション ID を特定します。

1. データベースの監査証跡表に出力されている実行日時「EXEC_DATE,EXEC_TIME,EXEC_TIME_MICRO」の列の時間と，性能解析トレースの実行時間とを突き合せて，監査対象のテーブルにアクセスしたリクエストのルートアプリケーション情報を特定します。

2. 1.で特定したルートアプリケーション情報と，アプリケーションのユーザログに出力されたルートアプリケーション情報とを突き合わせて，リクエストを実行したアプリケーションのログイン名およびセッション ID を特定します。

7.6.1，7.6.2，7.6.3 の例では，セッション ID「010D7AE670AEB55C022B6A4DA1C1E8B1CtEPmR9H」のリクエスト「10.209.15.130/1234/0x0000000000000001」（ルートアプリケーション情報）から，ユーザ ID「USER1」（ログイン名）で Bean の実行中（0x8405～0x8406）に，データベースのテーブル「Table1」に INSERT 文を実行したことが確認できます。

7.7　ルートアプリケーション情報をデータベースに設定する場合の注意事項

ルートアプリケーション情報をデータベースに設定するときに，データベースからエラーが返ってきた場合，KDJE54051-E のメッセージが出力されます。この場合，リクエストの処理は続行されますが，データベース監査証跡にルートアプリケーション情報は出力されません。

KDJE54051-E のメッセージが出力された場合は，Component Container およびデータベースの保守情報を取得して，保守員に連絡してください。また，KDJE54051-E のメッセージはデータベースから初めてエラーが返ってきた時だけ出力され，それ以降はエラーが発生しても出力されません。そのため，ログを調査する場合などには注意してください。

8 運用管理コマンドによる稼働情報の出力

この章では，運用管理コマンドによる稼働情報の出力について説明します。

8.1　この章の構成

この章の構成を次の表に示します。

表 8-1　この章の構成（運用管理コマンドによる稼働情報の出力）

分類	タイトル	参照先
解説	運用管理コマンドによる稼働情報の出力の概要	8.2
	サーバの稼働情報の出力方法	8.3
	稼働情報監視で確認できる項目	8.4

注　「実装」,「運用」,「設定」,「注意事項」について，この機能固有の説明はありません。

8.2　運用管理コマンドによる稼働情報の出力の概要

Management Server の運用管理コマンド（mngsvrutil）を使用して稼働情報を出力できます。稼働情報は，運用管理ドメイン単位でファイルに出力して確認できます。ファイルの形式は，CSV 形式または SNMP 連携用形式です。

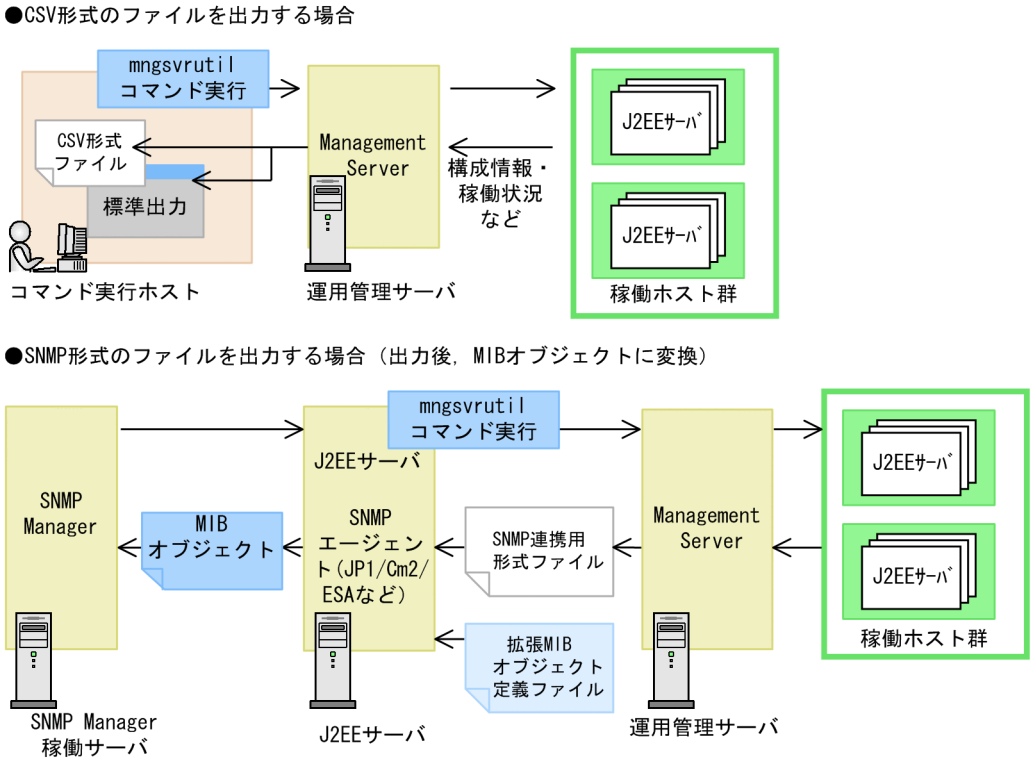
SNMP 連携用形式のファイルは，SNMP Manager 製品と連携する場合に使用できます。これらの製品で使用する場合は，SNMP 連携用形式のファイルを，JP1/Cm2/ESA※などを使用して MIB オブジェクトに変換して使用します。

注※

JP1/Cm2/ESA は UNIX の場合に使用します。

稼働情報のファイルへの出力の流れを次の図に示します。なお，稼働情報を MIB オブジェクトに変換する場合は，アプリケーションサーバで提供する拡張 MIB オブジェクト定義ファイルを使用します。

図 8-1　稼働情報のファイルへの出力の流れ



参考

なお，運用管理エージェントおよび Management Server の稼働状況については，運用管理コマンドまたは adminagentcheck コマンドを使用して確認できます。

ポイント

稼働情報の監視を行う場合，通常の運用では，稼働情報ファイルを使用した稼働監視を行ってください。稼働情報ファイルで取得できる情報より詳細な情報を取得したい場合だけ，運用管理コマンドを使用した稼働情報の監視を行ってください。稼働情報ファイルを使用した稼働情報の監視は，「3.2　稼働情報収集機能の概要」を参照してください。

8.3　サーバの稼働情報の出力方法

J2EE サーバ，またはバッチサーバの稼働情報は，運用管理コマンド（mngsvrutil）で監視できます。

稼働情報を監視するには，運用管理コマンド（mngsvrutil）にサブコマンド「get」を指定して実行します。これによって，J2EE サーバの稼働情報を標準出力に出力したり，CSV 形式または SNMP 連携用形式のファイルに出力したりできます。

「get」の引数に稼働情報を出力する対象を指定できます。

J2EE サーバの稼働情報を取得する場合の実行形式と実行例を次に示します。

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 <J2EEサーバ名> get <ドメイン名または取得するカテゴリ>
```

実行例

```
mngsvrutil -m mnghost -u user01 -p pw1 -t J2EEServer1 get j2eeContainer
```

mngsvrutil コマンド，そのサブコマンド，および取得できる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 コマンド編」の「mngsvrutil（Management Server の運用管理コマンド)」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 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。

8.4 稼働情報監視で確認できる項目

稼働情報監視では，J2EE サーバ，およびバッチサーバの稼働状況が確認できます。

稼働情報として監視できる項目を次の表に示します。

表 8-2 稼働情報監視で表示できる項目（J2EE サーバの場合）

監視の対象		監視できる項目
J2EE サーバ	J2EE コンテナ	• J2EE サーバ名 • Naming Service ホスト • Naming Service ポート番号 • コンテナ起動時刻
	EJB コンテナ	• コンテナ名
	Web コンテナ	**基本情報** • コンテナ名 • Web コンテナ起動時刻 • 管理用サーバのポート番号 • Web サーバとの通信に使用するポート番号 • Web サーバとの通信ソケットのバックログ設定上限値 • 稼働中（リクエスト処理の実行中）スレッド数設定上限値 • 最大同時実行スレッド数 • デフォルトの実行待ちキューサイズ **稼働情報** • Web サーバと Web コンテナとのコネクション数 • 稼働中スレッド数（HTTP サーバコネクタ） • Web コンテナ単位の実行待ちしているリクエスト数 • 同時実行可能スレッド数上限値 • 稼働スレッド数 • デフォルトの実行待ちリクエスト数 • デフォルトの実行待ちキューからあふれたリクエスト数 • 稼働中スレッド数（HTTP サーバ）
	JavaVM	**基本情報** • JavaVM 名 • JavaVM バージョン • JavaVM の最大メモリ使用量 **稼働情報** • JavaVM の空きメモリ量 • JavaVM の総メモリ容量
アプリケーション	J2EE アプリケーション	• アプリケーション名（display name） • 説明 • デプロイされた時刻

監視の対象		監視できる項目
	EJB アプリケーション	• EJB-JAR 名 • 説明
	Enterprise Bean（Stateful Session Bean）	**基本情報** • Enterprise Bean 名 • 説明 • Bean 名（内部識別用） • Home インタフェース名 • Local Home インタフェース名 • Component インタフェース名 • Local Component インタフェース名 • EJB クラス名 • トランザクションタイプ • 同時接続最大値 • 同時実行最大値 • 非活性セッションのタイムアウト • 実行中セッションのタイムアウト **稼働情報** • 現在の接続セッション数 • 実行中のセッション数 • 非活性セッション数 • 接続待ちセッション数 **Home インタフェース** • Home インタフェース名 • レスポンス • EJB メソッド実行時間 **Local Home インタフェース** • Home インタフェース名 • レスポンス • EJB メソッド実行時間 **Component インタフェース** • Component インタフェース名 • レスポンス • EJB メソッド実行時間 **Local Component インタフェース** • Component インタフェース名 • レスポンス • EJB メソッド実行時間
	Enterprise Bean（Stateless Session Bean）	**基本情報** • Enterprise Bean 名 • 説明 • Bean 名（内部識別用）

監視の対象		監視できる項目
		• Home インタフェース名 • Local Home インタフェース名 • Component インタフェース名 • Local Component インタフェース名 • EJB クラス名 • トランザクションタイプ • Bean インスタンスプール **稼働情報** • 接続待ちセッション数 • Bean インスタンスプール現在値 • 使用中の Session Bean 数 • 未使用の Session Bean 数 **Home インタフェース** Stateful Session Bean と同じ **Local Home インタフェース** Stateful Session Bean と同じ **Component インタフェース** Stateful Session Bean と同じ **Local Component インタフェース** Stateful Session Bean と同じ
	Enterprise Bean（Entity Bean）	**基本情報** • Enterprise Bean 名 • 説明 • Bean 名（内部識別用） • Home インタフェース名 • Local Home インタフェース名 • Component インタフェース名 • Local Component インタフェース名 • EJB クラス名 • EntityBean の永続化タイプ • EntityBean のキャッシュモデル • 同時接続最大値 • Bean インスタンスプール • 接続タイムアウト **稼働情報** • 現在の接続セッション数 • 接続待ちセッション数 • Bean インスタンスプール現在値 • 使用中の Entity Bean 数 • 未使用の Entity Bean 数 **Home インタフェース** Stateful Session Bean と同じ

監視の対象		監視できる項目
		Local Home インタフェース Stateful Session Bean と同じ Component インタフェース Stateful Session Bean と同じ Local Component インタフェース Stateful Session Bean と同じ
	Enterprise Bean （Message-driven Bean）	基本情報 • Enterprise Bean 名 • 説明 • Bean 名（内部識別用） • EJB クラス名 • トランザクションタイプ • デスティネーションタイプ※1 • Bean インスタンスプール 稼働情報 • 現在の接続セッション数 • Bean インスタンスプール現在値 • EJB メソッド実行時間
	Web アプリケーション	基本情報 • コンテキストルート • 占有スレッド数 • 最大同時実行スレッド数 • Web アプリケーション単位の実行待ちキューサイズ 稼働情報 • 有効なセッション数 • 同時実行可能スレッド数上限値 • 稼働スレッド数 • Web アプリケーション単位の実行待ちリクエスト数 • Web アプリケーション単位の実行待ちキューからあふれたリクエスト数
	サーブレット	• サーブレット名 • サーブレットの実装クラス名 • サーブレット実行回数 • サーブレット失敗回数 • サーブレット実行時間 • 出力データサイズ
	URL	• URL • URL 呼び出し回数 • URL 呼び出し失敗回数 • URL 実行時間 • 出力データサイズ

監視の対象		監視できる項目
リソース	データソース（SimpleJTA）	**基本情報** • リソース名 • リソースタイプ • 説明 • 認証タイプ • ログインタイムアウト • ユーザ ID • コネクションプール **稼働情報** • リソース名 • プール現在値（総数） • 使用中のコネクション数 • 未使用のコネクション数
	データソース（FullJTA）	**基本情報** • リソース名 • リソースタイプ • 説明 • ログインタイムアウト • ユーザ ID • コネクションプール **稼働情報** • リソース名 • プール現在値 • 使用中のコネクション数 • 未使用のコネクション数 • getConnection()メソッドの実行時間 • getXAConnection()メソッドの実行時間 • getConnection()メソッドの失敗回数 • Connection で FATAL エラーが発生した回数
	リソースアダプタ	**基本情報** • リソース名 • リソースタイプ • 説明 • リソースアダプタ提供ベンダ名 • 準拠する JCA 仕様のバージョン • リソースアダプタのバージョン • 接続先 EIS のタイプ • ConnectionFactory のインタフェース名 • ConnectionFactory の実装クラス名 • ManagedConnectionFactory の実装クラス名 • Connection のインタフェース名 • Connection の実装クラス名

監視の対象		監視できる項目
		• トランザクションサポートモデル • 設定プロパティ情報 • ユーザ ID • コネクションプール※2 **稼働情報** • リソース名 • プール現在値（総数）※2 • 使用中のコネクション数※2 • 未使用のコネクション数※2 • ManagedConnectionFactory の createManagedConnection()メソッドの実行回数※2 • ManagedConnection の getConnection()メソッドの実行回数※2 • ManagedConnection の cleanup()メソッドの実行回数※2 • ManagedConnection の destroy()メソッドの実行回数※2 • ConnectionManager の allocateConnection()メソッドの実行時間※2 • ManagedConnectionFactory の createManagedConnection()メソッドの実行時間※2 • ConnectionManager の allocateConnection()メソッドの失敗回数※2 • ManagedConnection で FATAL エラーが発生した回数※2
サービス	トランザクション	**基本情報** • サービス名 • サービスタイプ • トランザクションタイムアウトデフォルト値 **稼働情報** • アクティブトランザクション数 • 平均トランザクション時間

注　使用する機能（最大同時実行スレッド数制御機能を使用するかどうか），オプションの設定，使用するリソースの違いなどによっては表示されない項目があります。

注※1　Connector 1.5 仕様に準拠したリソースアダプタを使用した場合は，「Other」が出力されます。

注※2　Reliable Messaging を使用している場合，これらの項目は Reliable Messaging 側の稼働情報として出力されます。Management Server の稼働情報監視では確認できません。

表 8-3　稼働情報監視で表示できる項目（バッチサーバの場合）

監視の対象		監視できる項目
バッチサーバ	J2EE コンテナ	• バッチサーバ名 • Naming Service ホスト • Naming Service ポート番号

監視の対象		監視できる項目
		• コンテナ起動時刻
	JavaVM	**基本情報** • JavaVM 名 • JavaVM バージョン • JavaVM の最大メモリ使用量 **稼働情報** • JavaVM の空きメモリ量 • JavaVM の総メモリ容量
リソース	データソース（SimpleJTA）	**基本情報** • リソース名 • リソースタイプ • 説明 • 認証タイプ • ログインタイムアウト • ユーザ ID • コネクションプール **稼働情報** • リソース名 • プール現在値（総数） • 使用中のコネクション数 • 未使用のコネクション数
	データソース（FullJTA）	**基本情報** • リソース名 • リソースタイプ • 説明 • ログインタイムアウト • ユーザ ID • コネクションプール **稼働情報** • リソース名 • プール現在値 • 使用中のコネクション数 • 未使用のコネクション数 • getConnection()メソッドの実行時間 • getXAConnection()メソッドの実行時間 • getConnection()メソッドの失敗回数 • Connection で FATAL エラーが発生した回数
	リソースアダプタ	**基本情報** • リソース名 • リソースタイプ • 説明

監視の対象		監視できる項目
		• リソースアダプタ提供ベンダ名 • 準拠する JCA 仕様のバージョン • リソースアダプタのバージョン • 接続先 EIS のタイプ • ConnectionFactory のインタフェース名 • ConnectionFactory の実装クラス名 • ManagedConnectionFactory の実装クラス名 • Connection のインタフェース名 • Connection の実装クラス名 • トランザクションサポートモデル • 設定プロパティ情報 • ユーザ ID • コネクションプール※ **稼働情報** • リソース名 • プール現在値（総数）※ • 使用中のコネクション数※ • 未使用のコネクション数※ • ManagedConnectionFactory の createManagedConnection()メソッドの実行回数※ • ManagedConnection の getConnection()メソッドの実行回数※ • ManagedConnection の cleanup()メソッドの実行回数※ • ManagedConnection の destroy()メソッドの実行回数※ • ConnectionManager の allocateConnection()メソッドの実行時間※ • ManagedConnectionFactory の createManagedConnection()メソッドの実行時間※ • ConnectionManager の allocateConnection()メソッドの失敗回数※ • ManagedConnection で FATAL エラーが発生した回数※
サービス	トランザクション	**基本情報** • サービス名 • サービスタイプ • トランザクションタイムアウトデフォルト値 **稼働情報** • アクティブトランザクション数 • 平均トランザクション時間

注　使用する機能（最大同時実行スレッド数制御機能を使用するかどうか），オプションの設定，使用するリソースの違いなどによっては表示されない項目があります。

注※　Reliable Messaging を使用している場合，これらの項目は Reliable Messaging 側の稼働情報として出力されます。Management Server の稼働情報監視では確認できません。

ポイント

統計情報のサンプリング時間の設定

稼働情報を表示する場合，統計情報のサンプリング時間を指定してください。

サンプリング時間を指定すると，画面を表示した時刻または更新した時刻からさかのぼって，サンプリング時間（これを**N 秒**といいます）に指定した時間内のデータを統計情報として表示できます。「N 秒ピーク」や「N 秒平均値」とはこの統計情報です。また，最大値，最小値などは，サンプリングを開始した以降のデータの中から値が抽出されます。

運用管理コマンド（mngsvrutil）にサブコマンド「set」を指定してサンプリング時間を設定できます。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「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。

9 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

この章では，Management イベントによる処理の自動実行について説明します。

9.1　この章の構成

この章の構成を次の表に示します。

表 9-1　この章の構成（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）

分類	タイトル	参照先
解説	Management イベントの通知と Management アクションの概要	9.2
	Management アクションの実行制御とは	9.3
設定	Management イベントによる処理の自動実行の設定	9.4

注　「実装」,「運用」,「注意事項」について，この機能固有の説明はありません。

9.2 Management イベントの通知と Management アクションの概要

Management イベントとは，運用管理ドメイン内の J2EE サーバ，またはバッチサーバで発生する障害やリソース枯渇などの事象を Management Server に通知するためのイベントです。J2EE サーバ，またはバッチサーバが稼働中に出力するメッセージを契機にして Management イベントを発行できます。Management Server 側では，Management イベントが通知されたときの動作を定義しておくことで，Management イベントが発生すると自動的にアクションを実行できるようになります。このアクションを**Management アクション**といいます。

注意事項

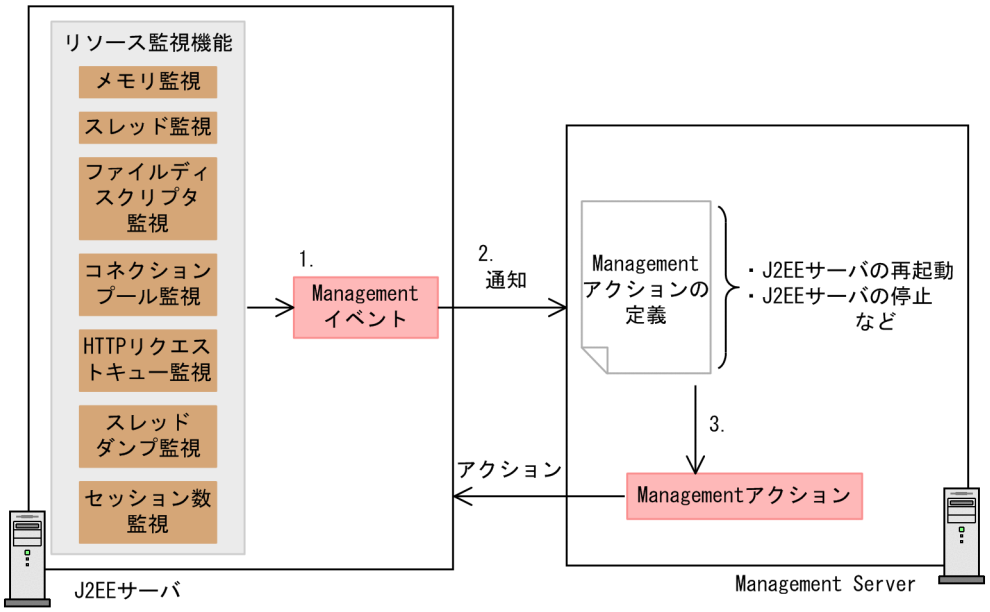
J2EE サーバ，およびバッチサーバが出力するメッセージのうち，次に示すメッセージは Management イベントの発行対象外となります。

- KDJE90001-E
- KDJE90002-E
- KDJE90003-E
- KDJE90005-W
- KDJE90006-W
- KDJE90009-W

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)」を参照してください。

Management イベントが発行されてから，Management Server で Management アクションが実行されるまでの流れを次の図に示します。

図 9-1　Management イベントと Management アクション



この例では，リソース枯渇監視機能で設定したしきい値を超えた場合に出力されるメッセージを契機に，Management イベントを実行しています。図中の処理の流れについて説明します。

1. 監視しているリソースで設定しているしきい値を超えると，Management イベントが発行されます。
2. Mnagement イベントは，Management Server に通知されます。
3. Management Server 内で定義された Management アクションの定義に従って，処理が自動的に実行されます。

   Management アクションの定義では，J2EE サーバまたはバッチサーバから挙がる Management イベントに対応したアクションを定義します。Management アクションはあらかじめ定義しておく必要があります。

なお，リソース枯渇監視機能を使用してメモリ監視をしている場合，FullGC の予兆検知ができます。メモリ監視中に，FullGC の予兆を検知すると，その情報は Management イベントとして Management Server に通知されます。このとき，Management イベントで FullGC が通知されたときの動作として，J2EE サーバまたはバッチサーバでサービスを閉塞して再起動するアクションが定義されていると，J2EE サーバまたはバッチサーバに対してこのアクションが自動的に実行されます。これによって，リクエスト処理の停止を回避できるようになります。

9.3 Management アクションの実行制御とは

ここでは，Management アクションの実行制御について説明します。

同一の Management アクションの実行を一定時間抑止したり，同時実行数の上限を設定したりすることによって，Management アクションの実行を制御できます。これによって，同一の Management アクションが重複して実行されたり，Management アクションの実行が集中したりすることを防ぎます。

なお，Management アクションの実行とは，Management アクションとして指定されたコマンドを実行してから，コマンドが終了またはタイムアウトするまでのことを指します。

Management アクションの実行制御方式を次の表に示します。

表 9-2 Management アクションの実行制御方式

制御方式	説明
抑止時間制御	Management アクションの実行後に，同一の Management アクションが実行されるのを一定時間抑止します。これによって，一定時間内に発生する Management イベントに対して実行される，Management アクションが集約されます。
同時実行数制御	同一の Management アクションの同時実行数を制限します。これによって，Management アクション実行中に，同一の Management アクションが過度に重複して実行されることを防ぎます。同時実行数は Management アクション ID ごとに適用されます。

なお，上記二つの方式を同時に指定した場合，同時実行数の上限値に達していないときでも，抑止時間内の Management アクションの実行は抑止されます。

Management アクションの制御方式について例を使って説明します。

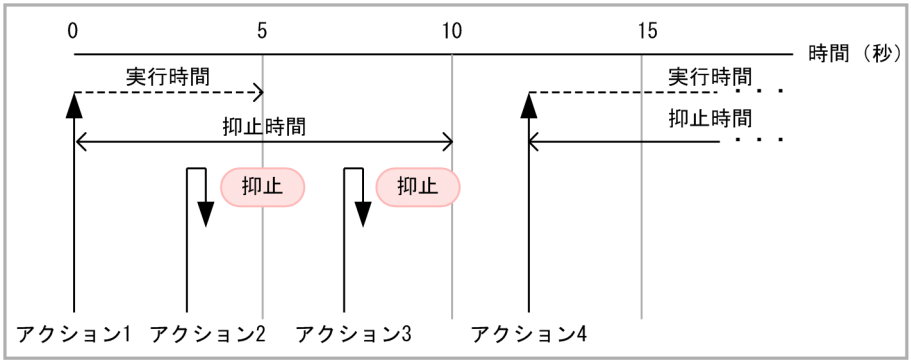
- **設定例 1**

  次の内容が設定されていることとします。

  - Management アクションの実行時間：5 秒
  - 抑止時間：10 秒
  - 同時実行数の上限値：2

  Management アクションの制御例を次の図に示します。

図 9-2　Management アクションの制御例 1



（凡例）
：Managementアクションの実行時間
：Managementアクションの抑止時間
：Managementアクション

この図の例では，アクション 1 のあと，10 秒間は同一アクションの実行を抑止するため，アクション 2 およびアクション 3 の実行は抑止されます。抑止時間が過ぎたあと，アクション 4 が実行できるようになります。

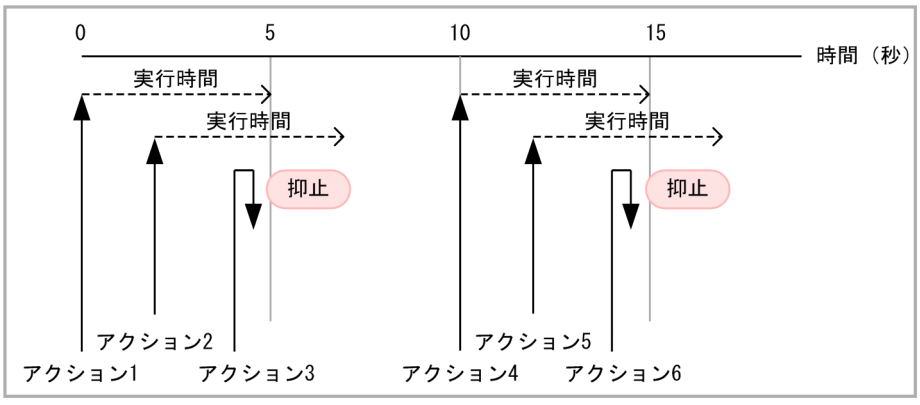
- **設定例 2**

  次の内容が設定されていることとします。

  - Management アクションの実行時間：5 秒
  - 抑止時間：0 秒（抑止しない）
  - 同時実行数の上限値：2

  Management アクションの制御例を次の図に示します。

図 9-3　Management アクションの制御例 2



（凡例）
：Managementアクションの実行時間
：Managementアクション

この図の例では，抑止時間の制限がないため，アクション 1 の実行後に，アクション 2 が実行されます。ただし，この時点で同一 Management アクションの同時実行数の上限値に達しているため，アクション 3 の実行は抑止されます。アクション 1 の実行が終了したあと，アクション 4 を実行できるようになります。アクション 5，6 については，アクション 2，3 と同様になります。

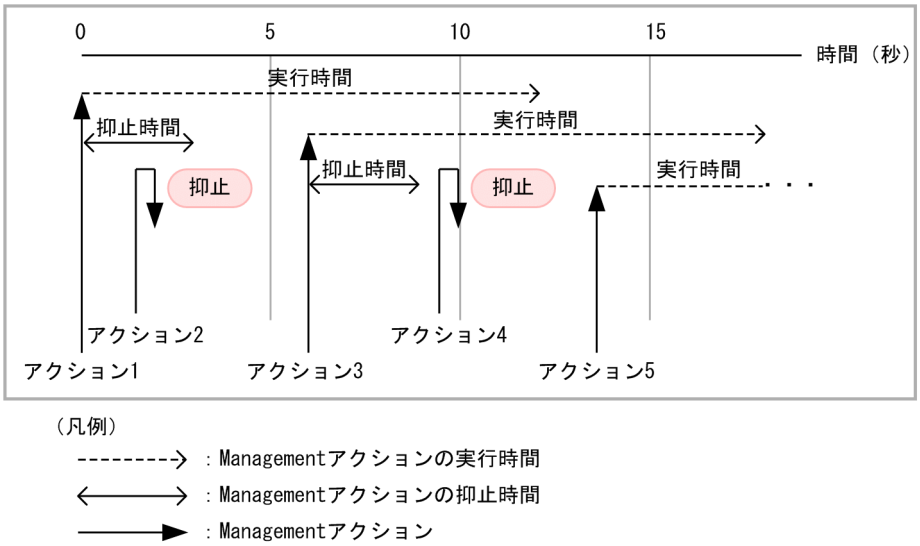
- **設定例 3**

  次の内容が設定されていることとします。

  - Management アクションの実行時間：12 秒
  - 抑止時間：3 秒
  - 同時実行数の上限値：2

  Management アクションの制御例を次の図に示します。

図 9-4　Management アクションの制御例 3



この図の例では，アクション 1 の実行後，3 秒間は同一の Management アクションの実行が抑止されるため，アクション 2 の実行は抑止されます。アクション 3 の実行後，同一 Management アクションの同時実行数の上限値に達したため，アクション 4 の実行が抑止されます。アクション 1 の実行終了後，アクション 5 を実行できようになります。

9.4　Management イベントによる処理の自動実行の設定

この節では，Management イベントによる処理の自動実行を行うための設定方法について説明します。

Management イベントは，運用管理ドメイン内の J2EE サーバ，またはバッチサーバで発生する障害やリソース枯渇などの事象を Management Server に通知するためのイベントです。J2EE サーバ，およびバッチサーバが稼働中に出力するすべてのメッセージを契機にして Management イベントを発行できます。Management Server 側では，Management イベントが通知されたときの動作を定義しておくことで，Management イベントが発生すると自動的にアクションを実行できるようになります。このアクションを**Management アクション**といいます。

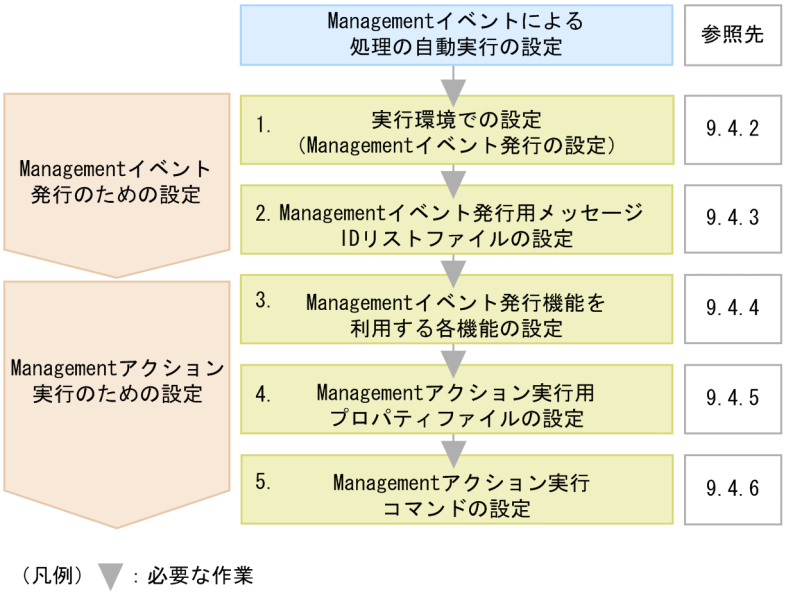
例えば，リソース枯渇監視機能を使用してメモリの使用状態を監視している場合に，しきい値を超えたらアラートを発生させてメッセージを出力し，FullGC が発生する前に，Management イベントを通知するようにします。また，J2EE サーバまたはバッチサーバでサービスを閉塞して再起動するという Management アクションを定義しておくことで，FullGC の予兆を検知し，自動的にアクションが実行されて，リクエスト処理の停止を回避できるようになります。

Management イベントを使用して障害などの事象を検知し，Management イベントに対応する動作を Management アクションとして定義しておくことで，障害を回避するための対処など，発生した事象に対する処理を自動的に実行できるようになります。

9.4.1　Management イベントによる処理の自動実行の設定手順

Management イベントによる処理の自動実行の設定手順を次の図に示します。

図 9-5　Management イベントによる処理の自動実行の設定手順



図中の 1.～5.について説明します。

1. 実行環境で Management イベント発行の設定をします。

   Management イベントの発行を有効にして，Management イベント発行時の動作を設定します。詳細については，「9.4.2　実行環境での設定」を参照してください。

2. Management イベント発行用メッセージ ID リストファイルを設定します。

   リストファイルで，Management イベント発行対象となるメッセージを指定します。詳細については，「9.4.3　Management イベント発行用メッセージ ID リストファイルの設定」を参照してください。

3. Management イベント発行機能を利用する各機能の設定をします。

   Management イベント発行機能を利用して，Management Server に Management イベントを通知する機能の設定をします。

   次に，設定した Management イベントに対応する Management アクションを実行するための設定をします。詳細については，「9.4.4　Management イベント発行機能を利用する各機能の設定」を参照してください。

4. Management アクション実行用プロパティファイルを設定します。

   プロパティファイルで，Management アクションの定義，メッセージ ID と Management アクションとのマッピングなどをします。詳細については，「9.4.5　Management アクション実行用プロパティファイルの設定」を参照してください。

5. Management アクション実行コマンドの設定をします。

   Management アクションとして実行するコマンドを記述するコマンドファイル（バッチファイルまたはシェルスクリプトファイル）を作成します。詳細については，「9.4.6　Management アクション実行コマンドの設定」を参照してください。

9.4.2　実行環境での設定

Management イベントを発行する場合，J2EE サーバの設定が必要です。

(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。Management イベントの発行の定義は，簡易構築定義ファイルの論理 J2EE サーバ（j2ee-server）の<configuration>タグ内に指定します。

簡易構築定義ファイルでの Management イベントの発行の定義について次の表に示します。

表 9-3　簡易構築定義ファイルでの Management イベントの発行の定義（J2EE サーバ）

項目	指定するパラメタ	設定内容
Management イベント発行の有効化	ejbserver.manager.agent.MEventAgent.enabled	Management イベントの発行を有効にするかどうかを指定します。Management イベントを発行するには，「true」を指定します。

項目	指定するパラメタ	設定内容
Management イベント発行時の動作の設定	manager.mevent.send.timeout	Management イベントの送信タイムアウトの時間（秒）を指定します。
	manager.mevent.retry.limit	Management イベントの再送期限を指定します。
	manager.mevent.retry.interval	Management イベントの再送間隔（秒）を指定します。
	manager.mevent.send.max	Management イベントの同時発行最大数を指定します。
	manager.mevent.message_id.list	メッセージ ID リストファイルのパスを指定します。
	manager.mevent.sender.bind.host	ローカルアドレスの固定の有無を指定します。

簡易構築定義ファイル，および指定するパラメタの詳細は，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「4.3　簡易構築定義ファイル」を参照してください。

9.4.3　Management イベント発行用メッセージ ID リストファイルの設定

Management イベントを発行したいメッセージのメッセージ ID は，Management イベント発行用メッセージ ID リストファイルに記述します。Management イベント発行用メッセージ ID リスト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「8.2.12 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。

(1)　ファイルの指定内容

Management イベント発行用メッセージ ID リストファイルに，Management イベントを発行したいメッセージのメッセージ ID を記述してください。

なお，Management イベントを発行できるのは，J2EE サーバ，およびバッチサーバが稼働中に出力するメッセージです。起動処理中，停止処理中のメッセージおよびユーザログは，Management イベント発行の対象外です。

Management イベント発行用メッセージ ID リストファイルの設定を省略すると，デフォルトのメッセージ ID で Management イベントを発行します。デフォルトの設定のままで利用する場合には，Management イベント発行用メッセージ ID リストファイルは設定不要です。デフォルトのメッセージ ID については，Management イベント発行用メッセージ ID リストファイルの説明を参照してください。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。

(2) ファイルの格納場所

Management イベント発行用メッセージ ID リストファイルのサンプルは，次の場所に格納されています。サンプルのファイルをコピーして，リストファイルを作成してください。

- Windows の場合

  <Application Server のインストールディレクトリ>¥manager¥config¥templates¥mevent.midlist.conf

- UNIX の場合

  /opt/Cosminexus/manager/config/templates/mevent.midlist.conf

作成したリストファイルのパスは，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，manager.mevent.message_id.list パラメタでリストファイルのパスを指定してください。

(3) ファイルの作成例

Management イベント発行用メッセージ ID リストファイルの作成例を次に示します。

```
# Monitoring of resources
# : Status of memory(Java Heap)
KDJE34500-W

# : Number of file descriptors
KDJE34520-W

# : Number of threads
KDJE34540-W
# : Number of thread dump files
-KDJE34580-W
KDJE34581-E

# : Number of HTTP requests in queue
-KDJE34621-W

# : Number of HTTP sessions
KDJE34640-W

# : Status of connection pool
-KDJE34660-W
KDJE34661-W

# Monitoring of execution time of user program
KDJE52702-W
KDJE52703-W
KDJE52705-W
KDJE52713-E
```

メッセージ ID の前に「+」を記述した場合，そのメッセージは Management イベントを発行します。「-」を記述した場合，そのメッセージは Management イベントを発行しません。「+」または「-」を省略した場合，そのメッセージは Management イベントを発行します。

9.4.4　Management イベント発行機能を利用する各機能の設定

Management イベントを発行する機能の設定をしてください。

ここでは，例として，次の機能で Management イベントを発行させるために必要な設定について説明します。

- リソース枯渇監視機能
- J2EE アプリケーション実行時間の監視機能

(1)　リソース枯渇監視の設定

リソースの監視間隔やしきい値を指定してリソースを監視し，しきい値を超えた場合に Management イベントが発行されるように設定します。

リソース枯渇監視の機能および設定については，「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。

(2)　J2EE アプリケーション実行時間の監視の設定

リクエストの実行時間を監視して，無限ループなどの障害が発生した場合に Management イベントが発行されるようにします。J2EE アプリケーションの実行時間の監視機能および設定については，「5.3　J2EE アプリケーションの実行時間の監視とキャンセル」を参照してください。

なお，メソッドタイムアウト機能の対象となる，Web アプリケーションのリクエスト処理や EJB のメソッド呼び出し処理にもタイムアウトを設定します。設定方法については，「5.3.8　実装時の注意事項」を参照してください。

9.4.5　Management アクション実行用プロパティファイルの設定

Management アクションの定義，メッセージ ID と Management アクションとのマッピングなどは，**Management アクション実行用プロパティファイル（maction.properties）**に定義します。maction.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「8.2.10　maction.properties（Management アクション実行用プロパティファイル)」を参照してください。

(1) ファイルの格納場所

maction.properties 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¥maction.properties
- UNIX の場合
  /opt/Cosminexus/manager/config/maction.properties

(2) ファイルの設定例

Windows の場合の Management アクション実行用プロパティファイルの設定例を次に示します。

```
# Managementアクションの定義
maction.restart.command=c¥:¥¥tmp¥¥command1.bat
maction.restart.timeout=12
maction.restart.timeout.forced_stop=true
maction.restart.exclusive_time=60
maction.restart.max_executable_actions=1

# メッセージIDとManagementアクションのマッピング
maction.message.KDJE11111-E.mactions=restart
maction.message.KDJE22222-E.mactions=restart

# 論理サーバとManagementアクションのマッピング
maction.server.j2ee1.mactions=restart
maction.server.j2ee2.mactions=restart
maction.server.j2eeClstr1.mactions=restart
```

この設定例では，Management アクションを識別する ID（Management アクション ID）として，「restart」を定義しています。「restart」の Management アクションの動作および設定について説明します。

- Management アクション実行コマンドとして「command1.bat」というコマンドファイルを実行します。
- Management アクション実行コマンドのタイムアウトは，「12」秒とします。
- Management アクション実行コマンドのタイムアウトが発生すると，コマンドを強制終了します。
- Management アクションの抑止時間は「60」秒，Management アクションの同時実行数は，「1」とします。※
- J2EE サーバ「j2ee1」，「j2ee2」および「j2eeClstr1」から，メッセージ「KDJE11111-E」と「KDJE22222-E」が出力された場合に，この Management アクションを実行します。

注※

Management アクションは，Management アクション ID で区別され，複数のサーバや異なるメッセージ ID に対して同一の Management アクションを実行できます。抑止時間や同時実行数を設定して Management アクションの実行を制御することで，Management アクションの集約や重複実行の

抑止ができます。実行制御については，「9.3　Management アクションの実行制御とは」を参照してください。

(3)　注意事項

Management アクション実行用プロパティファイルに関する注意事項を次に示します。

- プロパティの優先順位

  次に示すキーで指定する論理サーバやクラスタには，「J2EE サーバ ＜ J2EE サーバクラスタ ＜ サービスユニット ＜ 物理ティア」という包含関係があります。

  - maction.server.<論理サーバ名>.mactions
  - maction.unit.<Web システム名>.<サービスユニット名>.mactions
  - maction.tier.<Web システム名>.<物理ティア種別名>.mactions

  このため，J2EE サーバクラスタとその J2EE サーバクラスタの構成要素となる J2EE サーバのように，包含関係を持つ論理サーバそれぞれに対して別々の Management アクションを定義した場合，次に示す優先順位で，どれか一つの Management アクションを実行します。

  1. Management イベントを発行した J2EE サーバ
  2. Management イベントを発行した J2EE サーバを含む J2EE サーバクラスタ
  3. Management イベントを発行した J2EE サーバを含むサービスユニット
  4. Management イベントを発行した J2EE サーバを含む物理ティア

  なお，Web システム，サービスユニットや物理ティアは，Smart Composer 機能を使用してシステムを構築する場合の概念です。Smart Composer 機能については，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「1.1.3　Smart Composer 機能とは」を参照してください。

- Management アクションの指定順序

  次に示す二つのキーで Management アクションの指定順序が異なる場合は，maction.message.<メッセージ ID>.mactions キーでの指定順序を優先します。

  - maction.message.<メッセージ ID>.mactions
    メッセージ ID と Management アクションとをマッピングするキーです。
  - maction.server.<論理サーバ名>.mactions
    論理サーバと Management アクションとをマッピングするキーです。

  （例）

  この例では，「act1」が優先されます。

  maction.message.KDJE99999-E.mactions=act1,act2

  maction.server.J2EE01.mactions=act3,act2,act1

9.4.6 Management アクション実行コマンドの設定

Management アクションとして実行するコマンドは，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。必要に応じて，コマンドファイルを作成してください。なお，バッチサーバの場合，CTM によるリクエストのスケジューリングは使用できないため，CTM に関する記述は該当しません。

ここでは，コマンドファイルで使用できる環境変数と，コマンドファイルのサンプルについて説明します。

(1) コマンドファイルで使用できる環境変数

コマンドファイルでは，次の表に示す環境変数を使用できます。

表 9-4 Management アクション実行コマンドのコマンドファイルで使用できる環境変数

環境変数	説明
COSMI_MNG_MACT_LSNAME	Management イベントを発行した論理サーバ名。
COSMI_MNG_MACT_HOST	Management イベントを発行した論理サーバのホスト名。
COSMI_MNG_MACT_MSG_ID	Management イベントで通知されたメッセージ ID。
COSMI_MNG_MACT_MSG_TEXT	Management イベントで通知されたメッセージテキスト。
COSMI_MNG_MACT_CTM	Management イベントを発行した論理サーバに設定された論理 CTM 名。
COSMI_MNG_MACT_WEBSYSTEM	Management イベントを発行した論理サーバが所属する Web システム名。
COSMI_MNG_MACT_UNIT	Management イベントを発行した論理サーバが所属するサービスユニット名。
COSMI_MNG_MACT_TIER	Management イベントを発行した論理サーバが所属する物理ティア種別。
COSMI_MNG_MACT_OPT<N>	Management イベントで通知されたオプション文字列。オプション文字列としてメッセージの埋め字が使用されます。N は，0 以上の整数であり，オプション文字列の 1 番目を表す環境変数は「COSMI_MNG_MACT_OPT0」となります。 設定される文字列の詳細については，Management イベント発行の対象となるメッセージのメッセージテキスト中の可変値に表示される情報を参照してください。
COSMI_MNG_MACT_MNGSVR_PORT	Management Server の HTTP ポート番号。
COSMI_MNG_MACT_RNAME	Management イベントを発行した論理サーバの実サーバ名。
COSMI_MNG_MACT_NS_HOSTPORT	Management イベントを発行した論理サーバが使用するネーミングサービスのホスト名とポート番号。例えば，ホスト名が「HostA」でポート番号が「900」の場合には，「HostA:900」となります。

(2) コマンドファイルのサンプル

コマンドファイルのサンプルが提供されていますので，サンプルを参考にしてコマンドファイルを作成してください。サンプ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¥examples¥maction
- UNIX の場合
  /opt/Cosminexus/manager/examples/maction

なお，サンプルを使用する場合は，mngsvrutil コマンドの引数（-m，-u，-p オプションなど）やログファイルの格納先を環境に合わせて適宜変更してください。

(a) Management イベント発行サーバを再起動するサンプル

サンプルのファイル名

- Windows の場合：mActionSample_restartServer.bat
- UNIX の場合：mActionSample_restartServer.sh

サンプルの動作

Management イベントを発行した論理サーバを再起動します。

このサンプルは，リソース枯渇監視機能で Management イベントを発行する場合に使用できます。

1. リソース枯渇監視機能では，JavaVM でのメモリの使用状況を監視し，設定されているしきい値を超えて，FullGC が発生しそうな場合にメッセージを出力します。
2. Management イベント発行機能は，出力されたメッセージに対応した Management イベントを発行したあと Management アクションを実行し，Management イベントを発行した J2EE サーバを再起動します。

Management イベントを発行した J2EE サーバを停止することで，J2EE アプリケーションを閉塞できます。また，J2EE サーバが CTM によってリクエストを負荷分散させている場合は，J2EE サーバを停止することで，新規リクエストをほかの J2EE サーバに分散できます。

Management イベント発行サーバを再起動するサンプルを次に示します。

- Windows の場合

```
       （省略）
         :
rem Management action sample for restart server.

setlocal

set LSNAME=%COSMI_MNG_MACT_LSNAME%
set MSGID=%COSMI_MNG_MACT_MSG_ID%

set CJCLDELLOG=%COSMINEXUS_HOME%¥CC¥client¥bin¥cjcldellog.bat
set LOG_ROOT_DIR=%SystemDrive%¥<MyLogDir>
```

```
if not exist "%LOG_ROOT_DIR%" mkdir "%LOG_ROOT_DIR%"
call "%CJCLDELLOG%" -t 30d -f "%LOG_ROOT_DIR%"
set LOG_DIR=%LOG_ROOT_DIR%¥%date:/=%
if not exist "%LOG_DIR%" mkdir "%LOG_DIR%"
set LOG=%LOG_DIR%¥%time::=%_%LSNAME%_%MSGID%.txt

echo %0 > "%LOG%"

set MNGSVR=localhost:%COSMI_MNG_MACT_MNGSVR_PORT%
set UID=<User-id>
set PWD=<Password>

echo mngsvrutil.exe -m %MNGSVR% -u %UID% -p %PWD% -t %LSNAME% -s stop server >> "%LOG%" 2
>&1
mngsvrutil.exe -m %MNGSVR% -u %UID% -p %PWD% -t %LSNAME% -s stop server >> "%LOG%" 2>&1
set RET=%ERRORLEVEL%
echo %RET% >> "%LOG%"

if not %RET% == 0 goto END
echo mngsvrutil.exe -m %MNGSVR% -u %UID% -p %PWD% -t %LSNAME% -s start server >> "%LOG%"
2>&1
mngsvrutil.exe -m %MNGSVR% -u %UID% -p %PWD% -t %LSNAME% -s start server >> "%LOG%" 2>&1
set RET=%ERRORLEVEL%
echo %RET% >> "%LOG%"

:END
exit %RET%
```

- UNIX の場合

```
       (省略)
         :
# Management action sample for restart server.

LSNAME=${COSMI_MNG_MACT_LSNAME}
MSGID=${COSMI_MNG_MACT_MSG_ID}

LOG_ROOT_DIR=/tmp/<MyLogDir>
if [ ! -d ${LOG_ROOT_DIR} ]; then
  /bin/mkdir ${LOG_ROOT_DIR}
fi
/usr/bin/find ${LOG_ROOT_DIR}/* -depth -mtime +30 | /usr/bin/xargs /bin/rm -fr
LOG_DIR=${LOG_ROOT_DIR}/`/bin/date +%y%m%d`
if [ ! -d ${LOG_DIR} ]; then
  /bin/mkdir ${LOG_DIR}
fi
LOG=${LOG_DIR}/`/bin/date +%H%M%S`_${LSNAME}_${MSGID}.txt

echo $0 > "${LOG}"

MNGSVR=localhost:${COSMI_MNG_MACT_MNGSVR_PORT}
USERID=<User-id>
PASSWD=<Password>

echo ./mngsvrutil -m ${MNGSVR} -u ${USERID} -p ${PASSWD} -t ${LSNAME} -s stop server >> "
${LOG}" 2>&1
./mngsvrutil -m ${MNGSVR} -u ${USERID} -p ${PASSWD} -t ${LSNAME} -s stop server >> "${LOG
}" 2>&1
```

```
RET=$?
echo ${RET} >> "${LOG}"

if [ $RET -eq 0 ]; then
  echo ./mngsvrutil -m ${MNGSVR} -u ${USERID} -p ${PASSWD} -t ${LSNAME} -s start server >
> "${LOG}" 2>&1
  ./mngsvrutil -m ${MNGSVR} -u ${USERID} -p ${PASSWD} -t ${LSNAME} -s start server >> "${
LOG}" 2>&1
  RET=$?
  echo ${RET} >> "${LOG}"
fi

exit ${RET}
```

(b) Management イベント発行サーバの所属サービスユニットを閉塞するサンプル

Management イベント発行サーバの所属サービスユニットを閉塞するサンプルが提供されていますので，サンプルを参考にして作成してください。サンプルのファイル名と動作を次に示します。

サンプルのファイル名

- Windows の場合：mActionSample_closeUnit.bat
- UNIX の場合：mActionSample_closeUnit.sh

サンプルの動作

サービスユニットに所属する J2EE サーバにリソース枯渇などの障害が発生した場合に，Management イベントを発行したサーバを閉塞します。

Management イベント発行サーバの所属サービスユニットを閉塞するサンプルを次に示します。

- Windows の場合

```
       (省略)
         :
rem Management action sample for close unit.

setlocal

set LSNAME=%COSMI_MNG_MACT_LSNAME%
set MSGID=%COSMI_MNG_MACT_MSG_ID%
set WEBSYSTEM=%COSMI_MNG_MACT_WEBSYSTEM%
set UNIT=%COSMI_MNG_MACT_UNIT%

set CJCLDELLOG=%COSMINEXUS_HOME%¥CC¥client¥bin¥cjcldellog.bat
set LOG_ROOT_DIR=%SystemDrive%¥<MyLogDir>
if not exist "%LOG_ROOT_DIR%" mkdir "%LOG_ROOT_DIR%"
call "%CJCLDELLOG%" -t 30d -f "%LOG_ROOT_DIR%"
set LOG_DIR=%LOG_ROOT_DIR%¥%date:/=%
if not exist "%LOG_DIR%" mkdir "%LOG_DIR%"
set LOG=%LOG_DIR%¥%time::=%_%LSNAME%_%MSGID%.txt

echo %0 > "%LOG%"

if "%WEBSYSTEM%" == "" goto :ERR
if "%UNIT%" == "" goto :ERR
```

```
set MNGSVR=localhost:%COSMI_MNG_MACT_MNGSVR_PORT%
set UID=<User-id>
set PWD=<Password>

echo cmx_stop_target.exe -m %MNGSVR% -u %UID% -p %PWD% -mode ALL -s %WEBSYSTEM% -unit %UN
IT% >> "%LOG%" 2>&1
cmx_stop_target.exe -m %MNGSVR% -u %UID% -p %PWD% -mode ALL -s %WEBSYSTEM% -unit %UNIT% >
> "%LOG%" 2>&1
set RET=%ERRORLEVEL%
echo %RET% >> "%LOG%"
goto :END

:ERR
set RET=2

:END
exit %RET%
```

- UNIX の場合

```
       (省略)
         :
# Management action sample for close unit.

LSNAME=${COSMI_MNG_MACT_LSNAME}
MSGID=${COSMI_MNG_MACT_MSG_ID}
WEBSYSTEM=${COSMI_MNG_MACT_WEBSYSTEM}
UNIT=${COSMI_MNG_MACT_UNIT}

LOG_ROOT_DIR=/tmp/<MyLogDir>
if [ ! -d ${LOG_ROOT_DIR} ]; then
  /bin/mkdir ${LOG_ROOT_DIR}
fi
/usr/bin/find ${LOG_ROOT_DIR} -depth -mtime +30 -mindepth 1 -exec /bin/rm -fr {} ¥;
LOG_DIR=${LOG_ROOT_DIR}/`/bin/date +%y%m%d`
if [ ! -d ${LOG_DIR} ]; then
  /bin/mkdir ${LOG_DIR}
fi
LOG=${LOG_DIR}/`/bin/date +%H%M%S`_${LSNAME}_${MSGID}.txt

echo $0 > "${LOG}"

if [ "${WEBSYSTEM}" = "" -o "${UNIT}" = "" ]; then
  exit 2
fi

MNGSVR=localhost:${COSMI_MNG_MACT_MNGSVR_PORT}
USERID=<User-id>
PASSWD=<Password>

echo ./cmx_stop_target -m ${MNGSVR} -u ${USERID} -p ${PASSWD} -mode ALL -s ${WEBSYSTEM} -
unit ${UNIT} >> "${LOG}" 2>&1
./cmx_stop_target -m ${MNGSVR} -u ${USERID} -p ${PASSWD} -mode ALL -s ${WEBSYSTEM} -unit
${UNIT} >> "${LOG}" 2>&1
RET=$?
echo ${RET} >> "${LOG}"
```

```
exit ${RET}
```

(3) Management アクション実行コマンドの動作

Management アクション実行コマンドの動作を次に示します。

- Management アクション実行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。
- Management アクション実行コマンドは Management Server によって実行されます。実行ユーザは，Management Server の実行ユーザになります。また，環境変数は Management Server に設定された環境変数を引き継ぎます。
- Management アクション実行コマンドのワーキングディレクトリは，<Application Server のインストールディレクトリ>¥manager¥bin（Windows の場合），または/opt/Cosminexus/manager/bin（UNIX の場合）です。

10 CTM の稼働統計情報の収集

この章では，CTM が提供するプロセス群（CTM デーモンや CTM レギュレータなど）が出力する稼働統計情報を使用した，CTM の処理性能の監視について説明します。

10.1　この章の構成

この章の構成を次の表に示します。

表 10–1　この章の構成（CTM の稼働統計情報の収集）

分類	タイトル	参照先
解説	CTM の稼働統計情報の概要	10.2
	稼働統計情報ファイルの利用方法	10.3

注　「実装」,「設定」,「運用」,「注意事項」について，この機能固有の説明はありません。

10.2　CTM の稼働統計情報の概要

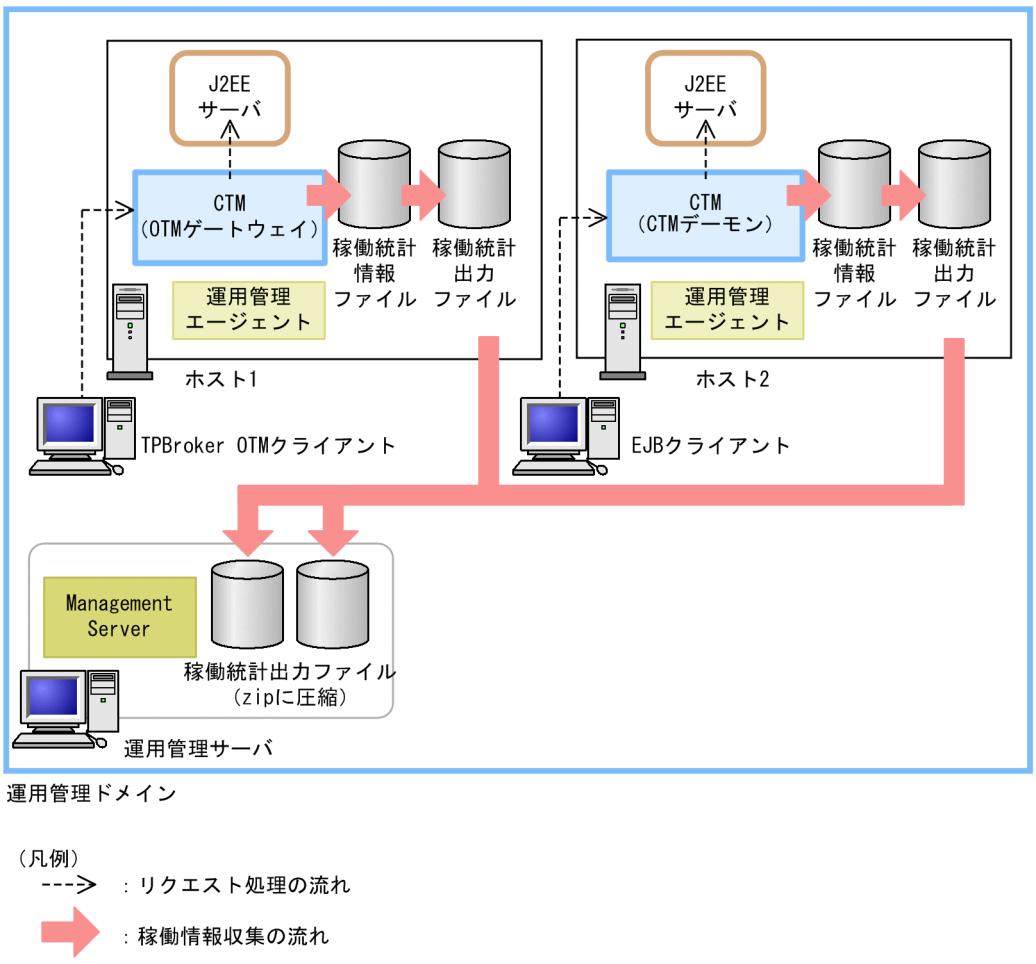
CTM が提供する CTM デーモンや CTM レギュレータなどのプロセス群では，リクエストを処理する過程で稼働統計情報をバッファに出力します。

このバッファの情報は，CTM 単位にホスト上のファイル（稼働統計情報ファイル）に一定間隔で出力されます。バイナリ形式の稼働統計情報ファイルから，CSV 形式などのテキストファイル（稼働統計出力ファイル）に変換したファイルを使用して，CTM の処理性能を解析します。

CTM では，CTM でリクエストを受け付けてから J2EE サーバにリクエストを送信するまで，およびその処理結果が CTM に返却されるまでの一連の処理の過程で稼働統計情報を出力します。稼働統計情報を基に，CTM が提供するプロセス群の性能処理性能を解析したり，稼働状態を確認したりしてシステムの安定化を図ることができます。

CTM の稼働統計情報の収集の概要を次の図に示します。

図 10-1　CTM の稼働統計情報の収集の概要



性能に関する処理の CTM の稼働統計情報は，CTM が提供するプロセス群によって，CTM 単位に，ホスト上のファイル（稼働統計情報ファイル）に出力されます。

Management Server を利用して運用している場合，各ホストに出力された稼働統計情報ファイルの内容は，Management Server のコマンドによって，運用管理サーバに一括収集できます。なお，CTM が出力した稼働統計情報ファイルは，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，稼働統計情報ファイルをテキスト（CSV）形式のファイルに編集して，それを圧縮したものを収集します。テキスト形式に編集した稼働統計情報ファイルを稼働統計出力ファイルといいます。

運用管理者は，収集した稼働統計出力ファイルを基に，運用管理ドメイン内の CTM に関する性能解析およびボトルネックの分析ができるようになります。

なお，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稼働統計情報ファイルです。

以降で，運用管理コマンド（mngsvrutil）を使用して，CTM の稼働統計情報を収集する方法について説明します。また，収集した情報を確認する方法についても説明します。

次の単位で CTM の稼働統計情報を収集できます。

- 運用管理ドメイン内のすべての CTM
- 運用管理ドメイン内の指定した CTM
- 運用管理ドメイン内の指定したホスト内の CTM

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「2.2.1　製品と構成ソフトウェアの対応」を参照してください。

10.2.1　CTM の稼働統計情報の収集方法

CTM の稼働統計情報の出力と収集は，mngsvrutil コマンドにサブコマンド「collect」を指定して実行します。

CTM の稼働統計情報は，次のどれかの単位で収集できます。

- 運用管理ドメイン内のすべての CTM
- 運用管理ドメイン内の指定した CTM
- 運用管理ドメイン内の指定したホスト内の CTM

なお，収集対象になるのは稼働中の CTM だけです。

CTM の稼働統計情報を出力，収集する場合の mngsvrutil コマンド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すべての CTM 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 allCtmStatistics
```

実行例

```
mngsvrutil -m mnghost -u user01 -p pw1 collect allCtmStatistics
```

運用管理ドメイン内の指定した CTM，または指定したホスト内の CTM 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CTM名またはホスト名>[-k host] collect ctmStatistics
```

実行例

- 指定した CTM の稼働統計情報を収集する場合

  mngsvrutil -m mnghost -u user01 -p pw1 -t ctm01 collect ctmStatistics
- 指定したホスト内の CTM の稼働統計情報を収集する場合

  mngsvrutil -m mnghost -u user01 -p pw1 -t host01 -k host collect ctmStatistics

コマンドの実行結果は，標準出力またはファイルに出力されます。

mngsvrutil 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。

10.2.2 CTM の稼働統計情報の出力先と出力情報

収集した稼働統計情報は CSV 形式で出力されて，ZIP 形式で圧縮されます。稼働統計情報が出力される CSV ファイルを，**稼働統計出力ファイル**といいます。

収集した稼働統計出力ファイルの出力先と出力情報について説明します。

収集した稼働統計出力ファイルの出力先

- Windows の場合

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

  <Manager のログ出力ディレクトリ>/ctm

なお，稼働統計出力ファイルは，稼働統計情報を収集する対象ごとに次の表に示すファイル名で出力されます。

表 10–2　収集対象ごとの稼働統計出力ファイル名

稼働統計情報の収集対象	ファイル名
運用管理ドメイン内のすべての CTM	<ドメイン名>-<日時>.zip
運用管理ドメイン内の指定した CTM	<論理サーバ名>-<日時>.zip
指定したホスト内の CTM	<ホスト名>-<日時>.zip

注　出力された稼働統計出力ファイル（CSV 形式）は，ZIP 形式で圧縮されます。

また，CSV 形式に出力される稼働統計情報は，CTM ノード単位，キュー単位，およびメソッド単位でそれぞれ異なるファイルで出力されます。出力単位ごとの稼働統計情報のファイル名を次に示します。

表 10–3　出力単位ごとの稼働統計出力ファイル名

出力単位	ファイル名
CTM ノード単位	noddata.csv
キュー単位	quedata.csv
メソッド単位	mtddata.csv

稼働統計情報の出力情報

稼働統計出力ファイルに出力される情報を次の表に示します。

表 10–4　稼働統計出力ファイルに出力される情報

統計情報種別	情報	説明
リクエスト情報	CTM レギュレータレスポンス時間	TPBroker クライアント，および EJB クライアントからの要求を受信した CTM レギュレータが，J2EE サーバに対してリクエストを送信してから応答を受信するまでの時間の差です※1。 CTM レギュレータで出力されます。 発生件数，および最大値・最小値・平均値がマイクロ秒単位で出力されます。
	CTM レギュレータリクエストタイムアウト	CTM レギュレータがリクエストを送信してから応答が返されるまでの間に発生したタイムアウト障害の発生件数です。CTM レギュレータで出力されます。
	CTM レギュレータリクエスト障害	CTM レギュレータがリクエストを送信してからその応答が返されるまでの処理で発生した障害の発生件数です。CTM レギュレータで出力されます。
	OTM ゲートウェイレスポンス時間	TPBroker OTM クライアントからの要求を受信した OTM ゲートウェイが，J2EE サーバに対してリクエストを送信してから，応答を受信するまでの時間の差です。OTM ゲートウェイで出力されます。 発生件数，および最大値・最小値・平均値がマイクロ秒単位で出力されます。

統計情報種別	情報	説明
	OTM ゲートウェイリクエストタイムアウト	OTM ゲートウェイでのリクエストの応答待ちで発生したタイムアウト障害の件数です。OTM ゲートウェイで出力されます。
	OTM ゲートウェイリクエスト障害	OTM ゲートウェイがリクエストを送信してからその応答が返されるまでの処理で発生した障害の件数です。OTM ゲートウェイで取得します。
スケジュール情報	スケジュール待ち件数	スケジュールキューに滞留したリクエストの数です。 発生件数，および最大値・最小値・平均値が出力されます。
	スケジュール待ち時間	リクエストがスケジュールキューに登録されてから取り出されるまでの滞留時間です。 発生件数，および最大値・最小値・平均値がマイクロ秒単位で出力されます。
	スケジュールできなかったリクエスト数	スケジュールキューに同時に登録できる数を超えたため，スケジュールできなかったリクエストの数です。
プロセス情報	CTM レギュレータ異常終了件数	CTM レギュレータの異常終了が発生した件数です。 異常終了が発生するたびに，この情報が出力されます。
	OTM ゲートウェイ異常終了件数	OTM ゲートウェイの異常終了が発生した件数です。 異常終了が発生するたびに，この情報を取得します。
コネクション情報	CTM レギュレータとクライアント間	CTM レギュレータとクライアントとのコネクション情報です。ctmstsstart コマンドの-CTMCheckInterval オプションに指定した間隔で出力されます。 最大値・最小値・平均値がマイクロ秒単位で出力されます。
	OTM ゲートウェイと TPBroker OTM クライアント間	OTM ゲートウェイと TPBroker OTM クライアントとのコネクション情報です。ctmstsstart コマンドの-CTMCheckInterval オプションに指定した間隔でこの情報を取得します。 最大値・最小値・平均値がマイクロ秒単位で出力されます。
	CTM デーモンと CTM レギュレータ間	CTM デーモンと CTM レギュレータとのコネクション情報です。ctmstsstart コマンドの-CTMCheckInterval オプションに指定した間隔でこの情報を取得します。 最大値・最小値・平均値がマイクロ秒単位で出力されます。
	CTM デーモンと OTM ゲートウェイ間	CTM デーモンと OTM ゲートウェイとのコネクション情報です。ctmstsstart コマンドの-CTMCheckInterval オプションに指定した間隔でこの情報を取得します。 最大値・最小値・平均値がマイクロ秒単位で出力されます。
	CTM デーモンと J2EE サーバ※2 間(制御用)	CTM デーモンと J2EE サーバ※2 間の制御用コネクション情報です。ctmstsstart コマンドの-CTMCheckInterval オプションに指定した時間間隔でこの情報を取得します。
	CTM デーモンと J2EE サーバ※2 間(リクエスト用)	CTM デーモンと J2EE サーバ※2 間のリクエスト要求用処理コネクション情報です。ctmstsstart コマンドの-CTMCheckInterval オプションに指定した時間間隔でこの情報を取得します。

注※1

バッチアプリケーションを使用するシステムの場合は，cjexecjob コマンドからの要求を受信した CTM レギュレータが，バッチサーバに対して要求を送信してから応答を受信するまでの時間の差です。

注※2

バッチアプリケーションを実行するシステムでは，J2EE サーバはバッチサーバとなります。

10.2.3　CTM の稼働統計情報の出力例

運用管理コマンド（mngsvrutil）を利用して収集した稼働統計出力ファイルの出力例を次に示します。

CTM ノード単位で出力した稼働統計情報の出力例

```
*** Statistics Information by CTMNode ***
CTMDomain, CTMID, start, end, Event, Count, Maximum, Minimum, Average, unit
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Request Response Time(Reg),      26
525,      94000,          0,       9099,(usec)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Request Time Out(Reg),          0,,
,,(count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Request Error(Reg),          2,,,,(
count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Request Response Time(OTM),
  0,          0,          0,          0,(usec)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Request Time Out(OTM),          0,,
,,(count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Request Error(OTM),          0,,,,(
count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Schedule Wait Count,      26526,
       1,          1,          1,(count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Schedule Wait Time,      26526,
5688000,          0,        560,(usec)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Request Overflow,          0,,,,(co
unt)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Regulator down,          0,,,,(coun
t)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,OTMGateway down,          0,,,,(cou
nt)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,Regulator-Client,,          2,
     0,          1,(count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,OTMGateway-Client,,          0,
      0,          0,(count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,CTMDaemon-J2EEServer(Ctr),,
  2,          0,          1,(count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,CTMDaemon-J2EEServer(Req),,
  1,          0,          1,(count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,CTMDaemon-Regulator,,          1,
        1,          1,(count)
CTMDOMAIN,10.209.15.55,2004/11/05 09:57,2004/11/05 10:07,CTMDaemon-OTMGateway,,          5,
        5,          5,(count)
```

キュー単位で出力した稼働統計情報の出力例

```
*** Statistics Information by Queue ***
CTMDomain, CTMID, QueueName, start, end, Event, Count, Maximum, Minimum, Average, unit
CTMDOMAIN,10.209.15.55,converter,2004/11/05 09:57,2004/11/05 10:07,Schedule Wait Count,
 26526,          1,          1,          1,(count)
CTMDOMAIN,10.209.15.55,converter,2004/11/05 09:57,2004/11/05 10:07,Schedule Wait Time,
26526,    5688000,          0,        560,(usec)
CTMDOMAIN,10.209.15.55,converter,2004/11/05 09:57,2004/11/05 10:07,Request Overflow,
  0,,,,(count)
CTMDOMAIN,10.209.15.55,converter,2004/11/05 09:57,2004/11/05 10:07,CTMDaemon-J2EEServer(Req)
,,          1,          1,          1,(count)
```

メソッド単位で出力した稼働統計情報の出力例

```
*** Statistics Information by Method ***
CTMDomain, CTMID, Interface, Method, start, end, Event, Count, Maximum, Minimum, Average, un
it
CTMDOMAIN,10.209.15.55,Converter,dollarToYen,2004/11/05 09:57,2004/11/05 10:07,Request Respo
nse Time(Reg),      10393,      78000,          0,       9143,(usec)
CTMDOMAIN,10.209.15.55,Converter,dollarToYen,2004/11/05 09:57,2004/11/05 10:07,Request Time
Out(Reg),          0,,,,(count)
CTMDOMAIN,10.209.15.55,Converter,dollarToYen,2004/11/05 09:57,2004/11/05 10:07,Request Erro
r(Reg),          0,,,,(count)
CTMDOMAIN,10.209.15.55,Converter,dollarToYen,2004/11/05 09:57,2004/11/05 10:07,Request Respo
nse Time(OTM),          0,          0,          0,          0,(usec)
CTMDOMAIN,10.209.15.55,Converter,dollarToYen,2004/11/05 09:57,2004/11/05 10:07,Request Time
Out(OTM),          0,,,,(count)
CTMDOMAIN,10.209.15.55,Converter,dollarToYen,2004/11/05 09:57,2004/11/05 10:07,Request Erro
r(OTM),          0,,,,(count)
CTMDOMAIN,10.209.15.55,Converter,yenToEuro,2004/11/05 09:57,2004/11/05 10:07,Request Respons
e Time(Reg),      10393,      94000,          0,       9119,(usec)
CTMDOMAIN,10.209.15.55,Converter,yenToEuro,2004/11/05 09:57,2004/11/05 10:07,Request Time Ou
t(Reg),          0,,,,(count)
CTMDOMAIN,10.209.15.55,Converter,yenToEuro,2004/11/05 09:57,2004/11/05 10:07,Request Error(R
eg),          2,,,,(count)
CTMDOMAIN,10.209.15.55,Converter,yenToEuro,2004/11/05 09:57,2004/11/05 10:07,Request Respons
e Time(OTM),          0,          0,          0,          0,(usec)
CTMDOMAIN,10.209.15.55,Converter,yenToEuro,2004/11/05 09:57,2004/11/05 10:07,Request Time Ou
t(OTM),          0,,,,(count)
CTMDOMAIN,10.209.15.55,Converter,yenToEuro,2004/11/05 09:57,2004/11/05 10:07,Request Error(O
TM),          0,,,,(count)
```

10.3　稼働統計情報ファイルの利用方法

CTM の稼働統計情報ファイルの利用方法について説明します。

CTM の稼働統計情報ファイルには，CTM が提供する次に示すプロセス群の処理の状態，処理の開始・終了時間や，処理の回数などの情報がバイナリ形式で自動的に出力されます。

- CTM デーモン
- CTM レギュレータ
- OTM ゲートウェイ

バイナリ形式で出力された稼働統計情報ファイルは，Management Server の運用管理コマンド（mngsvrutil）を実行して編集します。編集した情報は，CSV 形式のファイル，または標準出力に出力されます。

稼働統計出力ファイルの取得方法，および稼働統計出力ファイルの出力形式については，「10.2　CTM の稼働統計情報の概要」を参照してください。また，稼働統計出力ファイルの取得に使用する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「ctmstsed（稼働統計情報の編集と出力)」を参照してください。

11 コンソールログの出力

この章では，運用管理エージェントが起動したプロセスの標準出力や標準エラー出力などコンソールログの出力機能について説明します。

11.1　この章の構成

この章の構成を次の表に示します。

表 11–1　この章の構成（コンソールログの出力）

分類	タイトル	参照先
解説	コンソールログの出力対象	11.2
設定	コンソールログの取得の設定	11.3

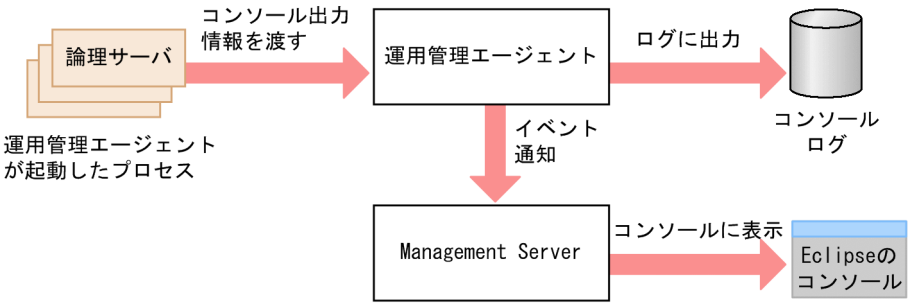
注　「実装」,「運用」,「注意事項」について，この機能固有の説明はありません。

11.2　コンソールログの出力対象

この節では，コンソールログの出力対象について説明します。

Management Server を利用して運用する場合，運用管理エージェントが起動したプロセスの標準出力や標準エラー出力などを，コンソール出力情報として取得できます。コンソール出力情報の取得を次の図に示します。

図 11-1　コンソール出力情報の取得



運用管理エージェントが各プロセスから取得したコンソール出力情報は，ログファイル（コンソールログ）に出力されます。

また，Management Server を利用する Eclipse プラグインを使用している場合，コンソール出力情報は，Management Server を経由して Eclipse のコンソールにも表示されます。コンソール出力情報は，イベントとして Management Server に通知されます。イベントはイベントキューに滞留され，順次イベントキューからイベントが取り出されて Management Server に通知されます。Management Server が起動していないなどの理由でイベントを通知できない場合には，通知できるまでイベントが再送されます。イベントキュー内に滞留しているイベントがキューのサイズを超えた場合には，古いイベントから破棄されます。

11.2.1　コンソールログの出力対象となる操作

次の方法で論理サーバ，J2EE アプリケーション，リソースを操作する場合には，運用管理エージェントがプロセスを起動および停止するため，コンソールログを取得できます。

- Smart Composer 機能のコマンドを使用する場合
- Management Server の運用管理コマンド（mngsvrutil）を使用する場合
- Management Server を利用する Eclipse プラグインを使用する場合

11.2.2　コンソールログの出力対象となるプロセス

コンソールログの出力対象となるのは，次のプロセスです。

- 論理サーバを構成するプロセス
- J2EE アプリケーションやリソースを操作するプロセス

ただし，Windows の場合には，間接起動のプロセスからはコンソールログを取得できません。間接起動される論理サーバについては，「2.3　論理サーバの起動・停止の仕組み」を参照してください。

注意事項

運用管理エージェントが起動したプロセスが一度に大量の文字列を出力する場合，運用管理エージェントの GC などのタイミングによっては，対象となるプロセスが一時停止することがあります。

11.3　コンソールログの取得の設定

Management Server を利用している場合には，運用管理エージェントが起動したプロセスの標準出力や標準エラー出力などを，コンソール出力情報として取得できます。この節では，コンソールログを取得するための設定の変更について説明します。コンソール出力情報のログファイル（コンソールログ）への出力の設定を変更する場合は，adminagent.properties で次のキーを設定してください。

- adminagent.process.consolelog.enabled
  コンソール出力情報を出力するかどうかを指定します。デフォルトの設定では，true が指定されていて，出力するようになっています。
- adminagent.j2ee.process.console_log.enabled
  論理 J2EE サーバのコンソール出力情報をコンソールログに出力するかどうかを指定します。デフォルトの設定では，false が指定されていて，出力されないようになっています。
  また，adminagent.j2ee.process.console_log.enabled キーに true を指定する場合，adminagent.process.consolelog.enabled キーにも true を指定する必要があります。
- adminagent.userserver.process.console_log.enabled
  論理ユーザサーバのコンソール出力情報をコンソールログに出力するかどうかを指定します。デフォルトの設定では，false が指定されていて，出力されないようになっています。
  また，adminagent.userserver.process.console_log.enabled キーに true を指定する場合，adminagent.process.consolelog.enabled キーにも true を指定する必要があります。
- adminagent.j2ee.process.console_event.enabled
  論理 J2EE サーバのコンソール出力情報を，Management Server を利用する Eclipse プラグインで表示するかどうかを指定します。デフォルトの設定では，false が指定されていて，表示されないようになっています。
  また，adminagent.j2ee.process.console_event.enabled キーに true を指定する場合，adminagent.process.consolelog.enabled キーにも true を指定する必要があります。
- adminagent.userserver.process.console_event.enabled
  論理ユーザサーバのコンソール出力情報を，Management Server を利用する Eclipse プラグインで表示するかどうかを指定します。デフォルトの設定では，false が指定されていて，表示されないようになっています。
  また，adminagent.userserver.process.console_event.enabled キーに true を指定する場合，adminagent.process.consolelog.enabled キーにも true を指定する必要があります。
- adminagent.process.consolelog.filenum
  コンソールログの面数を指定します。
- adminagent.process.consolelog.filesize
  コンソールログの 1 面当たりの最大サイズを指定します。

12 JP1 と連携したシステムの運用

この章では，JP1 と連携したシステムの運用について説明します。

JP1 と連携してアプリケーションサーバのシステムを運用することで，アプリケーションサーバのシステムを含んだ業務システム全体の監視から問題の検知，調査対策までの一連の運用を，一貫した操作で効率良くできるようになります。また，サーバやアプリケーションの起動・停止を自動化することで，システムの日常運用の効率化を図れるほか，システムリソースの有効活用もできるようになります。

なお，この章で説明する JP1 との連携機能は，Management Server を利用して運用することが前提になります。

12.1　この章の構成

この章の構成を次の表に示します。

表 12–1　この章の構成（JP1 と連携したシステムの運用）

分類	タイトル	参照先
解説	JP1 との連携	12.2
	JP1 と連携するシステムでできること	12.3

注　「実装」,「設定」,「運用」,「注意事項」について，この機能固有の説明はありません。

12.2 JP1 との連携

業務システムは，アプリケーションサーバ以外にも，サーバ，ネットワーク，ストレージなどのハードウェアや，Web サーバ，メールサーバ，データベースなどのソフトウェアなど，さまざまなリソースによって構成されます。業務システムの運用管理では，これら全体をできるだけ効率良く確実に運用する方法が求められます。

JP1 は，このような複合的な業務システムの統合運用管理を実現する，ミドルウェアです。JP1 と連携して実現できる運用を次に示します。これらは，業務システムの運用形態に応じて，組み合わせて運用できます。なお，JP1 との連携は，J2EE アプリケーションの実行環境，およびバッチアプリケーションの実行環境で使用できます。

- システム全体を対象に，障害を調査して対策できます（**システムの集中監視**）。
- Smart Composer 機能でのシステム構築時にシステムの構成情報を定義したり，定義したシステムの構成情報を収集したりできます（**システムの構成定義および管理**）※。
- アプリケーションサーバで管理しているサーバやプロセスの起動／停止を，ジョブを使用して自動化できます（**ジョブによるシステムの自動運転**）。
- Smart Composer 機能で構築したシステムを，シナリオを使用して自動運転できます（**シナリオによるシステムの自動運転**）※。
- システムの監査ログを自動で収集して一元管理できます（**監査ログの収集／一元管理**）。

注※

システムの構成定義・管理，およびシナリオを使用した自動運転は，Smart Composer 機能を使用して構築したシステムが前提となります。Smart Composer 機能を使用したシステム構築方法については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」を参照してください。

また，アプリケーションサーバでは，Management Server の機能として，JP1 と連携する環境の構築を支援したり，JP1 と運用管理ポータルを一連の流れの中でスムーズに運用したりするための，次の機能を提供しています。

- JP1/IM の統合スコープに表示する監視ツリーの自動生成
- JP1/IM の統合コンソールまたは統合スコープからの Management Server の運用管理ポータルの直接起動（Windows の場合）
- JP1/AJS でジョブを定義するときに使用するカスタムジョブ（Windows の場合）

12.3　JP1 と連携するシステムでできること

この節では，JP1 との連携の設定の概要について説明します。

Management Server を利用して JP1 と連携することで，JP1 の集中監視，システム構成の定義・管理，運用の自動化，稼働状況の分析などの機能を使用して，アプリケーションサーバ以外で構築されたシステムも含めた業務システム全体を一括管理できるようになります。JP1 との連携では，業務システムの運用形態に応じて，次の設定を組み合わせて環境を構築してください。

- システムの集中監視および構成情報の定義・管理の設定（JP1/IM との連携）
- システムの自動運転の設定（JP1/AJS との連携）

12.3.1　システムの集中監視

JP1/IM と連携することで，システムの集中監視の定義・管理ができます。

システムの集中監視は，業務システム全体のリソースの状態を監視することで，障害の発生を検知し，原因を究明して対策したりします。なお，JP1/IM と連携したシステムの運用の詳細については，「13.2　システムの集中監視（JP1/IM との連携）の概要」を参照してください。

JP1/IM と連携してシステムを集中監視する場合の設定内容を次に示します。

- **監視ツリーの自動生成の設定**

  JP1/IM でアプリケーションサーバの障害を監視するために必要な設定をして，JP1/IM - Manager の統合スコープに表示するアプリケーションサーバ用の監視ツリーを自動生成します。設定方法については，「13.4.1　監視ツリーの自動生成の設定」を参照してください。

- **障害監視の設定**

  JP1/IM - Manager の統合コンソールまたは統合スコープから，アプリケーションサーバのシステムで発生した障害を監視するための設定をします。設定方法については，「13.4.2　障害監視の設定」を参照してください。

> **参考**
>
> JP1/Base のログファイルトラップを使用すると，アプリケーションサーバのログファイルに出力される情報を JP1 イベントに変換して，JP1/IM に通知できます。この場合，JP1/Base のログファイルトラップで監視するログファイルの設定が必要になります。詳細は，「13.4.3　監視するログファイルの設定」を参照してください。

12.3.2　システムの自動運転

システムの自動運転は，サーバやアプリケーションの開始や停止のスケジュールをあらかじめ定義しておくことで，効率的なリソースの配分や業務の効率化，省略化を図る運用方法です。アプリケーションサーバでは，JP1/AJS と連携して，自動運転をジョブに定義します。Windows の場合には，アプリケーションサーバ用の**カスタムジョブ**を利用してジョブを定義できます。カスタムジョブは，JP1/AJS 以外のプログラムと JP1/AJS が連携するジョブを定義する場合に，目的のジョブを容易に作成するために使用できるジョブのテンプレートです。

JP1/AJS と連携したシステムの運用の詳細については，「14.2　ジョブによるシステムの自動運転（JP1/AJS との連携）の概要」を参照してください。JP1/AJS と連携してシステムを自動運転化するための設定方法については，「14.3　ジョブによるシステムの自動運転の設定」を参照してください。

12.3.3　JP1 で監視できるアプリケーションサーバのプロセス

この節では，JP1 との連携で稼働情報などを監視できるアプリケーションサーバのプロセスについて，OS ごとに説明します。

JP1/IM と連携して，障害検知や対策を実施するシステムの集中監視については，「13.　システムの集中監視（JP1/IM との連携）」を参照してください。

- Windows の場合

  アプリケーションサーバのプロセスについて次の表に示します。

表 12-2　アプリケーションサーバのプロセス（Windows の場合）

<table>
<tr><th>項番</th><th>プロセス</th><th colspan="4">起動するプロセスの名称
（上位プロセス←→下位プロセス）</th><th>ユーザの権限</th><th>プロセスの説明</th></tr>
<tr><td>1</td><td>J2EE サーバ</td><td>adminagentsv</td><td>adminagent</td><td>aapgw</td><td>cjstartsv</td><td>SYSTEM</td><td>業務プログラムを実行します。
停止が要求されるまで常に存在します。</td></tr>
<tr><td>2</td><td rowspan="4">Web サーバ</td><td colspan="4">httpsd</td><td>SYSTEM</td><td>HTTP Server を制御します。</td></tr>
<tr><td>3</td><td>httpsd</td><td colspan="3">httpsd</td><td>SYSTEM</td><td>HTTP Server のサーバとして機能します。</td></tr>
<tr><td>4</td><td rowspan="2"></td><td colspan="3">rotatelogs</td><td>SYSTEM</td><td rowspan="2">HTTP Server のログをラップアラウンドで取得します。
httpsd.conf で指定した数だけ起動します。</td></tr>
<tr><td>5</td><td colspan="3">rotatelogs2</td><td>SYSTEM</td></tr>
<tr><td>6</td><td>Management Server</td><td>mngsvr.exe</td><td colspan="3">cjstartsv.exe</td><td>SYSTEM</td><td>運用管理エージェントに指示を出して，運用管理ドメイン全体を運用管理します。
停止が要求されるまで常に存在します。
なお，cjstartsv.exe の引数に指定したサーバが監視対象となります。デフォ</td></tr>
</table>

項番	プロセス	起動するプロセスの名称 （上位プロセス←→下位プロセス）		ユーザの権限	プロセスの説明
					ルトのサーバ名は「cosmi_m」です。mngsvrctl setup コマンドでサーバ名を指定している場合，指定したサーバ名を指定します。
7	運用管理エージェント	adminagentsv.exe	adminagent.exe	SYSTEM	運用管理者の代わりに，論理サーバの起動・停止，および設定ファイルの更新を実行します。 停止が要求されるまで常に存在します。
8	サーバ通信エージェント	sinaviagentsv.exe	sinaviagent.exe	SYSTEM	仮想サーバ上のアプリケーションサーバをセットアップします。 停止が要求されるまで常に存在します。
9	ネーミングサービス	nameserv		＜OS のアカウント＞	ネーミングサービスをアウトプロセスで起動する場合に実行します。
10	CORBA クライアントアプリケーション	vbj		＜OS のアカウント＞	TPBroker が提供する CORBA アプリケーションを実行します。
11	スマートエージェント	osagent		＜OS のアカウント＞	CTM を使用する場合にスマートエージェントを実行します。
12	CJMS プロバイダ	java.exe		＜OS のアカウント＞	CJMSP ブローカーを実行します。 停止が要求されるまで常に存在します。
13	Java アプリケーション	cjclstartap		＜OS のアカウント＞	cjclstartap コマンドを実行して Java アプリケーションを開始します。アプリケーションを終了するまで存在します。
14	PRF	cprfd.exe		SYSTEM	パフォーマンストレーサ（PRF デーモン）のトレースファイルを出力します。
15	CTM	ctmdmd.exe		SYSTEM	CTM デーモンの情報を管理します。
16		ctmd.exe		SYSTEM	リクエストをスケジューリングします。
17		ctmd.exe	ctmregltd.exe	SYSTEM	CTM デーモンへのリクエストを集約します。
18			ctmtscgwd.exe	SYSTEM	TPBroker，OTM クライアントのリクエストを受け付けるゲートウェイとして機能します。

- UNIX の場合

  アプリケーションサーバのプロセスについて次の表に示します。

表 12-3　アプリケーションサーバのプロセス（UNIX の場合）

項番	プロセス	起動するプロセスの名称 （上位プロセス←→下位プロセス）		ユーザの権限	プロセスの説明
1	J2EE サーバ	cjstartsv		＜ OS のアカウント＞	業務プログラムを実行します。 停止が要求されるまで常に存在します。
2	Web サーバ	httpsd		root	HTTP Server を制御します。
3		httpsd	httpsd	各ディレクティブの指定値	HTTP Server のサーバとして機能します。 最大で MaxClient に指定した数だけ起動します。
4			rotatelogs	root	HTTP Server のログをラップアラウンドで取得します。 httpsd.conf で指定した数だけ起動します。
5			rotatelogs2	root	
6			gcache	root	SSL セションキャッシュを取得する。 HTTP Server の SSL セション管理機能有効時に起動する。
7	Management Server	cjstartsv		root	運用管理エージェントに指示を出して，運用管理ドメイン全体を運用管理します。 なお，cjstartsv の引数に指定したサーバが監視対象となります。デフォルトのサーバ名は「cosmi_m」です。 mngsvrctl setup コマンドでサーバ名を指定している場合，指定したサーバ名を指定します。
8	運用管理エージェント	adminagent		root	運用管理者の代わりに，論理サーバの起動・停止，および設定ファイルの更新を実行します。 停止が要求されるまで常に存在します。
9	サーバ通信エージェント	sinaviagent		root	仮想サーバ上のアプリケーションサーバをセットアップします。 停止が要求されるまで常に存在します。
10	ネーミングサービス	nameserv		＜ OS のアカウント＞	ネーミングサービスをアウトプロセスで起動する場合に実行します。
11	CORBA クライアントアプリケーション	vbj		＜ OS のアカウント＞	TPBroker が提供する CORBA アプリケーションを実行します。
12	スマートエージェント	osagent		＜ OS のアカウント＞	CTM を使用する場合にスマートエージェントを実行します。
13	CJMS プロバイダ	java		＜ OS のアカウント＞	CJMSP ブローカーを実行します。

項番	プロセス	起動するプロセスの名称 （上位プロセス←→下位プロセス）		ユーザの権限	プロセスの説明
					停止が要求されるまで常に存在します。
14	Java アプリケーション	cjclstartap		＜OS のアカウント＞	cjclstartap コマンドを実行して Java アプリケーションを開始します。アプリケーションを終了するまで存在します。
15	PRF	cprfd		root	パフォーマンストレーサ（PRF デーモン）のトレースファイルを出力します。
16	CTM	ctmdmd		root	CTM デーモンの情報を管理します。
17		ctmd		root	リクエストをスケジューリングします。
18		ctmd	ctmregltd	root	CTM デーモンへのリクエストを集約します。
19			ctmtscgwd	root	TPBroker，OTM クライアントのリクエストを受け付けるゲートウェイとして機能します。

13 システムの集中監視（JP1/IM との連携）

この章では，JP1/IM と連携したシステムの集中監視の運用について説明します。システムの集中監視では，システム全体を対象に，障害を調査して対策できます。システムの集中監視の運用に必要な機能の設定についても説明します。

なお，この章で説明する JP1 との連携機能は，Management Server を利用して運用することが前提になります。

13.1　この章の構成

この章の構成を次の表に示します。

表 13–1　この章の構成（システムの集中監視（JP1/IM との連携））

分類	タイトル	参照先
解説	システムの集中監視（JP1/IM との連携）の概要	13.2
	システムの集中監視の仕組み	13.3
設定	システムの集中監視のための設定	13.4
運用	Web フロントシステムでの運用例	13.5.1

注　「実装」,「注意事項」について，この機能固有の説明はありません。

13.2 システムの集中監視（JP1/IM との連携）の概要

Management Server を利用して JP1/IM と連携することで，JP1 の集中監視ができます。

この節では，システムの集中監視の概要について説明します。

システムの集中監視は，業務システム全体のリソースの状態を監視することで，障害の発生を検知して原因を究明，対策したりできる運用方法です。JP1/IM と連携することで実現できます。**JP1/IM** は，企業情報システム全体を統合管理する基盤になるプログラム群です。JP1/IM でシステムの集中監視をする場合は，次のプログラムを利用します。

- JP1/IM - Manager
  - **統合コンソール**

    システムで発生した事象を JP1 イベントによって集中管理することで，システムの統合管理を実現するプログラムです。**JP1 イベント**とは，システムで発生した事象に，JP1 で管理するための属性を付与された情報です。

    JP1/IM - Manager では，発生した事象に関連するエラーメッセージなどが表示されるので，事象の詳細を確認できます。
  - **統合スコープ**

    システムを監視するための画面を，管理者の目的に合わせて表示できるようにすることで，目的指向型のシステム監視を実現するプログラムです。

    発生した事象の影響範囲が業務単位またはサーバ単位のツリー形式で確認できるので，影響範囲の切り分けや特定ができます。
- JP1/IM - View（ビューアー）

  JP1/IM でシステム運用管理をするときに使用する，監視・操作画面を提供するプログラムです。

  統合コンソールおよび統合スコープの操作画面を表示するホストに必要です。

JP1/IM のシステムの集中監視の詳細については，マニュアル「JP1/Integrated Management - Manager 運用ガイド」を参照してください。

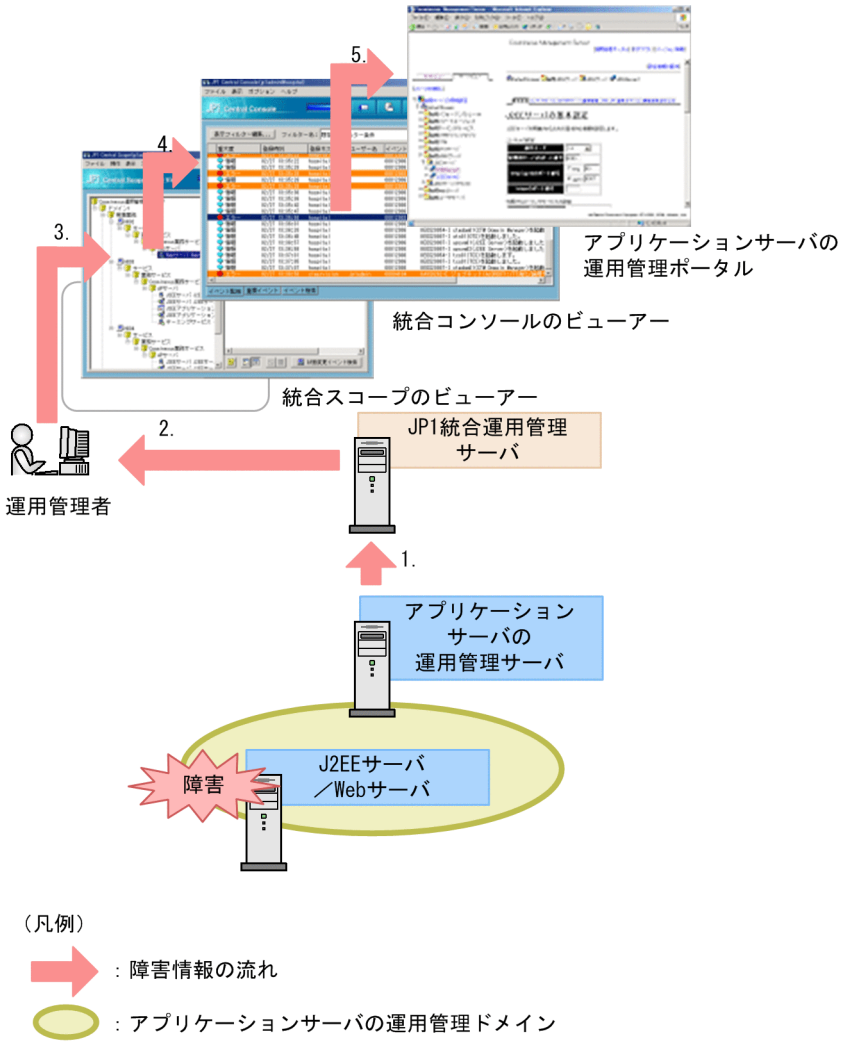
13.3　システムの集中監視の仕組み

JP1/IM と連携してシステムを集中監視すると，システムの次の事象を統合コンソールと統合スコープのビューアーで監視できるようになります。

- J2EE アプリケーションが開始，停止または実行されたことの通知や，J2EE アプリケーション内で発生した障害を監視できます。J2EE アプリケーションの監視をするには，監視ツリーの自動生成で取得する内容にアプリケーション情報を含めるかどうかの設定が必要です。詳細については，「13.3.2(1)　業務指向ツリー」の J2EE アプリケーション監視の説明を参照してください。
- 論理サーバが開始，停止されたことの通知や，論理サーバで発生した障害を監視できます。
- J2EE アプリケーションやバッチアプリケーション内で出力されたユーザログの情報を監視できます。

JP1/IM を使用したシステムの集中監視の概要を次に示します。

図 13–1　JP1/IM を使用したシステムの集中監視の概要



1. J2EE サーバ／Web サーバで障害が発生した場合，その事象が，JP1 統合運用管理サーバに通知されます。

2. JP1 統合運用管理サーバから，業務システムの運用管理者に事象が通知されます。

   通知方法としては，事象を選別して携帯電話に自動的に通知する設定などができます。このような設定をしておくと，運用管理者自身が常にコンソールを監視している必要がなくなります。

3. 通知を受けた運用管理者が，JP1/IM の統合スコープのビューアーを参照して，事象を確認します。

   統合スコープのビューアーでは，監視ツリーを使用して障害などの影響範囲を確認できます。例えば，特定の J2EE サーバで障害が発生している場合，監視ツリーでは，ツリーのルートノードから該当する J2EE サーバまでのノードが，障害を表す色で表示されます。このツリーをルートノードからたどれば，障害が発生している J2EE サーバとその影響範囲をすぐに確認できます。また，その障害によって業務が完全に停止してしまうのか，またはクラスタ構成によってシステムの停止にはなっていないのかなどを確認して，緊急性の有無を判断できます。

   なお，アプリケーションサーバでは，業務指向ツリーとサーバ指向ツリーの 2 種類の監視ツリーを作成する機能を提供しています。詳細については，「13.3.2　監視ツリーの自動生成」を参照してください。

4. 障害が発生した個所を特定したら，該当するノードから統合コンソールを表示して，エラー情報などを確認します。

5. 障害の原因がアプリケーションサーバシステムの設定にあった場合は，統合コンソールからアプリケーションサーバシステムの運用管理ポータルを表示して，障害の原因に対処します※。

   運用管理ポータルが表示される場合は，統合コンソールで選択したイベントに応じて，その事象に対処するための画面が直接表示されます。

   注※

   運用管理ポータルの表示は，Windows の場合だけできます。

アプリケーションサーバシステムでは，このような運用を実現するために，次の機能を提供しています。

- JP1/IM で使用するアプリケーションサーバ用の監視ツリーを自動生成する機能（アダプタコマンド）
- アプリケーションサーバシステムで発生したトラブルを JP1/IM に通知するための JP1 イベントを発行する機能
- JP1/IM から Management Server の運用管理ポータルを表示する機能（モニタ起動コマンド）※

注※

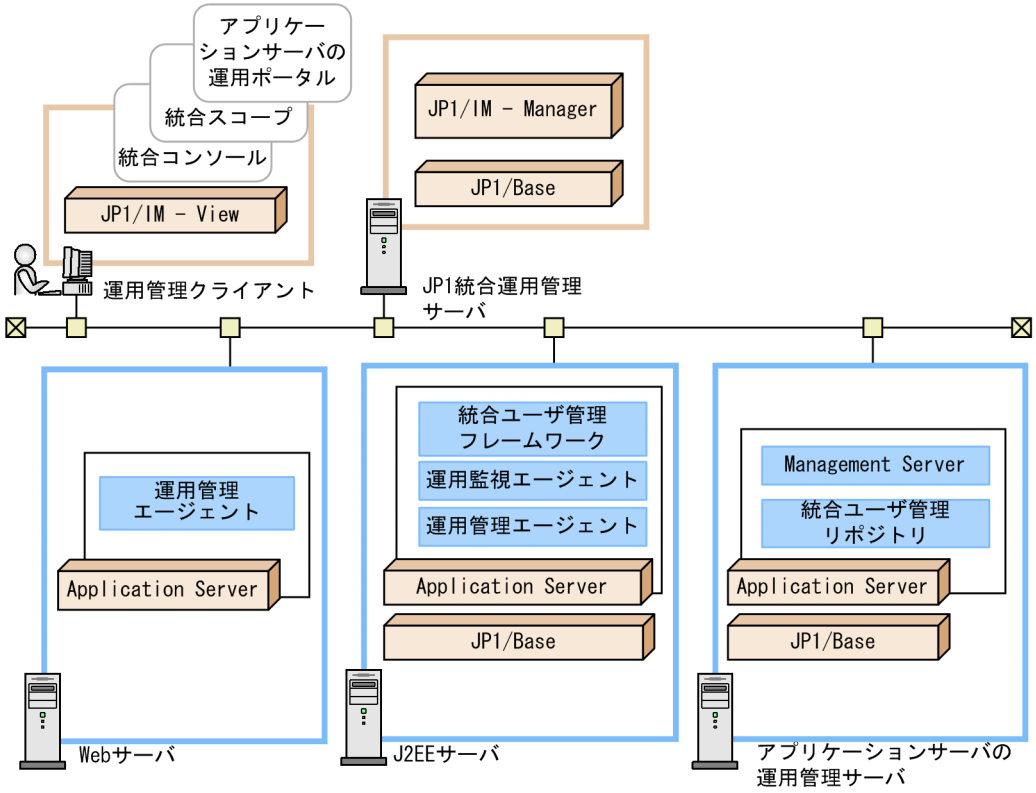
モニタ起動コマンドは Windows の場合だけ使用できます。

13.3.1　システムの集中監視に必要なプログラム

システムの集中監視は，JP1/IM を中心とした JP1 のプログラム群と，アプリケーションサーバの運用管理機能を組み合わせて実現します。

JP1/IM と連携してシステムの集中監視をする場合のシステム構成例を次の図に示します。

図 13-2　JP1/IM と連携してシステムの集中監視をする場合のシステム構成例



JP1 の各プログラムの主な機能を次に示します。なお，アプリケーションサーバの運用管理機能については，「1.2　システムの目的と機能の対応」を参照してください。

表 13-2　システムの集中監視をするために使用する JP1 プログラムの機能

JP1 プログラム	機能	配置するサーバ／クライアント
JP1/Base	JP1/IM の基盤機能を提供します。JP1 イベントの送受信などをするほか，JP1/IM システムのエージェントとして機能します。	• J2EE サーバ • アプリケーションサーバの運用管理サーバ • JP1 統合運用管理サーバ
JP1/IM - Manager	**統合コンソール**機能を提供します。	• JP1 統合運用管理サーバ
	統合スコープ機能を提供します。	
JP1/IM - View	**ビューアー**です。	• 運用管理クライアント

13.3.2　監視ツリーの自動生成

JP1/IM の統合スコープでは，監視ツリーを使用してシステムを監視します。監視ツリーは，JP1/IM の機能を使用して，自動生成できます。

アプリケーションサーバシステムを集中監視する場合，次の 2 種類の監視ツリーが自動生成できます。これによって，JP1/IM と連携したアプリケーションサーバシステムを含んだシステム全体の障害監視が容易に実現できるようになります。

- 業務指向ツリー
- サーバ指向ツリー

なお，監視ツリーは，運用開始前にあらかじめ自動生成しておきます。

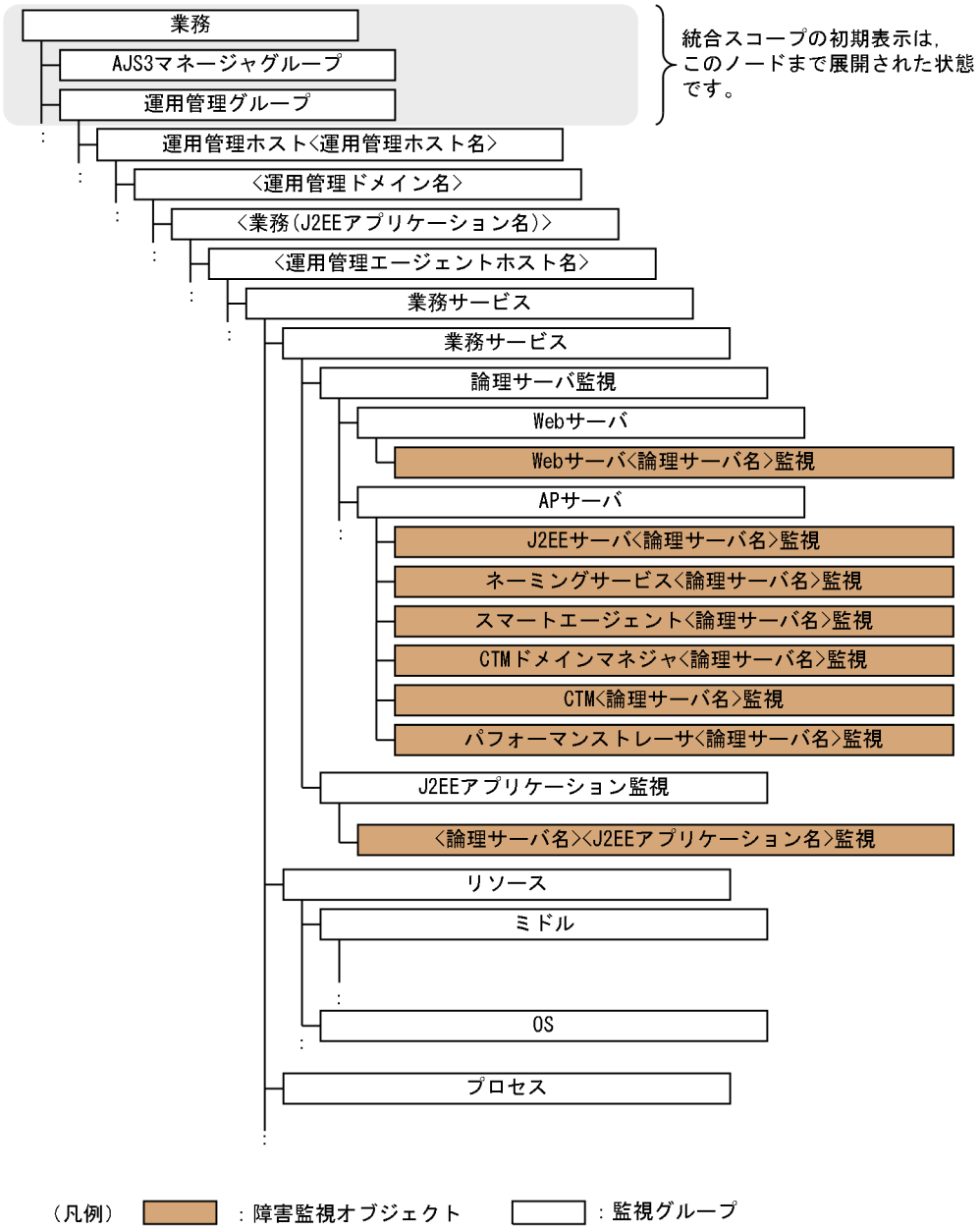
監視ツリーの自動生成の設定については，「13.4.1　監視ツリーの自動生成の設定」を参照してください。

(1)　業務指向ツリー

業務指向ツリーとは，アプリケーションサーバシステム上で稼働している J2EE アプリケーションを主体に構成されている監視ツリーです。どの業務で問題が発生しているのか，業務の視点で障害が監視できるようになります。

業務指向ツリーの構成を次に示します。初期表示は，「運用管理グループ」ノードまでが展開された状態になります。

図 13-3　業務指向ツリーの構成



業務指向ツリーの各ノードについて説明します。

表 13-3　業務指向ツリーのノード

ノード名※1※2	説明
業務	業務指向ツリーのルートノードです。
AJS3 マネージャグループ	JP1/AJS 用のルートノードです。
運用管理グループ	アプリケーションサーバ用のルートノードです。
運用管理ホスト<運用管理ホスト名>	運用管理ホストの監視グループです。運用管理ホストの名称が表示されます。

ノード名※1※2	説明
<運用管理ドメイン名>	運用管理ドメインの監視グループです。運用管理ドメインの名称が表示されます。
<業務 J2EE アプリケーション名>※3	業務に対応する監視グループです。J2EE アプリケーション（EAR）のプロパティである Display Name プロパティの値が表示されます。統合スコープから，わかりやすい業務名に必要に応じて変更できます。
<運用管理エージェントホスト名>	論理サーバが稼働するホストの監視グループです。アプリケーションサーバの運用管理エージェントが配置されているホストのホスト名が表示されます。
業務サービス	JP1/IM で管理しているほかの業務指向ツリーと階層を合わせるためのノードです。
アプリケーションサーバ業務サービス	サーバ指向ツリーを使用する場合に JP1/AJS の監視ノードと区別するためのノードです。
論理サーバ監視	論理サーバに対応する監視グループです。論理サーバを監視する Web サーバ，および AP サーバの監視グループは，このノードの下に配置されます。
Web サーバ※3	Web サーバに対応する監視グループです。Web サーバを監視するオブジェクトは，このノード下に配置されます。
<Web サーバ名>監視※3	Web サーバの障害監視オブジェクトです。 該当する Web サーバに障害が発生すると，この監視オブジェクトの状態が変わります。
AP サーバ	アプリケーションサーバの監視グループです。Web サーバ以外を監視するオブジェクトは，このノード下に配置されます。
J2EE サーバ<論理サーバ名>監視	J2EE サーバまたはバッチサーバの障害監視オブジェクトです（バッチサーバの場合も「J2EE サーバ」と表示されます）。 J2EE アプリケーションが稼働している J2EE サーバの表示名，またはバッチサーバの表示名が表示されます。 該当する J2EE サーバまたはバッチサーバで障害が発生すると，この監視オブジェクトの状態が変わります。なお，J2EE サーバの場合，該当する J2EE アプリケーションが 1 ホスト内の複数の J2EE サーバに配置されている場合，この監視オブジェクトは複数生成されます。
ネーミングサービス<論理サーバ名>監視※3	ネーミングサービスの障害監視オブジェクトです。J2EE サーバに関連するネーミングサービスの表示名が表示されます。 該当するネーミングサービスに障害が発生すると，この監視オブジェクトの状態が変わります。
スマートエージェント<論理サーバ名>監視※3	スマートエージェントの障害監視オブジェクトです。同じ監視グループ内の OTS や CTM に関連するスマートエージェントの表示名が表示されます。 該当するスマートエージェントに障害が発生すると，この監視オブジェクトの状態が変わります。
CTM ドメインマネジャ名<論理サーバ名>監視※3	CTM ドメインマネジャの障害監視オブジェクトです。J2EE サーバに関連する CTM ドメインマネジャの表示名が表示されます。

ノード名※1※2			説明
			該当する CTM ドメインマネジャに障害が発生すると，この監視オブジェクトの状態が変わります。
		CTM<論理サーバ名>監視※3	CTM の障害監視オブジェクトです。J2EE サーバに関連する CTM の表示名が表示されます。 該当する CTM に障害が発生すると，この監視オブジェクトの状態が変わります。
		パフォーマンストレーサ<論理サーバ名>監視	パフォーマンストレーサの障害監視オブジェクトです。J2EE サーバに関連するパフォーマンストレーサの表示名が表示されます。 該当するパフォーマンストレーサに障害が発生すると，この監視オブジェクトの状態が変わります。
J2EE アプリケーション監視※3※4			J2EE アプリケーションの監視グループです。J2EE アプリケーションを監視するオブジェクトはこのノードの下に配置されます。
	<論理サーバ名><J2EE アプリケーション名>監視※3		J2EE サーバ内の J2EE アプリケーション監視オブジェクトです。J2EE サーバのサーバ名と J2EE アプリケーション名が表示されます。 該当する J2EE サーバ内の J2EE アプリケーションで障害が発生すると，この監視オブジェクトの状態が変わります。なお，この監視オブジェクトは，各 J2EE サーバ内の J2EE アプリケーションの数だけ生成されます。
ミドル			ミドル製品用の監視グループです。ミドル製品のリソースを監視するオブジェクトです。

注※1　＜　＞で囲まれている名称は，環境に応じて設定される名称です。

注※2　各ノードの名称は，統合スコープ上で任意の名称に変更できます。

注※3　バッチサーバの場合，該当しません。

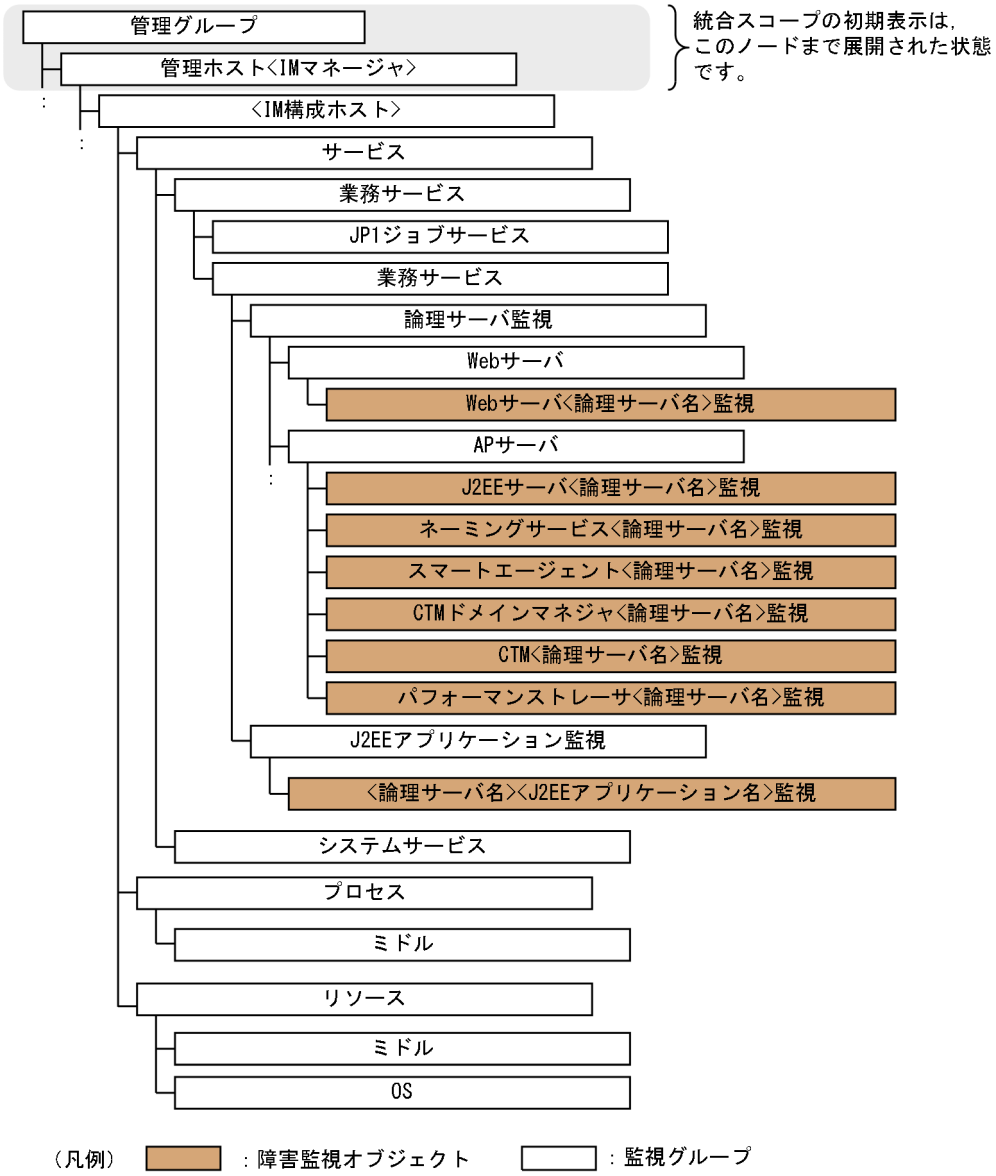
注※4　J2EE アプリケーション監視をするには，監視ツリーの自動生成で取得する内容にアプリケーション情報を含めるかどうかの設定（mngsvrutil.compat.monitoring_tree）が必要です。また，J2EE サーバが起動されていることが前提となります。

(2)　サーバ指向ツリー

サーバ指向ツリーとは，サーバマシン（ホスト）を主体に構成されている監視ツリーです。どのホストで問題が発生しているのか，ホストの視点で障害が監視できるようになります。

サーバ指向ツリーの構成を次に示します。初期表示は，「管理ホスト<IM マネージャ>」ノードまでが展開された状態になります。なお，IM 構成ホストとは，JP1/IM マネジャホストまたはエージェントホストのことです。

図 13-4　サーバ指向ツリーの構成



(3)　監視ツリー自動生成の処理の流れ

監視ツリーの自動生成処理の流れについて説明します。

JP1/IM - View で表示した統合スコープのビューアーから監視ツリーの自動生成を実行すると，JP1/IM から JP1/Base を介してアダプタコマンドが実行されます。**アダプタコマンド**とは，監視ツリーの自動生成に使用する定義情報を収集するために，JP1/Base によって実行されるコマンドです。あらかじめ JP1/Base に登録しておく必要があります。

アプリケーションサーバシステムに対してアダプタコマンドが実行されると，運用管理コマンド（mngsvrutil）が実行されます。J2EE アプリケーションを実行しているシステムでは，J2EE アプリケーションや J2EE サーバ，Web サーバなどの構成情報が取得されます。バッチアプリケーションを実行しているシステムでは，バッチサーバ，パフォーマンストレーサなどの構成情報が取得されます。

JP1/IM では，それぞれのアダプタコマンドによって取得された情報を基に，監視ツリーの自動生成が実行されます。

13.3.3　統合機能メニューからの運用管理ポータルの表示（Windows の場合）

この機能は Windows の場合だけ使用できます。

JP1/IM では「統合機能メニュー」画面を提供しています。この画面には，JP1/IM と連携するプログラムの一覧が表示されます。それぞれのプログラムは，「統合機能メニュー」から起動できます。

JP1/IM と連携すると，アプリケーションサーバの運用管理ポータル画面が「統合機能メニュー」画面から表示できるようになります。運用管理ポータル画面は，自動ログインを設定しているかによって，次のどちらかの画面が表示されます。

- 運用管理ポータルへのログイン画面
  自動ログインを設定していない場合に表示されます。
- 運用管理ポータル
  自動ログインを設定している場合に表示されます。

自動ログインの設定は，JP1/IM 連携用モニタ起動設定ファイル（.mngsvrmonitorrc）で設定します。モニタ起動コマンドのセットアップについては，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「付録 E.1　モニタ起動コマンドのセットアップ」を参照してください。また，統合機能メニューの詳細については，マニュアル「JP1/Integrated Management - Manager 運用ガイド」を参照してください。

13.3.4　障害の監視の仕組み

JP1/IM の統合コンソールと統合スコープを使用して，アプリケーションサーバシステムで発生した障害を監視します。

(1)　障害の監視の仕組み

統合コンソールおよび統合スコープを使用することで，アプリケーションサーバシステム以外の要因で発生した障害も同じ画面で確認できるようになり，障害の影響範囲も特定しやすくなります。さらに，Windows の場合は，アプリケーションサーバシステムで発生した障害の，調査および対処をするとき，障害の内容に応じたアプリケーションサーバの運用管理ポータル画面を，統合コンソールまたは統合スコープから直接表示できます。

JP1/IM と連携して障害を監視する場合は，次の準備が必要です。

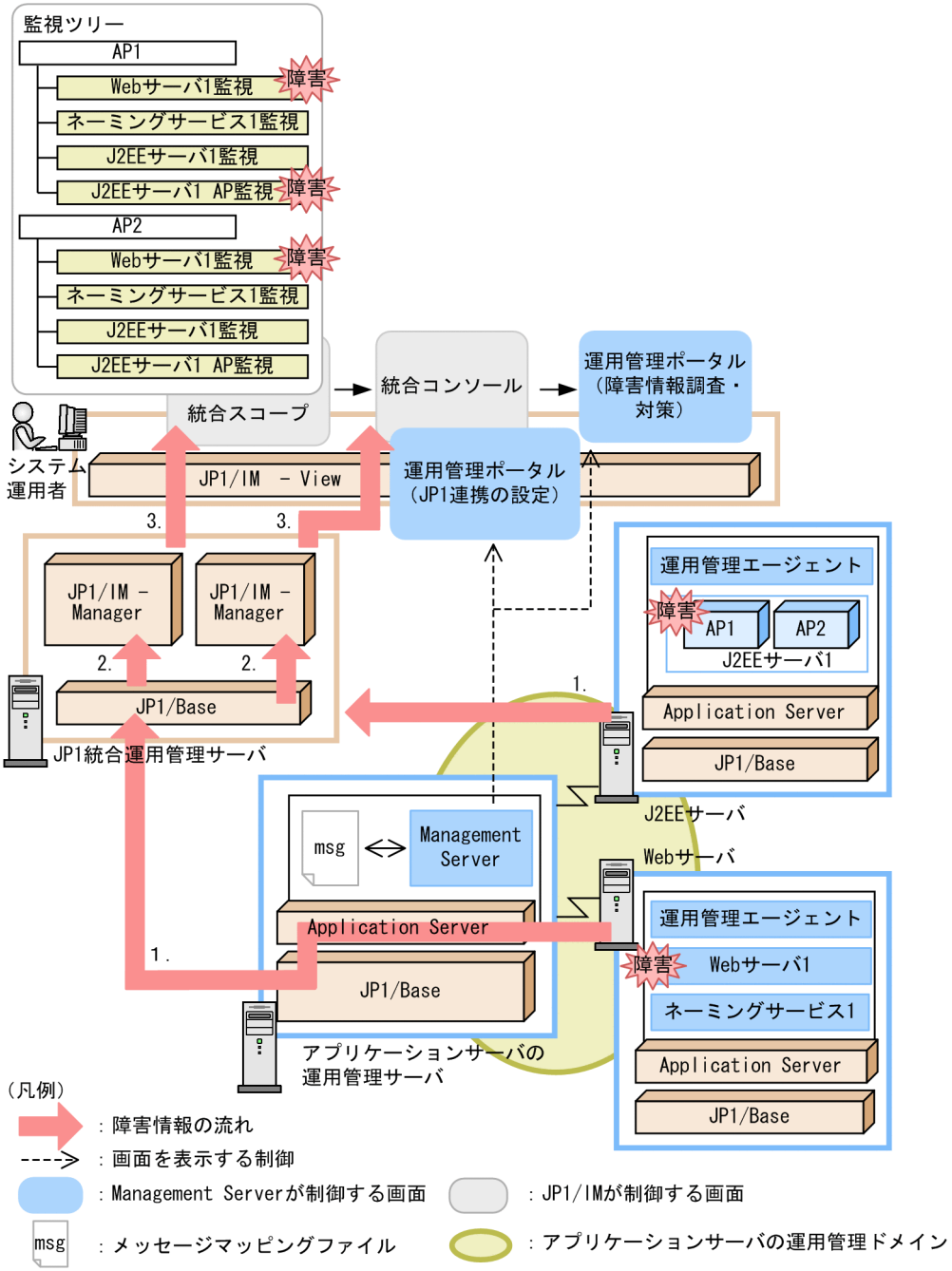
JP1/IM と連携して障害監視をするために必要な準備

- アプリケーションサーバシステムで発生する，どのようなエラーやインフォメーション情報を JP1/IM で監視したいのかを決めて，アプリケーションサーバの運用管理ポータル（「JP1 連携の設定」画面）から定義しておきます（**JP1 イベントのフィルタリング設定**）。
- アプリケーションサーバシステムが出力するどのメッセージを JP1 イベント発行対象にするかを運用方法に合わせて決定して，メッセージマッピング定義ファイルで設定，編集します。メッセージマッピング定義ファイルでの設定については，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「8.3　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」を参照してください。
- アプリケーションサーバシステムが出力するメッセージごとの重要度を運用方法に合わせて決定して，運用管理ポータル（論理サーバの環境設定）で設定，編集します。設定，編集は任意です。運用管理ポータル（論理サーバの環境設定）での設定については，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「8.2.7　JP1 連携の設定（Management Server の設定)」，および「10.8.25　JP1 連携の設定（J2EE サーバ)」を参照してください。
- JP1 イベント発行対象のメッセージ，およびメッセージごとの重要度の両方を設定したい場合，メッセージマッピング定義ファイルと運用管理ポータルの両方で設定が必要です。
- 統合スコープを使用した障害監視をする場合は，監視ツリーを生成しておきます。JP1/IM の統合スコープに表示する監視ツリーは，各ホストの JP1/Base によって収集されて JP1 統合運用管理サーバに集められた情報を基に自動生成されます。

  なお，統合コンソールだけを使用する場合は，監視ツリーの生成は不要です。

JP1/IM と連携した障害監視処理の流れを次の図に示します。

図 13-5　JP1/IM と連携した障害監視処理の流れ



1. アプリケーションサーバシステムで発生したトラブルに関する障害情報は，JP1/Base を経由して，JP1 イベントとして JP1 統合運用管理サーバの JP1/IM に通知されます。

   - J2EE サーバで発生したトラブルに関する障害情報は，J2EE サーバホスト内の JP1/Base を経由して JP1 統合運用管理サーバに発行されます。
   - J2EE サーバ以外の論理サーバ（Web サーバやネーミングサービスなど）で発生したトラブルに関する障害情報は，運用管理エージェントによってアプリケーションサーバの運用管理サーバに収集されます。その後，運用管理サーバの JP1/Base を経由して，JP1 統合運用管理サーバに発行されます。

なお，障害情報には，通知メッセージやJ2EEアプリケーションで出力されたユーザログのメッセージなども含まれます。

2. **J2EEサーバ，バッチサーバ，またはアプリケーションサーバの運用管理サーバからJP1イベントを受け取ったJP1統合運用管理サーバのJP1/IMによって，次のような処理が実行されます。**
   - JP1/IMの自動アクション機能を使って運用管理者への通知を設定している場合は，トラブルが発生したことが運用管理者に通知されます。
   - 統合スコープの監視ツリーの状態が変化します。例えば，Webサーバや J2EEサーバなど，業務に依存しない論理サーバレベルの障害の場合はすべての業務を対象にする該当論理サーバオブジェクトの状態が変化します。また，業務レベルの障害の場合は，該当するJ2EEアプリケーションの監視オブジェクトの状態が変化します。

3. **通知を受け取った運用管理者は，統合コンソールや統合スコープから障害に対処します。**
   障害の影響範囲を特定したい場合は，統合スコープを確認します。統合スコープには，障害の影響範囲が明示された監視ツリーが表示されます。また，詳細な障害の内容を知りたい場合は，統合コンソールを確認します。統合コンソールには，エラーメッセージなどが表示されます。さらに，Windowsの場合，アプリケーションサーバシステムで障害の原因究明や対策が必要なときは，Management Serverの運用管理ポータルの該当画面を表示して，調査や対策を実施します。

JP1イベントとして発行されるアプリケーションサーバシステムの障害情報を次に示します。

表13–4　JP1イベントとして発行されるアプリケーションサーバシステムの障害情報

イベントの種類	説明
運用管理システムJP1イベント	Management Serverが出力した通知・障害メッセージをJP1イベントに変換したものです。変換は，メッセージマッピングファイルで指定したルールに従って実行されます。 主な内容は，J2EEサーバ，バッチサーバ，Webサーバ，ネーミングサービス，CTMなどの論理サーバでプロセス停止レベルの障害が発生した時や，各サーバが起動または停止した時の通知メッセージです。 発行されるJP1イベントのIDは，00012000～0001207F（未割当てのIDあり）です。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「8.3.4　JP1イベントへの変換」を参照してください。 なお，Management Serverの起動中に出力されるメッセージについては，JP1イベントが発行されない場合があります。
J2EEサーバシステムJP1イベント	J2EEサーバまたはバッチサーバが出力した通知・障害メッセージをJP1イベントに変換したものです。変換は，メッセージマッピングファイルで指定したルールに従って実行されます。 発行されるJP1イベントのIDは，00012080～000120CF（未割当てのIDあり）です。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「8.3.4　JP1イベントへの変換」を参照してください。 なお，Management Serverの起動中に出力されるメッセージについては，JP1イベントが発行されない場合があります。
J2EEサーバユーザJP1イベント	J2EEアプリケーションが，ユーザログ出力機能※によって出力した通知・障害メッセージをJP1イベントに変換したものです。

注　JP1/Base のログファイルトラップを使用すると，アプリケーションサーバのログファイルに出力される情報を JP1 イベントに変換できます。この場合，JP1/Base のログファイルトラップで監視するログファイルの設定が必要になります。詳細は，「13.4.3　監視するログファイルの設定」を参照してください。

注※　ユーザログ出力機能については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「8.　アプリケーションのユーザログ出力」を参照してください。

(2)　運用管理ポータルの表示のされ方（Windows の場合）

統合コンソールまたは統合スコープからは，次のようにして運用管理ポータルの画面が表示されます。なお，統合コンソールまたは統合スコープからの，運用管理ポータル画面の表示は Windows の場合だけできます。

- ログイン方法の設定

  統合コンソールまたは統合スコープから運用管理ポータルを表示する場合，自動ログインするか，ログイン画面を表示して手動ログインするかを選択できます。自動ログインをするためには，あらかじめ設定ファイル（.mngsvrmonitorrc）に管理ユーザのユーザ ID とパスワードを指定しておく必要があります。ファイルの指定方法については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「8.2.17　.mngsvrmonitorrc（JP1/IM 連携用モニタ起動コマンドの設定ファイル)」を参照してください。

- 論理サーバに障害が発生した場合に表示される画面

  J2EE サーバ，バッチサーバ，Web サーバなどの論理サーバレベルの障害が発生した場合は，次に示す状態の「論理サーバの稼働状況」画面が表示されます。

  - ツリーペインには，サーバビューが表示されます。ツリーは，障害が発生した論理サーバまで展開された状態で表示されます。ただし，論理サーバを特定できない障害（一括起動操作に関する障害など）が発生している場合は，ツリーは展開されません。
  - ボディペインには，ツリーペインのドメインを選択した場合に表示される「論理サーバの稼働状況」画面が表示されます。

- J2EE アプリケーションに障害が発生した場合に表示される画面

  J2EE アプリケーションで障害が発生した場合は，次に示す状態の「論理サーバのアプリケーション管理」画面が表示されます。

  - ツリーペインのツリーは，障害が発生した J2EE アプリケーションが稼働している J2EE サーバまで展開された状態で表示されます。
  - ボディペインには，J2EE サーバ下のアプリケーションを選択した場合に表示される「J2EE アプリケーションの開始／停止」画面が表示されます。

- そのほかの障害が発生した場合に表示される画面

  Management Server の障害など，サーバや J2EE アプリケーション以外の障害が発生した場合は，「運用管理ポータル」画面が表示されます。

13.4　システムの集中監視のための設定

この節では，JP1/IM と連携してシステムを集中監視するための設定について説明します。

JP1/IM と連携することで，システムの集中監視ができます。システムの集中監視は，業務システム全体のリソースの状態を監視することで，障害の発生を検知し，原因を究明して対策したりします。

JP1/IM と連携してシステムを集中監視する場合の設定内容を次に示します。

- **監視ツリーの自動生成の設定**

  JP1/IM でアプリケーションサーバの障害を監視するために必要な設定をして，JP1/IM - Manager の統合スコープに表示するアプリケーションサーバ用の監視ツリーを自動生成します。設定方法については，「13.4.1　監視ツリーの自動生成の設定」を参照してください。

- **障害監視の設定**

  JP1/IM - Manager の統合コンソール，または統合スコープから，アプリケーションサーバシステムで発生した障害を監視するための設定をします。設定方法については，「13.4.2　障害監視の設定」を参照してください。

なお，JP1/Base のログファイルトラップを使用して，アプリケーションサーバのログファイルに出力される情報を JP1 イベントに変換する場合は，監視するログファイルを設定します。監視できるログファイルについては，「13.4.3　監視するログファイルの設定」を参照してください。

13.4.1　監視ツリーの自動生成の設定

この節では，JP1/IM の統合スコープに表示するアプリケーションサーバ用の監視ツリーを自動生成するための設定について説明します。

まず，次の表に示すサーバおよびクライアントに，監視ツリーの自動生成に必要な JP1 の製品またはアプリケーションサーバの製品の構成ソフトウェアをインストール，セットアップしておいてください。なお，JP1/IM と連携する場合のシステム構成については，「13.3.1　システムの集中監視に必要なプログラム」を参照してください。

表 13–5　監視ツリーの自動生成に必要な製品または構成ソフトウェア

サーバ／クライアント	製品／構成ソフトウェア
J2EE サーバ	• Component Container • Manager
アプリケーションサーバの運用管理サーバ	• Manager • JP1/Base
JP1 統合運用管理サーバ	• JP1/Base • JP1/IM - Manager

サーバ／クライアント	製品／構成ソフトウェア
運用管理クライアント	• JP1/IM - View

アプリケーションサーバ用の監視ツリーを自動生成するための手順を次に示します。

1. アプリケーションサーバの運用管理サーバで，次の作業を実施します。
   - アダプタコマンドのセットアップ（(1)参照）
   - アダプタコマンドの実行環境の設定（(2)参照）
   - Management Server での環境設定（(3)参照）
2. JP1 統合運用管理サーバで，次の作業を実施します。
   - JP1/Base の構成定義の作成（(4)参照）
3. 運用管理クライアントで，次の作業を実施します。
   - 監視ツリーの自動生成（(5)参照）

(1) アダプタコマンドのセットアップ

監視ツリーの自動生成に使用する定義情報を収集するために，JP1/Base によって実行されるアダプタコマンドを JP1/Base に登録します。

アプリケーションサーバの運用管理サーバで mngsvr_adapter_setup コマンドを実行して，JP1/Base の環境にアプリケーションサーバ用のアダプタコマンド設定ファイルを追加します。なお，mngsvr_adapter_setup コマンドを実行するユーザには Administrator 権限（UNIX の場合は，root 権限）が必要です。mngsvr_adapter_setup コマンドの実行例を次に示します。

- Windows の場合

  <Application Server のインストールディレクトリ>¥manager¥bin¥mngsvr_adapter_setup -i -t IM_CS
- UNIX の場合

  /opt/Cosminexus/manager/bin/mngsvr_adapter_setup -i -t IM_CS

mngsvr_adapter_setup 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「mngsvr_adapter_setup（アダプタコマンドのセットアップとアンセットアップ)」を参照してください。

(2) アダプタコマンドの実行環境の設定

アダプタコマンドを実行するための環境は，Management Server の運用管理コマンドである mngsvrutil コマンドのクライアント側定義ファイル（.mngsvrutilrc）に設定します。

アプリケーションサーバの運用管理サーバで，OS ユーザ（SYSTEM ユーザ）のホームディレクトリ（UNIX の場合は，JP1/Base を起動した OS ユーザのホームディレクトリ）に，クライアント側定義ファイルを作成して，次のパラメタを指定します。

- mngsvrutil.connect.host
  Management Server のホスト名とポート番号を指定します。
- mngsvrutil.connect.userid
  Management Server の mngsvrctl setup コマンドで設定した管理ユーザ ID を指定します。
- mngsvrutil.connect.password
  Management Server の mngsvrctl setup コマンドで設定したパスワードを指定します。パスワードを設定していない場合，このパラメタの指定は省略できます。

クライアント側定義ファイルは，mngsvrutil コマンドのオプションのデフォルト値を設定するファイルです。なお，mngsvrutil コマンドの実行環境を変更したい場合は，Management Server の運用管理コマンドである mngsvrutil コマンドのサーバ側定義ファイル（mngsvrutil.properties）で変更します。

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「mngsvrutil（Management Server の運用管理コマンド）」を参照してください。クライアント側定義ファイル（.mngsvrutilrc）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「8.2.14　.mngsvrutilrc（mngsvrutil コマンドのクライアント側定義ファイル）」を参照してください。サーバ側定義ファイル（mngsvrutil.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「8.2.15　mngsvrutil.properties（mngsvrutil コマンドのサーバ側定義ファイル）」を参照してください。

参考

クライアント側定義ファイルはクライアントごとに個別のデフォルト値を設定する場合に使用します。すべてのクライアント共通のデフォルト値を設定したい場合は，クライアント側共通設定ファイル（mngsvrutilcl.properties）を使用してください。なお，両方のファイルを使用している場合は，クライアント側定義ファイルが適用されます。クライアント側共通設定ファイルは読み込まれません。

(3)　Management Server での環境設定

監視ツリーを自動生成する前に，Management Server で次の設定をしてください。

- Web システムの構成定義
  決定したシステム構成パターンで，Web システムを構築してください。Web システムの構築には，Smart Composer 機能のファイルとコマンドを使用します。
  Smart Composer 機能の簡易構築定義ファイルで，Web システムの構成や，Web システム内の論理サーバの環境を定義し，Smart Composer 機能のコマンドで，定義したファイルを基に Web システ

ムを構築します。詳細については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「4.6　バッチアプリケーションを実行するシステムの構築」を参照してください。

- J2EE サーバへの J2EE アプリケーションのインポート

  J2EE アプリケーションのインポートの詳細については，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「4.1.29　業務アプリケーションを設定して開始する（CUI 利用時）」を参照してください。

運用管理エージェントと Management Server の起動および停止については，「2.6　システムの起動と停止の設定」を参照してください。

なお，論理サーバの構成や J2EE アプリケーションの数を変更した場合は，もう一度監視ツリーを自動生成する必要があります。

(4)　JP1/Base の構成定義の作成

JP1/IM が管理するシステムの構成を定義する JP1/Base の構成定義ファイルを作成します。また，JP1/Base の構成定義ファイルに定義したホストへ，定義情報を配布します。

JP1/Base の構成定義の作成手順を次に示します。なお，詳細については，マニュアル「JP1/Integrated Management - Manager 構築ガイド」のシステムの構成定義情報の設定に関する説明を参照してください。

1. JP1/Base の構成定義ファイルを作成して，システムの構成を定義します。

   構成定義ファイル（jbs_route.conf ファイル）は次の場所にインストールされています。

   - Windows の場合

     <JP1/Base のインストールディレクトリ>¥jp1base¥conf¥route¥jbs_route.conf

   - UNIX の場合

     /etc/opt/jp1base/conf/route/jbs_route.conf

   構成定義ファイルの内容を次に示します。

   構成定義ファイルの内容

   [JP1 統合運用管理サーバマシンのホスト名]
   J2EE サーバマシンのホスト名
   アプリケーションサーバの運用管理サーバマシンのホスト名

2. jbsrt_distrib コマンドを実行して，構成定義情報を配布します。

(5)　監視ツリーの自動生成

運用管理クライアントにある JP1/IM - View で，監視ツリーを自動生成します。

監視ツリーを自動生成するための手順を次に示します。監視ツリーの自動生成は，JP1/IM - View がインストールされた運用管理クライアント（Windows）で実施してください。JP1/IM の起動と停止，ログイ

ンとログアウトなどの操作については，マニュアル「JP1/Integrated Management - Manager 運用ガイド」を参照してください。

1. Windows の［スタート］メニューから［JP1_Integrated Manager - View］-［監視ツリー編集］を選択します。

   JP1/IM - View の［監視ツリー（編集中）］画面が起動されます。

2. ［オプション］ −［ツリーの自動生成］を選択します。

   統合スコープにログインするための［ログイン］画面が表示されるため，ログイン処理をすると，［自動生成-構成選択］画面が表示されます。

3. 自動生成する監視ツリーのモデルとして［業務指向ツリー］または［サーバ指向ツリー］を選択して，［生成］ボタンをクリックします。

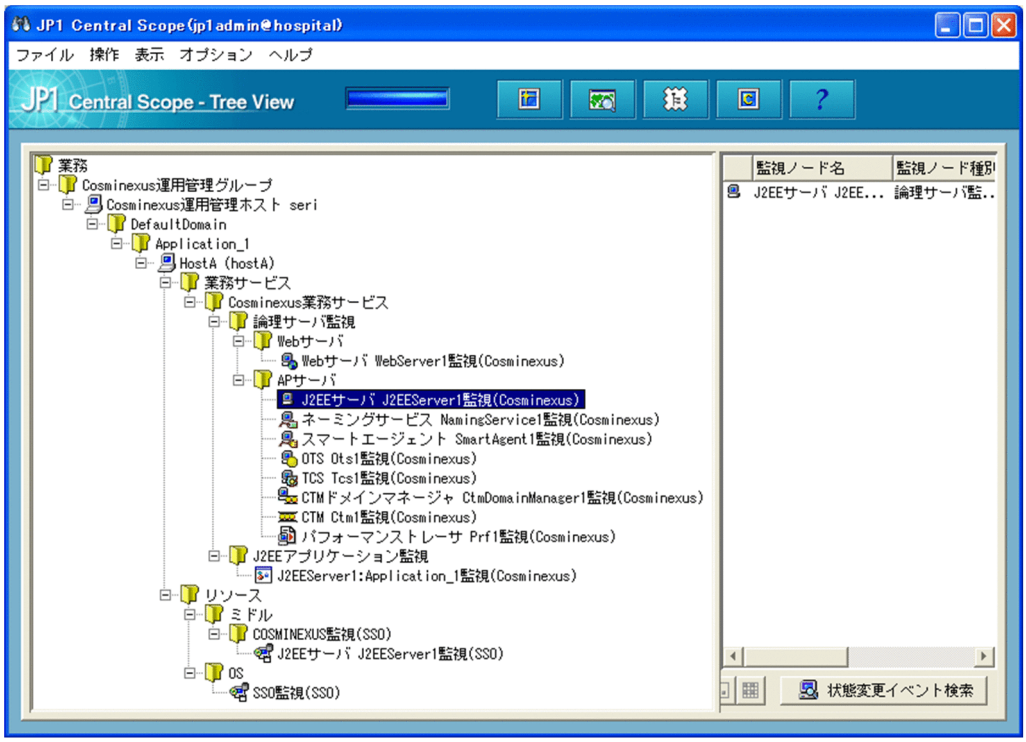
   監視ツリーが自動生成されます。

4. ［監視ツリー(編集中)］画面で，［ファイル］ −［サーバのツリーを更新］を選択します。

   確認ダイアログが表示されるので，［はい］ボタンをクリックすると，生成された監視ツリーの情報が JP1 統合運用管理サーバに反映されます。

   JP1/IM - View でアプリケーションサーバの業務指向ツリーを生成および編集した場合の例を次の図に示します。

図 13-6　アプリケーションサーバの業務指向ツリーの生成，編集例



13.4.2　障害監視の設定

ここでは，JP1/IM の統合コンソールまたは統合スコープから，アプリケーションサーバシステムで発生した障害を監視するための設定について説明します。

まず，次の表に示すサーバおよびクライアントに，障害監視に必要な JP1 の製品またはアプリケーションサーバの製品の構成ソフトウェアをインストール，セットアップしておいてください。なお，JP1/IM と連携する場合のシステム構成については，「13.3.1　システムの集中監視に必要なプログラム」を参照してください。

表 13-6　障害監視に必要な製品または構成ソフトウェア

サーバ／クライアント	製品／構成ソフトウェア
J2EE サーバ	• Component Container • JP1/Base
アプリケーションサーバの運用管理サーバ	• JP1/Base
JP1 統合運用管理サーバ	• JP1/Base • JP1/IM - Manager
運用管理クライアント	• JP1/IM - View

障害監視の設定手順を次に示します。

1. アプリケーションサーバの運用管理サーバで，次の作業を実施します。
   - アプリケーションサーバの JP1 イベント発行の設定（(1)参照）
   - JP1/Base のイベントサーバ名の設定（(2)参照）
2. J2EE サーバで，次の作業を実施します。
   - JP1/Base のイベントサーバ名の設定（(2)参照）
3. JP1 統合運用管理サーバで，次の作業を実施します。
   - JP1/Base の構成定義の作成（(3)参照）

(1)　アプリケーションサーバの JP1 イベント発行の設定

アプリケーションサーバシステムで発生するエラーやインフォメーション情報のうち，どのようなエラーやインフォメーション情報を JP1 イベントとして発行するかを設定します。

- Management Server 用 JP1 イベントの設定
- J2EE サーバ用 JP1 イベントの設定
- J2EE ユーザ用 JP1 イベントの設定

それぞれの設定方法について説明します。また，ユーザ JP1 イベントの推奨設定についても説明します。簡易構築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「4.3　簡易構築定義ファイル」を参照してください。

(a)　Management Server 用 JP1 イベントの設定

Management Server 用 JP1 イベントの設定手順を次に示します。

1. システム JP1 イベント発行の設定をします。

   mserver.properties で，com.cosminexus.mngsvr.jp1event.enabled キーで，Management Server が検知した障害や通知を JP1 イベントとして発行するかどうかを指定します。また，JP1 イベントの重大度ごとに，JP1 イベントを発行するかどうかを指定します。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。

2. メッセージマッピングファイルを設定します。

   Management Server が検知した障害や通知を，メッセージマッピングファイルで，メッセージごとにどの重大度の JP1 イベントに変換するかを定義します。なお，マッピング定義がなく JP1 イベントの重大度を割り当てられないメッセージは，JP1 イベントとして発行されません。

   ファイル名は「mserver.jp1event.system.mapping.properties」です。マッピング関係を定めた標準的なファイルを提供しているので，必要に応じて，テキストエディタなどで編集してください。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

   - UNIX の場合

     /opt/Cosminexus/manager/config

   Management Server 用メッセージマッピング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「8.3.1　mserver.jp1event.system.mapping.properties（Management Server 用メッセージマッピングファイル)」を参照してください。

3. Management Server を再起動します。

   再起動後に，設定した情報が有効になります。

(b)　J2EE サーバ用 JP1 イベントの設定

J2EE サーバ用 JP1 イベントの設定手順を次に示します。

1. システム JP1 イベント発行の設定をします。

   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.manager.agent.JP1EventAgent.enabled パラメタで「true」を指定します。デフォルトでは，「false」が指定されています。また，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 JP1 イベントの重大度ごとに，JP1 イベントを発行するかどうかを指定します。

- manager.jp1event.system.filtering.severity.emergency
- manager.jp1event.system.filtering.severity.alert
- manager.jp1event.system.filtering.severity.critical
- manager.jp1event.system.filtering.severity.error
- manager.jp1event.system.filtering.severity.warning
- manager.jp1event.system.filtering.severity.notice
- manager.jp1event.system.filtering.severity.information

2. Management Server および運用管理エージェントを再起動します。

再起動後に，設定した情報が有効になります。

3. メッセージマッピングファイルを設定します。

J2EE サーバが検知した障害や通知を，メッセージマッピングファイルで，メッセージごとにどの重大度の JP1 イベントに変換するかを定義します。なお，マッピング定義がなく JP1 イベントの重大度を割り当てられないメッセージは，JP1 イベントとして発行されません。メッセージマッピングファイルには，J2EE サーバ共通用と J2EE サーバ個別用があります。

J2EE サーバ共通用メッセージマッピングファイル

マシン内の J2EE サーバ共通用で，J2EE サーバが検知した障害や通知を JP1 イベントに変換するときに使用するファイルです。ファイル名は「manager.jp1event.system.mapping.properties」です。マッピング関係を定めた標準的なファイルを提供しているので，必要に応じて，テキストエディタなどで編集してください。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

UNIX の場合

/opt/Cosminexus/manager/config

J2EE サーバ個別用メッセージマッピングファイル

J2EE サーバごとに，J2EE サーバが検知した障害や通知を JP1 イベントに変換したいときに使用するファイルです。このファイルと J2EE サーバ共通用メッセージマッピングファイルが作成されている場合は，このファイルでの指定が優先されます。

ファイル名は「manager.<論理サーバ名>.jp1event.system.mapping.properties」です。このファイルは，テキストエディタなどを使用してユーザが作成してください。パラメタの記述形式は，J2EE サーバ共通用メッセージマッピングファイルと同じです。

メッセージマッピング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「8.3　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」を参照してください。

4. J2EE サーバを再起動します。

再起動後に，設定した情報が有効になります。

(c) J2EE ユーザ用 JP1 イベントの設定

J2EE ユーザ用 JP1 イベントの設定手順を次に示します。

1. ユーザ JP1 イベント発行の設定をします。

   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，次のパラメタで J2EE アプリケーションのユーザログを JP1 イベントとして発行するかどうかを指定します。マッピングの設定で，ユーザログのログレベルに JP1 イベントの重大度を割り当てたログが JP1 イベントの発行対象になります。

   - manager.jp1event.user.mapping.level.severe
   - manager.jp1event.user.mapping.level.warning
   - manager.jp1event.user.mapping.level.info
   - manager.jp1event.user.mapping.level.config
   - manager.jp1event.user.mapping.level.fine
   - manager.jp1event.user.mapping.level.finer
   - manager.jp1event.user.mapping.level.finest

   また，出力するユーザログをメッセージ ID などでフィルタリングできます。フィルタリングは，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，次のパラメタで指定します。

   - manager.jp1event.user.filtering.enabled
   - manager.jp1event.user.filtering.filter

   なお，ユーザ JP1 イベントの推奨設定については，「13.4.2(1)(d) ユーザ JP1 イベントの推奨設定」を参照してください。

2. ロガーおよびハンドラを設定します。

   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，次のパラメタで J2EE アプリケーションのユーザログで使用するロガーおよびハンドラを指定します。ハンドラには JP1 イベント用のハンドラ（com.cosminexus.mngsvr.externals.jp1event.JP1EventHandler）を指定してください。

   - ejbserver.application.userlog.loggers
   - ejbserver.application.userlog.Logger.<ロガー名称>.level
   - ejbserver.application.userlog.Logger.<ロガー名称>.useParentHandlers
   - ejbserver.application.userlog.Logger.<ロガー名称>.filter
   - ejbserver.application.userlog.menu.handlers.
   - ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.path
   - ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.limit
   - ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.count

- ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.level
- ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.appname
- ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.msgid
- ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.separator
- ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.formatter
- ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.filter
- ejbserver.application.userlog.CJLogHandler.<ハンドラ名称>.encoding
- ejbserver.application.userlog.Logger.<ロガー名称>.handlers

3. Management Server および運用管理エージェントを再起動します。
   再起動後に，設定した情報が有効になります。

4. J2EE サーバを再起動します。
   再起動後に，設定した情報が有効になります。

(d) ユーザ JP1 イベントの推奨設定

「13.4.2(1)(c) J2EE ユーザ用 JP1 イベントの設定」で設定するユーザ JP1 イベントの推奨設定について説明します。なお，J2EE アプリケーションのユーザログ出力の詳細については，マニュアル「アプリケーションサーバ 機能解説 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。

- ユーザ JP1 イベントのマッピングの設定
  マッピングは次のように設定します。

ログレベル	JP1 イベント重大度
SEVERE	Error
WARNING	Warning
INFO	Information
CONFIG	Notice
FINE	なし
FINER	なし
FINEST	なし

- ユーザ JP1 イベントのフィルタリングの設定
  ログを取得するときにメッセージ ID を付けて，「MESSAGE_ID」でフィルタリングします。JP1 イベントを発行させたいメッセージ ID に対して「MESSAGE_ID EQU <メッセージ ID>」を OR で接続して記述します。

フィルタの設定

```
MESSAGE_ID EQU 0001
OR
MESSAGE_ID EQU 0003
```

J2EEアプリケーション

```
Logger logger = Logger.getLogger("user.application1");
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー１", "0001"));
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー２", "0002"));
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー３", "0003"));
```

この二つだけがJP1イベントとして発行される

また，J2EE アプリケーションごとに JP1 イベントの発行をフィルタリングしたい場合は，ログを取得するときに J2EE アプリケーション名を付けるか，デフォルトの J2EE アプリケーション名を設定して，「APPLICATION」でフィルタリングします。

フィルタの設定

```
APPLICATION EQU APP01
MESSAGE_ID EQU 0001
OR
APPLICATION EQU APP01
MESSAGE_ID EQU 0003
OR
APPLICATION EQU APP02
MESSAGE_ID EQU 0002
```

Application1（デフォルトのJ2EEアプリケーション名なし）

```
Logger logger = Logger.getLogger("user.application1");
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー１", "APP01", "0001"));
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー２", "APP02", "0002"));
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー３", "APP03", "0003"));
```

Application2（デフォルトのJ2EEアプリケーション名=APP02）

```
Logger logger = Logger.getLogger("user.application1");
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー１", "0001"));
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー２", "0002"));
logger.log(CJLogRecord.create(Level.SEVERE, "SEVEREレベルのエラー３", "0003"));
```

この三つだけがJP1イベントとして発行される

(2) JP1/Base のイベントサーバ名の設定

Windows Server Failover Cluster または HA モニタを使用したクラスタ構成の場合で，論理ホストの JP1/Base のイベントサーバを使用して JP1 イベントを発行するとき，アプリケーションサーバの運用管理サーバ，および J2EE サーバでイベントサーバ名を設定します。

(a) 運用管理サーバ用イベントサーバ名の設定

アプリケーションサーバの運用管理サーバでイベントサーバ名を設定するには，Management Server 環境設定ファイル（mserver.properties）の mngsvr.jp1event.event_server_name キーに，論理ホスト名または論理 IP アドレスを指定します。なお，設定した情報を有効にするために，設定後は，Management Server を再起動してください。

Management Server 環境設定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「8.2.6　mserver.properties（Management Server 環境設定ファイル)」を参照してください。

(b)　J2EE サーバ用イベントサーバ名の設定

J2EE サーバでイベントサーバ名を設定する場合，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，ejbserver.manager.jp1event.event_server_name パラメタで指定します。

(3)　JP1/Base の構成定義の作成

JP1/IM が管理するシステムの構成を定義する JP1/Base の構成定義ファイルを作成します。また，JP1/Base の構成定義ファイルに定義したホストへ，定義情報を配布します。

JP1/Base の構成定義の作成手順を次に示します。なお，詳細については，マニュアル「JP1/Integrated Management - Manager 構築ガイド」のシステムの構成定義情報の設定に関する説明を参照してください。

1. JP1/Base の構成定義ファイルを作成して，システムの構成を定義します。

   構成定義ファイル（jbs_route.conf）は次の場所にインストールされています。

   - Windows の場合
     <JP1/Base のインストールディレクトリ>¥jp1base¥conf¥route¥jbs_route.conf
   - UNIX の場合
     /etc/opt/jp1base/conf/route/jbs_route.conf

   構成定義ファイルの内容を次に示します。

   構成定義ファイルの内容

   [JP1 統合運用管理サーバマシンのホスト名]
   J2EE サーバマシンのホスト名
   アプリケーションサーバの運用管理サーバマシンのホスト名

2. jbsrt_distrib コマンドを実行して，構成定義情報を配布します。

13.4.3　監視するログファイルの設定

JP1/Base のログファイルトラップを使用して監視できる，アプリケーションサーバのログファイルについて説明します。JP1/Base のログファイルトラップでは，アプリケーションサーバのログファイルに出力される情報を JP1 イベントに変換します。JP1/Base のログファイルトラップの設定については，マニュアル「JP1/Base 運用ガイド」を参照してください。

アプリケーションサーバが出力するログのうち，JP1 イベントに変換できるログファイルと，そのファイルの内容を次の表に示します。

表 13–7　JP1/Base で監視できるログファイルとファイルの内容

プロセス	ログ出力先およびログファイル名※1	ファイル形式※2	改行コード	説明
J2EE サーバ	• Windows の場合 <ejb.server.log.directory>※3¥cjmessage[n].log • UNIX の場合 <ejb.server.log.directory>※3/cjmessage[n].log	WRAP2 新規出現タイプ	CR+LF	J2EE サーバの稼働ログ
	• Windows の場合 （Connector 1.0 仕様のリソースアダプタ） <ejb.server.log.directory>※3¥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>※3¥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>※3/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>※3/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	WRAP2 新規出現タイプ	CR+LF	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
	• 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※4 • UNIX の場合 <Application Server のインストールディレクトリ>/CC/admin/logs/cjmessage[n].log※4	WRAP2 新規出現タイプ	CR+LF	サーバ管理コマンドの稼働ログ
PRF	• Windows の場合 Eventlog • UNIX の場合 syslog	－	－	OS のログ
Management Server※7	<Manager のログ出力ディレクトリ>※6¥mngsvr.exe.[n].log	WRAP2 新規出現タイプ	CR+LF	Management Server サービス（mngsvr コマンド）のログ（Windows だけ）

プロセス	ログ出力先およびログファイル名※1	ファイル形式※2	改行コード	説明
	• Windows の場合 <Manager のログ出力ディレクトリ>※6¥mngsvr[n].log • UNIX の場合 <Manager のログ出力ディレクトリ>※6/mngsvr[n].log	WRAP2 新規出現タイプ	CR+LF	Management Server のログ
運用管理エージェント※8	<Manager のログ出力ディレクトリ>※6¥adminagentsv.exe.[n].log	WRAP2 新規出現タイプ	CR+LF	運用管理エージェントサービス（adminagentsv コマンド）のログ（Windows だけ）
	• Windows の場合 <Manager のログ出力ディレクトリ>※6¥adminagent[n].log • UNIX の場合 <Manager のログ出力ディレクトリ>※6/adminagent[n].log	WRAP2 新規出現タイプ	CR+LF	運用管理エージェントのログ
CTM	• Windows の場合 Eventlog • UNIX の場合 syslog	−	−	OS のログ
ネーミングサービス	• Windows の場合 <ejb.server.log.directory>※3¥ejb¥<J2EE サーバ名>¥logs¥TPB¥logj¥hvimsgj[n].txt • UNIX の場合 <ejb.server.log.directory>※3/ejb/<J2EE サーバ名>/logs/TPB/logj/hvimsgj[n].txt	WRAP1 新規出現タイプ	CR+LF	ネーミングサービスのメッセージログ（ネーミングサービスをインプロセスで起動した場合）
	• Windows の場合 <VBROKER_ADM ディレクトリ(vbroker_adm)>※5¥..¥logj¥hvimsgj[n].txt • UNIX の場合 <VBROKER_ADM ディレクトリ(vbroker_adm)>※5/../logj/hvimsgj[n].txt	WRAP1 新規出現タイプ	CR+LF	ネーミングサービスのメッセージログ（ネーミングサービスをアウトプロセスで起動した場合）

プロセス	ログ出力先およびログファイル名※1	ファイル形式※2	改行コード	説明
CORBA クライアントアプリケーション	• Windows の場合 <VBROKER_ADM ディレクトリ(vbroker_adm)>※5¥..¥logj¥hvimsgj[n].txt • UNIX の場合 <VBROKER_ADM ディレクトリ(vbroker_adm)>※5/../logj/hvimsgj[n].txt	WRAP1 新規出現タイプ	CR+LF	CORBA クライアントアプリケーションのメッセージログ
JavaVM	• Windows の場合 <ejb.server.log.directory>※3¥javalog[nn].log • UNIX の場合 <ejb.server.log.directory>※3/javalog[nn].log	WRAP2 新規出現タイプ	CR+LF	JavaVM の保守情報および GC のログ
	• Windows の場合 <ejb.server.log.directory>※3¥ehjavalog[nn].log • UNIX の場合 <ejb.server.log.directory>※3/ehjavalog[nn].log	WRAP2 新規出現タイプ	CR+LF	明示管理ヒープ機能のイベントログ
CJMS プロバイダ	• Windows の場合 **CJMSP ブローカーの起動コマンド（cjmsbroker コマンド）で-varhome オプションを指定しない場合** <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¥cjmsbroker_msg[n].log **CJMSP ブローカーの起動コマンド（cjmsbroker コマンド）で-varhome オプションを指定する場合** <-varhome オプションの設定値>¥instances¥<インスタンス名>¥log¥cjmsbroker_msg[n].log • UNIX の場合 **CJMSP ブローカーの起動コマンド（cjmsbroker コマンド）で-varhome オプションを指定しない場合** <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/cjmsbroker_msg[n].log **CJMSP ブローカーの起動コマンド（cjmsbroker コマンド）で-varhome オプションを指定する場合** <-varhome オプションの設定値>/instances/<インスタンス名>/log/cjmsbroker_msg[n].log	WRAP1 新規出現タイプ	CR+LF	CJMSP ブローカーのメッセージログ
	• Windows の場合 **CJMSP ブローカーの起動コマンド（cjmsbroker コマンド）で-varhome オプションを指定しない場合** <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¥cjmsbroker_err[n].log **CJMSP ブローカーの起動コマンド（cjmsbroker コマンド）で-varhome オプションを指定する場合**	WRAP1 新規出現タイプ	CR+LF	CJMSP ブローカーのエラーログ

プロセス	ログ出力先およびログファイル名※1	ファイル形式※2	改行コード	説明
	<-varhome オプションの設定値>¥instances¥<インスタンス名>¥log¥cjmsbroker_err[n].log • UNIX の場合 **CJMSP ブローカーの起動コマンド（cjmsbroker コマンド）で-varhome オプションを指定しない場合** <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/cjmsbroker_err[n].log **CJMSP ブローカーの起動コマンド（cjmsbroker コマンド）で-varhome オプションを指定する場合** <-varhome オプションの設定値>/instances/<インスタンス名>/log/cjmsbroker_err[n].log			
	• Windows の場合 <Application Server のインストールディレクトリ>¥CC¥cjmsp¥var¥admin¥log¥cjmsadmin_msg[n].log • UNIX の場合 <Application Server のインストールディレクトリ>/CC/cjmsp/var/admin/log/cjmsadmin_msg[n].log	WRAP1 新規出現タイプ	CR+LF	サーバ管理コマンド（cjmsicmd）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>¥CC¥cjmsp¥var¥admin¥log¥cjmsadmin_err[n].log • UNIX の場合 <Application Server のインストールディレクトリ>/CC/cjmsp/var/admin/log/cjmsadmin_err[n].log	WRAP1 新規出現タイプ	CR+LF	サーバ管理コマンド（cjmsicmd）のエラーログ
	• Windows の場合 <ejb.server.log.directory>※3¥cjms¥Cosminexus_JMS_Provider_RA¥cjmsra_msg[n].log • UNIX の場合 <ejb.server.log.directory>※3/cjms/Cosminexus_JMS_Provider_RA/cjmsra_msg[n].log	WRAP1 新規出現タイプ	CR+LF	CJMSP リソースアダプタのメッセージログ
	• Windows の場合 <ejb.server.log.directory>※3¥cjms¥Cosminexus_JMS_Provider_RA¥cjmsra_err[n].log • UNIX の場合 <ejb.server.log.directory>※3/cjms/Cosminexus_JMS_Provider_RA/cjmsra_err[n].log	WRAP1 新規出現タイプ	CR+LF	CJMSP リソースアダプタのエラーログ

プロセス	ログ出力先およびログファイル名※1	ファイル形式※2	改行コード	説明
Management Server および運用管理エージェント※7※8	• Windows の場合 <Manager のログ出力ディレクトリ>※6¥message¥mngmessage[n].log • UNIX の場合 <Manager のログ出力ディレクトリ>※6/ message/ mngmessage[n].log	WRAP2 新規出現タイプ	CR+LF	Management Server と運用管理エージェントの統合メッセージログ

（凡例）－：なし

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

読み込むログファイルのレコード形式です。

- WRAP1：ラップラウンドファイル（ラップラウンドして，再び先頭からデータを上書きするファイル）
- WRAP2：ラップラウンドファイル（ラップラウンドするとき，データを削除して再び先頭からデータを書き込むファイル）
- HTRACE：マルチプロセス対応トレースファイル（複数のプロセスが一組のトレースファイルを共有し，メモリマップドファイルを使用した固定サイズのファイル）。ログファイルへの書き込み方法は WRAP1 と同じです。
- 新規出現タイプ：ログファイルが通番に従い出現します。通番は[1]から始まり，順に[2]，[3]…とファイルが作成されます。なお，一度出現すれば，ファイル自体は削除されません。
- 全ファイル既出タイプ：定義したログファイル面数分のファイルがすべて作成されているタイプです。

注※3

<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。デフォルト値を OS ごとに次に示します。

Windows の場合：<Application Server のインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs

UNIX の場合：<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

注※4

ログファイルの出力先は，サーバ管理コマンド用オプション定義ファイルの usrconf.bat（Windows の場合），または usrconf（UNIX の場合）の ejbserver.log.directory キーで変更できます。

注※5

<VBROKER_ADM ディレクトリ(vbroker_adm)>は，環境変数 VBROKER_ADM に指定したディレクトリを指します。

注※6

<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。デフォルト値を OS ごとに次に示します。

Windows の場合：¥opt¥Cosminexus¥manager¥log です。

UNIX の場合：/opt/Cosminexus/manager/log です。

注※7

Management Server のログ，または統合メッセージログのどちらか一方を監視するようにしてください。

注※8

運用管理エージェントのログ，または統合メッセージログのどちらか一方を監視するようにしてください。

13.5　運用例

13.5.1　Web フロントシステムでの運用例

ここでは，Web フロントシステムで JP1/IM と連携し，アプリケーションサーバのログファイルを監視して，システムの集中監視をする運用例について説明します。

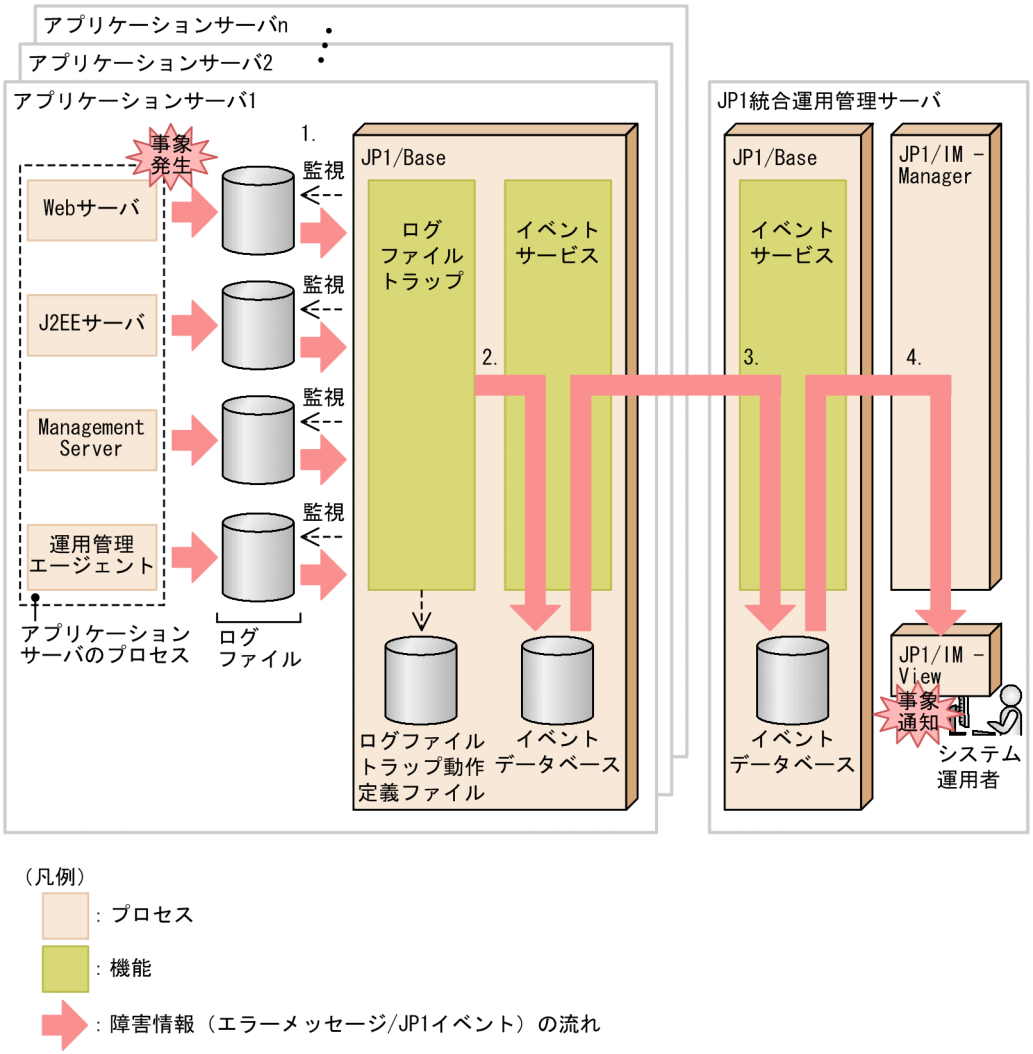
(1)　システム構成と運用例

JP1/IM と連携してシステムを集中監視すると，アプリケーションサーバのプロセスで発生した事象（エラーメッセージ）は，JP1 イベントに変換されて JP1 総合運用管理サーバへ転送され，システム運用者に通知されます。

この場合，アプリケーションサーバには JP1/Base，JP1/統合運用管理サーバには JP1/Base，JP1/IM - Manager，JP1/IM - View などを利用します。

Web フロントシステムでのシステム構成と運用例を次の図に示します。

図 13-7　Web フロントシステムでのシステム構成と運用例（JP1/IM を使用したシステムの集中監視）



1. アプリケーションサーバのプロセスで発生した事象（エラーメッセージ）が出力されたログファイルは，JP1/Base のログファイルトラップによって定期的に監視されます。
2. ログファイルトラップ動作定義ファイルの監視条件と一致すると，JP1/Base のログファイルトラップによってそのエラーメッセージは JP1 イベントに変換され，JP1/Base のイベントサービスによってイベントデータベースに蓄積されます。
3. JP1 イベントは，JP1/Base のイベントサービスによって JP1 統合運用管理サーバへ転送されます。
4. 通知を受けたシステム運用者は，JP1/IM - View を使って発生した事象を確認します。

このシステム構成を例にして，監視するログとメッセージ，および各ログを監視するためのログファイルトラップ動作定義ファイルの設定例を説明します。ここでは，代表的な OS の設定例について説明します。

(2)　Windows の場合の設定例

Windows の場合に監視するログファイルと，ログファイルトラップ動作定義ファイルの設定内容を次の表に示します。なお，ここで説明するログファイルの出力先はデフォルトになります。

表 13–8　監視するログファイルとログファイルトラップ動作定義ファイルの設定内容（Windows の場合）

項番	プロセス名	監視するログファイル名	ログファイルトラップ動作定義ファイルの設定内容			説明
			FILETYPE※1 の指定値	ACTDEF※2 の指定値	MARKSTR※3 の指定値	
1	Web サーバ（HTTP Server の場合）	<HWS のインストールディレクトリ>¥servers¥HWS_<サーバ名称>¥logs¥error.[n]	WRAP2 新規出現タイプ	"emerg" "alert" "crit" "error"	"File does not exist"	[n]は 001 から 005 まで作成されます。すべてのファイルを監視します。
		<アプリケーションサーバのインストールディレクトリ>¥CC¥web¥redirector¥logs¥hws_redirect[n].log	WRAP2 新規出現タイプ	"-E"	"KDJE39188-E" "KDJE41010-E"	[n]は 1 から 8 まで作成されます。すべてのファイルを監視します。
2	J2EE サーバ	<アプリケーションサーバのインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs¥cjmessage[n].log	WRAP2 新規出現タイプ	"-E" "KDJE52703-W" "KDJE31002-W" "KDJE34500-W"	"KDJE39022-E" "KDJE39051-E" "KDJE39057-E"	[n]は 1 から 2 まで作成されます。すべてのファイルを監視します。
		Oracle の場合 <アプリケーションサーバのインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs¥connectors¥DB_Connector_for_Oracle[n].log HiRDB の場合 <アプリケーションサーバのインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs¥connectors¥DB_Connector_for_HiRDB_Type4[n].log SQL Server の場合 <アプリケーションサーバのインストールディレクトリ>¥CC¥server¥public¥ejb¥<サーバ名称>¥logs¥connectors¥DB_	WRAP2 新規出現タイプ	"-E"	－	[n]は 1 から 4 まで作成されます。すべてのファイルを監視します。

項番	プロセス名	監視するログファイル名	ログファイルトラップ動作定義ファイルの設定内容			説明
			FILETYPE※1の指定値	ACTDEF※2の指定値	MARKSTR※3の指定値	
		Connector_for_SQLServer2005[n].log				
3	Management Server	<アプリケーションサーバのインストールディレクトリ>¥manager¥log¥message¥mngmessage[n].log	WRAP2 新規出現タイプ	"-E"	－	[n]は1から4まで作成されます。すべてのファイルを監視します。
4	運用管理エージェント					

（凡例）－：指定値がないことを示します。

注※1　読み込むログファイルのレコード形式を指定します。

- WRAP2：ラップラウンドファイル（ラップラウンドするとき，データを削除して再び先頭からデータを書き込むファイル）
- 新規出現タイプ：ログファイルが通番に従い出現します。通番は[1]から始まり，順に[2]，[3]…とファイルが作成されます。なお，一度出現すれば，ファイル自体は削除されません。

注※2　JP1イベントに変換するログデータを指定します。

注※3　存在しないURLへのアクセス，リクエストの中断など，ブラウザの操作で簡単に発生するエラーメッセージ（監視の対象外にしたいデータ）を指定します。

JP1/Baseのログファイルトラップの起動コマンド（jevlogstartコマンド）に，引数として設定するログファイルトラップ動作定義ファイルの作成例を次に示します。ログファイルトラップ動作定義ファイルは，監視するログファイルごとに作成します。

- error.[n]を監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
MARKSTR=”File does not exist”
ACTDEF=<Emergency>111A ”emerg”
ACTDEF=<Alert>111B ”alert”
ACTDEF=<Critical>111C ”crit”
ACTDEF=<Error>111D ”error”
```

- hws_redirect[n].logを監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
MARKSTR=”KDJE39188-E”
MARKSTR=”KDJE41010-E”
ACTDEF=<Error>112A ”-E”
```

- cjmessage[n].logを監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
MARKSTR=”KDJE39022-E”
MARKSTR=”KDJE39051-E”
MARKSTR=”KDJE39057-E”
ACTDEF=<Error>113A ”-E”
```

```
ACTDEF=<Warning>113B ”KDJE52703-W”
ACTDEF=<Warning>113C ”KDJE31002-W”
ACTDEF=<Warning>113D ”KDJE53850-W”
```

- DB_Connector_for_Oracle[n].log，DB_Connector_for_HiRDB_Type4[n].log，または DB_Connector_for_SQLServer2005[n].log を監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
ACTDEF=<Error>114A ”-E”
```

- mngmessage[n].log を監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
ACTDEF=<Error>115A ”-E”
```

(3) Linux（AMD64 & Intel EM64T）の場合の設定例

Linux（AMD64 & Intel EM64T）の場合に監視するログファイルと，ログファイルトラップ動作定義ファイルの設定内容を次の表に示します。なお，ここで説明するログファイルの出力先はデフォルトになります。

表 13-9 監視するログファイルとログファイルトラップ動作定義ファイルの設定内容（Linux（AMD64 & Intel EM64T）の場合）

項番	プロセス名	監視するログファイル名	ログファイルトラップ動作定義ファイルの設定内容			説明
			FILETYPE※1 の指定値	ACTDEF※2 の指定値	MARKSTR※3 の指定値	
1	Web サーバ（HTTP Server の場合）	<HWS のインストールディレクトリ>/servers/HWS_<サーバ名称>/logs/error.[n]	WRAP2 新規出現タイプ	"emerg" "alert" "crit" "error"	"File does not exist"	[n]は 001 から 005 まで作成されます。すべてのファイルを監視します。
		/opt/Cosminexus/CC/web/redirector/logs/hws_redirect[n].log	HTRACE 新規出現タイプ	"-E"	"KDJE39188-E" "KDJE41010-E"	[n]は 1 から 8 まで作成されます。すべてのファイルを監視します。
2	J2EE サーバ	/opt/Cosminexus/CC/server/public/ejb/<サーバ名称>/logs/cjmessage[n].log	WRAP2 新規出現タイプ	"-E" "KDJE52703-W" "KDJE31002-W" "KDJE34500-W"	"KDJE39022-E" "KDJE39051-E" "KDJE39057-E"	[n]は 1 から 2 まで作成されます。すべてのファイルを監視します。

項番	プロセス名	監視するログファイル名	ログファイルトラップ動作定義ファイルの設定内容			説明
			FILETYPE※1 の指定値	ACTDEF※2 の指定値	MARKSTR※3 の指定値	
		Oracle の場合 /opt/Cosminexus/CC/server/public/ejb/<サーバ名称>/logs/connectors/DB_Connector_for_Oracle[n].log HiRDB の場合 /opt/Cosminexus/CC/server/public/ejb/<サーバ名称>/logs/connectors/DB_Connector_for_HiRDB_Type4[n].log SQL Server の場合 /opt/Cosminexus/CC/server/public/ejb/<サーバ名称>/logs/connectors/DB_Connector_for_SQLServer2005[n].log	WRAP2 新規出現タイプ	"-E"	－	[n]は 1 から 4 まで作成されます。すべてのファイルを監視します。
3	Management Server	/opt/Cosminexus/manager/log/message/mngmessage[n].log	WRAP2 新規出現タイプ	"-E"	－	[n]は 1 から 4 まで作成されます。すべてのファイルを監視します。
4	運用管理エージェント					

（凡例）－：指定値がないことを示します。

注※1　読み込むログファイルのレコード形式を指定します。

- WRAP2：ラップラウンドファイル（ラップラウンドするとき，データを削除して再び先頭からデータを書き込むファイル）
- HTRACE：マルチプロセス対応トレースファイル（複数のプロセスが一組のトレースファイルを共有し，メモリマップドファイルを使用した固定サイズのファイル）。ログファイルへの書き込み方法は WRAP1 と同じです。
- 新規出現タイプ：ログファイルが通番に従い出現します。通番は[1]から始まり，順に[2]，[3]…とファイルが作成されます。なお，一度出現すれば，ファイル自体は削除されません。

注※2　JP1 イベントに変換するログデータを指定します。

注※3　存在しない URL へのアクセス，リクエストの中断など，ブラウザの操作で簡単に発生するエラーメッセージ（監視の対象外にしたいデータ）を指定します。

JP1/Base のログファイルトラップの起動コマンド（jevlogstart コマンド）に，引数として設定するログファイルトラップ動作定義ファイルの作成例を次に示します。ログファイルトラップ動作定義ファイルは，監視するログファイルごとに作成します。

- error.[n]を監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
MARKSTR=”File does not exist”
ACTDEF=<Emergency>111A ”emerg”
ACTDEF=<Alert>111B ”alert”
ACTDEF=<Critical>111C ”crit”
ACTDEF=<Error>111D ”error”
```

- hws_redirect[n].log を監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=HTRACE
MARKSTR=”KDJE39188-E”
MARKSTR=”KDJE41010-E”
ACTDEF=<Error>112A ”-E”
```

- cjmessage[n].log を監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
MARKSTR=”KDJE39022-E”
MARKSTR=”KDJE39051-E”
MARKSTR=”KDJE39057-E”
ACTDEF=<Error>113A ”-E”
ACTDEF=<Warning>113B ”KDJE52703-W”
ACTDEF=<Warning>113C ”KDJE31002-W”
ACTDEF=<Warning>1130 ”KDJE34500-W”
```

- DB_Connector_for_Oracle[n].log，DB_Connector_for_HiRDB_Type4[n].log，または DB_Connector_for_SQLServer2005[n].log を監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
ACTDEF=<Error>114A ”-E”
```

- mngmessage[n].log を監視するためのログファイルトラップ動作定義ファイルの作成例

```
FILETYPE=WRAP2
ACTDEF=<Error>115A ”-E”
```

14 ジョブによるシステムの自動運転（JP1/AJS との連携）

この章では，JP1/AJS と連携したシステムの自動運転の運用および設定について説明します。システムの自動運転では，サーバやアプリケーションの開始や停止のスケジュールをあらかじめ定義しておくことで，効率的なリソースの配分や業務の効率化，省略化を図れます。ジョブによるシステムの自動運転の運用に必要な機能の設定についても説明します。

なお，この章で説明する JP1 との連携機能は，Management Server を利用して運用することが前提になります。

14.1　この章の構成

この章の構成を次の表に示します。

表 14–1　この章の構成（ジョブによるシステムの自動運転（JP1/AJS との連携））

分類	タイトル	参照先
解説	ジョブによるシステムの自動運転（JP1/AJS との連携）の概要	14.2
設定	ジョブによるシステムの自動運転の設定	14.3

注　「実装」,「運用」,「注意事項」について，この機能固有の説明はありません。

14.2　ジョブによるシステムの自動運転（JP1/AJS との連携）の概要

システムの自動運転は，サーバやアプリケーションの開始や停止のスケジュールをあらかじめ定義しておくことで，効率的なリソースの配分や業務の効率化，省略化を図る運用方法です。JP1/AJS と連携することでジョブによるシステムの自動運転が実現できます。

JP1/AJS は，日々の業務の中から，定型的・定期的なものを自動化して，システム運用に掛かるコストを削減し，少ない人員で確実な運用を実現するためのプログラム群です。JP1/AJS では，システム運用の各作業を，**ジョブ**として扱います。ジョブは，コマンド，シェルスクリプト，バッチファイルなどの集まりに対応します。また，ジョブの実行順序を関連づけたものを，**ジョブネット**といいます。

JP1/AJS は，次の 3 種類のプログラムで構成されています。

- JP1/AJS - Manager
  ジョブネットやスケジュールの定義を保存して，ジョブネットの実行を管理するプログラムです。ジョブの実行時には，実行するジョブを JP1/AJS - Agent に転送し，実行状況，実行結果の情報を受け取ります。エージェントとしてジョブの実行もできます。
- JP1/AJS - Agent
  ジョブを実行するためのプログラムです。JP1/AJS - Manager から転送されたジョブを実行します。
- JP1/AJS - View
  GUI を使って JP1/AJS を操作するためのプログラムです。JP1/AJS - Manager に接続して，ジョブネットの定義や操作，実行状況や結果の表示などをします。

JP1/AJS の詳細については，マニュアル「JP1/Automatic Job Management System 導入ガイド」を参照してください。

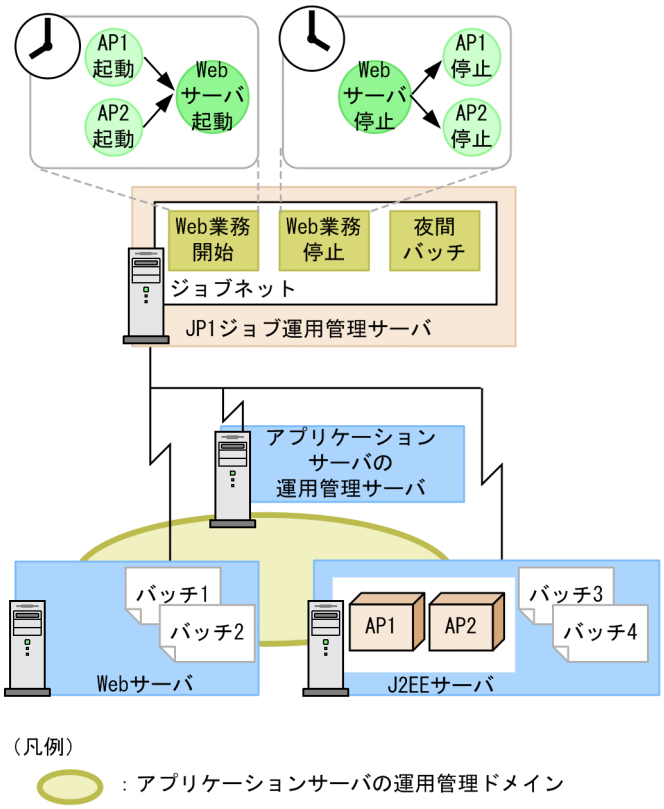
14.2.1　ジョブによるシステムの自動運転の仕組み

JP1/AJS と連携してシステムを自動運転にすると，アプリケーションサーバシステム内の論理サーバおよび J2EE アプリケーションの計画的な一括開始や一括停止ができるようになります。夜間に実行するバッチ処理，曜日によって異なる業務の切り替え，月末だけに実行する業務などをあらかじめスケジュール定義しておくことで，システムリソースを計画的に確保して，有効活用できるようになります。

(1)　JP1/AJS を使用したシステムの自動運転

JP1/AJS を使用したシステムの自動運転を次に示します。

図 14-1　JP1/AJS を使用したシステムの自動運転の概要



アプリケーションサーバシステムの運用を自動化するためには，まず，J2EE アプリケーションの開始や停止などの処理の一つ一つをジョブとして定義して，それらの実行順序をジョブネットとして定義します。アプリケーションサーバシステムの場合，ジョブとして，Management Server のコマンドである mngsvrutil コマンドを 1 実行単位で定義します。次に，運用計画に従って，ジョブネットごとに開始曜日や時刻などのスケジュールを設定して，JP1 ジョブ運用管理サーバの JP1/AJS に登録します。例えば，Web サーバと J2EE アプリケーション「AP1」「AP2」の起動と停止を，それぞれ AM8:00 と PM5:00 に定義します。設定した時刻になると，JP1/AJS によって，ジョブとして定義した mngsvrutil コマンドが実行されます。

また，JP1/IM と連携しておけば，JP1/AJS によって自動実行されたジョブに異常が発生した場合に，運用管理者の携帯電話などに通知する運用もできます。

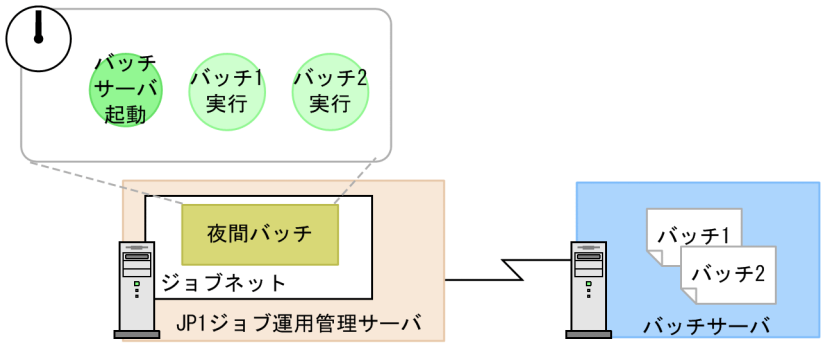
なお，Windows の場合，アプリケーションサーバでは，このような運用を実現するために，カスタムジョブの提供による，ジョブ定義支援機能を提供しています。

(2)　JP1/AJS を使用したバッチアプリケーションの自動実行

バッチアプリケーションは，アプリケーションサーバで構築するバッチサーバ上でも実行できます。バッチサーバで動作するバッチアプリケーションは，アプリケーションサーバで提供するバッチ実行コマンドで開始します。バッチ実行コマンドを JP1/AJS のジョブとして定義することで，バッチアプリケーションの自動実行が実現します。

JP1/AJS を使用したバッチアプリケーションの自動実行を次の図に示します。

図 14–2　JP1/AJS を使用したバッチアプリケーションの自動実行の概要



また，JP1/AJS では，BJEX または JP1/Advanced Shell と連携することもできます。BJEX または JP1/Advanced Shell と連携する場合は，JP1/AJS のジョブとして，BJEX または JP1/Advanced Shell のバッチジョブの実行コマンドを定義する必要があります。ジョブの定義については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「2.3.2　バッチアプリケーションの実行」を参照してください。

14.2.2　ジョブによるシステムの自動運転に必要なプログラム

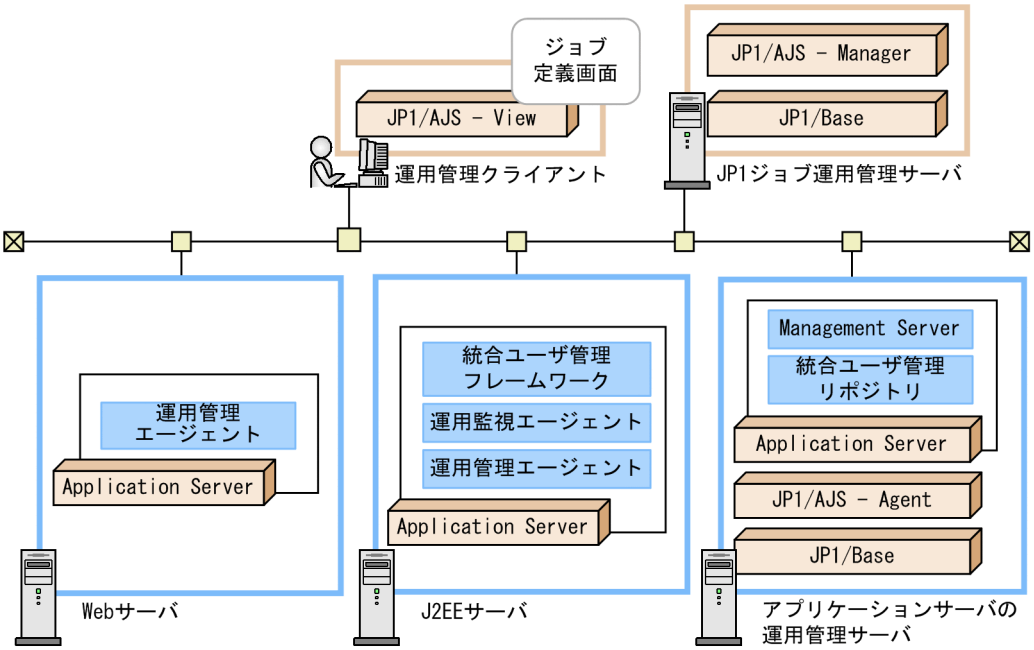
システムの自動運転は，JP1/AJS を中心とした JP1 のプログラム群と，アプリケーションサーバの運用管理機能を組み合わせて実現します。

JP1/AJS と連携してシステムの自動運転をする場合のシステム構築例を，J2EE サーバの場合とバッチサーバの場合に分けて説明します。

(1)　J2EE サーバの場合

J2EE アプリケーション実行環境で，システムを自動運転する場合のシステムの構成例を次の図に示します。

図 14-3　JP1/AJS と連携してシステムの自動運転をする場合のシステム構築例（J2EE サーバの場合）



それぞれの図中の JP1 の各プログラムの主な機能を次に示します。なお，アプリケーションサーバの管理機能については，「1.2　システムの目的と機能の対応」を参照してください。

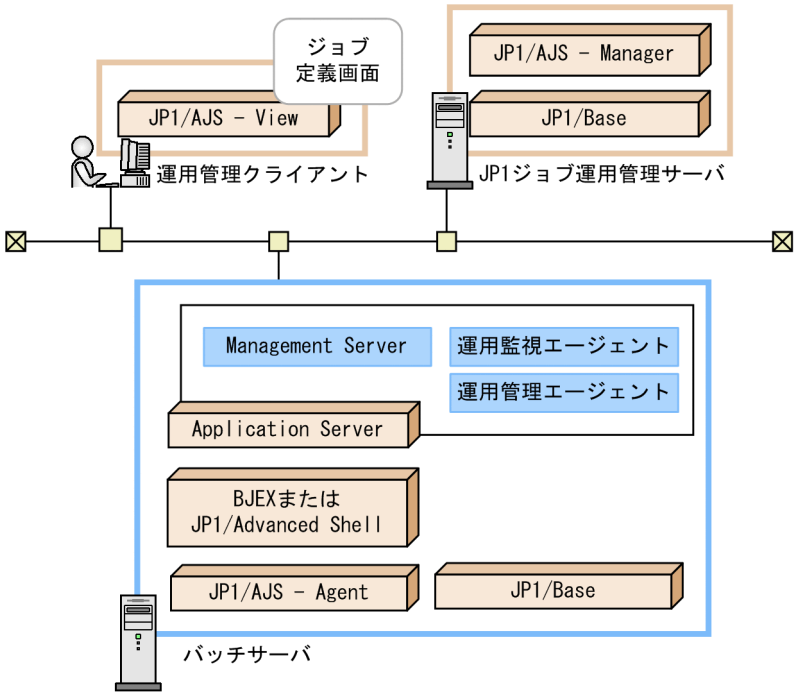
表 14-2　システムの自動運転をするために使用する JP1 プログラムの機能（J2EE サーバの場合）

JP1 プログラム	機能	配置するサーバ／クライアント
JP1/Base	JP1/AJS の基盤機能を提供します。ユーザ権限の管理や JP1 イベントの送受信をします。	• アプリケーションサーバの運用管理サーバ • JP1 ジョブ運用管理サーバ
JP1/AJS - Agent	JP1/AJS のエージェントです。JP1/AJS - Manager から転送されたジョブを実行します。	• アプリケーションサーバの運用管理サーバ
JP1/AJS - Manager	JP1/AJS のマネジャです。	• JP1 ジョブ運用管理サーバ
JP1/AJS - View	GUI を使って JP1/AJS を操作するための画面を提供します。JP1/AJS - Manager に接続して，ジョブネットの定義や操作，実行状況や結果の表示などをします。	• 運用管理クライアント

(2)　バッチサーバの場合

バッチアプリケーション実行環境で，システムを自動運転する場合のシステムの構成例を次の図に示します。

図 14-4　JP1/AJS と連携してシステムの自動運転をする場合のシステム構築例（バッチサーバの場合）



それぞれの図中の JP1 の各プログラムの主な機能を次に示します。なお，アプリケーションサーバの管理機能については，「1.2　システムの目的と機能の対応」を参照してください。

表 14-3　システムの自動運転をするために使用する JP1 プログラムの機能（バッチサーバの場合）

JP1 プログラム	機能	配置するサーバ／クライアント
JP1/Base	JP1/AJS の基盤機能を提供します。ユーザ権限の管理や JP1 イベントの送受信をします。	• バッチサーバ • JP1 ジョブ運用管理サーバ
JP1/AJS - Agent	JP1/AJS のエージェントです。JP1/AJS - Manager から転送されたジョブを実行します。	• バッチサーバ
JP1/AJS - Manager	JP1/AJS のマネジャです。	• JP1 ジョブ運用管理サーバ
JP1/AJS - View	GUI を使って JP1/AJS を操作するための画面を提供します。JP1/AJS - Manager に接続して，ジョブネットの定義や操作，実行状況や結果の表示などをします。	• 運用管理クライアント

BJEX と連携する場合は，BJEX をバッチサーバに配置してください。また，JP1/Advanced Shell と連携する場合は，JP1/Advanced Shell をバッチサーバに配置してください。

14.2.3　カスタムジョブによるジョブの定義（Windows の場合）

JP1/AJS では，運用に使用するコマンド，バッチファイルなどをジョブとして定義して，それぞれのジョブの実行順序を関連づけることで，システム運用を自動化します。

Windows の場合，ジョブを定義する際にカスタムジョブを使用すると，直接コマンドやバッチファイルをジョブに指定するのに比べて，容易，かつ間違いなく定義できます。

カスタムジョブとは，JP1/AJS 以外のプログラムと JP1/AJS が連携するジョブを定義する場合に，目的のジョブを容易に作成するために使用できるジョブのテンプレートです。

アプリケーションサーバでは，次の 2 種類のカスタムジョブを提供しています。

- 論理サーバ制御用カスタムジョブ
- アプリケーション制御用カスタムジョブ

これらのカスタムジョブを使用すると，GUI 操作でアプリケーションサーバ用のジョブが定義できます。

アプリケーションサーバが提供するカスタムジョブによるシステムの自動運転の設定については，「14.3 ジョブによるシステムの自動運転の設定」を参照してください。また，カスタムジョブを定義する場合の JP1/AJS - View の操作方法については，マニュアル「JP1/Automatic Job Management System 操作ガイド」を参照してください。

なお，カスタムジョブは Windows の場合だけ使用できます。このため，UNIX の場合は，JP1/AJS - View で，運用に使用するコマンドやバッチファイルなどを，ジョブとして定義する必要があります。ジョブの定義方法については，「14.3.5　ジョブの定義」を参照してください。

(1)　論理サーバ制御用カスタムジョブ

論理サーバの起動，停止を制御するジョブを定義するためのカスタムジョブです。監視時間の設定もできます。起動できる論理サーバは，次のとおりです。

- パフォーマンストレーサ
- スマートエージェント
- ネーミングサービス
- CTM ドメインマネジャ
- CTM
- J2EE サーバ
- バッチサーバ
- Web サーバ

(2)　アプリケーション制御用カスタムジョブ

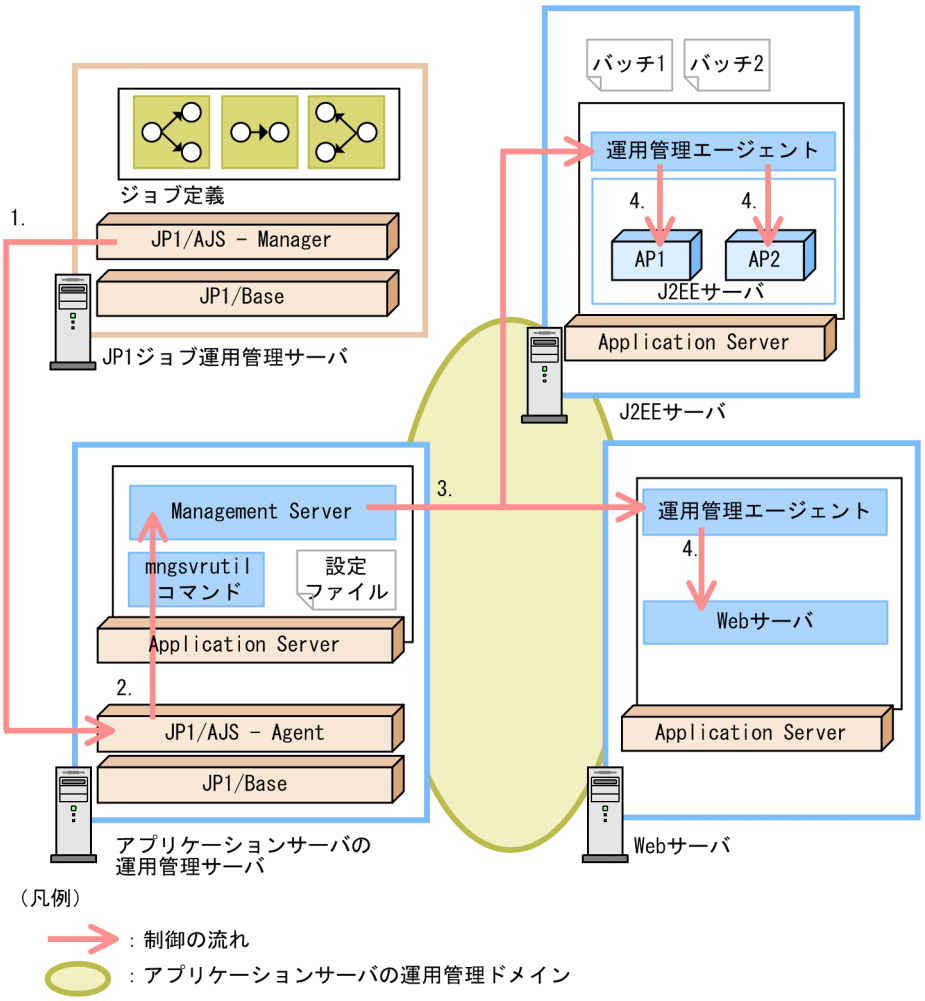
J2EE アプリケーション，バッチアプリケーション，および J2EE リソースアダプタの起動，停止を制御するジョブを定義するためのカスタムジョブです。監視時間の設定もできます。

14.2.4　論理サーバおよび J2EE アプリケーションの起動と停止の自動化の仕組み

JP1/AJS で定義したジョブおよびジョブネットを使用して，論理サーバと J2EE アプリケーションの起動と停止の運用を自動化します。

自動化した場合の処理の流れを次の図に示します。

図 14-5　論理サーバおよび J2EE アプリケーションの起動と停止を自動化した場合の処理の流れ



1. ジョブネットで設定した時間になると，ジョブに定義した論理サーバや J2EE アプリケーションの起動または停止を実行するためのコマンドとパラメタが，JP1/AJS - Manager から各ホストの JP1/AJS - Agent に送信されます。

2. ジョブの実行指示を受け取った JP1/AJS - Agent は，ジョブに定義されていたアプリケーションサーバの運用管理コマンド（mngsvrutil）を実行して，Management Server に処理を要求します。なお，JP1/AJS - Agent の処理は，要求が完了したところで終了します。

3. Management Server は，各ホストの運用管理エージェントに対して，処理の実行を要求します。

4. 運用管理エージェントによって，要求された処理が実行されます。

運用管理コマンド（mngsvrutil）の機能については，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。

14.3　ジョブによるシステムの自動運転の設定

この節では，JP1/AJS と連携して，ジョブによってアプリケーションサーバシステムの運用を自動化するための設定について説明します。

まず，次の表に示すサーバおよびクライアントに，システムの自動運転に必要な JP1 の製品または Application Server の製品の構成ソフトウェアをインストール，セットアップしておいてください。なお，JP1/AJS と連携する場合のシステム構成については，「14.2.2　ジョブによるシステムの自動運転に必要なプログラム」のジョブによるシステムの自動運転に必要なプログラムに関する説明を参照してください。

表 14-4　ジョブによるシステムの自動運転に必要な製品または構成ソフトウェア

サーバ／クライアント	製品／構成ソフトウェア
J2EE サーバ	• Component Container
アプリケーションサーバの運用管理サーバ	• JP1/Base • JP1/AJS - Agent
JP1 ジョブ運用管理サーバ	• JP1/Base • JP1/AJS - Manager
運用管理クライアント	• JP1/AJS - View

ジョブを使用して，システムを自動運転化するための手順を次に示します。

1. アプリケーションサーバの運用管理サーバで，次の作業を実施します。
   - mngsvrutil コマンドの実行環境の設定（14.3.1 参照）
   - J2EE サーバの運用監視の設定（14.3.2 参照）
2. 運用管理クライアントで，次の作業を実施します。
   - アプリケーションサーバの JP1/AJS 定義プログラムのインストール（14.3.3 参照）※
   - アプリケーションサーバのカスタムジョブの登録（14.3.4 参照）※
   - ジョブの定義（14.3.5 参照）
   - スケジュールの定義（14.3.6 参照）

注※　Windows の場合だけ実施してください。

14.3.1　mngsvrutil コマンドの実行環境の設定

JP1/AJS と連携してアプリケーションサーバシステムの運用を自動化する場合，論理サーバや J2EE アプリケーションの実行には，Management Server の運用管理コマンドである mngsvrutil コマンドを利用します。mngsvrutil コマンドの実行環境は，mngsvrutil コマンドのクライアント側定義ファイル（.mngsvrutilrc）に設定します。

アプリケーションサーバの運用管理サーバで，JP1/Base のユーザマッピングで設定した OS ユーザ（JP1/AJS が mngsvrutil コマンドを起動するときの OS ユーザ）のホームディレクトリに，クライアント側定義ファイルを作成して，次のパラメタを指定します。

- mngsvrutil.connect.host
  Management Server のホスト名とポート番号を指定します。
- mngsvrutil.connect.userid
  Management Server の mngsvrctl setup コマンドで設定した管理ユーザ ID を指定します。
- mngsvrutil.connect.password
  Management Server の mngsvrctl setup コマンドで設定したパスワードを指定します。パスワードを設定していない場合，このパラメタの指定は省略できます。

Windows の場合，アプリケーションサーバのカスタムジョブを利用して，論理サーバや J2EE アプリケーションを実行する場合は，これら以外のパラメタの指定は無効になります。

クライアント側定義ファイルは，mngsvrutil コマンドのオプションのデフォルト値を設定するファイルです。

なお，mngsvrutil コマンドの実行環境を変更したい場合は，Management Server の運用管理コマンドである mngsvrutil コマンドのサーバ側定義ファイル（mngsvrutil.properties）で変更します。

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「8.2.15　mngsvrutil.properties（mngsvrutil コマンドのサーバ側定義ファイル）」を参照してください。

参考

クライアント側定義ファイルはクライアントごとに個別のデフォルト値を設定する場合に使用します。すべてのクライアント共通のデフォルト値を設定したい場合は，クライアント側共通設定ファイル（mngsvrutilcl.properties）を使用してください。なお，両方のファイルを使用している場合は，クライアント側定義ファイルが適用されます。クライアント側共通設定ファイルは読み込まれません。

14.3.2　J2EE サーバの運用監視の設定

J2EE サーバの運用監視の設定手順を次に示します。

1. J2EE サーバの運用監視エージェントで運用監視をする設定をします。
   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.instrumentation.enabled パラメタで「true」を指定します。デフォルトの設定では，運用監視エージェントで運用監視をする設定になっています。
2. Management Server および運用管理エージェントを再起動します。

再起動後に，設定した情報が有効になります。

運用管理エージェントと Management Server の起動および停止については，「2.6　システムの起動と停止の設定」を参照してください。

14.3.3　アプリケーションサーバの JP1/AJS 定義プログラムのインストール（Windows の場合）

JP1/AJS 定義プログラムを JP1/AJS - View 稼働マシンに格納します。

アプリケーションサーバが提供する次のファイルを JP1/AJS - View 稼働マシンの任意のディレクトリにコピーしてください。

- **論理サーバ起動／停止ジョブ定義用の定義プログラム**
  <Application Server のインストールディレクトリ>¥manager¥externals¥jp1¥ajs3_custom_jobs¥cmsvcjdf.exe
- **J2EE アプリケーション起動／停止ジョブ定義用の定義プログラム**
  <Application Server のインストールディレクトリ>¥manager¥externals¥jp1¥ajs3_custom_jobs¥cmapcjdf.exe

14.3.4　アプリケーションサーバのカスタムジョブの登録（Windows の場合）

次の 2 種類のカスタムジョブを JP1/AJS - View に登録します。

- 論理サーバ制御用カスタムジョブ
- アプリケーション制御用カスタムジョブ

カスタムジョブの登録手順を次に示します。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。

1. Windows の［スタート］メニューから，［JP1_Automatic Job Management System 3 - View］－［カスタムジョブ登録］を選択します。
   ［カスタムジョブの登録］ダイアログが表示されます。
2. ［新規登録］ボタンをクリックします。
   ［カスタムジョブ登録情報］ダイアログが表示されます。
3. 論理サーバ制御用カスタムジョブおよびアプリケーション制御用カスタムジョブの情報を登録します。

名前

カスタムジョブ名を任意の 1～8 バイトで指定します。

コメント

コメントを 1～40 バイトで任意の文字列で指定します。この指定は省略できます。

定義プログラム

JP1/AJS - View 稼働マシンにコピーしたアプリケーションサーバのカスタムジョブの定義プログラムを絶対パスで指定します。論理サーバ制御用カスタムジョブの場合は「cmsvcjdf.exe」，アプリケーション制御用カスタムジョブの場合は「cmapcjdf.exe」を指定してください。

実行プログラム

mngsvrutil.コマンドを指定します。次のパスを指定してください。

<Application Server のインストールディレクトリ>¥manager¥bin¥mngsvrutil.exe

バージョン

「0600」を指定します。

バージョンはアプリケーションサーバのバージョンではなく，JP1/AJS - View のカスタムジョブのインタフェースのバージョンです。

クラス名

論理サーバ制御用カスタムジョブの場合は「COSMNGSV」，アプリケーション制御用カスタムジョブの場合は「COSMNGAP」を指定してください。

ジョブ種別

「PC ジョブ」を選択してください。

4. [OK] ボタンをクリックします。

アプリケーションサーバのカスタムジョブが JP1/AJS - View に登録されます。

14.3.5 ジョブの定義

ジョブの定義について OS ごとに説明します。

(1) Windows の場合

カスタムジョブを登録すると，JP1/AJS - View でジョブネット中のジョブを定義するときにカスタムジョブが利用できるようになります。

アプリケーションサーバのカスタムジョブを使用したジョブの定義手順を次に示します。詳細については，マニュアル「JP1/Automatic Job Management System 操作ガイド」のジョブの定義に関する説明を参照してください。

1. Windows の［スタート］メニューから，［JP1_Automatic Job Management System 3 - View］－［ジョブシステム運用］を選択します。

   ［ログイン］画面が表示されるので，ユーザ名，パスワード，および接続ホスト名を指定してログインすると，［JP1/AJS3 - View］ウィンドウが表示されます。

2. ［編集］－［新規作成］－［ジョブネット］を選択します。

   ［詳細定義－［ジョブネット］］ダイアログが表示されます。

3. ジョブネットに属性などを定義して，［OK］ボタンをクリックします。

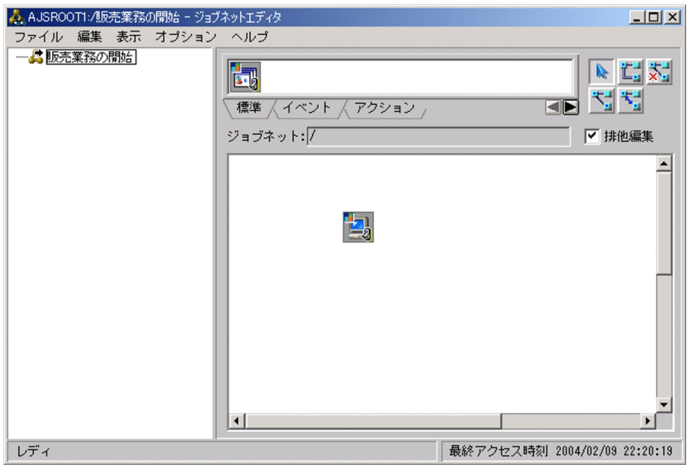
   ジョブネットが作成され，リストエリアに表示されます。

4. 作成したジョブネットをダブルクリックします。

   ［ジョブネットエディタ］ウィンドウが表示されます。

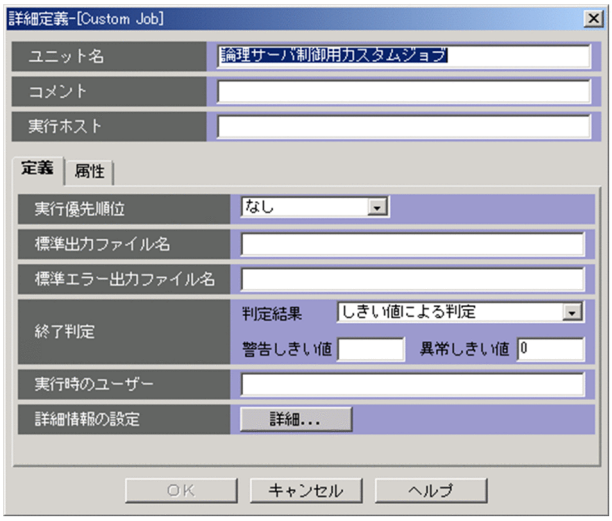
5. ジョブを定義したり関連づけたりするときにほかのユーザがアクセスできないように，［排他編集］をチェックします。

6. アイコンリストから必要なカスタムジョブのアイコンをマップエリアへドラッグします。



   ［詳細定義－［Custom Job］］ダイアログが表示されます。

7. ジョブに属性などを定義します。



ユニット名

ユニット名を 30 バイト以内の文字列で指定します。デフォルトは，カスタムジョブを登録したときのコメントが表示されます。

コメント

ユニットのコメントを 80 バイト以内の文字列で指定します。

実行ホスト

mngsvrutil コマンドを実行させる JP1/AJS - Manager，または JP1/AJS - Agent ホスト名を 255 バイト以内の文字列で指定します。

終了判定

正常終了と異常終了の判定基準とする値を定義します。mngsvrutil コマンドは終了コードが 0 以外は異常終了となるので次のようにデフォルトの値を指定してください。

判定結果：「しきい値による判定」

警告しきい値：空白

異常しきい値：「0」

8. **[詳細] ボタンをクリックします。**

［詳細］ボタンをクリックすると，アプリケーションサーバの［論理サーバの制御］ダイアログが表示されます。

9. **論理サーバ制御用カスタムジョブの場合は，アプリケーションサーバの［論理サーバの制御］ダイアログで，次の内容を指定します。**

論理サーバの指定

起動／停止の対象となる論理サーバ名（クラスタを含む）を指定します。すべての論理サーバを対象にしたいときは「全て」を選択します。

監視時間間隔

論理サーバの起動／停止の完了を監視する際の監視時間間隔を，「3 秒」，「10 秒」，「30 秒」，「1 分」から選択します。デフォルトでは，「3 秒」が選択されています。なお，起動／停止で障害が発生した場合，障害回復のためにリトライ処理が行われます。この監視時間間隔は，障害回復時のリトライ処理間隔としても利用されます。

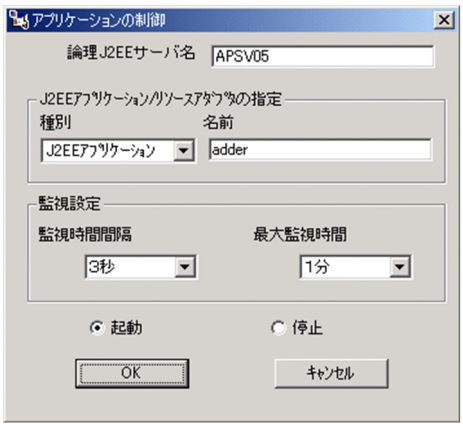
最大監視時間

論理サーバの起動／停止の完了を監視する際の最大監視時間を，「監視しない」，「1 分」，「5 分」，「10 分」，「60 分」，「120 分」から選択します。デフォルトでは，「1 分」が選択されています。

起動／停止

論理サーバの起動／停止を指定します。

10. アプリケーション制御用カスタムジョブの場合は，アプリケーションサーバの［アプリケーションの制御］ダイアログで次の内容を指定します。



論理 J2EE サーバ名

J2EE アプリケーションまたはリソースアダプタがインポートされている論理 J2EE サーバ名または J2EE サーバクラスタ名を指定します。

J2EE アプリケーション／リソースアダプタの指定

起動／停止の対象となる J2EE アプリケーション名またはリソースアダプタ名を指定します。「種別」で J2EE アプリケーションまたはリソースアダプタを選択してから，「名前」に J2EE アプリケーションまたはリソースアダプタ名を指定します。

監視時間間隔

J2EE アプリケーションまたはリソースアダプタの起動／停止の完了を監視する際の監視時間間隔を，「3 秒」，「10 秒」，「30 秒」，「1 分」から選択します。デフォルトでは，「3 秒」が選択されています。なお，起動／停止で障害が発生した場合，障害回復のためにリトライ処理が行われます。この監視時間間隔は，障害回復時のリトライ処理間隔としても利用されます。

最大監視時間

J2EE アプリケーションまたはリソースアダプタの起動／停止の完了を監視する際の最大監視時間を，「監視しない」，「1 分」，「5 分」，「10 分」，「60 分」から選択します。デフォルトでは，「1 分」が選択されています。

起動／停止

J2EE アプリケーションまたはリソースアダプタの起動／停止を指定します。

11. [OK] ボタンをクリックします。

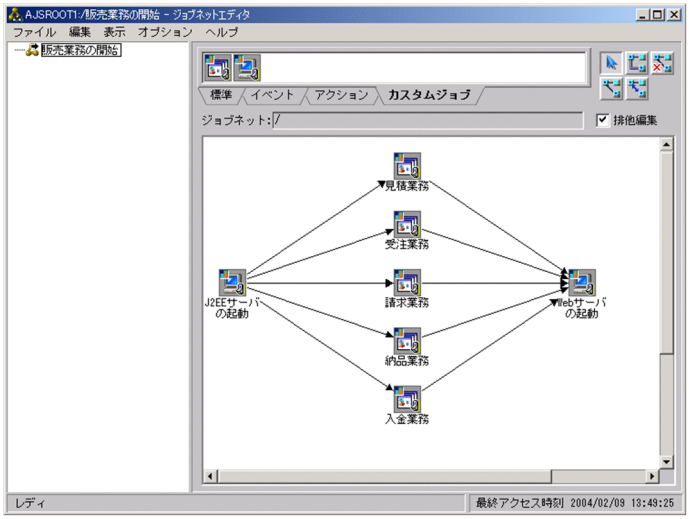
ジョブネットにジョブが定義されます。必要に応じて，同様の手順で，ジョブを定義してください。

12. ジョブ同士を実行順序に従って関連づけます。

ジョブを関連づけたら，ジョブの定義は完了です。

JP1/AJS - View でのジョブの定義例を次の図に示します。

図 14-6　ジョブの定義例



(2)　UNIX の場合

UNIX ジョブを使用したジョブの定義手順を次に示します。

UNIX ジョブを使用したジョブの定義には，JP1/AJS - View を使用します。JP1/AJS - View がインストールされた運用管理クライアント（Windows）で実施してください。詳細については，マニュアル「JP1/Automatic Job Management System 操作ガイド」のジョブの定義に関する説明を参照してください。

1. Windows の［スタート］メニューから，［JP1_Automatic Job Management System 3 - View］－［ジョブシステム運用］を選択します。

   ［ログイン］画面が表示されるので，ユーザ名，パスワード，および接続ホスト名を指定してログインすると，［JP1/AJS3 - View］ウィンドウが表示されます。

2. ［編集］－［新規作成］－［ジョブネット］を選択します。

   ［詳細定義－［ジョブネット］］ダイアログが表示されます。

3. ジョブネットに属性などを定義して，［OK］ボタンをクリックします。

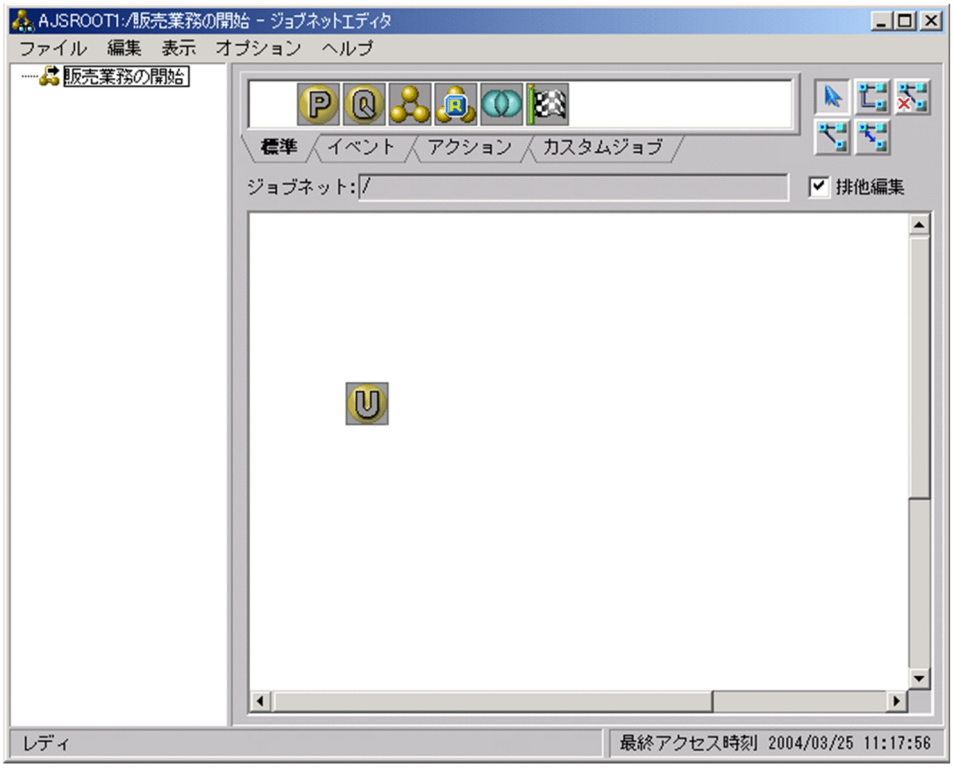
   ジョブネットが作成され，リストエリアに表示されます。

4. 作成したジョブネットをダブルクリックします。

   ［ジョブネットエディタ］ウィンドウが表示されます。

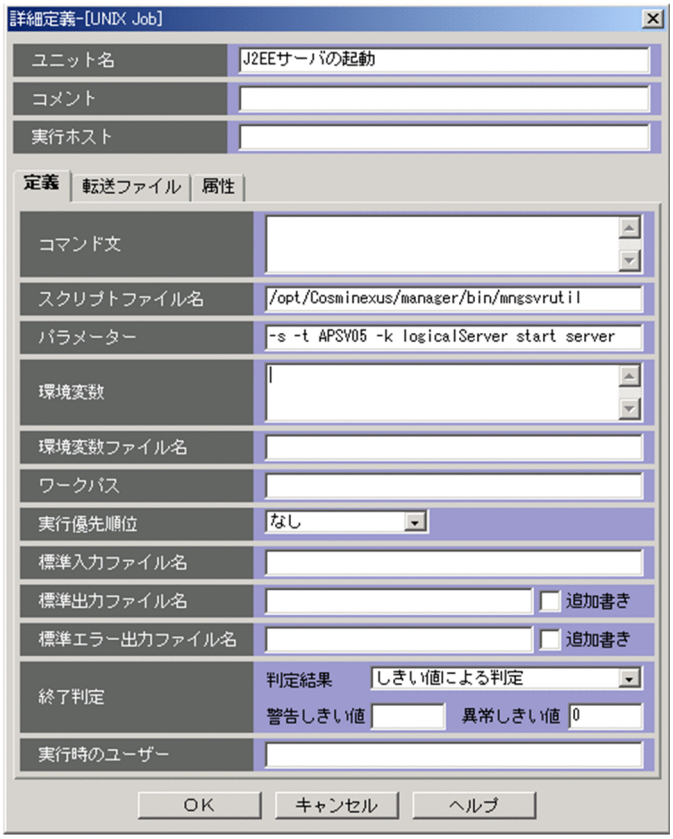
5. ジョブを定義したり関連づけたりするときにほかのユーザがアクセスできないように，［排他編集］をチェックします。

6. アイコンリストから必要な UNIX ジョブのアイコンをマップエリアへドラッグします。



   ［詳細定義－［UNIX Job］］ダイアログが表示されます。

7. ジョブに属性などを定義します。



ユニット名

ユニット名を 30 バイト以内の文字列で指定します。

コメント

ユニットのコメントを 80 バイト以内の文字列で指定します。

実行ホスト

mngsvrutil コマンドを実行させる JP1/AJS - Manager，または JP1/AJS - Agent のホスト名を 255 バイト以内の文字列で指定します。

スクリプトファイル名

mngsvrutil コマンドを指定します。

パラメーター

論理サーバの開始／停止やアプリケーションの開始／停止などを行うための mngsvrutil コマンドの引数を指定します。

終了判定

正常終了と異常終了の判定基準とする値を定義します。mngsvrutil コマンドは終了コードが 0 以外は異常終了となるので次のようにデフォルトの値を指定してください。

判定結果：「しきい値による判定」

警告しきい値：空白

異常しきい値：「0」

8. [OK] ボタンをクリックします。

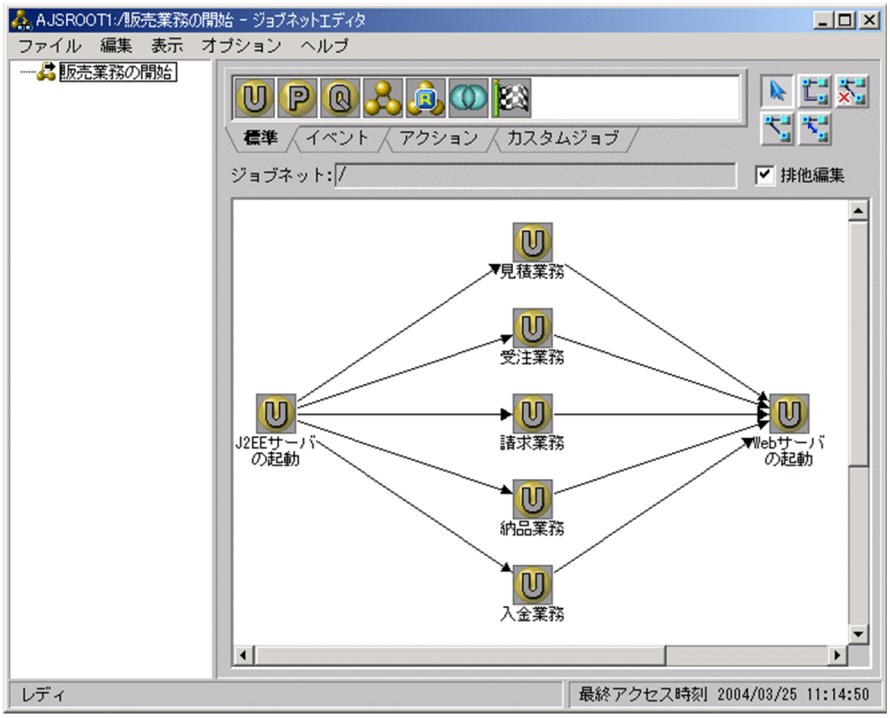
ジョブネットにジョブが定義されます。必要に応じて，同様の手順で，ジョブを定義してください。

9. **ジョブ同士を実行順序に従って関連づけます。**

   ジョブを関連づけたら，ジョブの定義は完了です。

   JP1/AJS - View でのジョブの定義例を次の図に示します。

   図 14-7　ジョブの定義例



14.3.6　スケジュールの定義

JP1/AJS - View でジョブネットにスケジュールを定義します。スケジュールの定義では，業務の運用日・休業日を設定したカレンダー，実行を開始する日時や実行間隔などを定義します。ジョブネットにスケジュールを定義しておくと，決められた日時にジョブネットを自動的に開始できるようになります。

スケジュールの定義に必要な作業について説明します。

- **ジョブグループにカレンダー情報を定義します。**

  JP1/AJS - View の［月間カレンダー編集］ウィンドウ，または［年間カレンダー編集］ウィンドウでジョブグループにカレンダー情報（運用日と休業日）を定義します。

  カレンダー情報は，ジョブグループ単位で定義します。定義したカレンダー情報は，そのジョブグループ中のすべてのジョブネットに有効です。ただし，各ジョブネットにスケジュールを定義する際，ほかのジョブグループのカレンダー情報を使用するように指定することもできます。ネストしたジョブグループの場合，上位のジョブグループのカレンダー情報を引き継げるので，カレンダー情報の定義は任意です。

- **ジョブネットにスケジュールを定義します。**

JP1/AJS - View の［スケジュールの設定］ダイアログ，および［スケジュールルール］ダイアログで，ジョブネットにスケジュールを定義します。スケジュールとして定義するのは，スケジュールの有効期限や，ジョブネットの実行開始日時，実行処理サイクル，使用するカレンダー情報などです。

スケジュールはジョブネット単位で定義します。ただし，ネストジョブネットの場合，上位のジョブネットのスケジュールルールを引き継ぐこともできるので，スケジュールルールの定義は任意です。

なお，詳細については，マニュアル「JP1/Automatic Job Management System 操作ガイド」のスケジュール定義に必要な作業に関する説明を参照してください。

15 クラスタソフトウェアとの連携

この章では，クラスタソフトウェアと連携したシステムの運用について説明します。

15.1　この章の構成

この章の構成を次の表に示します。

表 15–1　この章の構成（クラスタソフトウェアとの連携）

分類	タイトル	参照先
解説	クラスタソフトウェアと連携して実現できる運用	15.2
	クラスタソフトウェアと連携するための前提条件	15.3

注　「実装」,「設定」,「運用」,「注意事項」について，この機能固有の説明はありません。

15.2　クラスタソフトウェアと連携して実現できる運用

この節では，クラスタソフトウェアと連携するための設定の概要について説明します。

クラスタソフトウェアとは，システムの信頼性および稼働率の向上を目的とした，サーバプログラムを含めたシステムの切り替えを実現するプログラムです。アプリケーションサーバでは，次に示すクラスタソフトウェアと連携できます。

- Windows Server Failover Cluster

  Windows Server の場合に使用するクラスタソフトウェアです。詳細は，使用するクラスタソフトウェア/OS のマニュアルを参照してください。

  ご使用の OS に合わせて，次のように用語を読み替えてください。

マニュアルに記載の用語	Windows Server 2012 以降の用語
クラスタアドミニストレータ	フェールオーバー クラスター マネージャー
グループ，またはリソースグループ	クラスター化された役割

- HA モニタ

  AIX または Linux の場合に使用するクラスタソフトウェアです。

クラスタソフトウェアと連携してシステムを運用することで，アプリケーションサーバに障害が発生した場合に，直ちに自動でアプリケーションサーバを切り替えたり，待機しているリカバリサーバで，障害が発生したアプリケーションサーバのリカバリ処理をしたりできます。また，運用管理サーバに障害が発生した場合にも，待機させておいた運用管理サーバに切り替えることができます。これによって，システムの不稼働時間を短縮でき，信頼性や稼働率を高めることができます。

注意事項

運用管理エージェントと Management Server の障害を監視して，クラスタソフトウェアと連携してシステムを切り替える場合は，運用管理エージェントと Management Server に自動再起動を設定しないでください。

アプリケーションサーバのシステムでは，クラスタソフトウェアと連携して運用することで，障害によるシステムの不稼働時間を短縮したり，障害発生時でも業務の継続を可能にしたりできます。アプリケーションサーバでは，次に示すクラスタソフトウェアとの連携による系切り替え機能を使用します。

- 1:1 系切り替えシステム
  - アプリケーションサーバの 1:1 系切り替え

    実行系のアプリケーションサーバと待機系のアプリケーションサーバを 1:1 にした構成とすることで，実行系のアプリケーションサーバに障害が発生したときに系を切り替えて，業務を続行できます。
  - 運用管理サーバの 1:1 系切り替え

実行系の運用管理サーバと待機系の運用管理サーバを 1:1 にした構成とすることで，実行系の運用管理サーバに障害が発生したときに系を切り替えて，業務を続行できます。

- 相互系切り替え

  実行系のアプリケーションサーバと待機系のアプリケーションサーバを 1:1 にして，それぞれのアプリケーションサーバが実行系として動作しながらお互いの待機系として機能するシステムです。少ない台数のアプリケーションサーバマシンで，むだの少ない運用ができるようになります。

- N:1 リカバリシステム

  N 台の実行系のアプリケーションサーバに対して，1 台のリカバリ専用の待機系のアプリケーションサーバを配置するシステムです。J2EE サーバを冗長化した構成でグローバルトランザクションを使用する場合に，特定の J2EE サーバにトラブルが発生したとき，リカバリ専用サーバでトランザクションを決着できるようになります。

- 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

  アプリケーションサーバマシン（ホスト）の実行系複数（1～N 台）と待機系 1 台を配置し，それぞれに Management Server および運用管理エージェントを配置するシステムです。実行系のアプリケーションサーバマシンに障害が発生したときに待機系のアプリケーションサーバマシンに系を切り替えて，業務を続行できます。

参考

アプリケーションサーバで 1:1 系切り替えシステム，N:1 リカバリシステムおよび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を構築，および運用するためには，次のクラスタソフトウェアを使用します。

- Windows Server Failover Cluster（Windows の場合）
- HA モニタ（AIX または Linux の場合）

なお，クラスタソフトウェアと連携する場合には，次の用語を理解しておいてください。

クラスタソフトウェアでは，業務処理に必要なハードウェアのほか，実行するプログラムや通信機器を含めたシステム全体の総称を**系**といいます。システムで動作中の系を区別する呼び方に，実行系・待機系があります。

- **実行系**

  現在実行中のサーバがある系のことです。系切り替えが発生すると，待機系になります。

- **待機系**

  現在待機中のサーバがある系のことです。系切り替えが発生すると，実行系になります。

また，システムの設定などで，システムが動作していないときの系を区別する呼び方に，現用系・予備系があります。

- **現用系**

  起動したときに，最初に実行系として起動される系のことです。

- **予備系**
  起動したときに，最初に待機系として起動される系のことです。

アプリケーションサーバではクラスタソフトウェアと連携することによって，J2EE アプリケーションの実行環境の系切り替え構成でシステムを運用できます。

クラスタソフトウェアとの連携による系切り替え機能の参照先を次に示します。

表 15–2　クラスタソフトウェアとの連携による系切り替え機能

機能		参照先
1:1 系切り替えシステム	アプリケーションサーバの 1:1 系切り替え※	17.2
	運用管理サーバの 1:1 系切り替え	17.2
	相互系切り替え	18.2
N:1 リカバリシステム※		19.2
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		16.2

注※　これらのシステムは，Smart Composer 機能では構築できません。運用管理ポータルを使用して，システムを構築してください。

15.2.1　実行系と待機系を 1:1 で運用するシステムとは（1:1 系切り替えシステム）

1:1 系切り替えシステムとは，実行系と待機系が 1:1 で対応しているシステムです。アプリケーションサーバでは，アプリケーションサーバの 1:1 系切り替えシステムでの運用と，運用管理サーバの 1:1 系切り替えシステムでの運用をサポートしています。ただし，バッチアプリケーションの実行環境では，運用管理サーバを配置しないため，運用管理サーバの 1:1 系切り替えシステムの運用はできません。

1:1 系切り替えシステムでは，実行系に何らかの障害が発生すると，クラスタソフトウェアがこれを検知して，自動的に待機している系に切り替えて業務を続行します。また，障害が発生しなくても実行中のシステムに予防保守，その他必要があるときには，オペレータの操作によって待機している系に計画的に切り替えることができます。

1:1 系切り替えシステムの詳細については，「17.　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）」を参照してください。

15.2.2　相互スタンバイで運用するシステムとは（相互系切り替えシステム）

相互系切り替えシステムとは，1:1 系切り替えシステムの構成で，2 台のサーバがそれぞれ現用系として動作しながら，互いの予備系になるシステムです。アプリケーションサーバでは，アプリケーションサーバの相互切り替えシステムでの運用をサポートしています。

相互系切り替えシステムでは，各サーバでそれぞれ J2EE アプリケーションまたはバッチアプリケーションを動作させて，障害が発生した場合は自動的に相互の待機している系に切り替えて業務を続行します。また，1:1 系切り替えシステムの場合と同様に，障害が発生しなくても実行中のシステムに予防保守，その他必要があるときには，オペレータの操作によって待機している系に計画的に切り替えることができます。

相互系切り替えシステムの詳細については，「18.　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）」を参照してください。

15.2.3　待機系をリカバリ専用サーバとして運用するシステムとは（N:1 リカバリシステム）

N:1 リカバリシステムとは，クラスタ構成になっている実行系の複数（N 台）のアプリケーションサーバに対して，1 台のリカバリ専用サーバを待機系として配置したシステムです。

業務は，実行系のアプリケーションサーバで実行します。業務実行中のアプリケーションサーバに障害が発生すると，リカバリ専用サーバで，障害が発生したアプリケーションサーバの仕掛かり中のトランザクションを決着します。

なお，バッチサーバの場合，N:1 リカバリシステムは使用できません。

N:1 リカバリシステムの詳細については，「19.　N:1 リカバリシステム（クラスタソフトウェアとの連携）」を参照してください。

15.2.4　実行系と待機系のアプリケーションサーバマシン（ホスト）を N:1 で運用するシステムとは（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）

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とは，実行系のアプリケーションサーバ N 台と待機系のアプリケーション 1 台のそれぞれに Management Server および運用管理エージェントを配置する構成で，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での運用をサポートしています。

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では，実行系の Management Server や運用管理エージェントに障害が発生すると，クラスタソフトウェアがこれを検知して，待機している系に切り替えて業務を続行します。

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の詳細については，「16. 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）」を参照してください。

15.3　クラスタソフトウェアと連携するための前提条件

クラスタソフトウェアと連携してシステムを運用するための前提条件について説明します。

15.3.1　系切り替えの管理対象となるサーバ

クラスタソフトウェアと連携してシステムを運用する場合に，系切り替えの管理対象となるサーバを次の表に示します。

表 15–3　系切り替えの管理対象となるサーバ

系切り替えの種類	系切り替えの管理対象となるサーバ	クラスタソフトウェアと連携してシステムを運用する場合のシステム構成例
1:1 系切り替えシステム	運用管理サーバモデルのアプリケーションサーバ，運用管理サーバ	「17.3.1　1:1 系切り替えシステムのシステム構成例」を参照。
相互系切り替えシステム	運用管理サーバモデルのアプリケーションサーバ	「18.3.1　相互系切り替えシステムのシステム構成例」を参照。
N:1 系切り替えシステム	運用管理サーバモデルのアプリケーションサーバ	「19.3.1　N:1 リカバリシステムのシステム構成例」を参照。
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	ホスト単位管理モデルのアプリケーションサーバ	「16.2.1　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のシステム構成例」を参照。

15.3.2　ライトトランザクション機能の利用

クラスタソフトウェアと連携する場合，ライトトランザクション機能を有効にしているかどうかによって，使用できる系切り替えの種類が異なります。系切り替えの種類とライトトランザクション機能の利用について次の表に示します。なお，実行系と待機系は，同じ動作モードにすることが前提となります。

表 15–4　系切り替えの種類とライトトランザクション機能の利用

系切り替えの種類	ライトトランザクション	
	有効	無効
1:1 系切り替えシステム	○	○
相互系切り替えシステム	○	○
N:1 系切り替えシステム	×	○
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	○	○

（凡例）　○：使用できる　　×：使用できない

なお，バッチアプリケーション実行環境の場合，ライトトランザクション機能を有効にするため，N:1 系切り替えシステムは使用できません。

15.3.3　システムの運用方法

クラスタソフトウェアと連携する場合，系切り替えの種類にかかわらず，Management Server を利用してシステムを運用することが前提となります。

クラスタソフトウェアと連携する場合の設定については，「15.2　クラスタソフトウェアと連携して実現できる運用」を参照してください。

> **参考**
>
> アプリケーションサーバの 1:1 系切り替えシステムの場合，クラスタソフトウェアは，運用管理エージェント（アプリケーションサーバシステムを管理するプロセス）を監視します。
>
> 運用管理サーバの 1:1 系切り替えシステムの場合，クラスタソフトウェアは，Management Server を監視します。

Management Server または運用管理エージェントを含む系の切り替えの場合，システム起動時は，Management Server および運用管理エージェントの起動完了後に，システムを起動してください。

設定方法については，「2.6　システムの起動と停止の設定」を参照してください。

15.3.4　実行系の前提条件

N:1 系切り替えシステムの場合，実行系アプリケーションサーバは次に示す前提条件を満たしている必要があります。なお，1:1 系切り替えシステム，および相互系切り替えシステムの場合，実行系に前提条件はありません。

- グローバルトランザクションを使用します。
- CORBA ネーミングサービスおよびトランザクションサービスは，インプロセスで使用します。
- トランザクションサービスのステータスファイルは共有ディスク装置のディレクトリに格納します。

15.3.5　待機系の前提条件

待機系の前提条件を次に示します。

なお，クラスタソフトウェアと連携してシステムを運用する場合，待機系はコールドスタンバイで系切り替えを待ちます。コールドスタンバイとは，系切り替えが発生したあとに待機系のサーバが起動する方法です。

(1) 1:1 系切り替えシステム，相互系切り替えシステムおよび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の場合

1:1 系切り替えシステム，相互系切り替えシステム，および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の場合，待機系のアプリケーションサーバではアプリケーションサーバ関連のプロセスは起動できません。このため，次の操作ができないので注意してください。

- J2EE サーバへのアプリケーションのデプロイおよびアンデプロイ
  J2EE アプリケーションの入れ替えも含みます。
- Management Server および運用管理エージェントによる管理操作

また，待機系のアプリケーションサーバからは共有ディスク装置へのアクセスはできません。J2EE サーバを使用している場合でグローバルトランザクションのとき，共有ディスク装置にはトランザクションサービスなどの定義情報が保持されます。ただし，待機系のアプリケーションサーバからこの定義情報を変更することはできません。

(2) N:1 系切り替えシステムの場合

N:1 リカバリシステムの場合，待機系はリカバリ専用サーバとなります。待機系のリカバリ専用サーバの前提条件を次に示します。

- グローバルトランザクションを使用します。
- CORBA ネーミングサービスおよびトランザクションサービスはインプロセスで動作させます。
- 実行系のアプリケーションサーバ上の J2EE サーバと同じ名称の J2EE サーバが，待機系のリカバリ専用サーバ上で動作している必要があります。
- 実行系のアプリケーションサーバで使用しているリソースアダプタと同じ設定のリソースアダプタをインポートし，デプロイ，開始しておきます。なお，リカバリ専用サーバには実行系で使用しているリソースアダプタをすべてインポートする必要があります。
- リカバリ専用サーバは，障害が発生したアプリケーションサーバのトランザクションを決着するために使用するサーバであるため，J2EE アプリケーションをデプロイする必要はありません。
- バッチアプリケーションの実行環境では使用できません。

なお，待機系であるリカバリ専用サーバマシンには，リカバリ処理をするリソースアダプタを，あらかじめ設定しておく必要があります。また，N:1 リカバリが有効に動作するためには，次の二つの条件を満たしている必要があります。

- 実行系のアプリケーションサーバで起動している J2EE サーバと同じ名称の J2EE サーバが，待機系のリカバリ専用サーバにあること。

- 実行系の J2EE サーバでスタート状態にあるリソースアダプタの設定とまったく同じ設定のリソースアダプタが，待機系の J2EE サーバに常に存在すること。

16 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）

この章では，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で運用するシステムについて説明します。

16.1　この章の構成

この章の構成を次の表に示します。

表 16–1　この章の構成（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））

分類	タイトル	参照先
解説	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のシステム構成と動作	16.2
設定	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の設定	16.3
運用	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の起動と停止	16.4

注　「実装」,「注意事項」について，この機能固有の説明はありません。

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とは，複数（1～N 台）の実行系アプリケーションサーバマシン（ホスト）と 1 台の待機系アプリケーションサーバマシン（ホスト）を配置し，それぞれに Management Server および運用管理エージェントを配置するシステムです。

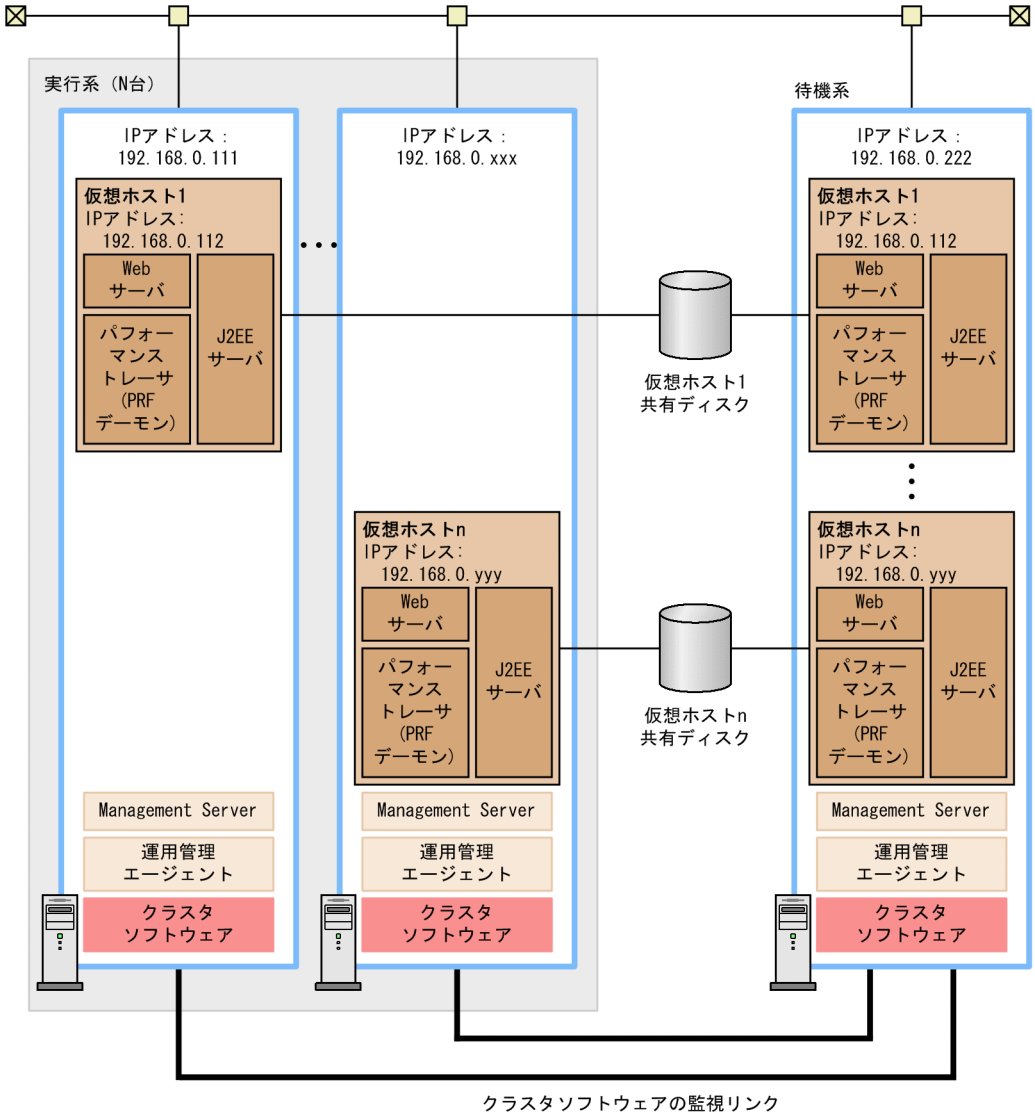
16.2　ホスト単位管理モデルを対象にした系切り替えシステムのシステム構成と動作

ここでは，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のシステム構成例と系切り替え処理の流れについて説明します。

16.2.1　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のシステム構成例

実行系 N 台の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の構成例を次の図に示します。

図 16–1　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のシステム構成例



図の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について説明します。

- アプリケーションサーバマシン（ホスト）の実行系 N 台と待機系 1 台を配置し，それぞれに Management Server および運用管理エージェントを配置するクラスタ構成のシステムです。実行系のアプリケーションサーバごとに仮想ホストが定義されたサーバが 1 台または複数台あります。これに対して，1 台の待機系アプリケーションサーバに複数の仮想ホストが定義されています。仮想ホストの説明については，「18.3.1　相互系切り替えシステムのシステム構成例」を参照してください。

  実行系のアプリケーションサーバで障害が発生すると待機系に切り替えられます。例えば，図中の実行系の仮想ホスト 1 の Management Server で障害が発生した場合，待機系の仮想ホスト 1 に系が切り替えられます。

- 実行系および待機系のそれぞれのアプリケーションサーバの運用に使用する IP アドレスは，クラスタソフトウェアによって割り当てられる仮想 IP アドレスを使用します。

  2 台以上の実行系を 1 台の待機系に対応させる場合は，Management Server，運用管理エージェントの IP アドレスには，系切り替えによって他系へ移動しない実 IP アドレスを使用します。1 台の実行系を 1 台の待機系に対応させる場合は，Management Server，運用管理エージェントの IP アドレスは，仮想 IP アドレス／実 IP アドレスのどちらでも指定可能です。

なお，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では次のような運用ができます。

共有ディスク装置の使用

共有ディスク装置は，系切り替え時に OTS のステータスなどのトランザクション情報を引き継ぐために使用します。なお，クラスタソフトウェアが HA モニタの場合，ローカルトランザクションを使用するときは不要となります。

16.2.2　系切り替えのタイミング

系切り替えは次のタイミングで実施されます。

- 運用管理エージェントまたは Management Server がダウンしたとき※1。
- 論理サーバの自動再起動回数がシステム構築時に設定した自動再起動回数を超えたとき※2。
- 系のハードウェアやクラスタソフトウェアで障害が発生したとき。

注※1

運用管理エージェントと Management Server に自動再起動を設定している場合，このタイミングでは系切り替えが実施されません。このタイミングで系切り替えを実施する場合は，運用管理エージェントと Management Server に自動再起動を設定しないでください。

注※2

mserver.properties（Management Server 環境設定ファイル）の com.cosminexus.mngsvr.logical_server_abnormal_stop.exit キーに論理サーバの稼働状況を示すステータスが異常停止状態（自動再起動回数を超えた状態，または自動再起動回数の設定が 0 の場合で障害を検知した状態）になったタイミングで系を切り替える設定（true）が必要です。mserver.properties（Management Server 環境設定ファイル）の設定については，「16.3.3　設定ファイルの編集」を参照

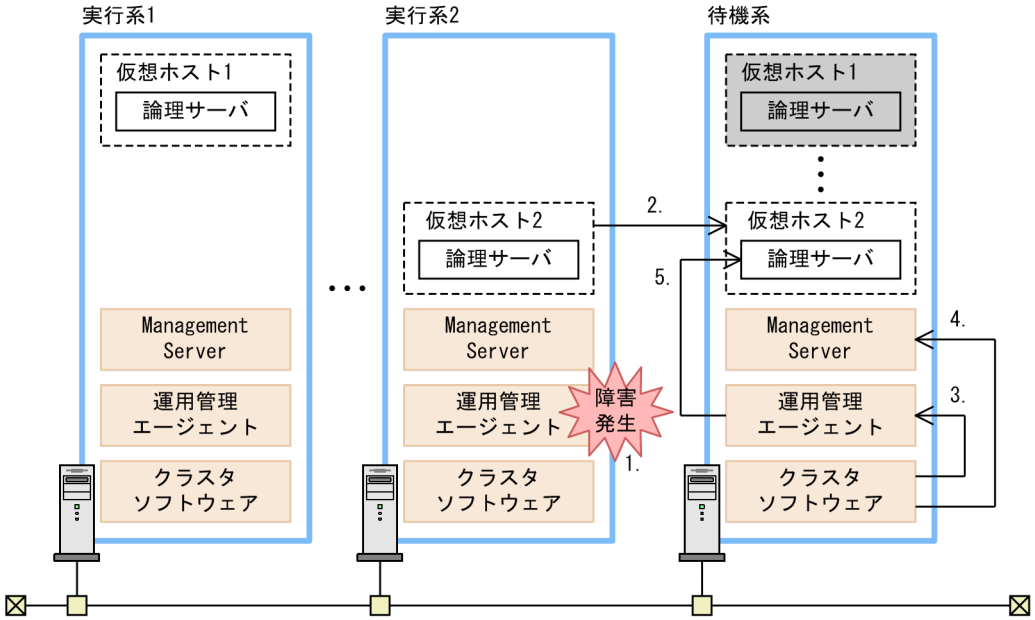
してください。Management Server に自動再起動を設定している場合，このタイミングでは系切り替えが実行されません。このタイミングで系切り替えを実行する場合は，Management Server に自動再起動を設定しないでください。自動再起動を設定している場合は，mngautorun コマンドで設定を解除してください。mngautorun コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「mngautorun（自動起動および自動再起動の設定／設定解除）」を参照してください。

ただし，実行系が 2 台以上ある場合は，待機系で複数の実行系のシステムが稼働する可能性があるため，com.cosminexus.mngsvr.logical_server_abnormal_stop.exit キーに true を指定しないでください。

16.2.3　系切り替え処理の流れ

ホスト単位管理モデルを対象にした系切り替え処理の流れを次の図に示します。

図 16–2　系切り替え処理の流れ



1. 障害を検知します。

   実行系 2 の運用管理エージェントで障害が発生し，ダウンすると，待機系のクラスタソフトウェアは実行系 2 の運用管理エージェントの障害を検知します。

2. 系の切り替えを実施します。

   待機系のクラスタソフトウェアは，実行系 2 の仮想ホスト 2 を待機系の仮想ホスト 2 に切り替えます。このとき，共有ディスク装置の切り替え，および仮想ホストの IP アドレスの引き継ぎをします。

3. 待機系の運用管理エージェントを起動します。

   運用管理エージェントに仮想 IP アドレスを使用している場合だけ，待機系のクラスタソフトウェアが運用管理エージェントを起動します。

4. 待機系の Management Server を起動します。

Management Server に仮想 IP アドレスを使用している場合だけ，待機系のクラスタソフトウェアが Management Server を起動します。

5. 論理サーバを起動します。

運用管理エージェントが待機系の仮想ホスト 2 の論理サーバを起動します。

運用管理エージェント，Management Server および論理サーバの起動はクラスタシステムに登録するサーバ起動スクリプトで実施されます。サーバ起動スクリプトについては，使用する OS のマニュアルを参照してください。

16.3　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の設定

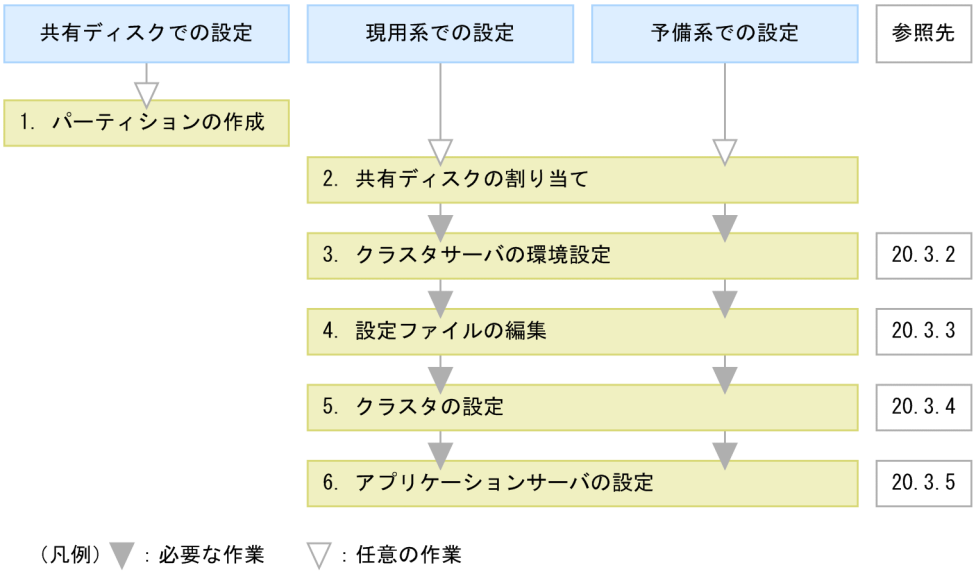
ホスト単位管理モデルを対象にした系切り替えシステムは，アプリケーションサーバマシン（ホスト）の実行系複数（1～N 台）と待機系 1 台の，それぞれに Management Server および運用管理エージェントを配置するシステムです。

この節では，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の設定について説明します。

16.3.1　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の設定手順

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の設定手順を次の図に示します。

図 16–3　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の設定手順



図中の 1.～6.について説明します。

1. 共有ディスクにパーティションを作成して，ファイルシステムを構築します。

   グローバルトランザクションを使用する場合には，トランザクション情報の格納場所を作成します。なお，HA モニタを使用している場合，ローカルトランザクションを使用するときは不要となります。

2. システムに共有ディスクを割り当てます。

   システムに共有ディスクを割り当てる際は，現用系 1 の仮想ホスト 1 と予備系の仮想ホスト 1，現用系 2 の仮想ホスト 2 と予備系の仮想ホスト 2 で，同じドライブ文字を割り当ててください。

3. Management Server でクラスタサーバの環境を設定します。

   Management Server の Smart Composer 機能の簡易構築定義ファイルで，クラスタサーバの環境（論理サーバを設置するホストおよび各論理サーバ）を設定します。詳細については，「16.3.2　クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

運用管理エージェントや Management Server で使用する各種定義ファイルを設定します。詳細については，「16.3.3　設定ファイルの編集」を参照してください。

5. クラスタの設定をします。

クラスタソフトウェアの環境設定や Management Sever と運用管理エージェントを監視するスクリプトファイルを作成します。詳細については，「16.3.4　クラスタの設定」を参照してください。

6. アプリケーションサーバを設定します。

アプリケーションサーバをクラスタ構成に配置して，J2EE アプリケーションはリソースアダプタを設定します。詳細については，「16.3.5　アプリケーションサーバの設定」を参照してください。

現用系および予備系の両方に対して，上記の手順を実施します。

参考

論理サーバは別々のホスト内に配置されているが，外部からは同じ論理サーバとして動作するのに対し，Management Server は別々のホスト内で別々に動作し，それぞれ運用管理ドメインを持ちます。一つの運用管理ドメイン内に一つの仮想ホストを定義し，それぞれを現用系アプリケーションサーバのホスト，予備系アプリケーションサーバのホストとして構築します。仮想ホストとは，運用管理エージェントによってアプリケーションサーバの起動，停止を制御しますが，運用に使用する IP アドレスと同じ IP アドレスを割り当て，見かけ上同じホストであるかのように定義したものです。

16.3.2　クラスタサーバの環境設定

クラスタソフトウェアと連携する場合の，クラスタサーバの環境設定時の Management Server での留意点について説明します。

参考

Management Server を初めて使用するホストでは，Management Server をセットアップする必要があります。Management Server のセットアップについては，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「4.1.14　運用管理機能を構築する」を参照してください。

(1)　簡易構築定義ファイルでの設定

Management Server の Smart Composer 機能の簡易構築定義ファイルで，論理サーバを設置するホストおよび各論理サーバを定義します。

なお，クラスタサーバの環境設定は運用管理ポータルを使用しても設定できます。運用管理ポータルでの操作については，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。

(a) ホストの定義

運用管理ドメイン内のホストを定義します。現用系および予備系の仮想ホストで，<host-name>タグにそれぞれのサーバの運用に使用する仮想 IP アドレスを設定します。

(b) 論理サーバの設定

論理 J2EE サーバ（j2ee-server）および論理 Web サーバ（web-server）設定時の留意点を次に示します。

- 論理 J2EE サーバ（j2ee-server）の設定

  論理 J2EE サーバ（j2ee-server）の<configuration>タグ内のパラメタに，必要に応じて仮想 IP アドレスおよび実 IP アドレスを設定してください。

  論理 J2EE サーバ（j2ee-server）の<configuration>タグ内で設定するパラメタを次の表に示します。

  表 16–2 論理 J2EE サーバ（j2ee-server）の<configuration>タグ内で設定するパラメタ

パラメタ名	内容	設定する IP アドレス
vbroker.se.iiop_tp.host	J2EE サーバ単位で EJB コンテナのホスト名または IP アドレス	仮想 IP アドレス
webserver.connector.http.bind_host	Web コンテナの管理用サーバで使用するローカル IP アドレス，または解決できるローカルホスト名称	仮想 IP アドレス
webserver.connector.nio_http.bind_host	NIO HTTP サーバで使用する IP アドレスまたはホスト名	仮想 IP アドレス
webserver.connector.http.permitted.hosts	Web コンテナの管理用サーバへのアクセスを許可するホストの IP アドレスまたはホスト名	実 IP アドレス，仮想 IP アドレス，localhost※1
add.jvm.arg	JavaVM の起動パラメタ※2	-Djava.rmi.server.hostname=<仮想 IP アドレス>

  注※1 ここで示している設定は，論理 J2EE サーバと同じホストからのアクセスを許可するものです。運用上，ほかのホストからのアクセスを受け付ける場合は，そのホストの IP アドレスまたはホスト名を追加してください。

  注※2 予備系のすべての J2EE サーバに対して設定してください。予備系で設定しない場合，J2EE サーバの稼働監視が失敗して，J2EE サーバの稼働状況が異常停止になることがあります。現用系での設定は不要です。また，1：1 系切り替えシステムでは，現用系および予備系のどちらの設定も不要です。

- 論理 Web サーバの設定

  論理 Web サーバ（web-server）の<configuration>タグ内の次のパラメタを設定して，ホストを固定してください。なお，IP アドレスには仮想 IP アドレスを設定してください。

  - Listen パラメタ（リクエストを受け付ける IP アドレスまたはホスト名）

16.3.3　設定ファイルの編集

クラスタソフトウェアと連携する場合の，Management Server で使用する設定ファイルを編集するときの留意点について説明します。なお，adminagent.properties 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「8.2.1　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。

(1)　運用管理エージェントの設定

adminagent.properties（運用管理エージェントプロパティファイル）に設定する項目のうち，クラスタソフトウェアと連携する場合に留意する設定項目について説明します。

- adminagent.adapter.bind_host

  2 台以上の実行系を 1 台の待機系に対応させる場合は，adminagent.adapter.bind_host キーに実 IP アドレスを指定します。

  1 台の実行系を 1 台の待機系に対応させる場合は，adminagent.adapter.bind_host キーに仮想 IP アドレス／実 IP アドレスのどちらでも指定できます。

(2)　Management Server の設定

mserver.properties（Management Server 環境設定ファイル）に設定する項目のうち，クラスタソフトウェアと連携する場合に留意する設定項目について説明します。

- com.cosminexus.mngsvr.logical_server_abnormal_stop.exit

  論理サーバの稼働状況を示すステータスが異常停止状態（自動再起動回数を超えた状態，または自動再起動回数の設定が 0 の場合で障害を検知した状態）になったタイミングで系を切り替えるための設定（true）をします。

  設定例を次に示します。

  com.cosminexus.mngsvr.logical_server_abnormal_stop.exit=true

  注意事項

  Management Server に自動再起動を設定している場合，このタイミングでは系切り替えが実行されません。このタイミングで系切り替えを実行する場合は，Management Server に自動再起動を設定しないでください。

  また，実行系が 2 台以上ある場合は，待機系で複数の実行系のシステムが稼働する可能性があるため，com.cosminexus.mngsvr.logical_server_abnormal_stop.exit=true を指定しないでください。

- webserver.connector.http.bind_host

  2 台以上の実行系を 1 台の待機系に対応させる場合は，webserver.connector.http.bind_host キーに実 IP アドレスを指定します。

1 台の実行系を 1 台の待機系に対応させる場合は，webserver.connector.http.bind_host キーに仮想 IP アドレス／実 IP アドレスのどちらでも指定できます。

- mngsvr.myhost.name

  2 台以上の実行系を 1 台の待機系に対応させる場合は，mngsvr.myhost.name キーに実 IP アドレスを指定します。

  1 台の実行系を 1 台の待機系に対応させる場合は，mngsvr.myhost.name キーに仮想 IP アドレス／実 IP アドレスのどちらでも指定できます。

(3) Management イベント発行用プロパティファイルの設定

mevent.<論理サーバ名>.properties（Management イベント発行用プロパティファイル）は J2EE サーバに関連する設定ファイルのため，Smart Composer 機能でシステムを構築したあとに作成されます。システム構築後に，次のキーを設定してください。

- manager.mevent.send.host

  2 台以上の実行系を 1 台の待機系に対応させる場合は，manager.mevent.send.host キーに実 IP アドレスを指定します。

  1 台の実行系を 1 台の待機系に対応させる場合は，manager.mevent.send.host キーに仮想 IP アドレス／実 IP アドレスのどちらでも指定できます。

(4) Management Server の運用管理コマンド（mngsvrutil）の設定

Management Server の運用管理コマンド（mngsvrutil）のクライアント側定義ファイル（.mngsvrutilrc）またはクライアント側共通定義ファイル（mngsvrutilcl.properties）に次のキーを設定してください。

- mngsvrutil.connect.host

  2 台以上の実行系を 1 台の待機系に対応させる場合は，mngsvrutil.connect.host キーに実 IP アドレスを指定します。

  1 台の実行系を 1 台の待機系に対応させる場合は，mngsvrutil.connect.host キーに仮想 IP アドレス／実 IP アドレスのどちらでも指定できます。

- mngsvrutil.target_name

  mngsvrutil.target_name キーに仮想 IP アドレスを指定します。なお，このキーはホスト名を指定するときだけ設定してください。

> **参考**
>
> .mngsvrutilrc ファイルはクライアントごとに個別のデフォルト値を設定する場合に使用します。すべてのクライアント共通のデフォルト値を設定したい場合は，mngsvrutilcl.properties ファイルを使用してください。なお，両方のファイルを使用している場合は，.mngsvrutilrc ファイルが適用されます。mngsvrutilcl.properties ファイルは読み込まれません。

このファイルは，Management Server を監視，起動，および停止するスクリプトで，mngsvrutil コマンドを実行する際に使用します。

(5) Smart Composer 機能で提供するコマンドの設定

クライアント設定プロパティファイル（.cmxrc）またはクライアント共通設定プロパティファイル（cmxclient.properties）に次のキーを設定してください。

- cmx.connect.host

  2 台以上の実行系を 1 台の待機系に対応させる場合は，cmx.connect.host キーに実 IP アドレスを指定します。

  1 台の実行系を 1 台の待機系に対応させる場合は，cmx.connect.host キーに仮想 IP アドレス／実 IP アドレスのどちらでも指定できます。

> **参考**
>
> .cmxrc ファイルはクライアントごとに個別のデフォルト値を設定する場合に使用します。すべてのクライアント共通のデフォルト値を設定したい場合は， cmxclient.properties ファイルを使用してください。なお，両方のファイルを使用している場合は，.cmxrc ファイルが適用されます。cmxclient.properties ファイルは読み込まれません。

(6) モニタ起動コマンドの設定（Windows の場合）

モニタ起動コマンドは Windows の場合だけ使用できます。

.mngsvrmonitorrc（JP1/IM 連携用モニタ起動設定ファイル）に次のキーを設定してください。なお，.mngsvrmonitorrc 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「8.2.17　.mngsvrmonitorrc（JP1/IM 連携用モニタ起動コマンドの設定ファイル)」を参照してください。

- mngsvrmonitor.connect.host

  2 台以上の実行系を 1 台の待機系に対応させる場合は，mngsvrmonitor.connect.host キーに実 IP アドレスを指定します。

  1 台の実行系を 1 台の待機系に対応させる場合は，mngsvrmonitor.connect.host キーに仮想 IP アドレス／実 IP アドレスのどちらでも指定できます。

16.3.4 クラスタの設定

クラスタに対して，次の設定を実施します。

- スクリプトファイルの作成
- クラスタソフトウェアの環境設定

(1) スクリプトファイルの作成

Management Server，運用管理エージェント，およびアプリケーションサーバを起動・停止するためのスクリプトファイルを作成します。また，スクリプトファイルでクラスタの監視方法を決めておく必要があります。スクリプトファイルの作成については，使用する OS のマニュアルを参照してください。

ここでは，ホスト単位管理モデルを対象にした系切り替えの監視対象および監視方法について説明します。

(a) 監視対象

- Management Server
- 運用管理エージェント
- 論理サーバ※

注※

論理サーバが異常停止状態（自動再起動回数を超えた状態，自動再起動回数の設定が 0 の場合で障害を検知した状態）になったタイミングで系切り替えが実行されます。mserver.properties（Management Server 環境設定ファイル）の次のキーを設定してください。

com.cosminexus.mngsvr.logical_server_abnormal_stop.exit=true

Management Server に自動再起動を設定している場合，このタイミングでは系切り替えが実行されません。このタイミングで系切り替えを実行する場合は，Management Server に自動再起動を設定しないでください。

ただし，実行系が 2 台以上ある場合は，待機系で複数の実行系のシステムが稼働する可能性があるため，com.cosminexus.mngsvr.logical_server_abnormal_stop.exit キーに true を指定しないでください。

(b) 監視方法

ホスト単位管理モデルを対象にした系切り替えで実施する監視方法を次に示します。

使用する運用または環境に合わせて，次のどちらかの方法を使用してください。

- プロセス監視
  監視処理にあまり負荷はかかりません。
- プロセス監視とヘルスチェック
  プロセス監視だけの場合より監視する精度は高くなりますが，監視処理に多少負荷が掛かります。

なお，論理サーバを監視対象とする場合は，「(a) 監視対象」の注意事項を参照してください。

- **プロセス監視**
  Management Server のプロセスの有無を監視します。監視するプロセス名を次に示します。

表 16-3　監視するプロセス

監視対象	プロセス名	
	Windows の場合	UNIX の場合
Management Server	mngsvr.exe	cjstartsv※1，※2
運用管理エージェント	adminagent.exe	adminagent
論理サーバ	mngsvr.exe	cjstartsv※1

注※1　cjstartsv プロセスは J2EE サーバのプロセス名と同一です。
サーバ名で Management Server のプロセスを識別してください。

注※2　Management Server のコマンドライン名は次の文字列で始まります。
/opt/Cosminexus/CC/server/bin/cjstartsv　cosmi_m
cosmi_m はデフォルトのサーバ名です。Management Server を cosmi_m 以外のサーバ名でセットアップした場合はサーバ名を変更してください。

- ヘルスチェック

  Management Server のプロセスの有無および存在しているプロセスが稼働しているかどうかを監視します。Management Server で次のコマンドを実行して確認します。

  - Management Server および論理サーバを監視する場合
    mngsvrutil コマンド
  - 運用管理エージェントを監視する場合
    adminagentcheck コマンド

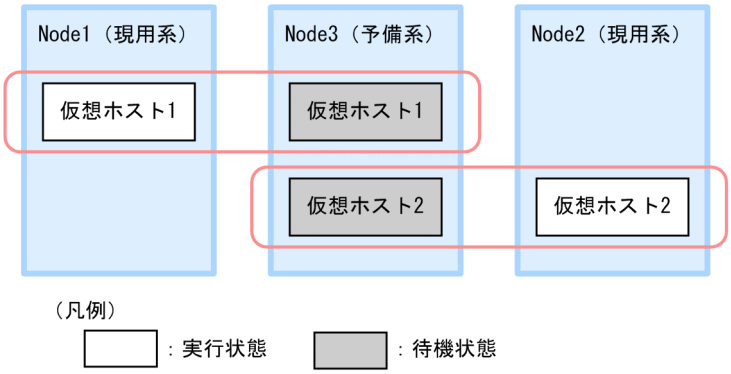
(2)　クラスタソフトウェアの環境設定

クラスタソフトウェアの環境設定については，使用するクラスタソフトウェアのマニュアルを参照してください。

16.3.5　アプリケーションサーバの設定

アプリケーションサーバの設定をするためには，現用系と予備系それぞれの運用管理エージェントと Management Server を起動します。起動後，論理サーバのセットアップを実施して設定ファイルを配布し，J2EE サーバにアプリケーションとリソースアダプタをインポートします。現用系 2 台（Node1，Node2），予備系 1 台（Node3）の場合のアプリケーションサーバの設定手順を次に示します。

1. Node1 と Node2 でクラスタソフトウェアから系を起動します。
   現用系のアプリケーションサーバ（Node1 の仮想ホスト 1 および Node2 の仮想ホスト 2）が起動されて，予備系のアプリケーションサーバ（Node3 の仮想ホスト 1 および仮想ホスト 2）が待機状態になっている，通常の運用状態になります。

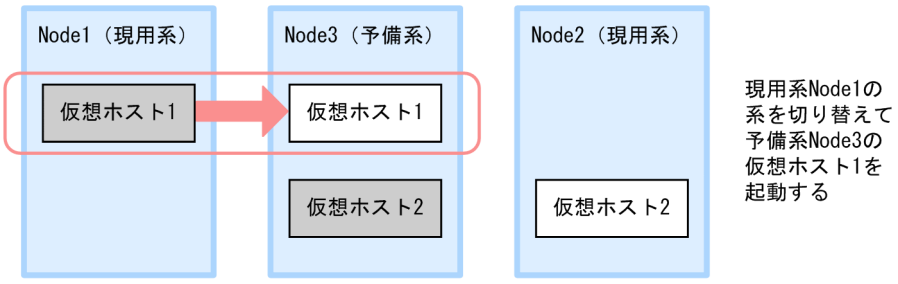


2. 現用系のアプリケーションサーバ（Node1 の仮想ホスト 1 および Node2 の仮想ホスト 2）に対して，Smart Composer 機能のコマンドで，設定済みの簡易構築定義ファイルを基に，Web システムをセットアップして起動します。また，サーバ管理コマンドを使用して，J2EE アプリケーションとリソースアダプタをインポートして，インポートした J2EE アプリケーションとリソースアダプタを開始します。

   サーバ管理コマンドでの操作については，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。

3. クラスタソフトウェアから仮想ホスト 1 の系を切り替えます。

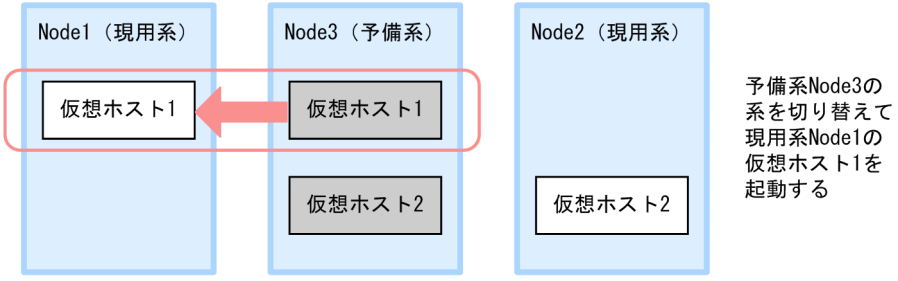
   現用系のアプリケーションサーバ（Node1 の仮想ホスト 1）の系を切り替えて，予備系のアプリケーションサーバ（Node3 の仮想ホスト 1）を起動します。



4. 予備系のアプリケーションサーバ（Node3 の仮想ホスト 1）に対して手順 2.を実施します。

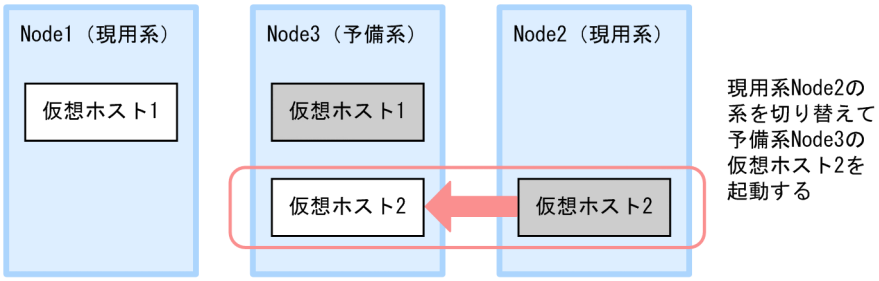
5. クラスタソフトウェアから仮想ホスト 1 の系を戻します。

   予備系のアプリケーションサーバ（Node3 の仮想ホスト 1）の系を切り替えて，現用系のアプリケーションサーバ（Node1 の仮想ホスト 1）を起動し，通常の運用状態に戻します。



6. クラスタソフトウェアから仮想ホスト 2 の系を切り替えます。

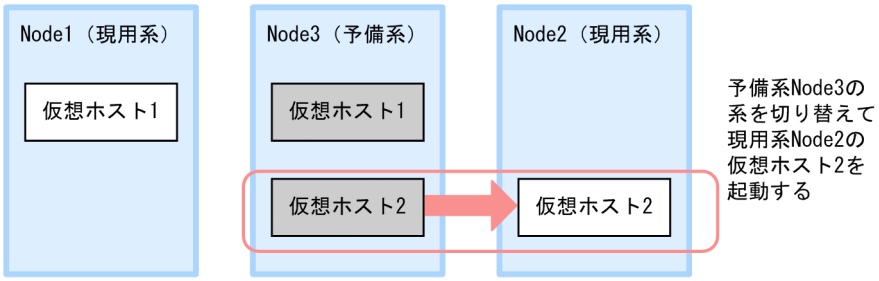
   現用系のアプリケーションサーバ（Node2 の仮想ホスト 2）の系を切り替えて，予備系のアプリケーションサーバ（Node3 の仮想ホスト 2）を起動します。



7. 予備系のアプリケーションサーバ（Node3 の仮想ホスト 2）に対して手順 2.を実施します。

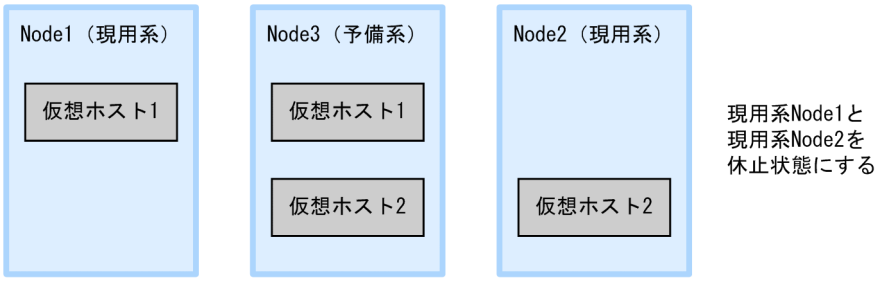
8. クラスタソフトウェアから仮想ホスト 2 の系を戻します。

   予備系のアプリケーションサーバ（Node3 の仮想ホスト 2）の系を切り替えて，現用系のアプリケーションサーバ（Node2 の仮想ホスト 2）を起動し，通常の運用状態に戻します。



9. Node1 と Node2 で，クラスタソフトウェアから系を停止します。

   現用系のアプリケーションサーバ（Node1 の仮想ホスト 1 および Node2 の仮想ホスト 2）を停止し，休止状態にします。



16.4　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の起動と停止

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を利用して運用する場合は，あらかじめ現用系 N 台と予備系 1 台のホストの用意やクラスタソフトウェアの監視対象となる Management Server や運用管理エージェントを監視・起動・停止するスクリプトの登録など，必要な環境設定をしておく必要があります。設定方法については「16.3.1　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の設定手順」を参照にしてください。

ここでは，現用系のアプリケーションサーバ（Node1 の現用系 1 および Node2 の現用系 2）と予備系のアプリケーションサーバ（Node3 の予備系 1 および予備系 2）のクラスタ構成の起動と停止の留意事項について説明します。

- 運用を開始する場合は，現用系のアプリケーションサーバ（Node1 の現用系 1 および Node2 の現用系 2）を起動して，予備系のアプリケーションサーバ（Node3 の予備系 1 および予備系 2）を待機状態にしてください。
- 現用系，予備系のアプリケーションサーバを同時に起動することはできません。J2EE アプリケーションの入れ替えが必要な場合は，サービスを停止して，現用系，予備系の系を切り替えてください。順次 J2EE アプリケーションを停止して J2EE アプリケーションを入れ替えて，必要に応じてセットアップ，配布を実行してください。

17 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）

この章では，実行系と待機系を 1:1 で運用するシステム（1:1 系切り替えシステム）について説明します。

17.1　この章の構成

この章の構成を次の表に示します。

表 17–1　この章の構成（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））

分類	タイトル	参照先
解説	1:1 系切り替えシステムの概要	17.2
	1:1 系切り替えシステムのシステム構成と動作	17.3
設定	アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合）	17.4
	運用管理サーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合）	17.5
	アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合）	17.6
	運用管理サーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合）	17.7
運用	アプリケーションサーバを対象にした 1:1 系切り替えシステムの起動と停止（Windows の場合）	17.8
	運用管理サーバを対象にした 1:1 系切り替えシステムの起動と停止（Windows の場合）	17.9
	1:1 系切り替えシステムの起動と停止（UNIX の場合）	17.10
設定の確認	系切り替えシステムの設定内容の確認方法	17.11

注　「実装」,「注意事項」について，この機能固有の説明はありません。

17.2　1:1 系切り替えシステムの概要

1:1 系切り替えシステムとは，実行系と待機系が 1:1 で対応しているシステムです。アプリケーションサーバでは，アプリケーションサーバの 1:1 系切り替えシステムと，運用管理サーバの 1:1 系切り替えシステムをサポートしています。ただし，バッチアプリケーションの実行環境では，運用管理サーバの 1:1 系切り替えシステムは使用できません。

- **アプリケーションサーバの 1:1 系切り替えシステム**

  実行系のアプリケーションサーバマシン 1 台に対して待機系のアプリケーションサーバマシンを 1 台用意しておく構成です。実行系のアプリケーションサーバマシンにトラブルが発生してマシンが終了した場合，または運用管理エージェントが終了した場合に，クラスタソフトウェアによって待機系のアプリケーションサーバマシンを起動して処理を切り替えます。

- **運用管理サーバの 1:1 系切り替えシステム**

  実行系の運用管理サーバマシン 1 台に対して待機系の運用管理サーバマシンを 1 台用意しておく構成です。実行系の運用管理サーバマシンにトラブルが発生してマシンが終了した場合，または Management Server のプロセスが終了した場合に，クラスタソフトウェアによって待機系の運用管理サーバマシンを起動して処理を切り替えます。

17.3　1:1 系切り替えシステムのシステム構成と動作

この節では，1:1 系切り替えシステムのシステムの構成例，系切り替え処理の流れなどについて説明します。

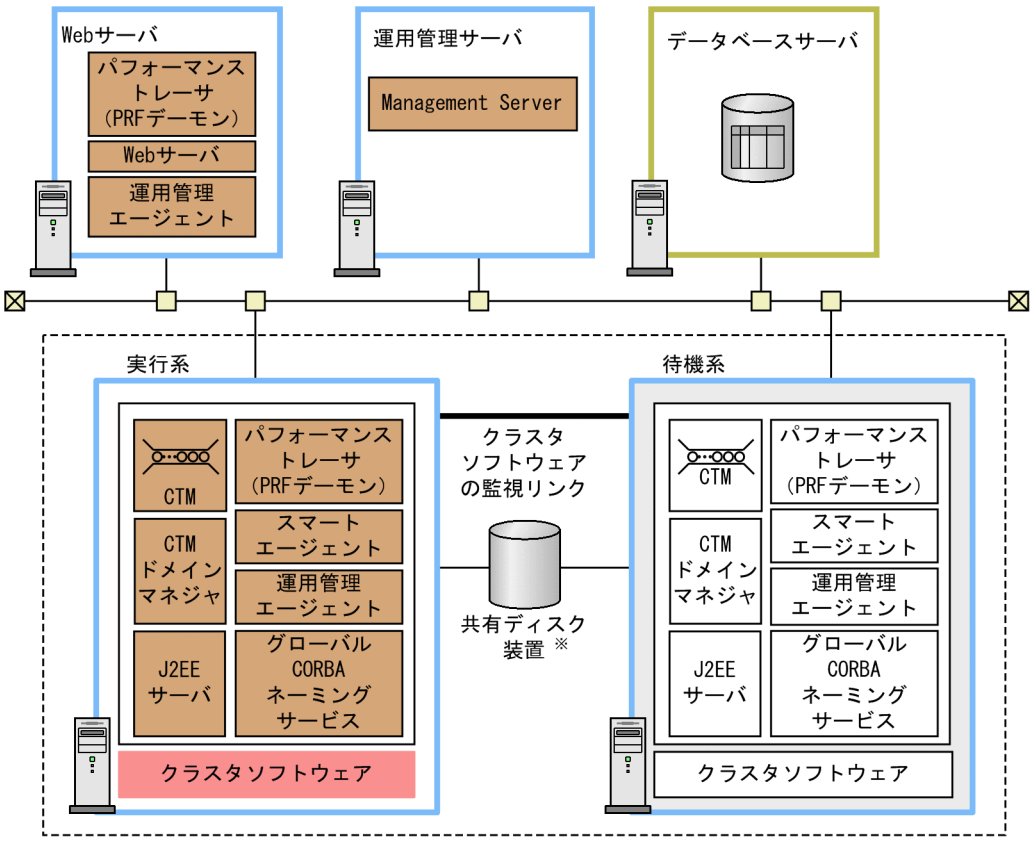
17.3.1　1:1 系切り替えシステムのシステム構成例

クラスタソフトウェアと連携して，アプリケーションサーバを 1:1 系切り替えシステムで運用する場合のシステム構成例と，運用管理サーバを 1:1 系切り替えシステムで運用する場合のシステム構成例について説明します。

- アプリケーションサーバを 1:1 系切り替えシステムで運用する場合

  アプリケーションサーバを 1:1 系切り替えシステムで運用する場合のシステム構成例を次の図に示します。系切り替え時，Web サーバや運用管理サーバからは，同じ論理アドレスでアプリケーションサーバが再起動したように見えます。

  図 17-1　アプリケーションサーバを 1:1 系切り替えシステムで運用する場合のシステム構成例

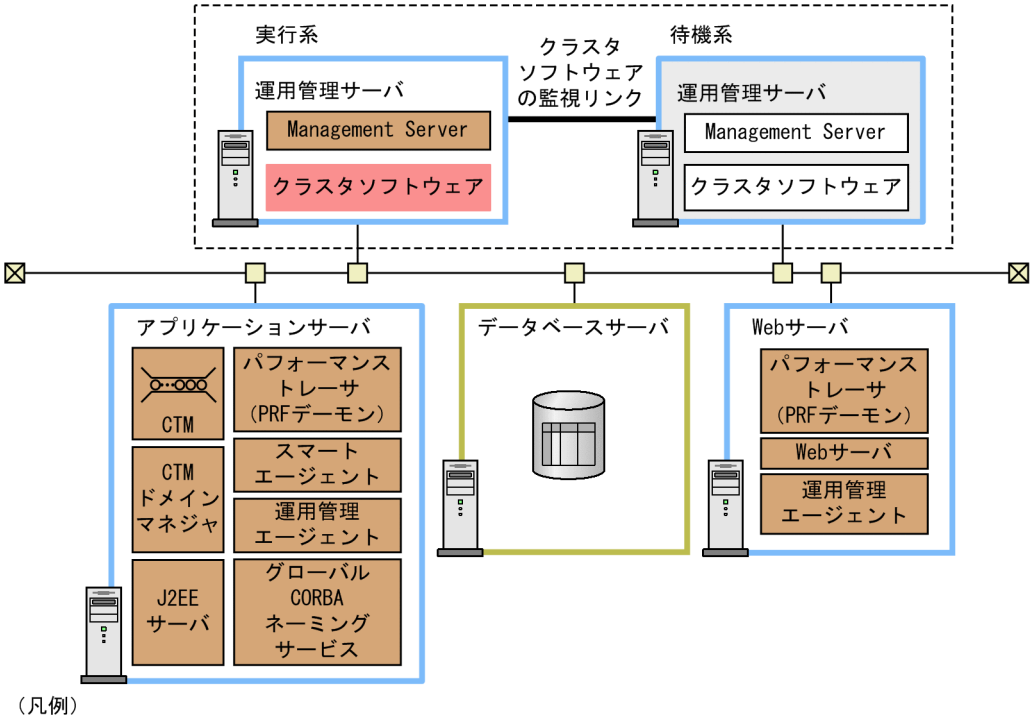


注※
グローバルトランザクションを使用するため必要となります。ローカルトランザクションを使用するときは不要となります。

- 運用管理サーバを 1:1 系切り替えシステムで運用する場合

運用管理サーバを 1:1 系切り替えシステムで運用する場合のシステム構成例を次の図に示します。系切り替え時，Web サーバやアプリケーションサーバからは，同じ論理アドレスで運用管理サーバが再起動したように見えます。

図 17-2　運用管理サーバを 1:1 系切り替えシステムで運用する場合のシステム構成例



参考

Management Server では，運用管理ドメインの構成情報などの Management Server に関する設定情報をファイル中に保持しています。運用管理サーバの 1:1 系切り替えシステムでは，運用管理ドメインの構成に変更がある場合，実行系の情報を待機系に引き継ぐために，システムの起動前に実行系で定義した情報（Management Server の各種定義ファイル，Management Server に登録した J2EE アプリケーションやリソースアダプタなど）を待機系にコピーする必要があります。Management Server に関する設定情報のコピーの手順については，「17.5.4 現用系から予備系への Management Server の設定情報のコピー」を参照してください。

なお，1:1 系切り替えシステムでは次のような運用ができます。

共有ディスク装置の使用

アプリケーションサーバを 1:1 系切り替えシステムで運用する場合，共有ディスク装置の使用については，ローカルトランザクションの場合とグローバルトランザクションの場合とで異なります。

- ローカルトランザクションの場合

  共有ディスク装置は不要です。ローカルトランザクションでは，実行系と待機系との間で引き継ぐセッション情報がないため，共有ディスク装置を使用しません。

- グローバルトランザクションの場合

共有ディスク装置が必要になります。共有ディスク装置は，系切り替え時に，OTS のステータスなどのトランザクション情報を引き継ぐために使用します。

JP1 連携をする場合

クラスタソフトウェアを使用した構成では，JP1 とも連携できます。

JP1 と連携する場合，アプリケーションサーバには，JP1/Base なども必要になります。クラスタソフトウェアでの JP1 の管理は，アプリケーションサーバとは別に行う必要があります。

データベースサーバでのクラスタソフトウェアとの連携

データベースサーバでもクラスタソフトウェアを使用した構成にすることもできます。この場合，アプリケーションサーバ側では，仮想アドレス（論理アドレス）だけを認識していれば，特にデータベースサーバがクラスタソフトウェアを使用していることを意識する必要はありません。

負荷分散機の適用

このシステム構成例では示していませんが，同一構成の Web サーバを複数用意して，負荷分散機を適用することもできます。これによって，Web サーバの信頼性・稼働率を上げることができます。

なお，アプリケーションサーバを 1:1 系切り替えシステムで運用する場合のシステム構成の詳細については，マニュアル「アプリケーションサーバ システム設計ガイド」の「3.11.1　アプリケーションサーバの実行系と待機系を 1:1 にする構成（トランザクションサービスを使用しない場合)」またはマニュアル「アプリケーションサーバ システム設計ガイド」の「3.11.2　アプリケーションサーバの実行系と待機系を 1:1 にする構成（トランザクションサービスを使用する場合)」を参照してください。運用管理サーバを 1:1 系切り替えシステムで運用する場合のシステム構成の詳細については，マニュアル「アプリケーションサーバ システム設計ガイド」の「3.11.3　運用管理サーバの実行系と待機系を 1:1 にする構成」を参照してください。

17.3.2　系切り替えのタイミング

系切り替えは次のタイミングで実施されます。

- クラスタソフトウェアがハード障害などの系障害※を検知したとき
- 監視対象プロセス（運用管理エージェントまたは Management Server）がダウンしたとき
- クラスタソフトウェアの系切り替えコマンドを使用したとき
- そのほかの，クラスタソフトウェアが系を切り替える事象が発生したとき

注※

サーバを除いた系に発生する障害を指します。系のハードウェア障害やクラスタソフトウェアの障害などがあります。

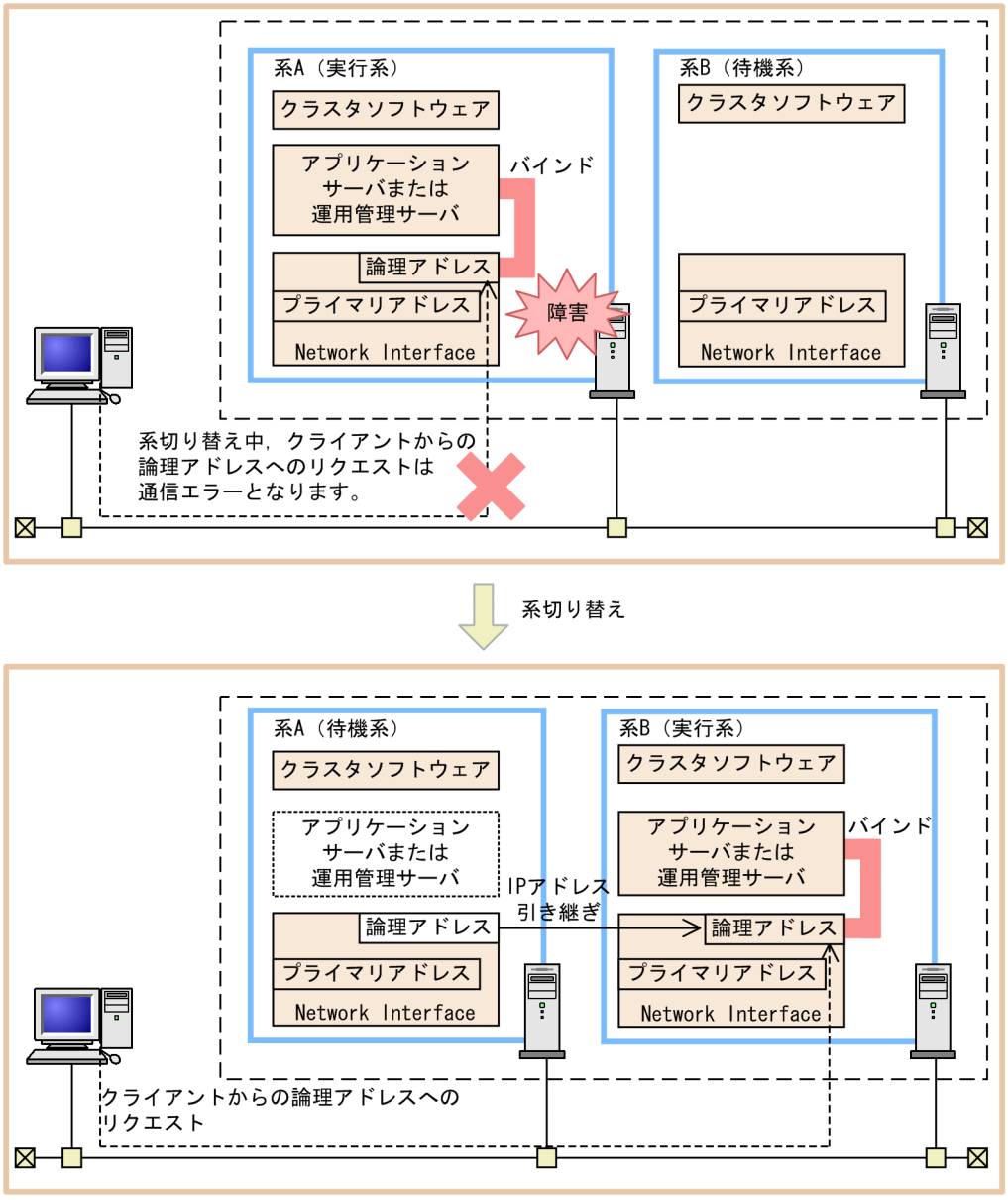
17.3.3　1:1 系切り替えシステムでの系切り替え処理の流れ

1:1 系切り替えシステムでの系切り替え時の動作と，処理の流れについて説明します。

(1)　系切り替えの動作

1:1 系切り替えシステムでの系切り替えは次の図のようにして行われます。この図では，系 A に障害が発生し，系 B に切り替える場合の例を示します。

図 17–3　1:1 系切り替えシステムでの系切り替えの動作



・系Aのサーバ（アプリケーションサーバまたは運用管理サーバ）は停止し，
　系Bのサーバ（アプリケーションサーバまたは運用管理サーバ）が起動します。
・論理アドレスのIPアドレスは，系Aから系Bに引き継がれます。

(2)　系切り替えの処理の流れ

1:1 系切り替えシステムで適用できる系切り替えには，自動系切り替えと計画系切り替えの二つがあります。**自動系切り替え**とは，クラスタソフトウェアが自動的に系を切り替える方法です。実行系に障害が発生したときは，自動系切り替えになります。一方，**計画系切り替え**とは，オペレータが系を保守する際に，計画的に系を切り替える方法です。実行系からクラスタソフトウェアのコマンドを使用して切り替えます。

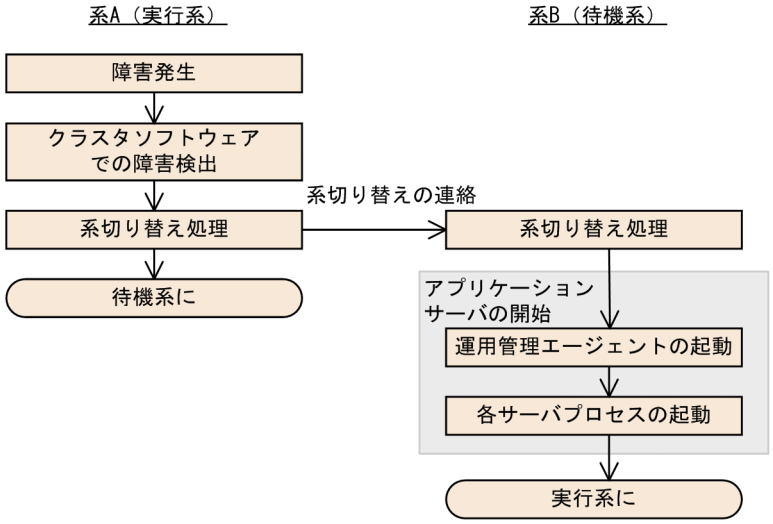
ここでは，系切り替え処理の流れについて，自動系切り替えの場合と計画系切り替えの場合に分けて説明します。

(a)　自動系切り替えの処理の流れ

1:1 系切り替えシステムでの自動系切り替えの処理の流れについて説明します。ここでは，実行系である系 A に障害が発生し，待機系である系 B に切り替えるときの流れについて説明します。

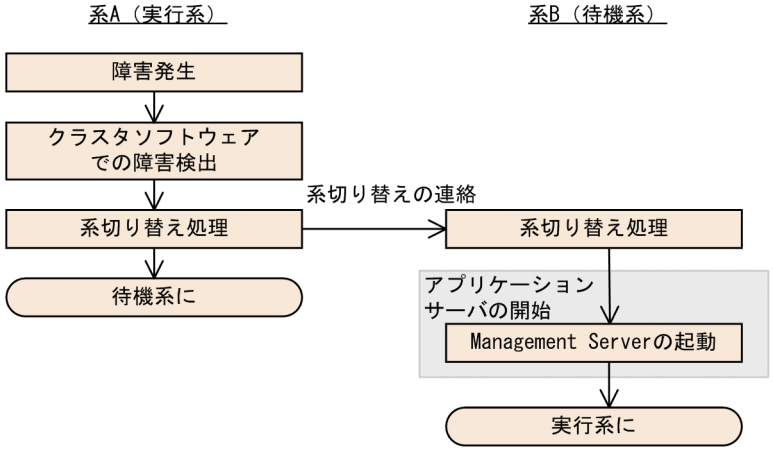
アプリケーションサーバの 1:1 系切り替えシステムでの自動系切り替えの処理の流れを次の図に示します。

図 17–4　アプリケーションサーバの 1:1 系切り替えシステムでの自動系切り替えの処理の流れ



運用管理サーバの 1:1 系切り替えシステムでの自動系切り替えの処理の流れを次の図に示します。

図 17–5　運用管理サーバの 1:1 系切り替えシステムでの自動系切り替えの処理の流れ

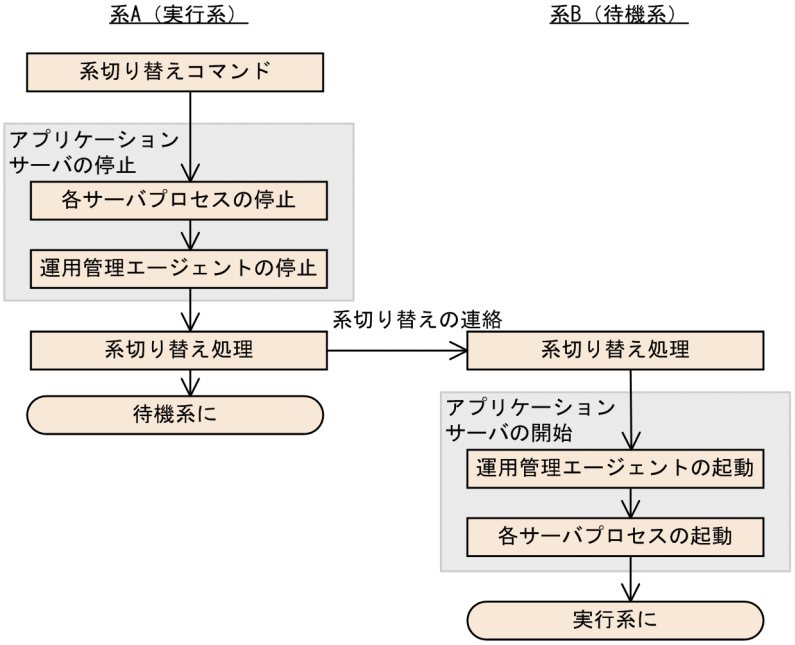


(b)　計画系切り替えの処理の流れ

1:1 系切り替えシステムでの計画系切り替えの処理の流れについて説明します。ここでは，クラスタソフトウェアのコマンドを使用して，実行系である系 A を待機系に，待機系である系 B を実行系に切り替えるときの流れについて説明します。

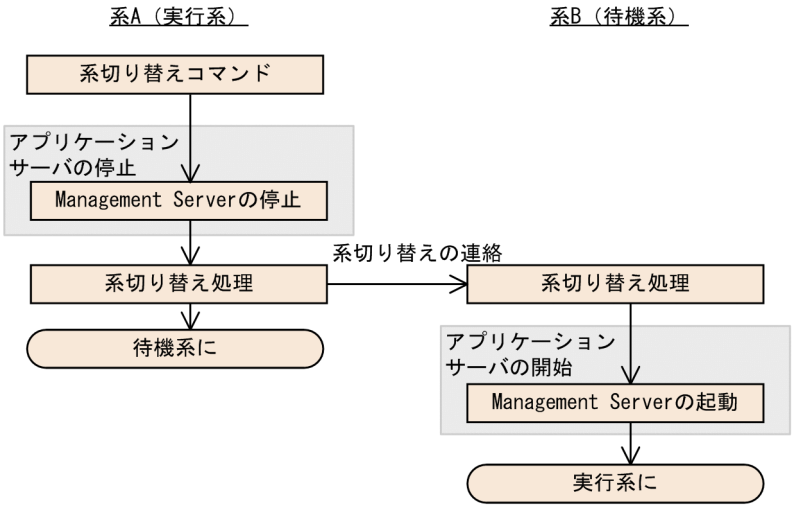
アプリケーションサーバの 1:1 系切り替えシステムでの計画系切り替えの処理の流れを次の図に示します。

図 17-6　アプリケーションサーバの 1:1 系切り替えシステムでの計画系切り替えの処理の流れ



運用管理サーバの 1:1 系切り替えシステムでの計画系切り替えの処理の流れを次の図に示します。

図 17-7　運用管理サーバの 1:1 系切り替えシステムでの計画系切り替えの処理の流れ



17.3.4　系切り替え時のシステムの動作

系切り替え時のシステムの動作について説明します。

(1)　HTTP 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を使用しない場合，Web コンテナが管理している HTTP のセッション情報※1 および ServletContext に設定したアトリビュート情報※2 は，J2EE サーバのメモリ中にしかなく，これらの情報は，実行系と待機系との間で共有されません。系の切り替え実施後，HTTP のセッション情報は失われます。このため，すでにセッションを開始していた Web アプリケーションおよび Web クライアントでは，不正な動作をするおそれがあります。これを防止するため，アプリケーションで，セッション情報に矛盾が発生した場合に適切に対処できるようにしておく必要があります。

セッションフェイルオーバ機能を使用する場合，系切り替え後も HTTP セッション情報を維持できます。ただし，維持できるのはセッションフェイルオーバ機能で引き継ぎの対象としたセッション情報だけです。

なお，系切り替え後もセッションを維持するためには，セッションを DBMS などで管理するようにアプリケーション側で独自に設定し，実行系および待機系の両方から参照できるようにする必要があります。

注※1

javax.servlet.http.HttpServletRequest#getSession メソッドで取得した，
javax.servlet.http.HttpSession オブジェクトに格納している情報を指します。

注※2

javax.servlet.ServletContext#getServletContext メソッドで取得した，
javax.servlet.ServletContext オブジェクトに格納している情報を指します。

(2)　EJB のセッション情報の引き継ぎ

EJB コンテナが管理しているセッション情報※は，J2EE サーバのメモリ中にしかないため，系の切り替え実施後はセッション情報が失われます（実行系と待機系の間で，セッション情報は共有されません）。このため，すでにセッションを開始していた EJB クライアントでは，正常に動作を継続できないおそれがあります。セッション情報が失われた場合は，Home の lookup からやり直すように，EJB クライアントを設計する必要があります。

なお，CTM 環境でも同様の問題が起こります。

注※

ホームインタフェース，およびコンポーネントインタフェースのオブジェクト情報を指します。

(3)　トランザクション情報の引き継ぎ

トランザクション情報の引き継ぎはトランザクションの種類によって異なります。トランザクション情報の引き継ぎについて，ローカルトランザクションの場合とグローバルトランザクションの場合を説明します。

(a)　ローカルトランザクションの場合

ローカルトランザクションの場合，系の切り替え時にはトランザクション情報は引き継がれません。

システムの障害によってダウンした場合や，J2EE サーバのプロセスダウンによって系の切り替えが発生した場合は，開始中のトランザクションは，コミットおよびロールバックのどちらも実施されません。

なお，計画系切り替えのように，J2EE サーバが正常に終了する場合は，トランザクションが開始中の状態で残ることはありません。

(b)　グローバルトランザクションの場合

グローバルトランザクションの場合，系切り替え後に引き継いだトランザクションを使用して，J2EE サーバの起動時に，トランザクションの回復処理をします。これによって，コミットまたはロールバックのどちらかの適切な処理が実施されます。

(4)　運用管理エージェントの動作

運用管理エージェントでは，実行系と待機系の間で直接情報を引き継ぐことはありません。系切り替え後には，運用管理エージェントの起動スクリプト中に記述した論理サーバが起動されます。なお，運用管理エージェントの起動スクリプトについては，「17.6.5　シェルスクリプトファイルの作成」を参照してください。

(5)　性能解析トレースで出力するキー情報

性能解析トレースの出力する情報には，一連の処理を識別するためのキー情報が付与されています。このキー情報には，作成時の IP アドレスが含まれます。IP アドレスを複数持つホストの場合，キー情報には，そのホストの持つ IP アドレスのどれか一つが含まれます。このため，実行系のホストで出力された性能解析トレースには，キー情報として，ステーショナリ IP アドレス（系切り替えによって他系へ移動しない IP アドレス）が含まれたり，エイリアス IP アドレス（クラスタソフトウェアによって動的に割り当てられる IP アドレス）が含まれたりする場合があります。これによって，系の切り替え前と切り替え後とで，キー情報に含まれる IP アドレスが変わる可能性があることに留意してください。

なお，性能解析トレースで出力するキー情報については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「7.2　性能解析トレースの概要」を参照してください。

17.3.5　系切り替え時の情報の引き継ぎ

系切り替え時に，待機系に引き継がれる情報と，引き継がれない情報について説明します。

- 引き継がれる情報
  - OTS トランザクション情報（グローバルトランザクションの場合）
    実行系から待機系のへの系切り替え後，OTS トランザクション情報を引き継ぎ，トランザクションの回復処理をします。

- **引き継がれない情報**

  系切り替えが発生したとき，次に示す情報は切り替え後の系に引き継がれないので注意してください。

  - Web コンテナが管理するアプリケーションの HTTP セッション情報※
  - Web コンテナの ServletContext に設定したアトリビュート情報
  - ホームインタフェースとコンポーネントインタフェースの Bean のセッション情報
  - アプリケーションの稼働状態，JNDI のキャッシュ，コネクションプール，CTM のキューなどの状態に関する情報

  注※

  セッションフェイルオーバ機能を使用する場合，引き継ぎの対象としたセッション情報だけ引き継げます。

17.4　アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合）

1:1 系切り替えシステムとは，実行系と待機系が 1:1 で対応しているシステムです。アプリケーションサーバでは，実行系のアプリケーションサーバと待機系のアプリケーションサーバを 1:1 に配置するシステムと，実行系の運用管理サーバと待機系の運用管理サーバを 1:1 に配置するシステムを構築・運用できます。アプリケーションサーバを 1:1 に配置するシステムは，運用管理ポータルを使用して構築します。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。この節では，実行系のアプリケーションサーバと待機系のアプリケーションサーバを 1:1 に配置して系を切り替えるシステムの設定について説明します。

アプリケーションサーバの実行系と待機系を 1:1 に配置して系を切り替えるシステムでは，実行系のアプリケーションサーバでマシンの障害や運用管理エージェントの終了などの障害が発生すると，クラスタソフトウェアがこれを検知して，自動的に待機している系に切り替えて業務を続行します。また，障害が発生しなくても実行中のシステムに予防保守などが必要な場合には，オペレータの操作によって待機している系に計画的に切り替えることができます。

なお，アプリケーションサーバの 1:1 系切り替えシステムの詳細については，「17.2　1:1 系切り替えシステムの概要」を参照してください。システム構成については，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「3.11.1　アプリケーションサーバの実行系と待機系を 1:1 にする構成（トランザクションサービスを使用しない場合）」またはマニュアル「アプリケーションサーバ システム設計ガイド」の「3.11.2　アプリケーションサーバの実行系と待機系を 1:1 にする構成（トランザクションサービスを使用する場合）」を参照してください。

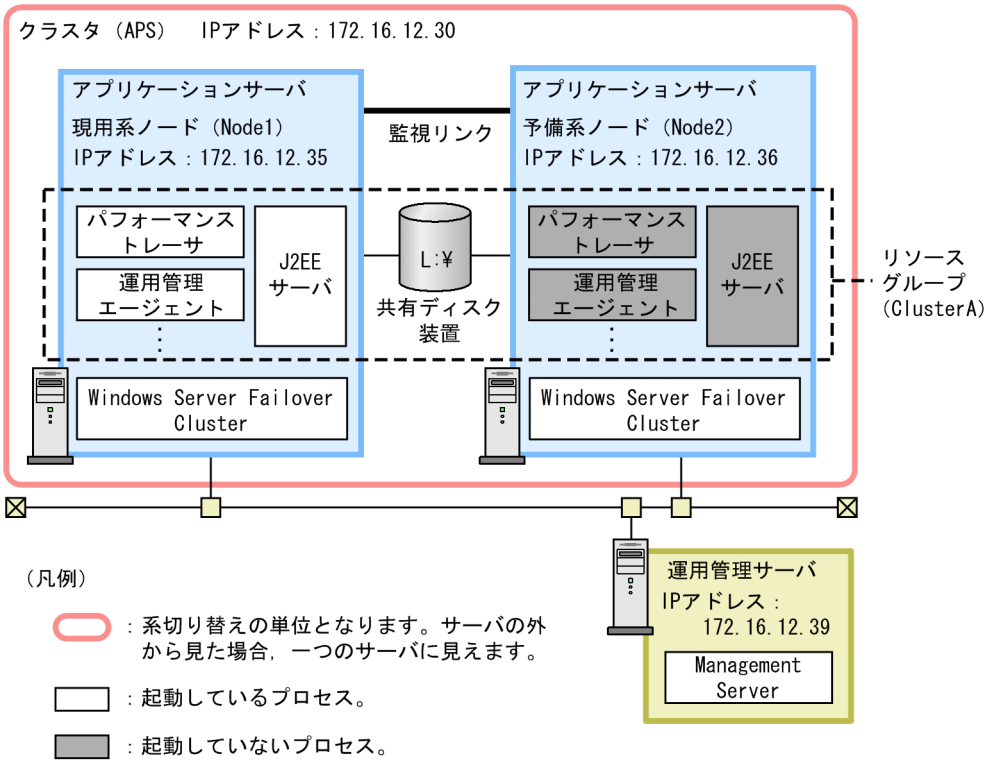
17.4.1　アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定手順

ここでは，アプリケーションサーバの 1:1 系切り替えシステムの構築例とシステムの設定手順について説明します。

(1)　システムの構成例

アプリケーションサーバの 1:1 系切り替えシステムの構成例を次の図に示します。なお，以降の項では，このシステムの構成例を使用したシステムの構築例を示します。

図 17-8　アプリケーションサーバを対象にした 1:1 系切り替えシステムの構成例（Windows の場合）



この例では，アプリケーションサーバの実行系（現用系）とアプリケーションサーバの待機系（予備系）を 1:1 で配置しています。また，Management Server は，運用管理サーバとして，アプリケーションサーバとは別のマシンに配置しています。

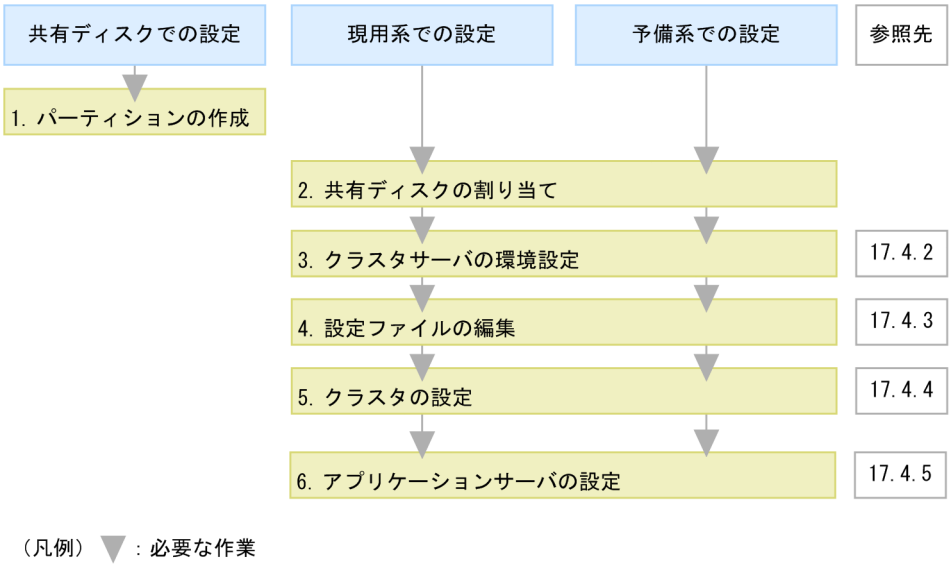
アプリケーションサーバの 1:1 系切り替えシステムでは，まずクラスタを構築して，アプリケーションサーバをクラスタ（例では APS）内に現用系ノード（例では Node1）と予備系ノード（例では Node2）として定義します。また，現用系ノードと予備系ノードは，「物理ディスクリソース」，「IP アドレスリソース」，「ネットワーク名リソース」，および「汎用スクリプトリソース」を含めた，一つのリソースグループ（例では ClusterA）として定義します。

なお，アプリケーションサーバをクラスタ構成に配置するためには，エイリアス IP アドレスを設けて，稼働中のノードがエイリアス IP アドレスを引き継ぐことで，クライアントがクラスタ内のノードを意識しないようにします。1:1 系切り替えシステムの場合，エイリアス IP アドレスは，Windows Server Failover Cluster によって動的に割り当てられるアドレス（クラスタ IP アドレス）となります。

(2)　システムの設定手順

Windows Server Failover Cluster と連携する場合には，運用管理ポータルやクラスタアドミニストレータの設定などが必要になります。アプリケーションサーバの 1:1 系切り替えシステムの設定手順を次の図に示します。

図 17-9　アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定手順（Windows の場合）



図中の 1.～6.について説明します。

1. 共有ディスクにパーティションを作成して，ファイルシステムを構築します。

   Windows Server Failover Cluster のクラスタ管理用ファイルの格納場所を作成します。また，グローバルトランザクションを使用する場合には，トランザクション情報の格納場所を作成します。なお，クラスタ管理用ファイルの格納場所とトランザクション情報の格納場所は，同じパーティションでもかまいません。

2. システムに共有ディスクを割り当てます。

   システムに共有ディスクを割り当てる際は，現用系と予備系で，同じドライブ文字を割り当ててください。

3. 運用管理ポータルでクラスタサーバの環境を設定します。

   運用管理ポータルの「運用管理ドメインの構成定義」，および「論理サーバの環境設定」で，Windows Server Failover Cluster を使用する場合の設定をします。詳細は，「17.4.2　クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

   運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。詳細は，「17.4.3　設定ファイルの編集」を参照してください。

5. クラスタを設定します。

   クラスタソフトウェアの環境設定や運用管理エージェントを監視するスクリプトファイルを作成します。詳細については，「17.4.4　クラスタの設定」を参照してください。

6. 運用管理ポータル，クラスタアドミニストレータなどで，アプリケーションサーバの設定をします。

運用管理ポータル，クラスタアドミニストレータなどを使用して，アプリケーションサーバをクラスタ構成に配置し，J2EE アプリケーションやリソースアダプタを設定します。詳細は，「17.4.5　アプリケーションサーバの設定」を参照してください。

17.4.2　クラスタサーバの環境設定

ここでは，Windows Server Failover Cluster と連携する場合の，運用管理ポータルでの設定の留意点について説明します。

参考

運用管理ポータルでシステムの動作環境を設定するには，あらかじめ，Management Server の起動などが必要です。また，Management Server を初めて使用するホストでは，Management Server をセットアップする必要があります。運用管理ポータルでの動作環境の設定や，操作手順および画面の詳細については，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)　「運用管理ドメインの構成定義」での設定

論理サーバを設置するホストおよび各論理サーバの構成を定義します。このとき，［ホストの定義］画面で，「ホスト名」にクラスタ IP アドレスを設定します。

(2)　「論理サーバの環境設定」での設定

論理 J2EE サーバ，論理 Web サーバ，および論理ネーミングサービスの設定での留意点を次に示します。

(a)　論理 J2EE サーバの設定

- ホストの固定の設定

  次の個所では，「ホストの固定」に「する」を設定してください。

  - ［J2EE コンテナの設定］画面にある「運用監視エージェントの設定」の「ホストの固定」
  - ［EJB コンテナの設定］画面にある「ホストの固定」
  - ［Web コンテナの設定］画面にある「Web サーバとの接続」および「管理用サーバの設定」の「ホストの固定」

  ホストを固定すると，運用管理ポータルの「運用管理ドメインの構成定義」で定義したホスト名が使用されます。

- グローバルトランザクションの設定

  グローバルトランザクションを使用する場合は，次の設定をしてください。

  - ［トランザクションの設定］画面にある「ライトトランザクション機能」を「無効」にします。

- ［トランザクションの設定］画面にある「インプロセス OTS のステータスファイル格納先」に，共有ディスク装置のディレクトリを設定します。

  設定例

  L:¥Group1¥otsstatus

(b)　論理 Web サーバの設定

［Web サーバの設定］画面で，「ホストの固定」に「する」を設定してください。

(c)　論理ネーミングサービスの設定

［ネーミングサービスの設定］画面で，「ホストの固定」に「する」を設定してください。このとき，J2EE サーバが使用するネーミングサービス（アウトプロセスでの起動）を設定する場合は，「運用管理エージェントのホストで固定」のチェックボックスをチェックしないでください。このように設定することで，「運用管理ドメインの構成定義」の［ネーミングサービスの追加］画面で指定したホスト名が使用されます。

17.4.3　設定ファイルの編集

運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。ここでは，Windows Server Failover Cluster と連携する場合に注意が必要なファイルの設定について説明します。

(1)　運用管理エージェントの設定

adminagent.properties（運用管理エージェントプロパティファイル）に設定する項目のうち，Windows Server Failover Cluster と連携する場合に留意する設定項目について説明します。なお，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.adapter.bind_host キー

  adminagent.adapter.bind_host キーでクラスタ IP アドレスを指定します。管理用のリクエストは，クラスタ IP アドレスだけで受け付けます。例では，次のように設定します。

  adminagent.adapter.bind_host=172.16.12.30

- adminagent.cluster.localaddress.check キー

  アプリケーションサーバの系切り替え時に，障害などの原因で停止しなかった待機系の論理サーバ，または運用管理エージェントを停止するための設定をします。

  adminagent.cluster.localaddress.check=true

(2) HTTP Server の設定（Web サーバと連携する場合）

httpsd.conf（HTTP Server 定義ファイル）に設定する項目のうち，Windows Server Failover Cluster と連携する場合に留意する設定項目について説明します。なお，httpsd.conf ファイルの各ディレクティブについては，マニュアル「HTTP Server」を参照してください。

- Listen ディレクティブ，BindAddress ディレクティブなど

  Listen ディレクティブ，BindAddress ディレクティブなどには，クラスタ IP アドレスを指定します。クライアントへのサービス提供は，クラスタ IP アドレスで実施します。

- ProxyPass ディレクティブ，ProxyPassReverse ディレクティブ

  転送先アドレスには，クラスタ IP アドレス，クラスタ IP アドレスに名前を解決できるホスト名，または"localhost"を指定します。

- ServerName ディレクティブ

  ServerName ディレクティブには，クラスタ IP アドレス，またはクラスタ IP アドレスに名前を解決できるホスト名を指定します。

(3) スマートエージェントの設定（CTM を利用する場合）

Management Server との連携で CTM を利用する場合には，次のファイルを編集する必要があります。

- localaddr

  localaddr を作成します。localaddr では，ステーショナリ IP アドレスとクラスタ IP アドレスを設定し，スマートエージェントがステーショナリ IP アドレスとエイリアス IP アドレスを通信に使用できるようにします。localaddr の設定については，「付録 A　クラスタソフトウェア連携時の TPBroker の設定（Windows の場合）」を参照してください。

17.4.4 クラスタの設定

クラスタに対して，次の設定を実施します。

- スクリプトファイルの作成
- クラスタソフトウェアの環境設定

(1) スクリプトファイルの作成

監視対象を起動・停止するためのスクリプトファイルを作成します。また，スクリプトファイルでクラスタの監視方法を決めておく必要があります。スクリプトファイルの作成については，使用する OS のマニュアルを参照してください。

ここでは，アプリケーションサーバの 1:1 系切り替えシステムの監視対象および監視方法について説明します。

(a) 監視対象

アプリケーションサーバの 1:1 系切り替えシステムの監視対象を次に示します。

- 運用管理エージェント

運用管理エージェントが終了したタイミングで系切り替えが実行されます。

(b) 監視方法

運用管理エージェントのプロセスの有無を監視します。監視するプロセスを次に示します。

- adminagent.exe

(2) Windows Server Failover Cluster の環境設定

Windows Server Failover Cluster の環境設定については，使用するクラスタソフトウェアのマニュアルを参照してください。

17.4.5 アプリケーションサーバの設定

アプリケーションサーバの設定では，運用管理ポータル，クラスタアドミニストレータなどを使用してアプリケーションサーバをクラスタ構成に配置します。また，論理サーバをセットアップして設定情報を配布し，J2EE サーバに J2EE アプリケーションとリソースアダプタをインポートします。アプリケーションサーバの設定手順を次に示します。

1. 運用管理サーバを起動します。
   運用管理サーバの起動方法については，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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

2. Node1 で，リソースグループ ClusterA をオンラインにします。

3. 運用管理ポータルで，J2EE サーバをセットアップします。
   「運用管理ドメインの構成定義」の［セットアップ］画面で，J2EE サーバをセットアップします。運用管理ポータルでの操作手順および画面の詳細については，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。

4. 運用管理ポータルで，J2EE サーバに設定した情報を Node1 に配布します。
   「論理サーバの環境設定」の［設定情報の配布］画面で，J2EE サーバに設定した情報を Node1 に配布します。

5. サーバ管理コマンドを使用して，Node1 の J2EE サーバに J2EE アプリケーションとリソースアダプタをインポートします。また，インポートした J2EE アプリケーションとリソースアダプタを設定および開始します。

J2EE 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 システム構築・運用ガイド」の「4.1.29　業務アプリケーションを設定して開始する（CUI 利用時)」，リソースアダプタの設定については，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」の次の個所を参照してください。

- 4.1.26　DB Connector を設定する（CUI 利用時）
- 4.1.27　DB Connector 以外のリソースアダプタを設定する（CUI 利用時）
- 4.1.28　リソースアダプタを開始する（CUI 利用時）

また，サーバ管理コマンドでの操作については，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。

なお，J2EE サーバおよびその前提プロセスを起動してからサーバ管理コマンドを実行してください。また，サーバ管理コマンドでの操作が終わったら，サーバ管理コマンドを停止させてください。

6. クラスタアドミニストレータで，リソースグループ ClusterA に対して「グループの移動」を実行して系を切り替え，リソースの所有者を Node2 に変更します。

7. Node2 で，手順 3.～手順 5.を実施します。

8. クラスタアドミニストレータで，リソースグループ ClusterA に対して「グループの移動」を実行して系を切り替え，リソースの所有者を Node1 に戻します。

17.5　運用管理サーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合）

1:1 系切り替えシステムとは，実行系と待機系が 1:1 で対応しているシステムです。アプリケーションサーバでは，実行系のアプリケーションサーバと待機系のアプリケーションサーバを 1:1 に配置するシステムと，実行系の運用管理サーバと待機系の運用管理サーバを 1:1 で配置するシステムを構築・運用できます。運用管理サーバを 1:1 に配置して系を切り替えるシステムは，Smart Composer 機能を使用して構築します。この節では，実行系の運用管理サーバと待機系の運用管理サーバを 1:1 に配置して系を切り替えるシステムの設定について説明します。

運用管理サーバを 1:1 に配置して系を切り替えるシステムでは，実行系の運用管理サーバでマシンの障害や Management Server のプロセスの終了が発生すると，クラスタソフトウェアがこれを検知して，自動的に待機している系に切り替えて業務を続行します。また，障害が発生しなくても実行中のシステムに予防保守などが必要な場合には，オペレータの操作によって待機している系に計画的に切り替えることができます。

なお，運用管理サーバの 1:1 系切り替えシステムの詳細については，「17.2　1:1 系切り替えシステムの概要」を参照してください。システム構成については，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「3.11.3　運用管理サーバの実行系と待機系を 1:1 にする構成」を参照してください。

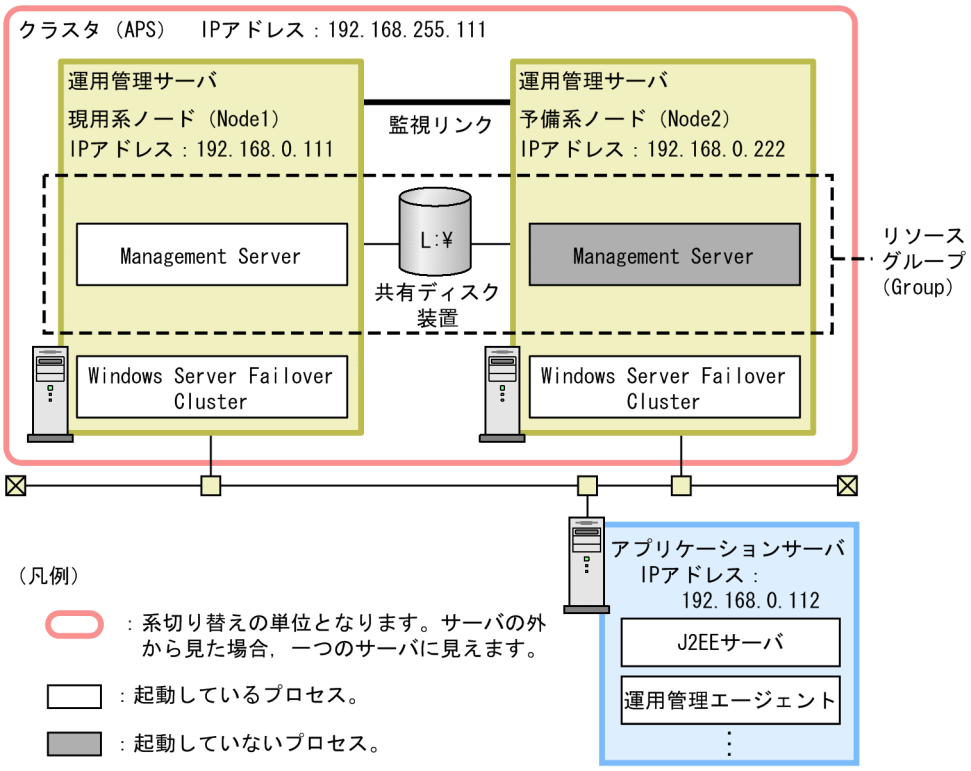
17.5.1　運用管理サーバを対象にした 1:1 系切り替えシステムの設定手順

ここでは，運用管理サーバの 1:1 系切り替えシステムの構成例とシステムの設定手順について説明します。

(1)　システムの構成例

運用管理サーバの 1:1 系切り替えシステムの構成例を次の図に示します。なお，以降の項では，このシステムの構成例を使用したシステムの構築例を示します。

図 17-10　運用管理サーバを対象にした 1:1 系切り替えシステムの構成例（Windows の場合）



運用管理サーバの実行系（現用系）と運用管理サーバの待機系（予備系）を 1:1 で配置しています。

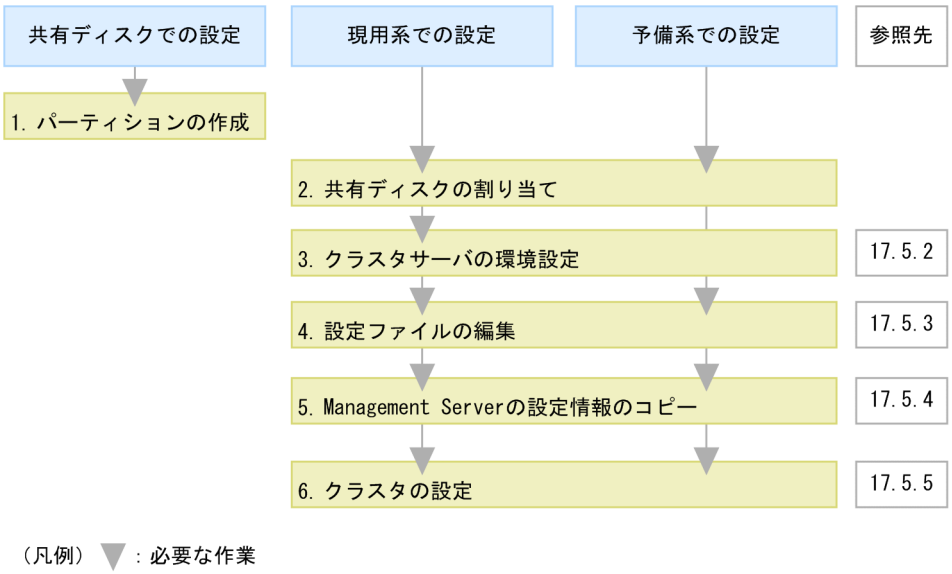
運用管理サーバの 1:1 系切り替えシステムでは，まずクラスタを構築して，運用管理サーバをクラスタ（例では APS）内に現用系ノード（例では Node1）と予備系ノード（例では Node2）として定義します。また，現用系ノードと予備系ノードは，「物理ディスクリソース」，「IP アドレスリソース」，「ネットワーク名リソース」，および「汎用スクリプトリソース」を含めた，一つのリソースグループ（例では Group）として定義します。

なお，アプリケーションサーバをクラスタ構成に配置するためには，エイリアス IP アドレスを設けて，稼働中のノードがエイリアス IP アドレスを引き継ぐことで，クライアントがクラスタ内のノードを意識しないようにします。1:1 系切り替えシステムの場合，エイリアス IP アドレスは，Windows Server Failover Cluster によって動的に割り当てられるアドレス（クラスタ IP アドレス）となります。

(2)　システムの設定手順

Windows Server Failover Cluster と連携する場合には，Management Server やクラスタアドミニストレータの設定などが必要になります。運用管理サーバの 1:1 系切り替えシステムの設定手順を次の図に示します。

図 17-11　運用管理サーバを対象にした 1:1 系切り替えシステムの設定手順（Windows の場合）



図中の 1.～6.について説明します。

1. 共有ディスクにパーティションを作成して，ファイルシステムを構築します。

   Windows Server Failover Cluster のクラスタ管理用ファイルの格納場所を作成します。また，グローバルトランザクションを使用する場合には，トランザクション情報の格納場所を作成します。なお，クラスタ管理用ファイルの格納場所とトランザクション情報の格納場所は，同じパーティションでもかまいません。

2. システムに共有ディスクを割り当てます。

   システムに共有ディスクを割り当てる際は，現用系と予備系で，同じドライブ文字を割り当ててください。

3. Management Server でクラスタサーバの環境を設定します。

   Management Server の Smart Composer 機能の簡易構築定義ファイルで，Windows Server Failover Cluster を使用する場合の設定をします。詳細は，「17.5.2　クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

   簡易構築定義ファイルでは設定できない，Management Server などの各種定義ファイルを設定します。詳細は，「17.5.3　設定ファイルの編集」を参照してください。

5. 現用系から予備系に Management Server の設定情報をコピーします。

   現用系と予備系を同じ環境にするために，現用系での Management Server に関する設定内容を予備系にコピーします。詳細は，「17.5.4　現用系から予備系への Management Server の設定情報のコピー」を参照してください。

6. クラスタを設定します。

クラスタソフトウェアの環境設定や Management Sever を監視するスクリプトファイルを作成します。詳細については，「17.5.5　クラスタの設定」を参照してください。

17.5.2　クラスタサーバの環境設定

ここでは，Windows Server Failover Cluster と連携する場合の，Management Server での設定時の留意点について説明します。

> **参考**
>
> Management Server を初めて使用するホストでは，Management Server をセットアップする必要があります。Management Server のセットアップについては，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「4.1.14　運用管理機能を構築する」を参照してください。

(1)　簡易構築定義ファイルでの設定

Management Server の Smart Composer 機能の簡易構築定義ファイルで，論理サーバを設置するホストおよび各論理サーバの構成を定義します。また，必要に応じて，各論理サーバの<configuration>タグ内に，論理サーバの環境や起動／停止の動作などを設定します。設定済みの簡易構築定義ファイルを基に，Smart Composer 機能のコマンドを使用して，Web システムをセットアップします。

必要に応じて，サーバ管理コマンドを使用して，J2EE アプリケーション，リソースアダプタをインポートして，開始します。

17.5.3　設定ファイルの編集

設定ファイルを編集します。ここでは，Windows Server Failover Cluster と連携する場合に注意が必要な設定ファイルについて説明します。なお，現用系と予備系で必要となる設定が異なります。

1. Management Server の設定
2. Management Server の運用管理コマンド（mngsvrutil）の設定

現用系では，1.と 2.を設定してください。

予備系では，2.だけ設定してください。1.については，現用系から Management Server の設定情報をコピーすることで設定できるため，ここでは設定不要です。

(1)　Management Server の設定

Management Server 環境設定ファイル（mserver.properties）に設定する項目のうち，Windows Server Failover Cluster と連携する場合に留意する設定項目について説明します。なお，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。

- mngsvr.myhost.name
  mngsvr.myhost.name キーでクラスタ IP アドレスを指定します。例では，次のように設定します。
  mngsvr.myhost.name=192.168.255.111

(2) Management Server の運用管理コマンド（mngsvrutil）の設定

ローカルシステムアカウントのホームディレクトリに，Management Server の運用管理コマンド（mngsvrutil）のクライアント側定義ファイル（.mngsvrutilrc）を用意し，Management Server の管理ユーザのユーザ ID とパスワードを設定してください。また，クライアント側定義ファイルには適切なアクセス権限を設定してください。

このファイルは，Management Server を監視，起動，および停止するスクリプトで，mngsvrutil コマンドを実行する際に使用します。

なお，mngsvrutil コマンドのクライアント側定義ファイル（.mngsvrutilrc）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「8.2.14　.mngsvrutilrc（mngsvrutil コマンドのクライアント側定義ファイル)」を参照してください。

17.5.4 現用系から予備系への Management Server の設定情報のコピー

現用系と予備系を同じ環境にするために，現用系での Management Server に関する設定情報を予備系にコピーします。Management Server の設定情報の退避／回復コマンドを使用して，現用系で設定した Management Server の各種定義ファイル，Management Server に登録した J2EE アプリケーションやリソースアダプタなどを，現用系から予備系にコピーしてください。

(1) Management Server の設定情報のコピー手順

Management Server の設定情報のコピー手順を次に示します。

1. 現用系で，mstrexport コマンドを実行します。
   mstrexport コマンドでは，mstrexport コマンドの実行対象となる現用系の Management Server の設定情報を収集し，収集した情報を ZIP 形式のファイルに保存します。mstrexport コマンドの引数には，保存する ZIP 形式のファイルのファイル名を指定してください。
   - コマンドの格納場所
     <Application Server のインストールディレクトリ>¥manager¥bin
   - 実行例
     mstrexport C:¥work¥mstruct.zip

なお，mstrexport コマンドは，mstrexport コマンドの実行対象となる Management Server の起動，停止に関係なく実行できます。

2. 手順 1.で保存された ZIP 形式のファイルを，現用系から予備系にコピーします。

3. 予備系で，mstrimport コマンドを実行します。

mstrimport コマンドでは，コピーしてきた ZIP 形式のファイルを，コマンドの実行対象となる予備系の Management Server に展開します。これによって，現用系の Management Server と予備系の Management Server を同じ設定にできます。

mstrimport コマンドの引数には，コピーしてきた ZIP 形式のファイルのファイル名を指定してください。

- コマンドの格納場所

  <Application Server のインストールディレクトリ>¥manager¥bin

- 実行例

  mstrimport D:¥recovery¥mstruct.zip

なお，mstrimport コマンドは，mstrimport コマンドの実行対象の Management Server が停止している場合だけ実行できます。

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「mstrexport（Management Server 管理ファイルの退避）」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 コマンド編」の「mstrimport（Management Server 管理ファイルの回復）」を参照してください。

(2) 収集対象のファイルの定義

mstrexport コマンドの引数として，mstrexport コマンドの収集対象を記述するファイル（Management Server 管理ファイル用退避対象定義ファイル）を指定できます。

実行例

```
mstrexport c:¥work¥mstruct.zip "C:¥Documents and Settings¥MyUser¥filelist.txt"
```

mstrexport コマンドでは，デフォルトの状態で，Management Server の各種定義ファイル，Management Server に登録した J2EE アプリケーションやリソースアダプタなど，Management Server でのシステム構築・運用に必要な情報が収集できます。これらの情報に加えて，ユーザ作成のコマンドなども mstrexport コマンドでの収集対象に追加したい場合には，収集対象とするファイルの絶対パスを Management Server 管理ファイル用退避対象定義ファイルに記述してください。

ファイルの記述例

```
${cosminexus.home}/manager/apps/MyApp.ear
D:/home/confdir/message1.conf
```

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「8.2.13　Management Server 管理ファイル用退避対象定義ファイル」を参照してください。

17.5.5　クラスタの設定

クラスタに対して，次の設定を実施します。

- スクリプトファイルの作成
- クラスタソフトウェアの環境設定

(1)　スクリプトファイルの作成

ここでは，運用管理サーバの 1:1 系切り替えシステムの監視対象および監視方法について説明します

監視対象を起動・停止するためのスクリプトファイルを作成します。また，スクリプトファイルでクラスタの監視方法を決めておく必要があります。スクリプトファイルの作成については，使用する OS のマニュアルを参照してください。

(a)　監視対象

運用管理サーバの 1:1 系切り替えシステムの監視対象を次に示します。

- Management Server

Management Server が終了したタイミングで系切り替えが実行されます。

(b)　監視方法

Management Server のプロセスの有無を監視します。監視するプロセスを次に示します。

- mngsvr.exe

(2)　Windows Server Failover Cluster の環境設定

Windows Server Failover Cluster の環境設定については，使用するクラスタソフトウェアのマニュアルを参照してください。

17.6　アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合）

1:1 系切り替えシステムとは，実行系と待機系が 1:1 で対応しているシステムです。アプリケーションサーバでは，実行系のアプリケーションサーバと待機系のアプリケーションサーバを 1:1 に配置するシステムと，実行系の運用管理サーバと待機系の運用管理サーバを 1:1 に配置するシステムを構築・運用できます。アプリケーションサーバを 1:1 に配置するシステムは，運用管理ポータルを使用して構築します。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。この節では，実行系のアプリケーションサーバと待機系のアプリケーションサーバを 1:1 に配置して系を切り替えるシステムの設定について説明します。

アプリケーションサーバの実行系と待機系を 1:1 に配置して系を切り替えるシステムでは，実行系のアプリケーションサーバでマシンの障害や運用管理エージェントの終了などの障害が発生すると，HA モニタがこれを検知して，自動的に待機している系に切り替えて業務を続行します。また，障害が発生しなくても実行中のシステムに予防保守などが必要な場合には，オペレータの操作によって待機している系に計画的に切り替えることができます。

なお，アプリケーションサーバの 1:1 系切り替えシステムの運用は，AIX または Linux の場合だけ使用できます。

機能の詳細については，「17.2　1:1 系切り替えシステムの概要」を参照してください。システム構成については，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「3.11.1　アプリケーションサーバの実行系と待機系を 1:1 にする構成（トランザクションサービスを使用しない場合）」またはマニュアル「アプリケーションサーバ システム設計ガイド」の「3.11.2　アプリケーションサーバの実行系と待機系を 1:1 にする構成（トランザクションサービスを使用する場合）」を参照してください。また，HA モニタについては，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

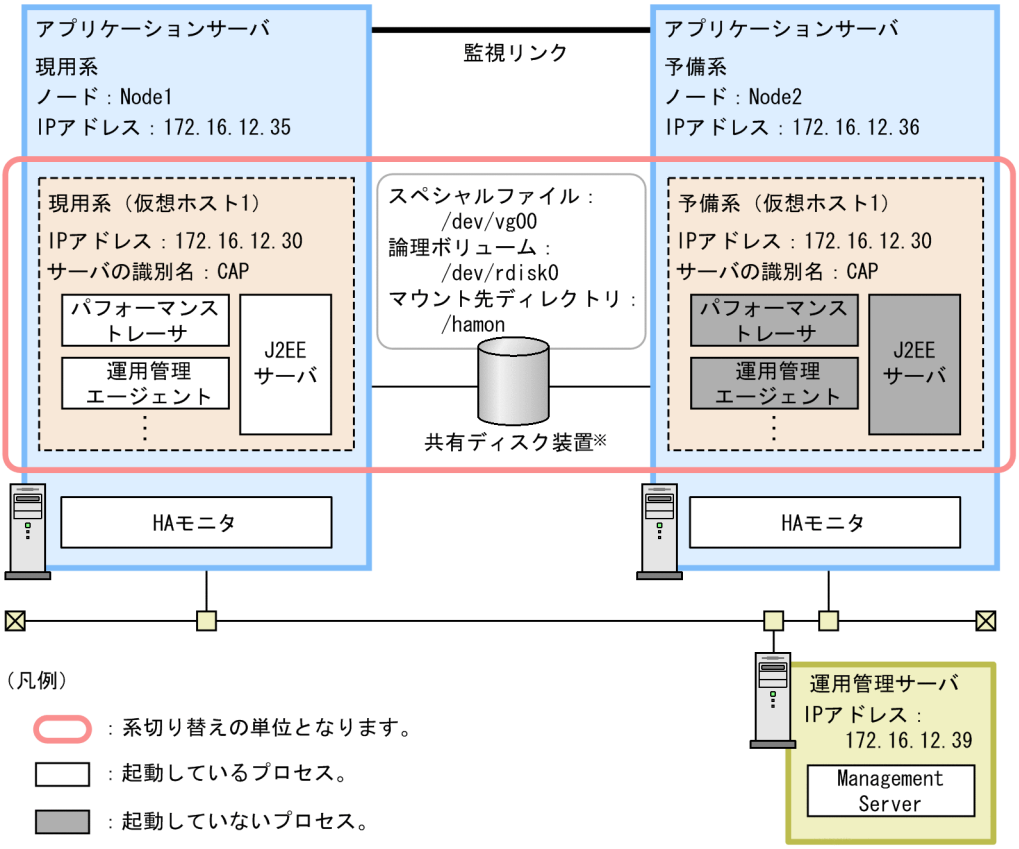
17.6.1　アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定手順

ここでは，システムの構築例とシステムの設定手順について説明します。

(1)　システムの構成例

アプリケーションサーバの 1:1 系切り替えシステムの構成例を次の図に示します。なお，以降の項では，このシステムの構成例を使用したシステムの構築例を示します。

図 17-12　アプリケーションサーバを対象にした 1:1 系切り替えシステムの構成例（UNIX の場合）



注※　グローバルトランザクションを使用するときに必要となります。
ローカルトランザクションを使用するときは不要となります。

この例では，アプリケーションサーバの実行系（現用系）とアプリケーションサーバの待機系（予備系）を 1:1 で配置しています。また，Management Server は，運用管理サーバとして，アプリケーションサーバとは別のマシンに配置しています。

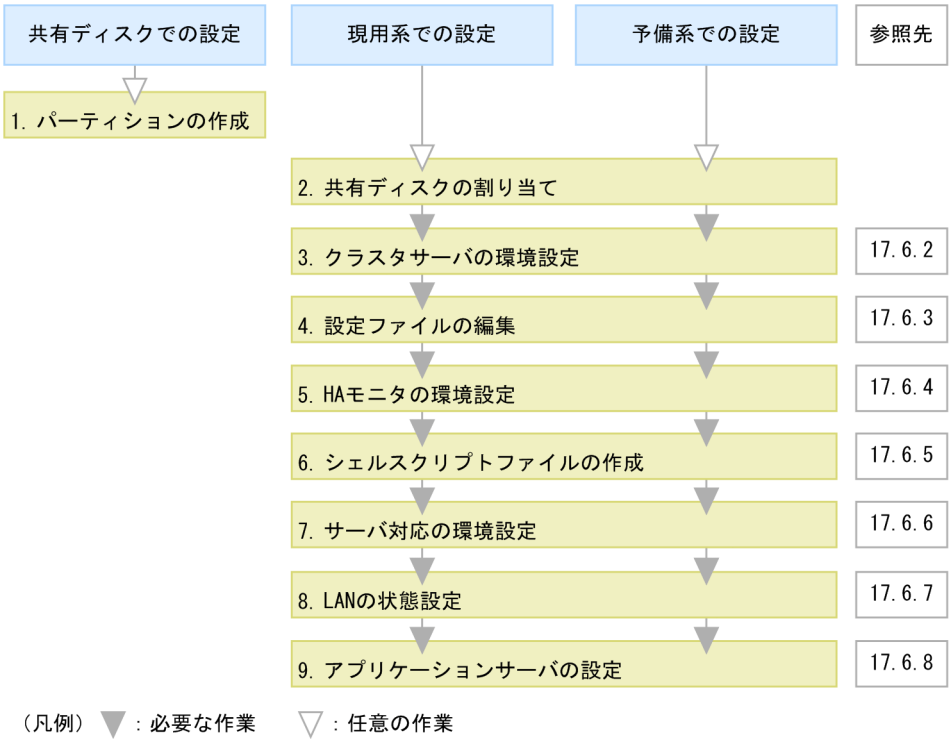
アプリケーションサーバの 1:1 系切り替えシステムでは，まずクラスタを構築して，アプリケーションサーバをクラスタ内に現用系と予備系として定義します。また，HA モニタで使用するコマンドや出力するメッセージのための，サーバの識別名（サーバの別名）を指定します。現用系と予備系とで同じ名称（例では CAP）を指定します。

なお，アプリケーションサーバをクラスタ構成に配置するためには，エイリアス IP アドレスを設けて，稼働中のノードがエイリアス IP アドレスを引き継ぐことで，クライアントがクラスタ内のノードを意識しないようにします。1:1 系切り替えシステムの場合，エイリアス IP アドレスは，HA モニタによって動的に割り当てられるアドレスとなります。

(2)　システムの設定手順

HA モニタと連携する場合には，運用管理ポータルや HA モニタのファイルの設定などが必要になります。アプリケーションサーバの 1:1 系切り替えシステムの設定手順を次の図に示します。

図 17-13　アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定手順（UNIX の場合）



図中の 1.～9.について説明します。

1. グローバルトランザクションを使用する場合は，共有ディスクにパーティションを作成して，ファイルシステムを構築します。

   グローバルトランザクションを使用するためには，トランザクション情報の格納場所を作成します。

2. グローバルトランザクションを使用する場合は，システムに共有ディスクを割り当てます。

   システムに共有ディスクを割り当てる際は，現用系と予備系で，同じマウント先ディレクトリにしてください。

3. 運用管理ポータルでクラスタサーバの環境を設定します。

   運用管理ポータルの「運用管理ドメインの構成定義」，および「論理サーバの環境設定」で，HA モニタを使用する場合の設定をします。詳細は，「17.6.2　クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

   運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。詳細は，「17.6.3　設定ファイルの編集」を参照してください。

5. HA モニタの環境を設定します。

   HA モニタの sysdef ファイルで，HA モニタの環境を定義します。詳細は，「17.6.4　HA モニタの環境設定」を参照してください。

6. シェルスクリプトファイルを作成します。

運用管理エージェントを監視，運用管理エージェントと論理サーバを起動および停止するためのシェルスクリプトファイルを作成します。詳細は，「17.6.5　シェルスクリプトファイルの作成」を参照してください。

7. サーバ対応の環境を設定します。

HA モニタの servers ファイルで，系で稼働させる現用系サーバや予備系サーバの環境を定義します。詳細は，「17.6.6　サーバ対応の環境設定」を参照してください。

8. LAN の状態を設定します。

HA モニタの LAN の状態設定ファイルで，LAN アダプタの IP アドレスなどを指定して HA モニタでの LAN の切り替えについて定義します。詳細は，「17.6.7　LAN の状態設定」を参照してください。

9. 運用管理ポータルと HA モニタのコマンドで，アプリケーションサーバの設定をします。

運用管理ポータル，HA モニタのコマンドを使用して，アプリケーションサーバをクラスタ構成に配置し，J2EE アプリケーションやリソースアダプタを設定します。詳細は，「17.6.8　アプリケーションサーバの設定」を参照してください。

注意事項

HA モニタと連携する場合のシステム設定時の注意について，次に示します。

- **HA モニタ連携のアプリケーションサーバに接続するクライアントの設定**

  HA モニタ連携のアプリケーションサーバに接続するクライアントでは，EJB のルックアップなどで，HA モニタと連携しているアプリケーションサーバを呼び出す場合，エイリアス IP アドレスでアプリケーションサーバを指定してください。

17.6.2　クラスタサーバの環境設定

ここでは，HA モニタと連携する場合の，運用管理ポータルでの設定の留意点について説明します。

参考

運用管理ポータルでシステムの動作環境を設定するには，あらかじめ，Management Server の起動などが必要です。また，Management Server を初めて使用するホストでは，Management Server をセットアップする必要があります。運用管理ポータルでの動作環境の設定や，操作手順および画面の詳細については，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)　「運用管理ドメインの構成定義」での設定

論理サーバを設置するホストおよび各論理サーバの構成を定義します。このとき，［ホストの定義］画面で，「ホスト名」にエイリアス IP アドレスを設定します。

(2) 「論理サーバの環境設定」での設定

論理 J2EE サーバ，論理 Web サーバ，および論理ネーミングサービスの設定での留意点を次に示します。

(a) 論理 J2EE サーバの設定

- ホストの固定の設定

  次の個所では，「ホストの固定」に「する」を設定してください。

  - ［J2EE コンテナの設定］画面にある「運用監視エージェントの設定」の「ホストの固定」
  - ［EJB コンテナの設定］画面にある「ホストの固定」
  - ［Web コンテナの設定］画面にある「Web サーバとの接続」および「管理用サーバの設定」の「ホストの固定」

  ホストを固定すると，運用管理ポータルの「運用管理ドメインの構成定義」で定義したホスト名が使用されます。

- グローバルトランザクションの設定

  グローバルトランザクションを使用する場合は，次の設定をしてください。

  - ［トランザクションの設定］画面にある「ライトトランザクション機能」を「無効」にします。
  - ［トランザクションの設定］画面にある「インプロセス OTS のステータスファイル格納先」に，共有ディスク装置のディレクトリを設定します。

    設定例

    /hamon

(b) 論理 Web サーバの設定

［Web サーバの設定］画面で，「ホストの固定」に「する」を設定してください。

(c) 論理ネーミングサービスの設定

［ネーミングサービスの設定］画面で，「ホストの固定」に「する」を設定してください。このとき，J2EE サーバが使用するネーミングサービス（アウトプロセスでの起動）を設定する場合は，「運用管理エージェントのホストで固定」のチェックボックスをチェックしないでください。このように設定することで，「運用管理ドメインの構成定義」の［ネーミングサービスの追加］画面で指定したホスト名が使用されます。

17.6.3 設定ファイルの編集

運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。ここでは，HA モニタと連携する場合に注意が必要なファイルの設定について説明します。

(1) 運用管理エージェントの設定

adminagent.properties（運用管理エージェントプロパティファイル）に設定する項目のうち，HA モニタと連携する場合に留意する設定項目について説明します。なお，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.adapter.bind_host キー

  adminagent.adapter.bind_host キーでエイリアス IP アドレスを指定します。管理用のリクエストは，エイリアス IP アドレスだけで受け付けます。例では，次のように設定します。

  adminagent.adapter.bind_host=172.16.12.30

- adminagent.cluster.localaddress.check キー

  アプリケーションサーバの系切り替え時に，障害などの原因で停止しなかった待機系の論理サーバ，または運用管理エージェントを停止するための設定をします。

  adminagent.cluster.localaddress.check=true

(2) HTTP Server の設定（Web サーバと連携する場合）

httpsd.conf（HTTP Server 定義ファイル）に設定する項目のうち，HA モニタと連携する場合に留意する設定項目について説明します。なお，httpsd.conf ファイルの各ディレクティブについては，マニュアル「HTTP Server」を参照してください。

- Listen ディレクティブ，BindAddress ディレクティブなど

  Listen ディレクティブ，BindAddress ディレクティブなどには，エイリアス IP アドレスを指定します。クライアントへのサービス提供は，エイリアス IP アドレスだけで実施します。

- ProxyPass ディレクティブ，ProxyPassReverse ディレクティブ

  転送先アドレスには，クラスタ IP アドレス，クラスタ IP アドレスに名前を解決できるホスト名，または"localhost"を指定します。

- ServerName ディレクティブ

  ServerName ディレクティブには，エイリアス IP アドレス，またはエイリアス IP アドレスに名前を解決できるホスト名を指定します。

(3) スマートエージェントの設定（CTM を利用する場合）

Management Server との連携で CTM を利用する場合には，次のファイルを編集する必要があります。

- localaddr

  localaddr を作成します。localaddr では，ステーショナリ IP アドレスとエイリアス IP アドレスを設定し，スマートエージェントがステーショナリ IP アドレスとエイリアス IP アドレスを通信に使用できるようにします。localaddr の設定については，「付録 B　クラスタソフトウェア連携時の TPBroker の設定（UNIX の場合)」を参照してください。

(4) Management Server の運用管理コマンド（mngsvrutil）の設定

現用系と予備系の root のホームディレクトリに，Management Server の運用管理コマンド（mngsvrutil）のクライアント側定義ファイル（.mngsvrutilrc）を用意し，Management Server の管理ユーザのユーザ ID とパスワードを設定してください。また，クライアント側定義ファイルには適切なアクセス権限を設定してください。

このファイルは，運用管理エージェントのプロセスを監視するスクリプト，運用管理エージェントと論理サーバを起動および停止するスクリプトで，mngsvrutil コマンドを実行する際に使用します。

なお，mngsvrutil コマンドのクライアント側定義ファイル（.mngsvrutilrc）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「8.2.14　.mngsvrutilrc（mngsvrutil コマンドのクライアント側定義ファイル)」を参照してください。

17.6.4 HA モニタの環境設定

使用しているシステムに従って，sysdef という定義ファイルに HA モニタの環境を定義してください。

HA モニタの環境設定については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

17.6.5 シェルスクリプトファイルの作成

運用管理エージェントのプロセスを監視し，運用管理エージェントと論理サーバを起動および停止するために，次のシェルスクリプトファイルを作成します。

- 運用管理エージェントのプロセスを監視するシェルスクリプトファイル
- 運用管理エージェントと論理サーバを起動するシェルスクリプトファイル
- 運用管理エージェントと論理サーバを停止するシェルスクリプトファイル

現用系のアプリケーションサーバと予備系のアプリケーションサーバで同一のシェルスクリプトファイルを使用し，同一のパスに配置してください。

(1) 運用管理エージェントのプロセスを監視するシェルスクリプトファイル

運用管理エージェントのプロセスを監視するシェルスクリプトファイルの例（manager_adminagent_monitor.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
AA=/opt/Cosminexus/manager/bin/adminagent
```

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

logg "### $0: started. ###"
while true
do
    CHECK=`ps -ef | grep $AA | grep -v grep`

    if [ "$CHECK" = "" ]
    then
        logg "### $0: stop. ###"
        exit 0
    fi
    sleep 10
done
```

(2) 運用管理エージェントと論理サーバを起動するシェルスクリプトファイル

運用管理エージェントと論理サーバを起動するシェルスクリプトファイルの例（manager_adminagent_start.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
SCRIPTDIR=/home/manager/hamon/bin
MNGDIR=/opt/Cosminexus/manager

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

# make adminagent.access.info
logg "### $0: make adminagent.access.info ###"
echo 172.16.12.30:28080,hostA:20295 > $MNGDIR/tmp/adminagent.access.info

# start Administration Agent
logg "### $0: starting Administration Agent. ###"
$MNGDIR/bin/adminagentctl start
if [ $? -eq 0 ] ; then
  logg "### $0: Administration Agent start normally. ###"
else
  logg "### $0: Administration Agent cannot start. ###"
  exit 1
fi

sleep 10

# start logical server
logg "### $0: starting logical servers. ###"
$MNGDIR/bin/mngsvrutil -m mnghost:28080 -t lserver1 -s start server
$MNGDIR/bin/mngsvrutil -m mnghost:28080 -t lserver2 -s start server
```

```
$MNGDIR/bin/mngsvrutil -m mnghost:28080 -t lserver3 -s start server

exit 0
```

このシェルスクリプトファイルでは，アクセス情報ファイル（/opt/Cosminexus/manager/tmp/adminagent.access.info）※の作成，運用管理エージェントの起動，論理サーバの起動をします。

注※

アクセス情報ファイルは，運用管理エージェントの起動後に削除され，Management Server からアクセスがあった場合に再度作成されます。

ここでは，Management Server からアクセスがあった場合に再度作成される内容と同じになるように設定してください。このファイルによって，系障害発生時に切り替わった場合に各論理サーバが再起動されます。

シェルスクリプトファイルでの設定内容のポイントを次に示します。

アクセス情報ファイルの作成

例のシェルスクリプトファイルで該当する記述

```
echo 172.16.12.30:28080,hostA:20295
                              > $MNGDIR/tmp/adminagent.access.info
```

フォーマット

```
<Mng_ip>:<Mng_port>,<AA_host>:<AA_port>
```

設定する内容について次に示します。

- Mng_ip
  Management Server のあるホストの IP アドレスを指定します。
- Mng_port
  Management Server 接続 HTTP ポート番号を指定します。
- AA_host
  運用管理エージェントのホスト名を指定します。Management Server の運用管理ポータルの「運用管理ドメインの構成定義」のホストに指定した値を指定してください。
- AA_port
  運用管理エージェントのポート番号を指定します。Management Server の運用管理ポータルの「運用管理ドメインの構成定義」の運用管理エージェントのポート番号に指定した値を指定してください。

論理サーバの起動

論理サーバを起動するように設定します。

論理サーバの起動には mngsvrutil コマンドを使用します。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「mngsvrutil（Management Server の運用管理コマンド)」を参照してください。

(3) 運用管理エージェントと論理サーバを停止するシェルスクリプトファイル

運用管理エージェントと論理サーバを停止するシェルスクリプトファイルの例（manager_adminagent_stop.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
MNGDIR=/opt/Cosminexus/manager

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

# stop logical server
logg "### $0: stop logical servers. ###"
$MNGDIR/bin/mngsvrutil -m mnghost:28080 -t lserver3 -s stop server
$MNGDIR/bin/mngsvrutil -m mnghost:28080 -t lserver2 -s stop server
$MNGDIR/bin/mngsvrutil -m mnghost:28080 -t lserver1 -s stop server

# stop Administration Agent
logg "### $0: stopping Administration Agent. ###"
$MNGDIR/bin/adminagentctl stop
```

17.6.6 サーバ対応の環境設定

HA モニタでのサーバ対応の環境設定では，系で稼働させる実行サーバや待機サーバの環境を定義します。

servers という定義ファイルに，アプリケーションサーバの 1:1 系切り替え用のサーバ対応の環境を定義してください。サーバ対応の環境設定での設定内容を次の表に示します。

表 17-2 サーバ対応の環境設定での設定内容（アプリケーションサーバの 1:1 系切り替えの場合）

オペランド	設定内容
name	運用管理エージェントと論理サーバを起動するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_adminagent_start.sh
alias	サーバの識別名を指定します。現用系と予備系で同じ名称を指定します。 指定例：CAP
acttype	サーバの起動方法を指定します。ここでは，HA モニタのコマンドでサーバを起動するため，「monitor」を指定します。
termcommand	運用管理エージェントと論理サーバを停止するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_adminagent_stop.sh
initial	サーバ起動時の状態を指定します。 • 現用系の場合

オペランド	設定内容
	「online」を指定します。 • 予備系の場合 「standby」を指定します。
disk	共有ディスク装置のキャラクタ型スペシャルファイル名を指定します。 指定例：/dev/vg00
lan_updown	LAN の状態設定ファイルを使用するかどうかを指定します。ここでは，LAN の状態設定ファイルを使用するため，「use」を指定します。
fs_name	切り替えるファイルシステムに対応する論理ボリュームの絶対パス名を指定します。なお，この設定は，$TPFS を UNIX ファイルで使用する場合だけ必要です。 指定例：/dev/rdisk0
fs_mount_dir	切り替えるファイルシステムのマウント先ディレクトリの絶対パス名を指定します。なお，この設定は，$TPFS を UNIX ファイルで使用する場合だけ必要です。 指定例：/hamon
patrolcommand	運用管理エージェントのプロセスを監視するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_adminagent_monitor.sh
servexec_retry	障害を検出した場合の再起動の回数を指定します。ここでは，障害を検出した場合に再起動をしないで系を切り替えるため，「0」を指定します。
waitserv_exec	運用管理エージェントと論理サーバの起動完了処理を実行するときに起動コマンドの実行完了を待つかどうかを指定します。ここでは，実行完了を待つため，「yes」を指定します。

サーバ対応の環境設定の詳細については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

17.6.7　LAN の状態設定

HA モニタの LAN の状態設定ファイルで，LAN アダプタの IP アドレスなどを指定して HA モニタでの LAN の切り替えについて定義します。

次のファイルにエイリアス IP アドレスを設定してください。

- **<サーバ識別名>.up ファイル**
  LAN を接続する場合に使用します。LAN アダプタに追加するエイリアス IP アドレスを指定します。
- **<サーバ識別名>.down ファイル**
  LAN を切り離す場合に使用します。LAN アダプタから削除するエイリアス IP アドレスを指定します。

<サーバ識別名>には，サーバ対応の環境設定（servers ファイル）の alias の値を指定してください。

LAN の状態設定の詳細については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

17.6.8　アプリケーションサーバの設定

アプリケーションサーバの設定では，運用管理ポータル，HA モニタのコマンドを使用してアプリケーションサーバをクラスタ構成に配置します。また，論理サーバをセットアップして設定情報を配布し，J2EE サーバに J2EE アプリケーションとリソースアダプタをインポートします。アプリケーションサーバの設定手順を次に示します。

1. 運用管理サーバを起動します。

   運用管理サーバの起動方法については，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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

2. Node1 および Node2 で HA モニタの monbegin コマンドを実行して，現用系のアプリケーションサーバを実行系として起動し，予備系のアプリケーションサーバを待機状態にします。

   - Node1 でのコマンドの実行例

     「# monbegin CAP」を実行して，Node1 の現用系を起動します。

   - Node2 でのコマンドの実行例

     「# monbegin CAP」を実行して，Node2 の予備系を待機状態にします。

3. 運用管理ポータルで，Node1 の現用系の J2EE サーバをセットアップします。

   「運用管理ドメインの構成定義」の［セットアップ］画面で，J2EE サーバをセットアップします。運用管理ポータルでの操作手順および画面の詳細については，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。

4. 運用管理ポータルで，J2EE サーバに設定した情報を Node1 の現用系に配布します。

   「論理サーバの環境設定」の［設定情報の配布］画面で，J2EE サーバに設定した情報を Node1 の現用系に配布します。

5. サーバ管理コマンドを使用して，Node1 の現用系の J2EE サーバに J2EE アプリケーションとリソースアダプタをインポートします。また，インポートした J2EE アプリケーションとリソースアダプタを設定および開始します。

   サーバ管理コマンドでの操作については，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。

   なお，J2EE サーバおよびその前提プロセスを起動してからサーバ管理コマンドを実行してください。また，サーバ管理コマンドでの操作が終わったら，サーバ管理コマンドを停止させてください。

6. Node1 で HA モニタの monswap コマンドを実行して，系を切り替えます。

   - Node1 でのコマンドの実行例

     「# monswap CAP」を実行して，サーバ識別名「CAP」で系を切り替えます。Node2 の予備系が起動され，Node1 の現用系が待機状態になります。

7. Node2 の予備系で，手順 3.～手順 5.を実施します。

予備系には，現用系とまったく同じ設定をするため，設定手順や内容については，現用系と同じです。

8. Node2 で HA モニタの monswap コマンドを実行して，系を切り替えて通常の運用状態に戻します。

   - Node2 でのコマンドの実行例

     「# monswap CAP」を実行して，サーバ識別名「CAP」で系を切り替えます。Node1 の現用系が起動され，Node2 の予備系が待機状態になります。

17.7　運用管理サーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合）

1:1 系切り替えシステムとは，実行系と待機系が 1:1 で対応しているシステムです。アプリケーションサーバでは，実行系のアプリケーションサーバと待機系のアプリケーションサーバを 1:1 に配置するシステムと，実行系の運用管理サーバと待機系の運用管理サーバを 1:1 で配置するシステムを構築・運用できます。運用管理サーバを 1:1 に配置して系を切り替えるシステムは，Smart Composer 機能を使用して構築します。この節では，実行系の運用管理サーバと待機系の運用管理サーバを 1:1 に配置して系を切り替えるシステムの設定について説明します。

運用管理サーバを 1:1 に配置して系を切り替えるシステムでは，実行系の運用管理サーバでマシンの障害や Management Server のプロセスの終了が発生すると，HA モニタがこれを検知して，自動的に待機している系に切り替えて業務を続行します。また，障害が発生しなくても実行中のシステムに予防保守などが必要な場合には，オペレータの操作によって待機している系に計画的に切り替えることができます。

なお，運用管理サーバの 1:1 系切り替えシステムの運用は，AIX または Linux の場合だけ使用できます。

機能の詳細については，「17.2　1:1 系切り替えシステムの概要」を参照してください。システム構成については，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「3.11.3　運用管理サーバの実行系と待機系を 1:1 にする構成」を参照してください。また，HA モニタについては，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

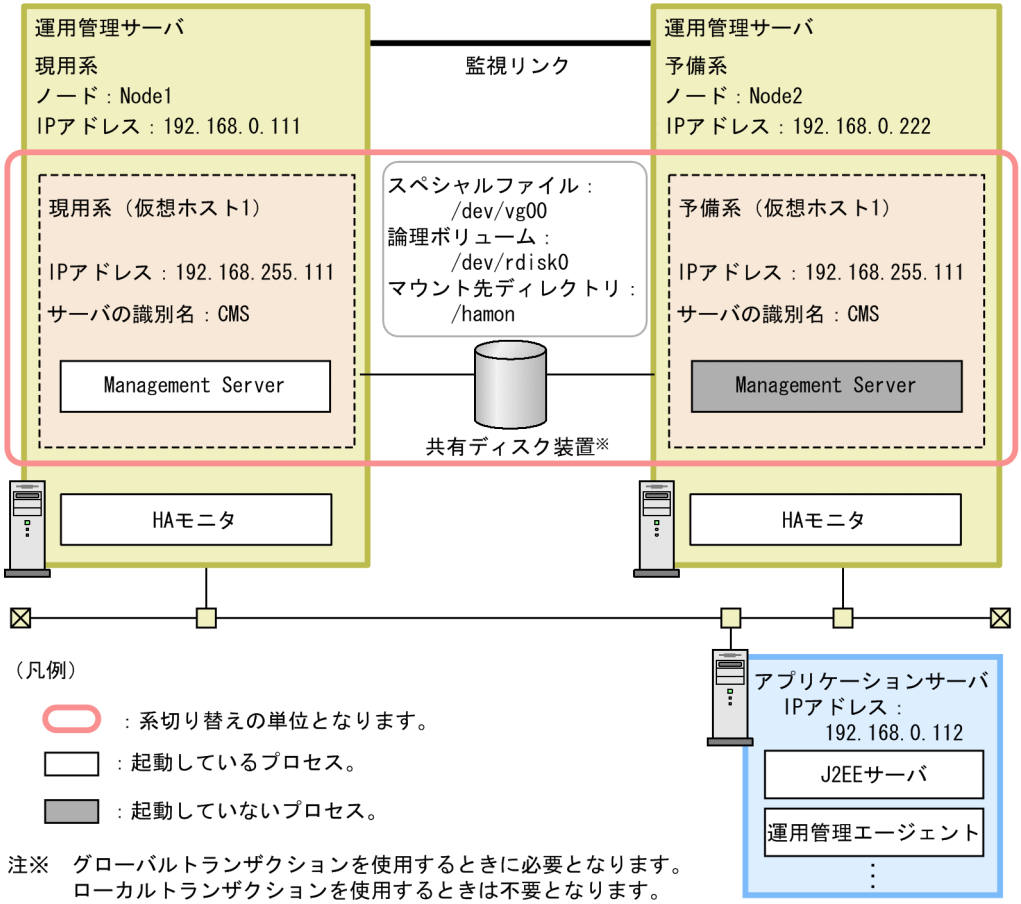
17.7.1　運用管理サーバを対象にした 1:1 系切り替えシステムの設定手順

ここでは，システムの構築例とシステムの設定手順について説明します。

(1)　システムの構成例

運用管理サーバの 1:1 系切り替えシステムの構成例を次の図に示します。なお，以降の項では，このシステムの構成例を使用したシステムの設定例を示します。

図 17-14　運用管理サーバを対象にした 1:1 系切り替えシステムの構成例（UNIX の場合）



運用管理サーバの実行系（現用系）と運用管理サーバの待機系（予備系）を 1:1 で配置しています。

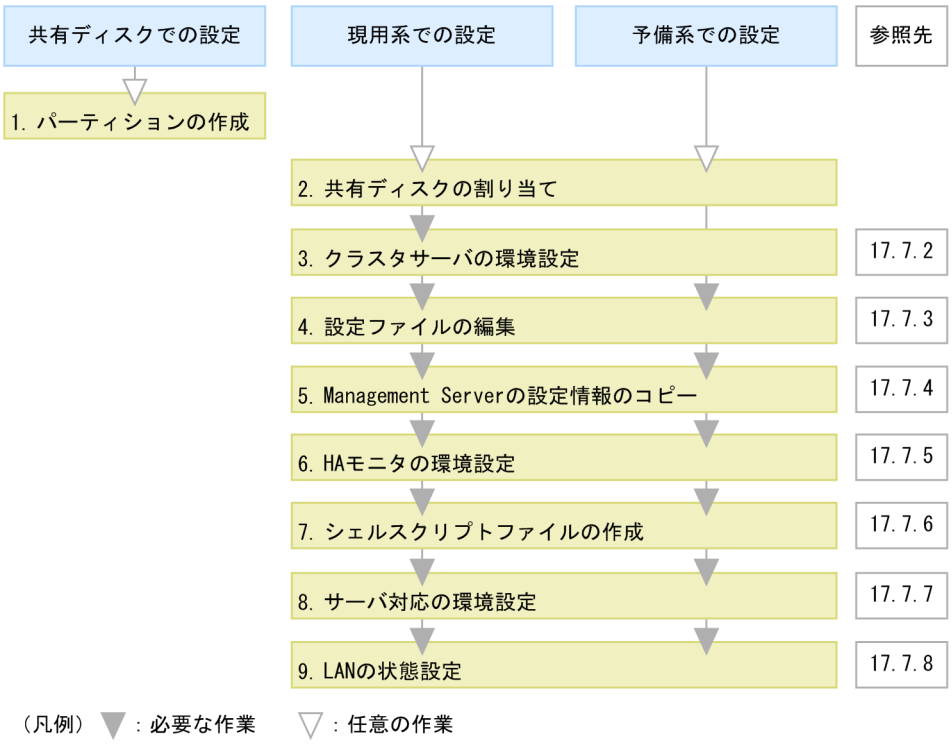
運用管理サーバの 1:1 系切り替えシステムでは，まずクラスタを構築して，運用管理サーバをクラスタ内に現用系と予備系として定義します。また，HA モニタで使用するコマンドや出力するメッセージのための，サーバの識別名（サーバの別名）を指定します。現用系と予備系とで同じ名称（例では CMS）を指定します。

なお，アプリケーションサーバをクラスタ構成に配置するためには，エイリアス IP アドレスを設けて，稼働中のノードがエイリアス IP アドレスを引き継ぐことで，クライアントがクラスタ内のノードを意識しないようにします。1:1 系切り替えシステムの場合，エイリアス IP アドレスは，HA モニタによって動的に割り当てられるアドレスとなります。

(2)　システムの設定手順

HA モニタと連携する場合には，Management Server や HA モニタのファイルの設定などが必要になります。運用管理サーバの 1:1 系切り替えシステムの設定手順を次の図に示します。

図 17-15　運用管理サーバを対象にした 1:1 系切り替えシステムの設定手順（UNIX の場合）



図中の 1.～9.について説明します。

1. グローバルトランザクションを使用する場合は，共有ディスクにパーティションを作成して，ファイルシステムを構築します。

   グローバルトランザクションを使用するためには，トランザクション情報の格納場所を作成します。

2. グローバルトランザクションを使用する場合は，システムに共有ディスクを割り当てます。

   システムに共有ディスクを割り当てる際は，現用系と予備系で，同じマウント先ディレクトリにしてください。

3. Management Server でクラスタサーバの環境を設定します。

   Management Server の Smart Composer 機能の簡易構築定義ファイルで，HA モニタを使用する場合の設定をします。詳細は，「17.7.2　クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

   簡易構築定義ファイルでは設定できない，Management Server などの各種定義ファイルを設定します。詳細は，「17.7.3　設定ファイルの編集」を参照してください。

5. 現用系から予備系に Management Server の設定情報をコピーします。

   現用系と予備系を同じ環境にするために，現用系での Management Server に関する設定内容を予備系にコピーします。詳細は，「17.7.4　現用系から予備系への Management Server の設定情報のコピー」を参照してください。

6. HA モニタの環境を設定します。

HA モニタの sysdef ファイルで，HA モニタの環境を定義します。詳細は，「17.7.5　HA モニタの環境設定」を参照してください。

7. シェルスクリプトファイルを作成します。

Management Server を監視，運用管理サーバを起動および停止するためのシェルスクリプトファイルを作成します。詳細は，「17.7.6　シェルスクリプトファイルの作成」を参照してください。

8. サーバ対応の環境を設定します。

HA モニタの servers ファイルで，系で稼働させる現用系サーバや予備系サーバの環境を定義します。詳細は，「17.7.7　サーバ対応の環境設定」を参照してください。

9. LAN の状態を設定します。

HA モニタの LAN の状態設定ファイルで，LAN アダプタの IP アドレスなどを指定して HA モニタでの LAN の切り替えについて定義します。詳細は，「17.7.8　LAN の状態設定」を参照してください。

17.7.2　クラスタサーバの環境設定

ここでは，HA モニタと連携する場合の，Management Server での設定時の留意点について説明します。

参考

Management Server を初めて使用するホストでは，Management Server をセットアップする必要があります。Management Server のセットアップについては，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「4.1.14　運用管理機能を構築する」を参照してください。

(1)　簡易構築定義ファイルでの設定

Management Server の Smart Composer 機能の簡易構築定義ファイルで，論理サーバを設置するホストおよび各論理サーバの構成を定義します。また，必要に応じて，各論理サーバの<configuration>タグ内に，論理サーバの環境や起動／停止の動作などを設定します。設定済みの簡易構築定義ファイルを基に，Smart Composer 機能のコマンドを使用して，Web システムをセットアップします。

セットアップした Web システムを Smart Composer 機能のコマンドで起動したあと，必要に応じて，サーバ管理コマンドを使用して，J2EE アプリケーション，リソースアダプタをインポートして，開始します。

17.7.3　設定ファイルの編集

設定ファイルを編集します。ここでは，HA モニタと連携する場合に注意が必要な設定ファイルについて説明します。なお，現用系と予備系で必要となる設定が異なります。

1. Management Server の設定
2. Management Server の運用管理コマンド（mngsvrutil）の設定

現用系では，1.と 2.を設定してください。

予備系では，2.だけ設定してください。1.については，現用系から Management Server の設定情報をコピーすることで設定できるため，ここでは設定不要です。

(1) Management Server の設定

Management Server 環境設定ファイル（mserver.properties）に設定する項目のうち，HA モニタと連携する場合に留意する設定項目について説明します。なお，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。

- mngsvr.myhost.name
  mngsvr.myhost.name キーでエイリアス IP アドレスを指定します。例では，次のように設定します。
  mngsvr.myhost.name=192.168.255.111

(2) Management Server の運用管理コマンド（mngsvrutil）の設定

ローカルシステムアカウントのホームディレクトリに，Management Server の運用管理コマンド（mngsvrutil）のクライアント側定義ファイル（.mngsvrutilrc）を用意し，Management Server の管理ユーザのユーザ ID とパスワードを設定してください。また，クライアント側定義ファイルには適切なアクセス権限を設定してください。

このファイルは，Management Server のプロセスを監視するスクリプト，運用管理サーバを起動および停止するスクリプトで，mngsvrutil コマンドを実行する際に使用します。

なお，mngsvrutil コマンドのクライアント側定義ファイル（.mngsvrutilrc）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「8.2.14　.mngsvrutilrc（mngsvrutil コマンドのクライアント側定義ファイル)」を参照してください。

17.7.4 現用系から予備系への Management Server の設定情報のコピー

現用系と予備系を同じ環境にするために，現用系での Management Server に関する設定情報を予備系にコピーします。Management Server の設定情報の退避／回復コマンドを使用して，現用系で設定した Management Server の各種定義ファイル，Management Server に登録した J2EE アプリケーションやリソースアダプタなどを，現用系から予備系にコピーしてください。

(1) Management Server の設定情報のコピー手順

Management Server の設定情報のコピー手順を次に示します。

1. 現用系で，mstrexport コマンドを実行します。

   mstrexport コマンドでは，mstrexport コマンドの実行対象となる現用系の Management Server の設定情報を収集し，収集した情報を ZIP 形式のファイルに保存します。mstrexport コマンドの引数には，保存する ZIP 形式のファイルのファイル名を指定してください。

   - コマンドの格納場所

     /opt/Cosminexus/manager/bin

   - 実行例

     mstrexport /tmp/work/mstruct.zip

   なお，mstrexport コマンドは，mstrexport コマンドの実行対象となる Management Server の起動，停止に関係なく実行できます。

2. 手順 1.で保存された ZIP 形式のファイルを，現用系から予備系にコピーします。

3. 予備系で，mstrimport コマンドを実行します。

   mstrimport コマンドでは，コピーしてきた ZIP 形式のファイルを，コマンドの実行対象となる予備系の Management Server に展開します。これによって，現用系の Management Server と予備系の Management Server を同じ設定にできます。

   mstrimport コマンドの引数には，コピーしてきた ZIP 形式のファイルのファイル名を指定してください。

   - コマンドの格納場所

     /opt/Cosminexus/manager/bin

   - 実行例

     mstrimport /tmp/recovery/mstruct.zip

   なお，mstrimport コマンドは，mstrimport コマンドの実行対象の Management Server が停止している場合だけ実行できます。

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「mstrexport (Management Server 管理ファイルの退避)」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 コマンド編」の「mstrimport（Management Server 管理ファイルの回復)」を参照してください。

(2) 収集対象のファイルの定義

mstrexport コマンドの引数として，mstrexport コマンドの収集対象を記述するファイル（Management Server 管理ファイル用退避対象定義ファイル）を指定できます。

実行例

```
mstrexport /tmp/work/mstruct.zip "/work/filelist.txt"
```

mstrexport コマンドでは，デフォルトの状態で，Management Server の各種定義ファイル，Management Server に登録した J2EE アプリケーションやリソースアダプタなど，Management Server でのシステム構築・運用に必要な情報が収集できます。これらの情報に加えて，ユーザ作成のコマンドな

ども mstrexport コマンドでの収集対象に追加したい場合には，収集対象とするファイルの絶対パスを Management Server 管理ファイル用退避対象定義ファイルに記述してください。

ファイルの記述例

```
${cosminexus.home}/manager/apps/MyApp.ear
/home/confdir/message1.conf
```

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「8.2.13　Management Server 管理ファイル用退避対象定義ファイル」を参照してください。

17.7.5　HA モニタの環境設定

使用しているシステムに従って，sysdef という定義ファイルに HA モニタの環境を定義してください。

HA モニタの環境設定については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

17.7.6　シェルスクリプトファイルの作成

Management Server のプロセスを監視し，運用管理サーバを起動および停止するために，次のシェルスクリプトファイルを作成します。

- Management Server のプロセスを監視するシェルスクリプトファイル
- 運用管理サーバを起動するシェルスクリプトファイル
- 運用管理サーバを停止するシェルスクリプトファイル

現用系の運用管理サーバと予備系の運用管理サーバで同一のシェルスクリプトファイルを使用し，同一のパスに配置してください。

(1)　Management Server のプロセスを監視するシェルスクリプトファイル

Management Server のプロセスを監視するシェルスクリプトファイルの例（manager_mngsvr_monitor.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
MNGDIR=/opt/Cosminexus/manager

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

```
logg "### $0: started. ###"
while true
do
$MNGDIR/bin/mngsvrutil -m 192.168.255.111:28080 check mngsvr
    if [ $? -ne 0 ]
    then
        logg "### $0: stop. ###"
        exit 0
    fi
    sleep 10
done
```

このシェルスクリプトファイルでは，mngsvrutil の check コマンドで Management Server の稼働状況を確認します。なお，mngsvrutil コマンドの-m オプションの引数のホスト名にはエイリアス IP アドレスを指定します。

(2) 運用管理サーバを起動するシェルスクリプトファイル

運用管理サーバを起動するシェルスクリプトファイルの例（manager_mngsvr_start.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
MNGDIR=/opt/Cosminexus/manager
RETRY_COUNT=20
RETRY_INTERVAL=10

logg()
{
    echo `date '+[%Y/%m/%d %H:%M:%S]'`"[$$]: $1" ¥
        >> ${LOGDIR}/mngsvr.log 2>&1
}
# start Management Server
logg "### $0: starting Management Server. ###"
$MNGDIR/bin/mngsvrctl start &

I=0
while [ $I -lt $RETRY_COUNT ] ; do
  $MNGDIR/bin/mngsvrutil -m 192.168.255.111:28080 check mngsvr
  if [ $? -eq 0 ] ; then
    break
  fi
  sleep $RETRY_INTERVAL
  I=`expr $I + 1`
done
exit 0
```

(3) 運用管理サーバを停止するシェルスクリプトファイル

運用管理サーバを停止するシェルスクリプトファイルの例（manager_mngsvr_stop.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
MNGDIR=/opt/Cosminexus/manager
logg()
{
    echo `date '+[%Y/%m/%d %H:%M:%S]'`"[$$]: $1" ¥
        >> ${LOGDIR}/mngsvr.log 2>&1
}
# stop Management Server
logg "### $0: stop Management Server. ###"
$MNGDIR/bin/mngsvrctl stop
exit 0
```

17.7.7　サーバ対応の環境設定

HA モニタでのサーバ対応の環境設定では，系で稼働させる実行サーバや待機サーバの環境を定義します。

servers という定義ファイルに，運用管理サーバの 1:1 系切り替え用のサーバ対応の環境を定義してください。サーバ対応の環境設定での設定内容を次の表に示します。

表 17-3　サーバ対応の環境設定での設定内容（運用管理サーバの 1:1 系切り替えの場合）

オペランド	設定内容
name	運用管理サーバを起動するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_mngsvr_start.sh
alias	サーバの識別名を指定します。現用系と予備系で同じ名称を指定します。 指定例：CMS
acttype	サーバの起動方法を指定します。ここでは，HA モニタのコマンドでサーバを起動するため，「monitor」を指定します。
termcommand	運用管理サーバを停止するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_mngsvr_stop.sh
initial	サーバ起動時の状態を指定します。 • 現用系の場合 「online」を指定します。 • 予備系の場合 「standby」を指定します。
disk	共有ディスク装置のキャラクタ型スペシャルファイル名を指定します。 指定例：/dev/vg00
lan_updown	LAN の状態設定ファイルを使用するかどうかを指定します。ここでは，LAN の状態設定ファイルを使用するため，「use」を指定します。
fs_name	切り替えるファイルシステムに対応する論理ボリュームの絶対パス名を指定します。なお，この設定は，$TPFS を UNIX ファイルで使用する場合だけ必要です。

オペランド	設定内容
	指定例：/dev/rdisk0
fs_mount_dir	切り替えるファイルシステムのマウント先ディレクトリの絶対パス名を指定します。なお，この設定は，$TPFS を UNIX ファイルで使用する場合だけ必要です。 指定例：/hamon
patrolcommand	Management Server のプロセスを監視するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_mngsvr_monitor.sh
servexec_retry	障害を検出した場合の再起動の回数を指定します。ここでは，障害が発生した場合に再起動しないで系を切り替えるため，「0」を指定します。
waitserv_exec	運用管理サーバの起動完了処理を実行するときに起動コマンドの実行完了を待つかどうかを指定します。ここでは，実行完了を待つため，「yes」を指定します。

servers ファイルの例を次に示します。

servers ファイルの例（現用系の場合）

現用系の場合の servers ファイルの例を次に示します。

```
server  name           /home/manager/hamon/bin/manager_mngsvr_start.sh,
        alias          CMS,
        acttype        monitor,
        termcommand    /home/manager/hamon/bin/manager_mngsvr_stop.sh,
        initial        online,
        disk           /dev/vg00,
        lan_updown     use,
        fs_name        /dev/rdisk0,
        fs_mount_dir   /hamon,
        patrolcommand  /home/manager/hamon/bin/manager_mngsvr_monitor.sh,
        servexec_retry 0,
        waitserv_exec  yes;
```

servers ファイルの例（予備系の場合）

予備系の場合の servers ファイルの例を次に示します。

```
server  name           /home/manager/hamon/bin/manager_mngsvr_start.sh,
        alias          CMS,
        acttype        monitor,
        termcommand    /home/manager/hamon/bin/manager_mngsvr_stop.sh,
        initial        standby,
        disk           /dev/vg00,
        lan_updown     use,
        fs_name        /dev/rdisk0,
        fs_mount_dir   /hamon,
        patrolcommand  /home/manager/hamon/bin/manager_mngsvr_monitor.sh,
        servexec_retry 0,
        waitserv_exec  yes;
```

サーバ対応の環境設定の詳細については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

17.7.8　LAN の状態設定

HA モニタの LAN の状態設定ファイルで，LAN アダプタの IP アドレスなどを指定して HA モニタでの LAN の切り替えについて定義します。

次のファイルにエイリアス IP アドレスを設定してください。

- <サーバ識別名>.up ファイル
  LAN を接続する場合に使用します。LAN アダプタに追加するエイリアス IP アドレスを指定します。
- <サーバ識別名>.down ファイル
  LAN を切り離す場合に使用します。LAN アダプタから削除するエイリアス IP アドレスを指定します。

<サーバ識別名>には，サーバ対応の環境設定（servers ファイル）の alias の値を指定してください。

LAN の状態設定の詳細については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

17.8　アプリケーションサーバを対象にした 1:1 系切り替えシステムの起動と停止（Windows の場合）

この節では，アプリケーションサーバを対象にした 1:1 系切り替えシステムを利用する場合の，システムの起動と停止の手順について説明します。稼働開始後に J2EE サーバ，またはバッチサーバをメンテナンスするときの，起動と停止の手順についても説明します。

アプリケーションサーバを対象にした 1:1 系切り替えシステムでは，クラスタサービスを使用します。クラスタサービスの詳細については，OS のマニュアルを参照してください。

1:1 系切り替えシステムを利用して運用する場合は，あらかじめ現用系と予備系の 2 種類のホストの用意や，クラスタサービスの対象となる運用管理エージェントを監視・起動・停止するスクリプトの登録など，必要な環境設定をしておく必要があります。設定方法については，「17.4　アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合）」を参照してください。

なお，この節では，手順内で計画的な系切り替えを実行する前には，現用系ホストが実行系ホストとして起動されていることを前提にして説明します。

注意事項

運用管理エージェントのダウンによって系切り替えが発生した場合，運用管理エージェントのダウンしたホストには各論理サーバのプロセスが残っている場合があります。プロセスが残っていると系の切り戻しが発生した時に，論理サーバの起動ができなくなってしまうため，系の切り戻しが発生する前にすべてのプロセスを停止させる必要があります。また，そのほかに，業務が正しく開始されるための作業をあらかじめ実施しておいてください。

ポイント

1:1 系切り替えシステムを利用する場合，Management Server は運用管理サーバで起動します。なお，個々のアプリケーションサーバのホストでは，運用管理エージェントを OS 起動と同時に起動する設定にしないでください。

17.8.1　アプリケーションサーバを対象にした 1:1 系切り替えシステムの起動

ここでは，アプリケーションサーバを対象にした 1:1 系切り替えシステムの起動方法について説明します。

アプリケーションサーバを対象にした 1:1 系切り替えシステムを起動するには，あらかじめ運用管理サーバを起動しておくことが必要です。運用管理サーバを起動していない場合には，運用管理サーバを先に起動しておいてください。運用管理サーバの起動方法については，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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

アプリケーションサーバを対象にした 1:1 系切り替えシステムの起動手順を次に示します。

1. [スタート] メニューの [コントロールパネル] − [パフォーマンスとメンテナンス] − [管理ツール] から，[クラスタ アドミニストレータ] を選択します。
   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）で現用系ノードを選択し，[ファイル] メニューの [クラスタサービスの開始] を選択します。
   現用系ノードのクラスタサービスが開始されます。

3. コンソールツリー（左ペイン）で予備系ノードを選択し，[ファイル] メニューの [クラスタサービスの開始] を選択します。
   予備系ノードのクラスタサービスが開始されます。

4. コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，[ファイル] メニューの [オンラインにする] を選択します。
   1:1 系切り替えシステムが起動します。

17.8.2 アプリケーションサーバを対象にした 1:1 系切り替えシステムの停止

ここでは，アプリケーションサーバを対象にした 1:1 系切り替えシステムの停止方法について説明します。

1. [スタート] メニューの [コントロールパネル] − [パフォーマンスとメンテナンス] − [管理ツール] から，[クラスタ アドミニストレータ] を選択します。
   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，[ファイル] メニューの [オフラインにする] を選択します。
   実行系ノードおよび待機系ノードが含まれているリソースグループがオフラインになります。

3. コンソールツリー（左ペイン）で予備系ノードを選択し，[ファイル] メニューの [クラスタサービスの停止] を選択します。
   予備系ノードのクラスタサービスが停止します。

4. コンソールツリー（左ペイン）で現用系ノードを選択し，[ファイル] メニューの [クラスタサービスの停止] を選択します。
   現用系ノードのクラスタサービスが停止します。

なお，運用管理サーバは最後に手動で停止します。運用管理サーバの停止方法については，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「4.1.3　システムの停止手順」およびマニュアル「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「4.1.4　システムの停止方法」を参照してください。

17.8.3　アプリケーションサーバを対象にした 1:1 系切り替えシステムで計画的に系を切り替えるときの起動と停止

トラブル発生時以外で，アプリケーションサーバを対象にした 1:1 系切り替えシステムの実行系と待機系を切り替える手順を次に示します。系を切り替えるときは，待機系のホストでアプリケーションサーバが待機状態になっている必要があります。

1. ［スタート］メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。
   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［グループの移動］を選択します。
   系が切り替わります。

17.8.4　アプリケーションサーバを対象にした 1:1 系切り替えシステムをメンテナンスする場合の起動と停止

アプリケーションサーバを対象にした 1:1 系切り替えシステムをメンテナンスする場合の，起動と停止の手順を次に示します。

なお，これらの手順は，すでに実行系のホストでアプリケーションサーバが起動している場合の手順です。

(1)　再起動が不要なメンテナンスの場合の手順

1. 実行系のホストで，動作中の J2EE アプリケーションとリソースアダプタを停止します。
   バッチサーバの場合は，バッチアプリケーションが実行中でないことを確認してからリソースアダプタを停止してください。
   停止方法については，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「付録 F.4 システムの停止方法」を参照してください。

2. サーバ管理コマンドを使用してメンテナンス処理を実行します。
   サーバ管理コマンドでの操作については，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。

3. 手順 1.で停止した J2EE アプリケーションとリソースアダプタを開始します。
   J2EE サーバの場合は J2EE アプリケーションとリソースアダプタを開始します。また，バッチサーバの場合はリソースアダプタを開始します。
   開始方法については，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「付録 C.2　システムの起動方法」を参照してください。

4. 必要に応じて J2EE アプリケーションとリソースアダプタの動作確認をします。

5. [スタート] メニューの [コントロールパネル] − [パフォーマンスとメンテナンス] − [管理ツール] から，[クラスタ アドミニストレータ] を選択します。

   クラスタアドミニストレータが開始されます。

6. コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，[ファイル] メニューの [グループの移動] を選択します。

   系が切り替わります。

7. 切り替え後の実行系のホストで手順 1.〜手順 6.を実行します。

8. 切り替え後の実行系のホストを実行系に戻す場合，コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，[ファイル] メニューの [グループの移動] を選択します。

   メンテナンスが完了します。

(2) 再起動が必要なメンテナンスの場合（両方の系を同時に停止する方法）

1. 論理サーバを一括停止する設定にしていない場合は，実行系のホストで各論理サーバを停止します。

   Management Server によって一括停止する設定の場合は，不要な手順です。停止方法については，次のマニュアルを参照してください。

   - J2EE アプリケーションを実行するシステムの場合

     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「4.1.3　システムの停止手順」およびマニュアル「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「4.1.4　システムの停止方法」を参照してください。

   - バッチアプリケーションを実行するシステムの場合

     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「6.1.3　システムの停止手順」およびマニュアル「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「6.1.4　システムの停止方法」を参照してください。

2. [スタート] メニューの [コントロールパネル] − [パフォーマンスとメンテナンス] − [管理ツール] から，[クラスタ アドミニストレータ] を選択します。

   クラスタアドミニストレータが開始されます。

3. コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，[ファイル] メニューの [オフラインにする] を選択します。

   実行系ノードおよび待機系ノードが含まれているリソースグループがオフラインになります。

4. コンソールツリー（左ペイン）で予備系ノードを選択し，[ファイル] メニューの [クラスタサービスの停止] を選択します。

   予備系ノードのクラスタサービスが停止します。

5. コンソールツリー（左ペイン）で現用系ノードを選択し，[ファイル] メニューの [クラスタサービスの停止] を選択します。

現用系ノードのクラスタサービスが停止します。

6. 現用系と予備系の両方のホストで，定義ファイルの変更など，メンテナンス作業を実行します。

7. コンソールツリー（左ペイン）で現用系ノードを選択し，［ファイル］メニューの［クラスタサービスの開始］を選択します。
   現用系ノードのクラスタサービスが開始されます。

8. コンソールツリー（左ペイン）で予備系ノードを選択し，［ファイル］メニューの［クラスタサービスの開始］を選択します。
   予備系ノードのクラスタサービスが開始されます。

9. コンソールツリー（左ペイン）で現用系ノードおよび予備系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［オンラインにする］を選択します。
   1:1 系切り替えシステムが再起動します。

10. 論理サーバを一括起動する設定にしていない場合は，実行系のホストで各論理サーバを起動します。
    Management Server によって一括起動する設定の場合は，不要な手順です。起動方法については，次のマニュアルを参照してください。

    - J2EE アプリケーションを実行するシステムの場合
      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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。
    - バッチアプリケーションを実行するシステムの場合
      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「6.1.1　システムの起動手順」およびマニュアル「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「6.1.2　システムの起動方法」を参照してください。

11. 実行系のホストで，必要に応じて，定義の変更に関連した動作確認を実行します。
    メンテナンスが完了します。

(3)　再起動が必要なメンテナンスの場合（両方の系を同時に停止しない方法）

1. 待機系のホストで，定義ファイルの変更など，メンテナンス作業を実行します。

2. ［スタート］メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。
   クラスタアドミニストレータが開始されます。

3. コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［グループの移動］を選択します。
   実行系のホストのアプリケーションサーバが停止してから，今まで待機していた系でアプリケーションサーバが起動します。これによって，メンテナンス済みのホストが実行系ホストになります。

4. 切り替え後の実行系のホストで，必要に応じて，定義の変更に関連した動作確認を実行します。

5. 切り替え後の待機系のホストで定義ファイルの編集など，メンテナンス作業を実行します。

6. 切り替え後の実行系のホストを実行系に戻す場合，コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［グループの移動］を選択します。

7. 切り替え後の実行系のホストで，必要に応じて，定義の変更に関連した動作確認を実行します。
   メンテナンスが完了します。

17.9　運用管理サーバを対象にした 1:1 系切り替えシステムの起動と停止（Windows の場合）

この節では，運用管理サーバを対象にした 1:1 系切り替えシステムを利用する場合の，システムの起動と停止の方法について説明します。

運用管理サーバを対象にした 1:1 系切り替えシステムでは，クラスタサービスを使用します。クラスタサービスの詳細については，OS のマニュアルを参照してください。

運用管理サーバを対象にした 1:1 系切り替えシステムを利用するには，あらかじめ現用系と予備系の 2 種類のホストの用意や，クラスタサービスの対象となる Management Server を監視・起動・停止するスクリプトの登録など，必要な環境設定をしておく必要があります。設定方法については，「17.5　運用管理サーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合）」を参照してください。現用系と予備系の詳細については，「15.2　クラスタソフトウェアと連携して実現できる運用」を参照してください。

17.9.1　運用管理サーバを対象にした 1:1 系切り替えシステムの起動

ここでは，運用管理サーバを対象にした 1:1 系切り替えシステムの起動方法について説明します。

運用管理サーバを対象にした 1:1 系切り替えシステムを起動するには，あらかじめ起動するアプリケーションサーバの運用管理エージェントを起動しておくことが必要です。運用管理エージェントを起動していない場合には，運用管理エージェントを先に起動しておいてください。運用管理エージェントの起動方法については，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「4.1.2　システムの起動方法」を参照してください。

運用管理サーバを対象にした 1:1 系切り替えシステムの起動手順を次に示します。

1. **［スタート］メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。**
   クラスタアドミニストレータが開始されます。

2. **コンソールツリー（左ペイン）で運用管理サーバの現用系ノードを選択し，［ファイル］メニューの［クラスタサービスの開始］を選択します。**
   現用系ノードのクラスタサービスが開始されます。

3. **コンソールツリー（左ペイン）で運用管理サーバの予備系ノードを選択し，［ファイル］メニューの［クラスタサービスの開始］を選択します。**
   予備系ノードのクラスタサービスが開始されます。

4. **コンソールツリー（左ペイン）で運用管理サーバの現用系ノードおよび予備系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［オンラインにする］を選択します。**
   運用管理サーバを対象にした 1:1 系切り替えシステムが起動します。

17.9.2　運用管理サーバを対象にした 1:1 系切り替えシステムの停止

ここでは，運用管理サーバを対象にした 1:1 系切り替えシステムの停止方法について説明します。

1. ［スタート］メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。

   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）で運用管理サーバの実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［オフラインにする］を選択します。

   運用管理サーバの実行系ノードおよび待機系ノードが含まれているリソースグループがオフラインになります。

3. コンソールツリー（左ペイン）で運用管理サーバの予備系ノードを選択し，［ファイル］メニューの［クラスタサービスの停止］を選択します。

   運用管理サーバの予備系ノードのクラスタサービスが停止します。

4. コンソールツリー（左ペイン）で運用管理サーバの現用系ノードを選択し，［ファイル］メニューの［クラスタサービスの停止］を選択します。

   運用管理サーバの現用系ノードのクラスタサービスが停止します。

5. アプリケーションサーバの運用管理エージェントを停止します。

   運用管理エージェントの停止方法については，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「4.1.4　システムの停止方法」を参照してください。

17.9.3　運用管理サーバを対象にした 1:1 系切り替えシステムで計画的に系を切り替えるときの起動と停止

トラブル発生時以外で，運用管理サーバを対象にした 1:1 系切り替えシステムの実行系と待機系を切り替える手順を次に示します。系を切り替えるときは，待機系のホストでアプリケーションサーバが待機状態になっている必要があります。

1. システムの運用中に Management Server の設定情報を変更した場合，実行系の Management Server の設定情報を収集して，実行系から待機系へコピーします。また，mstrimport コマンドを実行して，コピーした Management Server の設定情報を待機系に適用します。

   実行系の Management Server と待機系の Management Server が同じ設定になります。

   Management Server の設定情報を実行系から待機系へコピーして，待機系に適用する方法については，「17.7.4　現用系から予備系への Management Server の設定情報のコピー」の 1:1 系切り替えシステムの設定に関する説明を参照してください。

2. ［スタート］メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。

クラスタアドミニストレータが開始されます。

3. コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［グループの移動］を選択します。

   系が切り替わります。

17.10　1:1 系切り替えシステムの起動と停止（UNIX の場合）

この節では，HA モニタによる 1:1 系切り替えシステムを利用する場合の，システムの起動と停止の手順について説明します。稼働開始後に J2EE サーバ，バッチサーバ，Management Server などをメンテナンスする場合の，起動と停止の手順についても説明します。なお，HA モニタを利用できるのは，AIX または Linux の場合だけです。

1:1 系切り替えシステムを利用して運用する場合は，あらかじめ現用系と予備系の 2 種類のホストの用意や，HA モニタの監視の対象となる運用管理エージェント（アプリケーションサーバの 1:1 系切り替えシステムの場合）または Management Server（運用管理サーバの 1:1 系切り替えシステムの場合）を監視・起動・停止するスクリプトの登録など，必要な環境設定をしておく必要があります。設定方法については，「17.6　アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合）」および「17.7　運用管理サーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合）」を参照してください。

アプリケーションサーバの 1:1 系切り替えシステムと，運用管理サーバの 1:1 系切り替えシステムで実行できる運用操作およびその参照先を次の表に示します。

表 17–4　1:1 系切り替えシステムで実行できる運用操作

運用操作	アプリケーションサーバの 1:1 系切り替えシステム	運用管理サーバの 1:1 系切り替えシステム
システムの起動	17.10.1 参照	17.10.1 参照
システムの停止	17.10.2 参照	17.10.2 参照
実行系と待機系を計画的に系切り替えする場合のシステムの起動と停止	17.10.3 参照	17.10.3 参照
メンテナンスする場合のシステムの起動と停止	17.10.4 参照	17.10.5 参照

また，次のコマンドは，HA モニタが提供しているコマンドです。詳細は，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

- monbegin（HA モニタとのインタフェースを持たないサーバを起動）
- monend（HA モニタとのインタフェースを持たない実行サーバの停止連絡）
- monsbystp（待機サーバの停止）
- monswap（計画系切り替え）

17.10.1　1:1 系切り替えシステムの起動

1:1 系切り替えシステムを利用する場合のシステム起動時の留意事項，およびシステムの起動手順について説明します。

(1)　システム起動時の留意事項

1:1 系切り替えシステムを利用する場合のシステム起動時の留意事項を次に示します。

- 現用系と予備系のホストに配置された HA モニタは，OS の起動と同時に起動されています。
- アプリケーションサーバの 1:1 系切り替えシステムを起動する場合，運用管理サーバを使用するときは，運用管理サーバをあらかじめ起動しておいてください。運用管理サーバの起動方法については，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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。
- 運用管理サーバの 1:1 系切り替えシステムを起動する場合，アプリケーションサーバを別ホストに配置しているときは，運用管理ドメイン内のアプリケーションサーバの運用管理エージェントをあらかじめ起動しておいてください。運用管理エージェントの起動方法については，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「4.1.2　システムの起動方法」を参照してください。

(2)　システムの起動手順

1:1 系切り替えシステムを利用する場合のシステムの起動手順を次に示します。

1. 現用系のホストで monbegin コマンドを実行して，現用系のホストを実行系として起動します。

```
# 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」に指定されているサーバの識別名を指定します。

これによって，運用管理エージェントを起動するスクリプトファイルに定義されている処理が実行されて，現用系のホストが実行系として起動します。

2. 予備系のホストで monbegin コマンドを実行して，予備系のホストを待機系として起動します。

```
# 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」に指定されているサーバの識別名を指定します。

これによって，予備系のホストが待機系となり，実行系の障害に備えます。

注意事項

Management Server 起動時に運用管理ドメイン内の論理サーバを一括起動する設定にしていない場合は，現用系のホストの各論理サーバを起動してください。なお，Management Server 起動時に運用管理ドメイン内の論理サーバを一括起動するように設定にしている場合，論理サーバの手動起動は不要です。起動方法については，次のマニュアルを参照してください。

- J2EE アプリケーションを実行するシステムの場合

  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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

- バッチアプリケーションを実行するシステムの場合

  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「6.1.1　システムの起動手順」およびマニュアル「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「6.1.2 システムの起動方法」を参照してください。

参考

servers ファイルの定義（HA モニタでのサーバの環境設定）については，「17.6.6　サーバ対応の環境設定」，または「17.7.7　サーバ対応の環境設定」を参照してください。

17.10.2　1:1 系切り替えシステムの停止

1:1 系切り替えシステムの停止手順について説明します。

ここでは，次の二つの場合の停止手順について説明します。

- 実行系と待機系の両方のホストを停止する場合
- 待機系のホストだけ停止する場合

(1)　実行系と待機系の両方のホストを停止する場合

実行系と待機系の両方のホストを停止する場合の手順を次に示します。

1. **運用管理エージェントを停止するスクリプトで，運用管理ドメイン内の論理サーバを一括停止する設定にしていない場合は，実行系のホストの各論理サーバを停止します。**

   運用管理エージェントを停止するスクリプトで論理サーバを停止する設定にしている場合は，不要な手順です。論理サーバの停止方法については，次のマニュアルを参照してください。

   - J2EE アプリケーションを実行するシステムの場合

     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「4.1.4　システムの停止方法」を参照してください。

   - バッチアプリケーションを実行するシステムの場合

     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「6.1.4　システムの停止方法」を参照してください。

2. **実行系のホストで monend コマンドを実行して，実行系のホストの運用管理エージェントを停止します。**

   ```
   # monend 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」に指定されているサーバの識別名を指定します。

   これによって，実行系のホストの運用管理エージェントが停止します。また，HA モニタによって待機系に自動的に停止指示が出され，待機系のホストの運用管理エージェントも停止します。

なお，アプリケーションサーバの 1:1 系切り替えシステムの場合，運用管理サーバを使用しているときは，運用管理サーバは最後に手動で停止します。運用管理サーバの停止方法については，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「4.1.3　システムの停止手順」およびマニュアル「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「4.1.4　システムの停止方法」を参照してください。

参考

servers ファイルの定義（HA モニタでのサーバの環境設定）については，「17.6.6　サーバ対応の環境設定」，または「17.7.7　サーバ対応の環境設定」を参照してください。

(2)　待機系のホストだけ停止する場合

実行系は停止しないで待機系で待機状態をやめる場合は，待機系のホストで次のコマンドを実行します。

```
# monsbystp 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」に指定されているサーバの識別名を指定します。

17.10.3　1:1 系切り替えシステムで計画的に系を切り替える場合の起動と停止

1:1 系切り替えシステムでトラブル発生時以外の場合に，実行系と待機系を計画的に切り替えるときの手順について説明します。系を切り替えるときには，待機系のホストでアプリケーションサーバが待機状態になっている必要があります。

1:1 系切り替えシステムで実行系と待機系を計画的に切り替えるときの手順を次に示します。なお，現用系のホストが実行系として起動されていることを前提としています。

1. **運用管理エージェントを停止するスクリプトで，運用管理ドメイン内の論理サーバを一括停止する設定にしていない場合は，実行系のホストで各論理サーバを停止します。**

   運用管理エージェントを停止するスクリプトで論理サーバを停止する設定にしている場合は，不要な手順です。論理サーバの停止方法については，次のマニュアルを参照してください。

   - J2EE アプリケーションを実行するシステムの場合

     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「4.1.4　システムの停止方法」を参照してください。

   - バッチアプリケーションを実行するシステムの場合

     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「6.1.4　システムの停止方法」を参照してください。

2. **実行系のホストで monswap コマンドを実行して，系を切り替えます。**

   ```
   # monswap 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」に指定されているサーバの識別名を指定します。

運用管理エージェントを停止するスクリプトに定義されている処理が実行されて，実行系のホストのアプリケーションサーバが停止します。そのあと，今まで待機していた系でアプリケーションサーバが起動します。これで，計画的な系切り替えの手順は完了です。

参考

- 1:1 系切り替えシステムをメンテナンスする場合の起動と停止の手順については，「17.10.4 アプリケーションサーバの 1:1 系切り替えシステムをメンテナンスする場合の起動と停止」または「17.10.5　運用管理サーバの 1:1 系切り替えシステムをメンテナンスする場合の起動と停止」を参照してください。
- servers ファイルの定義（HA モニタでのサーバの環境設定）については，「17.6.6　サーバ対応の環境設定」，または「17.7.7　サーバ対応の環境設定」を参照してください。

17.10.4　アプリケーションサーバの 1:1 系切り替えシステムをメンテナンスする場合の起動と停止

アプリケーションサーバの 1:1 系切り替えシステムをメンテナンスする場合の，起動と停止の手順について説明します。

ここでは，次の三つの場合の停止手順について説明します。

- 再起動が不要なメンテナンスの場合
- 再起動が必要なメンテナンスの場合（両方の系を同時に停止する方法）
- 再起動が必要なメンテナンスの場合（両方の系を同時に停止しない方法）

なお，これらの手順は，すでに実行系のホストでアプリケーションサーバが起動している場合の手順です。

注意事項

運用管理エージェントのダウンによって系切り替えが発生した場合，運用管理エージェントのダウンしたホストには各論理サーバのプロセスが残っている場合があります。プロセスが残っていると系の切り戻しが発生した時に，論理サーバの起動ができなくなってしまうため，系の切り戻しが発生する前にすべてのプロセスを停止させる必要があります。また，そのほかに，業務が正しく開始されるための作業をあらかじめ実施しておいてください。

(1)　再起動が不要なメンテナンスの場合

再起動が不要なメンテナンスをする場合の起動と停止の手順を次に示します。

1. 実行系のホストで，動作中の J2EE アプリケーションとリソースアダプタを停止します。

バッチサーバの場合は，バッチアプリケーションが実行中でないことを確認してからリソースアダプタを停止してください。

停止方法については，次のマニュアルを参照してください。

- J2EE アプリケーションを実行するシステムの場合

  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「4.1.3　システムの停止手順」およびマニュアル「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「4.1.4　システムの停止方法」を参照してください。

- バッチアプリケーションを実行するシステムの場合

  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「6.1.3　システムの停止手順」およびマニュアル「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「6.1.4　システムの停止方法」を参照してください。

2. サーバ管理コマンドを使用してメンテナンス処理を実行します。

   サーバ管理コマンドでの操作については，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。

3. 手順 1.で停止した J2EE アプリケーションとリソースアダプタを開始します。

   J2EE サーバの場合は J2EE アプリケーションとリソースアダプタを開始します。また，バッチサーバの場合はリソースアダプタを開始します。

   開始方法については，次のマニュアルを参照してください。

   - J2EE アプリケーションを実行するシステムの場合

     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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

   - バッチアプリケーションを実行するシステムの場合

     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「6.1.1　システムの起動手順」およびマニュアル「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「6.1.2　システムの起動方法」を参照してください。

4. 必要に応じて J2EE アプリケーションとリソースアダプタの動作確認をします。

5. 実行系のホスト（現用系のホスト）で，monswap コマンドを実行して，系を切り替えます。

   実行系のホスト（現用系ホスト）が待機系のホストに切り替わり，待機系だったホスト（予備系ホスト）が実行系ホストになります。

6. 切り替え後の実行系のホスト（予備系のホスト）で手順 1.～手順 4.を実行します。

7. 切り替え後の実行系のホスト（予備系のホスト）で，必要に応じて，再度 monswap コマンドを実行します。

   現用系のホストを実行系に戻す場合に実行してください。

   これで，メンテナンス時の起動と停止の手順は完了です。

(2) 再起動が必要なメンテナンスの場合（両方の系を同時に停止する方法）

再起動が必要なメンテナンスをする場合に，両方の系を同時に停止するときの起動と停止の手順を次に示します。

1. 運用管理エージェントを停止するスクリプトで，運用管理ドメイン内の論理サーバを一括停止する設定にしていない場合は，実行系のホストで各論理サーバを停止します。

   運用管理エージェントを停止するスクリプトで論理サーバを停止する設定にしている場合は，不要な手順です。論理サーバの停止方法については，次のマニュアルを参照してください。

   - J2EE アプリケーションを実行するシステムの場合

     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「4.1.4　システムの停止方法」を参照してください。

   - バッチアプリケーションを実行するシステムの場合

     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「6.1.4　システムの停止方法」を参照してください。

2. 実行系のホストで monend コマンドを実行して，システムを停止します。

   ```
   # monend 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」に指定されているサーバの識別名を指定します。

   実行系のホストが停止します。また，HA モニタによって待機系に自動的に停止指示が出され，待機系のホストも停止します。

3. 現用系と予備系の両方のホストで，定義ファイルの変更など，メンテナンス作業を実行します。

4. 現用系のホストで monbegin コマンドを実行して，現用系のホストの運用管理エージェントを起動します。

   ```
   # 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」に指定されているサーバの識別名を指定します。

   これによって，現用系のホストが，実行系として起動します。

5. 予備系のホストで monbegin コマンドを実行します。

   ```
   # 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」に指定されているサーバの識別名を指定します。

   これによって，予備系のホストが，待機系として起動します。

6. 運用管理エージェントを起動するスクリプトで運用管理ドメイン内の論理サーバを一括起動する設定にしていない場合は，実行系のホストの各論理サーバを起動します。

   運用管理エージェントを起動するスクリプトで論理サーバを起動する設定にしている場合は，不要な手順です。論理サーバの起動方法については，次のマニュアルを参照してください。

- J2EE アプリケーションを実行するシステムの場合
  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「4.1.2　システムの起動方法」を参照してください。
- バッチアプリケーションを実行するシステムの場合
  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「6.1.2　システムの起動方法」を参照してください。

7. 実行系のホストで，必要に応じて，定義の変更に関連した動作確認を実行します。

参考

servers ファイルの定義（HA モニタでのサーバの環境設定）については，「17.6.6　サーバ対応の環境設定」を参照してください。

(3)　再起動が必要なメンテナンスの場合（両方の系を同時に停止しない方法）

再起動が必要なメンテナンスをする場合に，両方の系を同時に停止しないときの起動と停止の手順を次に示します。

1. 待機系のホストで，定義ファイルの変更など，メンテナンス作業を実行します。
2. 実行系のホストで monswap コマンドを実行して，系を切り替えます。

```
# monswap 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」に指定されているサーバの識別名を指定します。
実行系のホストのアプリケーションサーバが停止してから，今まで待機していた系でアプリケーションサーバが起動します。これによって，メンテナンス済みのホストが実行系ホストになります。

3. 切り替え後の実行系のホスト（予備系のホスト）で，必要に応じて，定義の変更に関連した動作確認を実行します。
4. 切り替え後の待機系のホスト（現用系のホスト）で定義ファイルの編集など，メンテナンス作業を実行します。
5. 切り替え後の実行系のホスト（予備系のホスト）で，必要に応じて，再度 monswap コマンドを実行します。
   現用系のホストを実行系に戻す場合に実行してください。
6. 切り替え後の実行系のホスト（現用系のホスト）で，必要に応じて，定義の変更に関連した動作確認を実行します。
   これで，メンテナンス時の起動と停止の手順は完了です。

参考

servers ファイルの定義（HA モニタでのサーバの環境設定）については，「17.6.6　サーバ対応の環境設定」を参照してください。

17.10.5　運用管理サーバの 1:1 系切り替えシステムをメンテナンスする場合の起動と停止

Management Server の設定を変更するために，運用管理サーバの 1:1 系切り替えシステムをメンテナンスする場合の起動と停止の手順について説明します。

なお，これらの手順は，すでに実行系のホストで Management Server が起動している場合の手順です。

1. 実行系のホスト（現用系のホスト）で，mstrexport コマンドを実行します。

```
# mstrexport mstruct.zip
```

現用系のホストの Management Server の設定情報が自動収集されて，収集された情報が ZIP 形式のファイルに保存されます。

下線部分には，自動収集された設定情報を保存する ZIP 形式のファイルのパスを指定します。

2. 手順 1.で保存された ZIP 形式のファイルを，実行系から待機系にコピーします。

3. 待機系のホスト（予備系のホスト）で，mstrimport コマンドを実行します。

```
# mstrimport mstruct.zip
```

下線部分には，手順 2 でコピーした ZIP 形式のファイルのパスを指定します。

これによって，手順 2.でコピーした Management Server の設定情報が待機系のホストに展開されて，実行系の Management Server と待機系の Management Server が同じ設定になります。

4. 実行系のホストで monswap コマンドを実行して，系を切り替えます。

```
# monswap 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」に指定されているサーバの識別名を指定します。

実行系のホストの Management Server が停止してから，今まで待機していた系で Management Server が起動します。Management Server の起動時に実行系の設定情報がシステムに反映されます。

5. 切り替え後の実行系のホスト（予備系のホスト）で，必要に応じて，定義の変更に関連した動作確認を実行します。

6. 切り替え後の実行系のホスト（予備系のホスト）で，必要に応じて，再度 monswap コマンドを実行します。

現用系のホストを実行系に戻す場合に実行してください。

7. 切り替え後の実行系のホスト（現用系のホスト）で，必要に応じて，定義の変更に関連した動作確認を実行します。

これで，メンテナンス時の起動と停止の手順は完了です。

参考

servers ファイルの定義（HA モニタでのサーバの環境設定）については，「17.7.7　サーバ対応の環境設定」を参照してください。

17.11　系切り替えシステムの設定内容の確認方法

現用系と予備系の J2EE サーバおよび HTTP Server の設定内容は，基本的に同一の設定内容になります。現用系と予備系の次に示す設定ファイルを比較して確認してください。

注意事項

J2EE サーバは，使用するコンポーネントによって定義が異なる場合がありますので，それらのプロパティの設定内容は各コンポーネントの仕様に従ってください。

17.11.1　J2EE サーバの設定

表 17–5　J2EE サーバの設定ファイル

項番	設定ファイル	ファイル格納先
1	usrconf.cfg (J2EE サーバ用オプション定義ファイル)	• 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>/
2	usrconf.properties※1 (J2EE サーバ用ユーザプロパティファイル)	
3	server.policy (J2EE サーバ用セキュリティポリシーファイル)	
4	Connector 属性ファイル	cjgetrarprop（RAR ファイルの属性の取得）コマンドで取得する。※2

注※1 「webserver.connector.http.permitted.hosts」は異なっていても問題ありません。

注※2 ユーザ名とパスワードは取得できないため，設定の内容は確認できません。

17.11.2　HTTP Server の設定

表 17–6　HTTP Server の設定ファイル

項番	設定ファイル	ファイル格納先
1	httpsd.conf	• Windows の場合 <HTTP Server インストールディレクトリ>¥servers¥HWS_<論理サーバ名>¥conf¥ • UNIX の場合 /opt/hitachi/httpsd/servers/HWS_<論理サーバ名>/conf/
2	mime.types	

17.11.3　運用管理サーバの設定

運用管理サーバの設定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「8.1　Managerで使用するファイルの一覧」を参照してください。

18 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）

この章では，クラスタソフトウェアと連携して，相互系切り替えシステムで運用するシステムについて説明します。

18.1　この章の構成

この章の構成を次の表に示します。

表 18–1　この章の構成（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））

分類	タイトル	参照先
解説	相互系切り替えシステムの概要	18.2
	相互系切り替えシステムのシステム構成と動作	18.3
設定	相互系切り替えシステムの設定（Windows の場合）	18.4
	相互系切り替えシステムの設定（UNIX の場合）	18.5
運用	相互系切り替えシステムの起動と停止（Windows の場合）	18.6
	相互系切り替えシステムの起動と停止（UNIX の場合）	18.7

注　「実装」,「注意事項」について，この機能固有の説明はありません。

18.2　相互系切り替えシステムの概要

相互系切り替えシステムとは，1:1 系切り替えシステムの構成で，2 台のサーバがそれぞれ現用系として動作しながら，互いの予備系になるシステムです。

この節では，相互スタンバイシステムのシステムの構成例と系切り替え処理の流れについて説明します。系切り替えのタイミングや系切り替え時のシステムの動作などは 1:1 系切り替え構成の場合と同じです。参照先を次の表に示します。

表 18–2　系切り替え時の動作と参照先

系切り替え時の動作	参照先
系切り替えのタイミング	17.3.2
系切り替え時のシステムの動作	17.3.4
系切り替え時の情報の引き継ぎ	17.3.5

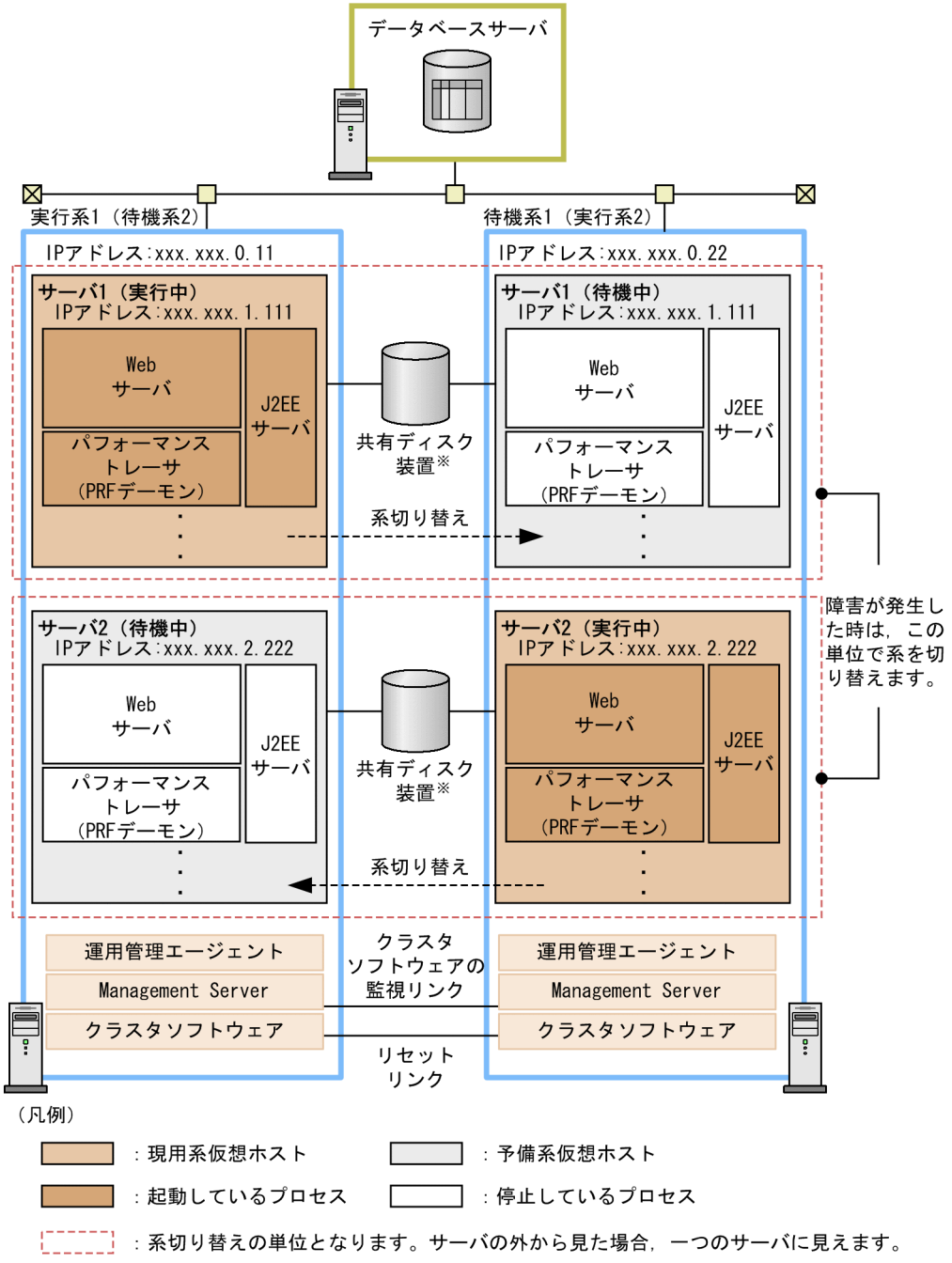
18.3　相互系切り替えシステムのシステム構成と動作

この節では，相互系切り替えシステムのシステムの構成例，系切り替え処理の流れなどについて説明します。

18.3.1　相互系切り替えシステムのシステム構成例

クラスタソフトウェアと連携した相互系切り替えシステムの構成例を次の図に示します。系切り替え時，クライアントやデータベースなどサーバの外からは，同じ論理アドレスでアプリケーションサーバが再起動したように見えます。

図 18-1　相互系切り替えシステムの構成例



注※　グローバルトランザクションを使用するため必要となります。ローカルトランザクションを使用するときは不要となります。

図に示す，相互系切り替えシステムについて説明します。

- 実行系 1（待機系 2）および待機系 1（実行系 2）の各アプリケーションサーバは，異なる運用管理ドメインで管理されています。
- それぞれの運用管理ドメイン内に，現用系および予備系の二つの仮想ホストを定義しています。現用系および予備系の仮想ホストは，それぞれ実行系アプリケーションサーバ，待機系アプリケーションサーバとして使用します。

- 障害発生時は，相互の運用管理ドメイン内の仮想ホストが切り替えられます。例えば，図中のサーバ 1 の現用系仮想ホストで障害が発生した場合，サーバ 1 の予備系仮想ホストに系が切り替えられます。
- 各サーバの運用に使用する IP アドレスは，クラスタソフトウェアによって動的に割り当てられる IP アドレス（エイリアス IP アドレス）を使用します。図中の相互系切り替えシステムの場合，「xxx.xxx.1.111」および「xxx.xxx.2.222」がエイリアス IP アドレスとなります。なお，クラスタソフトウェアは IP アドレス単位に LAN を切り替えるため，サーバ 1 およびサーバ 2 のクラスタ IP アドレスには一意となる値が割り当てられます。系切り替え時は，現用系仮想ホストでエイリアス IP アドレスが削除され，予備系仮想ホストでエイリアス IP アドレスが追加されることで処理が引き継がれます。
- Management Server から運用管理エージェントに対するリクエストの送信には，系切り替えによって他系へ移動しない IP アドレス（ステーショナリ IP アドレス）を使用します。図中の相互系切り替えシステムの場合，現用系 1 の「xxx.xxx.0.11」，および予備系 1 の「xxx.xxx.0.22」がステーショナリ IP アドレスとなります。

> **参考**
>
> 仮想ホストとは，一つのマシンに複数の異なる IP アドレスを割り当てて，複数の物理ホストとして使用できる構成です。同一運用管理ドメイン内の仮想ホストは，一つの運用管理エージェントによってアプリケーションサーバの起動や停止などの運用操作を制御できますが，運用に使用する IP アドレスは異なるため，見かけ上は異なる物理ホストとして扱われます。

なお，相互系切り替えシステムでは次のような運用ができます。

共有ディスク装置の使用

共有ディスク装置の使用については，ローカルトランザクションの場合とグローバルトランザクションの場合とで異なります。

- ローカルトランザクションの場合

  共有ディスク装置は不要です。ローカルトランザクションでは，実行系と待機系との間で引き継ぐセッション情報がないため，共有ディスク装置を使用しません。

- グローバルトランザクションの場合

  共有ディスク装置が必要になります。共有ディスク装置は，系切り替え時に，OTS のステータスなどのトランザクション情報を引き継ぐために使用します。

JP1 との連携

クラスタソフトウェアを使用した構成では，JP1 とも連携できます。

JP1 と連携する場合，アプリケーションサーバには，JP1/Base なども必要になります。クラスタソフトウェアでの JP1 の管理は，アプリケーションサーバとは別に行う必要があります。

データベースサーバでのクラスタソフトウェアとの連携

データベースサーバでもクラスタソフトウェアを使用した構成にすることもできます。この場合，アプリケーションサーバ側では，仮想アドレス（論理アドレス）だけを認識していれば，特にデータベースサーバがクラスタソフトウェアを使用していることを意識する必要はありません。

負荷分散機の適用

このシステム構成例では示していませんが，同一構成の Web サーバを複数用意して，負荷分散機を適用することもできます。これによって，Web サーバの信頼性・稼働率を上げることができます。

相互系切り替えシステムのシステム構成の詳細については，マニュアル「アプリケーションサーバ システム設計ガイド」の「3.11.4　アプリケーションサーバの実行系と待機系を相互スタンバイにする構成」を参照してください。

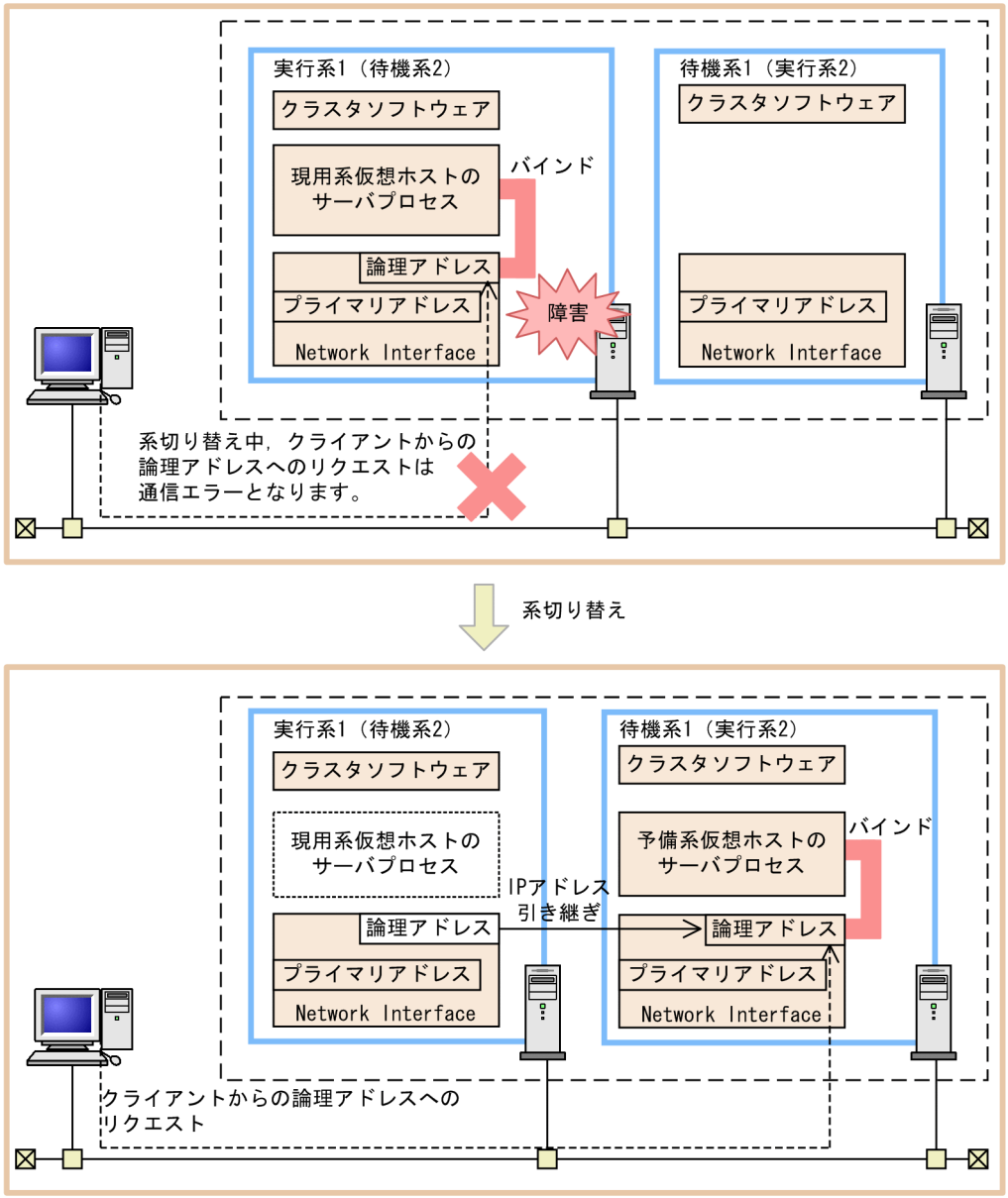
18.3.2　相互系切り替えシステムでの系切り替え処理の流れ

相互系切り替えシステムでの系切り替え時の動作と，処理の流れについて説明します。

(1)　系切り替えの動作

相互系切り替えシステムでの系切り替えは次の図のようにして行われます。この図では，実行系 1（待機系 2）の現用系仮想ホストに障害が発生し，待機系 1（実行系 2）の予備系仮想ホストに系を切り替える場合の例を示します。

図 18-2　相互系切り替えシステムでの系切り替えの動作



・現用系仮想ホストのサーバプロセスは停止して，予備系仮想ホストのサーバプロセスが起動します。
・論理アドレスのIPアドレスは，現用系仮想ホストから予備系仮想ホストに引き継がれます。

(2)　系切り替えの処理の流れ

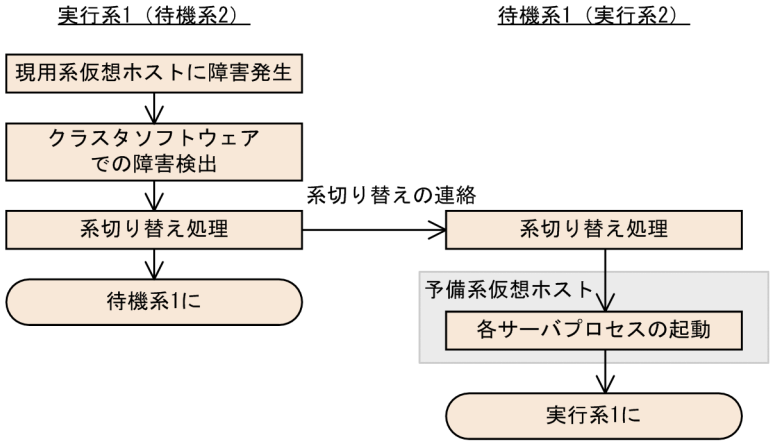
相互系切り替えシステムに適用できる系切り替えには，自動系切り替えと計画系切り替えの二つがあります。**自動系切り替え**とは，クラスタソフトウェアが自動的に系を切り替える方法です。実行系に障害が発生したときは，自動系切り替えになります。一方，**計画系切り替え**とは，オペレータが系を保守する際に，計画的に系を切り替える方法です。実行系からクラスタソフトウェアのコマンドを使用して切り替えます。

ここでは，系切り替え処理の流れについて，自動系切り替えの場合と計画系切り替えの場合に分けて説明します。

(a)　自動系切り替えの処理の流れ

相互系切り替えシステムでの自動系切り替えの処理の流れを次の図に示します。この図では，実行系 1（待機系 2）の現用系仮想ホストに障害が発生して，待機系 1（実行系 2）の予備系仮想ホストに系を切り替えるときの処理の流れを示します。

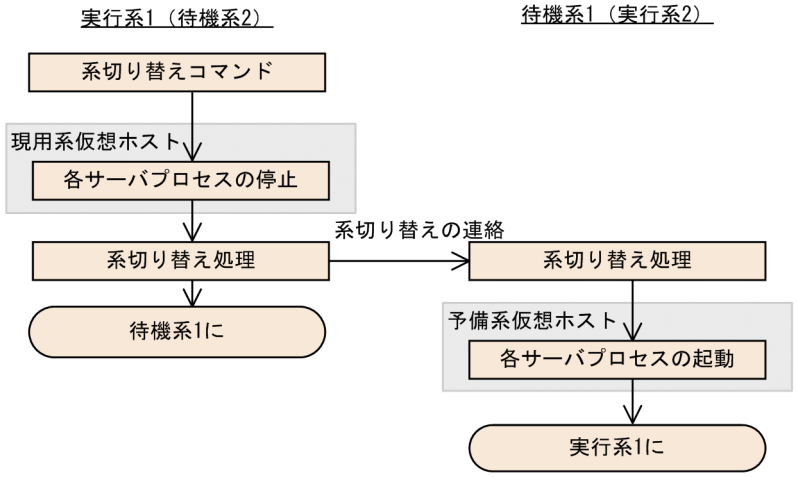
図 18–3　相互系切り替えシステムでの自動系切り替えの処理の流れ



(b)　計画系切り替えの処理の流れ

相互系切り替えシステムでの計画系切り替えの処理の流れを次の図に示します。この図では，クラスタソフトウェアのコマンドを使用して，実行系 1（待機系 2）の現用系仮想ホストを予備系仮想ホストに，待機系 1（実行系 2）の予備系仮想ホストを現用系仮想ホストに切り替えるときの流れについて示します。

図 18–4　相互系切り替えシステムでの計画系切り替えの処理の流れ



18.4　相互系切り替えシステムの設定（Windows の場合）

相互系切り替えシステムは，アプリケーションサーバの実行系と待機系を 1:1 にする構成の一つで，それぞれのアプリケーションサーバを実行系として稼働させながら，お互いの待機系にする構成（**相互スタンバイ構成**）のシステムです。それぞれのアプリケーションサーバに同じ種類の J2EE サーバを配置して，異なる J2EE サーバを起動しておくことで，それぞれのアプリケーションサーバが実行系として動作しながらお互いの待機系として機能します。どちらか一方の系に障害が発生すると，もう一方の系に切り替えられます。これによって，少ない台数のアプリケーションサーバマシンで，むだの少ない運用ができるようになります。このシステムは，Smart Composer 機能を使用して構築します。この節では，相互系切り替えシステムの設定について説明します。

なお，相互系切り替えシステムの詳細については，「18.2　相互系切り替えシステムの概要」を参照してください。システム構成については，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「3.11.4　アプリケーションサーバの実行系と待機系を相互スタンバイにする構成」を参照してください。

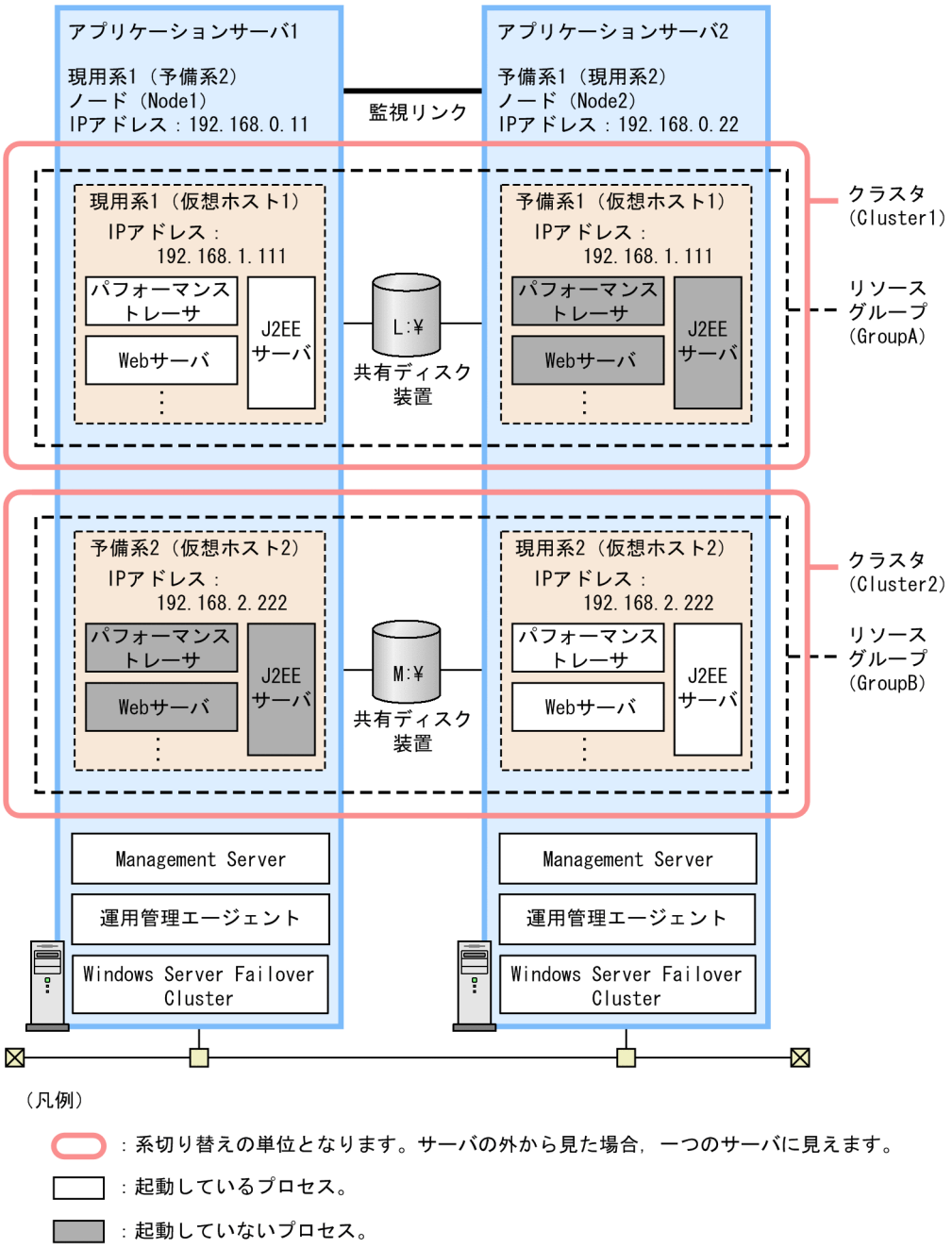
18.4.1　相互系切り替えシステムの設定手順

ここでは，Windows Server Failover Cluster と連携する場合の，システムの構成例とシステムの設定手順について説明します。

(1)　システムの構成例

相互系切り替えシステムの構成例を次の図に示します。なお，以降の項では，このシステムの構成例を使用したシステムの構築例を示します。

図 18-5　相互系切り替えシステムの構成例（Windows の場合）



この例では，2 種類のアプリケーションサーバ（アプリケーションサーバ 1，アプリケーションサーバ 2 とします）を使用しています。アプリケーションサーバ 1 の実行系（現用系 1）と待機系（予備系 2），アプリケーションサーバ 2 の実行系（現用系 2）と待機系（予備系 1）をそれぞれ 1:1 で配置しています。それぞれ別の運用管理ドメインを持つ Management Server を配置し，両方のマシンで Management Server を起動させています。

相互系切り替えシステムでは，一つの運用管理ドメイン内に二つの仮想ホストを定義し，それぞれを現用系アプリケーションサーバのホスト，予備系アプリケーションサーバのホストとして構築します。この例では，次のような組み合わせになります。

- Node1 の現用系 1（仮想ホスト 1）と Node2 の予備系 1（仮想ホスト 1）

- Node2 の現用系 2（仮想ホスト 2）と Node1 の予備系 2（仮想ホスト 2）

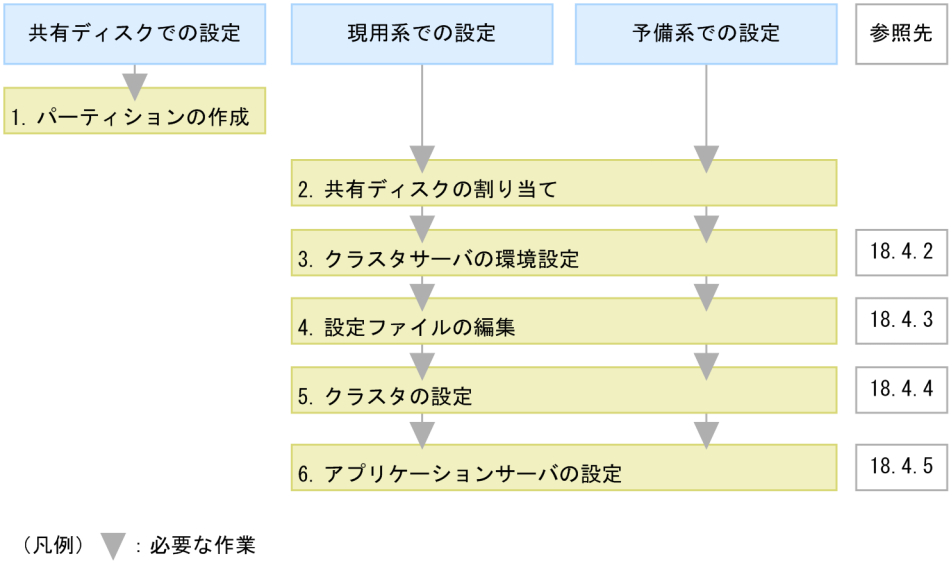
仮想ホストでは，一つの運用管理エージェントによってアプリケーションサーバの起動，停止を制御しますが，運用で使用する IP アドレスには異なる IP アドレスを割り当てて，見かけ上は異なるホストであるように定義しています。

アプリケーションサーバをクラスタ構成に配置するためには，エイリアス IP アドレスを設けて，稼働中のノードがエイリアス IP アドレスを引き継ぐことで，クライアントがクラスタ内のノードを意識しないようにします。相互系切り替えシステムの場合，エイリアス IP アドレスは，クラスタソフトウェアによって動的に割り当てられるアドレス（クラスタ IP アドレス）となります。アプリケーションサーバの運用に使用する IP アドレスには，クラスタ IP アドレスを使用し，Management Server から運用管理エージェントに対するリクエストの送信には，系切り替えによって他系に移動しない IP アドレス（ステーショナリ IP アドレス）を使用します。

(2) システムの設定手順

Windows Server Failover Cluster と連携する場合には，Management Server やクラスタアドミニストレータの設定などが必要になります。相互系切り替えシステムの設定手順を次の図に示します。

図 18-6 相互系切り替えシステムの設定手順（Windows の場合）



図中の 1.～6.について説明します。

1. 共有ディスクにパーティションを作成して，ファイルシステムを構築します。

   Windows Server Failover Cluster のクラスタ管理用ファイルの格納場所を作成します。また，グローバルトランザクションを使用する場合には，トランザクション情報の格納場所を作成します。なお，クラスタ管理用ファイルの格納場所とトランザクション情報の格納場所は，同じパーティションでもかまいません。

2. システムに共有ディスクを割り当てます。

システムに共有ディスクを割り当てる際は，現用系 1 と予備系 1，現用系 2 と予備系 2 で，同じドライブ文字を割り当ててください。

3. Management Server でクラスタサーバの環境を設定します。

Management Server の Smart Composer 機能の簡易構築定義ファイルで，Windows Server Failover Cluster を使用する場合の設定をします。詳細は，「18.4.2　クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

簡易構築定義ファイルでは設定できない，運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。詳細は，「18.4.3　設定ファイルの編集」を参照してください。

5. クラスタを設定します。

クラスタソフトウェアの環境設定や Management Sever と運用管理エージェントを監視するスクリプトファイルを作成します。詳細については，「18.4.4　クラスタの設定」を参照してください。

6. Management Server とクラスタアドミニストレータで，アプリケーションサーバの設定をします。

Management Server とクラスタアドミニストレータを使用して，アプリケーションサーバをクラスタ構成に配置し，J2EE アプリケーションやリソースアダプタを設定します。詳細は，「18.4.5　アプリケーションサーバの設定」を参照してください。

18.4.2　クラスタサーバの環境設定

ここでは，Windows Server Failover Cluster と連携する場合の，Management Server での設定時の留意点について説明します。

参考

Management Server を初めて使用するホストでは，Management Server をセットアップする必要があります。Management Server のセットアップについては，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「4.1.14　運用管理機能を構築する」を参照してください。

(1)　簡易構築定義ファイルでの設定

Management Server の Smart Composer 機能の簡易構築定義ファイルで，Web システムおよびホストの定義をします。論理サーバを設置するホストおよび各論理サーバの構成を定義するときの留意点を次に示します。

- Web システムの定義

Smart Composer 機能で用いる Web システムの定義をします。仮想ホストごとに，異なる Web システムを設定します。

- **ホストの定義**

  運用管理ドメイン内のホストを定義します。現用系および予備系の仮想ホストで，<host-name>タグにそれぞれのサーバの運用に使用するクラスタ IP アドレスを設定します。例では，次のように設定します。

  - Node1 の現用系 1 と Node2 の予備系 1 の場合：192.168.1.111
  - Node2 の現用系 2 と Node1 の予備系 2 の場合：192.168.2.222

  また，<agent-host>タグには，自ホストのステーショナリ IP アドレスを設定します。例では，次のように設定します。

  - Node1 の場合：192.168.0.11
  - Node2 の場合：192.168.0.22

また，論理 J2EE サーバ（j2ee-server），論理 Web サーバ（web-server）の<configuration>タグの設定での留意点を次に示します。

(a) 論理 J2EE サーバの設定

- **ホストの固定の設定**

  論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の次のパラメタに，任意の値を設定して，管理用ホストおよび運用用ホストを固定してください。

  - mngagent.connector.host パラメタ※（運用管理エージェントのホスト名または IP アドレス）
  - vbroker.se.iiop_tp.host パラメタ（J2EE サーバ単位で EJB コンテナのホスト名または IP アドレス）
  - webserver.connector.nio_http.bind_host パラメタ（NIO HTTP サーバで使用するホスト名または IP アドレス）

  注※　mngagent.connector.host パラメタとあわせて，mngagent.connector.port パラメタを指定する場合は，仮想ホストごとに異なるポート番号を指定してください。

- **グローバルトランザクションの設定**

  グローバルトランザクションを使用する場合は，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の次のパラメタに，次の設定をしてください。

  - ejbserver.distributedtx.XATransaction.enabled パラメタ（ライトトランザクション機能の有効または無効の設定）に「false」を設定します。デフォルトでは「false」が設定されています。
  - ejbserver.distributedtx.ots.status.directory1 パラメタ（インプロセス OTS のステータスファイルのバックアップの格納先）に，共有ディスク装置のディレクトリを設定します。デフォルトでは「otsstatus」が設定されています。

    設定例

    L:¥Group1¥otsstatus

(b) 論理 Web サーバの設定

論理 Web サーバ（web-server）の<configuration>タグ内に，次の設定が必要です。

- ホストの固定の設定

  Listen パラメタで，リクエストを受け付ける IP アドレスまたはホスト名を設定して，Web サーバのホストを固定してください。

18.4.3　設定ファイルの編集

設定ファイルを編集します。ここでは，Windows Server Failover Cluster と連携する場合に注意が必要な設定ファイルについて説明します。

(1)　運用管理エージェントの設定

adminagent.properties（運用管理エージェントプロパティファイル）に設定する項目のうち，Windows Server Failover Cluster と連携する場合に留意する設定項目について説明します。なお，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.adapter.bind_host キー

  adminagent.adapter.bind_host キーでステーショナリ IP アドレス（例では，192.168.0.11，または 192.168.0.22）を指定します。管理用のリクエストは，ステーショナリ IP アドレスだけで受け付けます。

(2)　Management Server の運用管理コマンド（mngsvrutil）の設定

<Application Server のインストールディレクトリ>¥manager¥config ディレクトリ下に，Management Server の運用管理コマンド（mngsvrutil）のクライアント側共通定義ファイル（mngsvrutilcl.properties）を用意し，Management Server の管理ユーザのユーザ ID とパスワードを設定してください。また，クライアント側共通定義ファイルには適切なアクセス権限を設定してください。

このファイルは，運用管理エージェントを監視するスクリプト，および論理サーバを起動，停止するスクリプトで，mngsvrutil コマンドを実行する際に使用します。

なお，mngsvrutil コマンドのクライアント側共通定義ファイル（mngsvrutilcl.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「8.2.16 mngsvrutilcl.properties（mngsvrutil コマンドのクライアント側共通定義ファイル)」を参照してください。

(3)　Management Server の設定

mserver.properties（Management Server 環境設定ファイル）に設定する項目のうち，Windows Server Failover Cluster と連携する場合に留意する設定項目について説明します。なお，mserver.properties 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「8.2.6 mserver.properties（Management Server 環境設定ファイル)」を参照してください。

- com.cosminexus.mngsvr.logical_server_abnormal_stop.exit
  論理サーバの稼働状況を示すステータスが異常停止状態（自動再起動回数を超えた状態，または自動再起動回数の設定が 0 の場合で障害を検知した状態）になったタイミングで系を切り替えるための設定をします。
  設定例を次に示します。
  com.cosminexus.mngsvr.logical_server_abnormal_stop.exit=true

  注意事項

  Management Server に自動再起動を設定している場合，このタイミングでは系切り替えが実行されません。このタイミングで系切り替えを実行する場合は，Management Server に自動再起動を設定しないでください。

また，次の項目を必要に応じて設定してください。

- webserver.connector.http.bind_host
  webserver.connector.http.bind_host キーに実 IP アドレスを指定します。
- mngsvr.myhost.name
  mngsvr.myhost.name キーに実 IP アドレスを指定します。
- com.cosminexus.mngsvr.management.host
  com.cosminexus.mngsvr.management.host キーに仮想 IP アドレスを指定します。

18.4.4　クラスタの設定

クラスタに対して，次の設定を実施します。

- スクリプトファイルの作成
- クラスタソフトウェアの環境設定

(1)　スクリプトファイルの作成

監視対象を起動・停止するためのスクリプトファイルを作成します。また，スクリプトファイルでクラスタの監視方法を決めておく必要があります。スクリプトファイルの作成については，使用する OS のマニュアルを参照してください。

ここでは，相互系切り替えシステムの監視対象および監視方法について説明します

(a)　監視対象

相互系切り替えシステムの監視対象を次に示します。

- Management Server

- 運用管理エージェント

Management Server が終了したタイミング，または運用管理エージェントが終了したタイミングで系切り替えが実行されます。

(b)　監視方法

監視対象のプロセスの有無を監視します。監視するプロセスを次に示します。

表 18-3　監視するプロセス

監視対象	プロセス名
Management Server	mngsvr.exe
運用管理エージェント	adminagent.exe

(2)　Windows Server Failover Cluster の環境設定

Windows Server Failover Cluster の環境設定については，使用するクラスタソフトウェアのマニュアルを参照してください。

18.4.5　アプリケーションサーバの設定

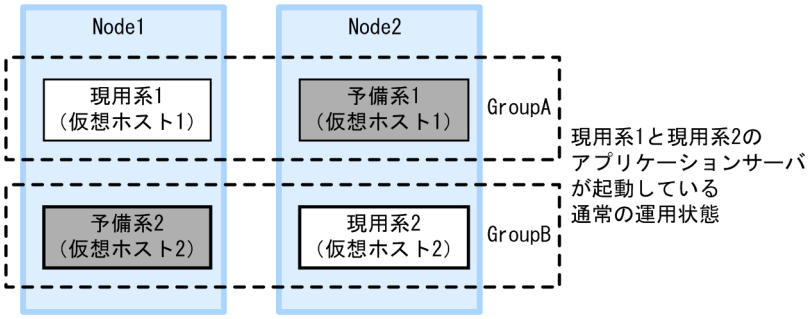
アプリケーションサーバの設定では，Management Server とクラスタアドミニストレータを使用してアプリケーションサーバをクラスタ構成に配置します。また，J2EE サーバに J2EE アプリケーションとリソースアダプタをインポート，および開始します。アプリケーションサーバの設定手順を次に示します。

1. Node1 および Node2 で，それぞれ運用管理エージェント，および Management Server を起動します。

   運用管理エージェント，および Management Server の起動方法については，「2.6　システムの起動と停止の設定」を参照してください。

2. クラスタサービスを開始して，リソースグループ GroupA と GroupB をそれぞれオンラインにします。

   現用系のアプリケーションサーバ（Node1 の現用系 1 と Node2 の現用系 2）が起動されて，予備系のアプリケーションサーバ（Node2 の予備系 1 と Node1 の予備系 2）が待機状態になっている，通常の運用状態になります。



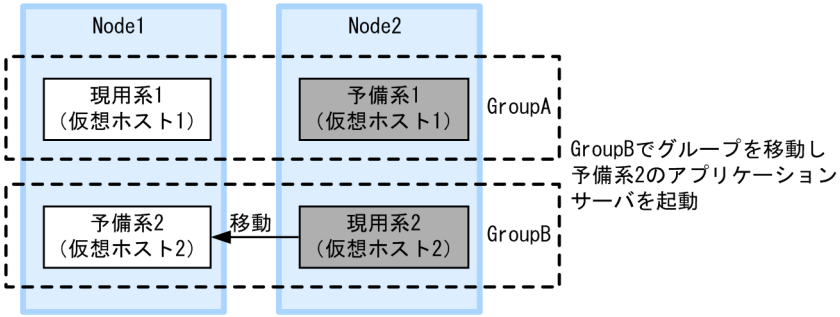
3. 現用系のアプリケーションサーバ（Node1 の現用系 1 と Node2 の現用系 2）に対して，Management Server の Smart Composer 機能のコマンドで，設定済みの簡易構築定義ファイルを基に，Web システムをセットアップして起動します。また，サーバ管理コマンドを使用して，J2EE アプリケーションとリソースアダプタをインポートして，インポートした J2EE アプリケーションとリソースアダプタを開始します。

   J2EE 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 システム構築・運用ガイド」の「4.1.29　業務アプリケーションを設定して開始する（CUI 利用時）」，リソースアダプタの設定については，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」の次の個所を参照してください。

   - 4.1.26　DB Connector を設定する（CUI 利用時）
   - 4.1.27　DB Connector 以外のリソースアダプタを設定する（CUI 利用時）
   - 4.1.28　リソースアダプタを開始する（CUI 利用時）

   また，サーバ管理コマンドでの操作については，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。

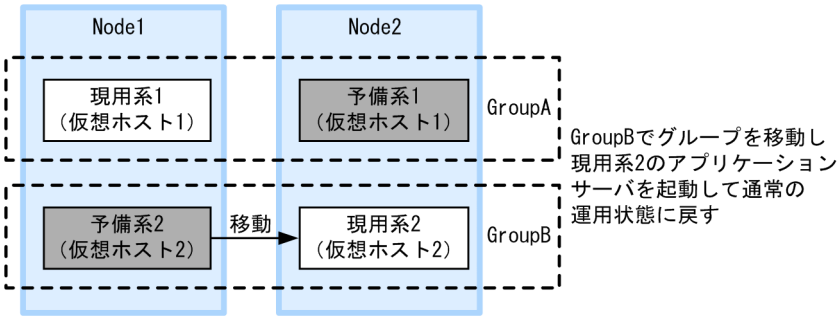
4. クラスタアドミニストレータで，リソースグループ GroupB に対して「グループの移動」を実行して系を切り替え，Node1 の予備系 2 のアプリケーションサーバを起動します。



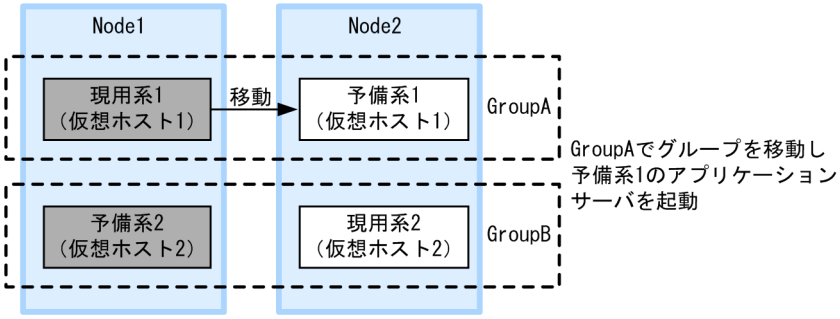
5. Node1 の予備系 2 のアプリケーションサーバに対して，Management Server の Smart Composer 機能のコマンドで，設定済みの簡易構築定義ファイルを基に，Web システムをセットアップして起動します。また，サーバ管理コマンドを使用して，J2EE アプリケーションとリソースアダプタをインポートして，インポートした J2EE アプリケーションとリソースアダプタを開始します。

   Node1 の予備系 2 には，Node2 の現用系 2 と同じ J2EE アプリケーションとリソースアダプタをインポートします。

6. クラスタアドミニストレータで，リソースグループ GroupB に対して「グループの移動」を実行して系を切り替え，Node2 の現用系 2 のアプリケーションサーバを起動し，通常の運用状態に戻します。



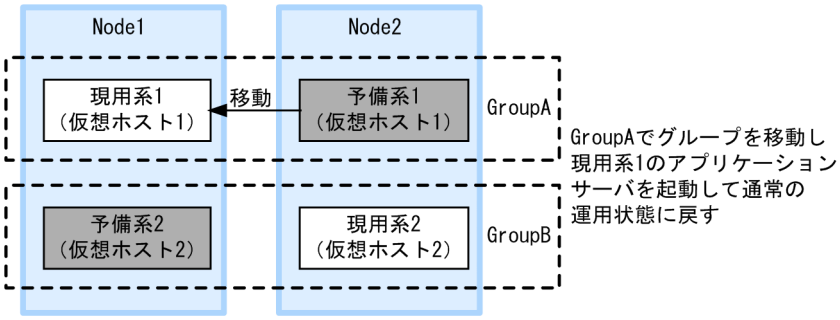
7. リソースグループ GroupA に対して「グループの移動」を実行して系を切り替え，Node2 の予備系 1 のアプリケーションサーバを起動します。



8. Node2 の予備系 1 のアプリケーションサーバに対して，Management Server の Smart Composer 機能のコマンドで，設定済みの簡易構築定義ファイルを基に，Web システムをセットアップして起動します。また，サーバ管理コマンドを使用して，J2EE アプリケーションとリソースアダプタをインポートして，インポートした J2EE アプリケーションとリソースアダプタを開始します。

   Node2 の予備系 1 には，Node1 の現用系 1 と同じ J2EE アプリケーションとリソースアダプタをインポートします。

9. リソースグループ GroupA に対して「グループの移動」を実行して系を切り替え，Node1 の現用系 1 のアプリケーションサーバを起動し，通常の運用状態に戻します。



18.5　相互系切り替えシステムの設定（UNIX の場合）

相互系切り替えシステムは，アプリケーションサーバの実行系と待機系を 1:1 にする構成の一つで，それぞれのアプリケーションサーバを実行系として稼働させながら，お互いの待機系にする構成（**相互スタンバイ構成**）のシステムです。それぞれのアプリケーションサーバに同じ種類の J2EE サーバを配置して，異なる J2EE サーバを起動しておくことで，それぞれのアプリケーションサーバが実行系として動作しながらお互いの待機系として機能します。どちらか一方の系に障害が発生すると，もう一方の系に切り替えられます。これによって，少ない台数のアプリケーションサーバマシンで，むだの少ない運用ができるようになります。このシステムは，Smart Composer 機能を使用して構築します。この節では，相互系切り替えシステムの設定について説明します。

なお，相互系切り替えシステムの運用は，AIX または Linux の場合だけ使用できます。

機能の詳細については，「18.2　相互系切り替えシステムの概要」を参照してください。システム構成については，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「3.11.4　アプリケーションサーバの実行系と待機系を相互スタンバイにする構成」を参照してください。また，HA モニタについては，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

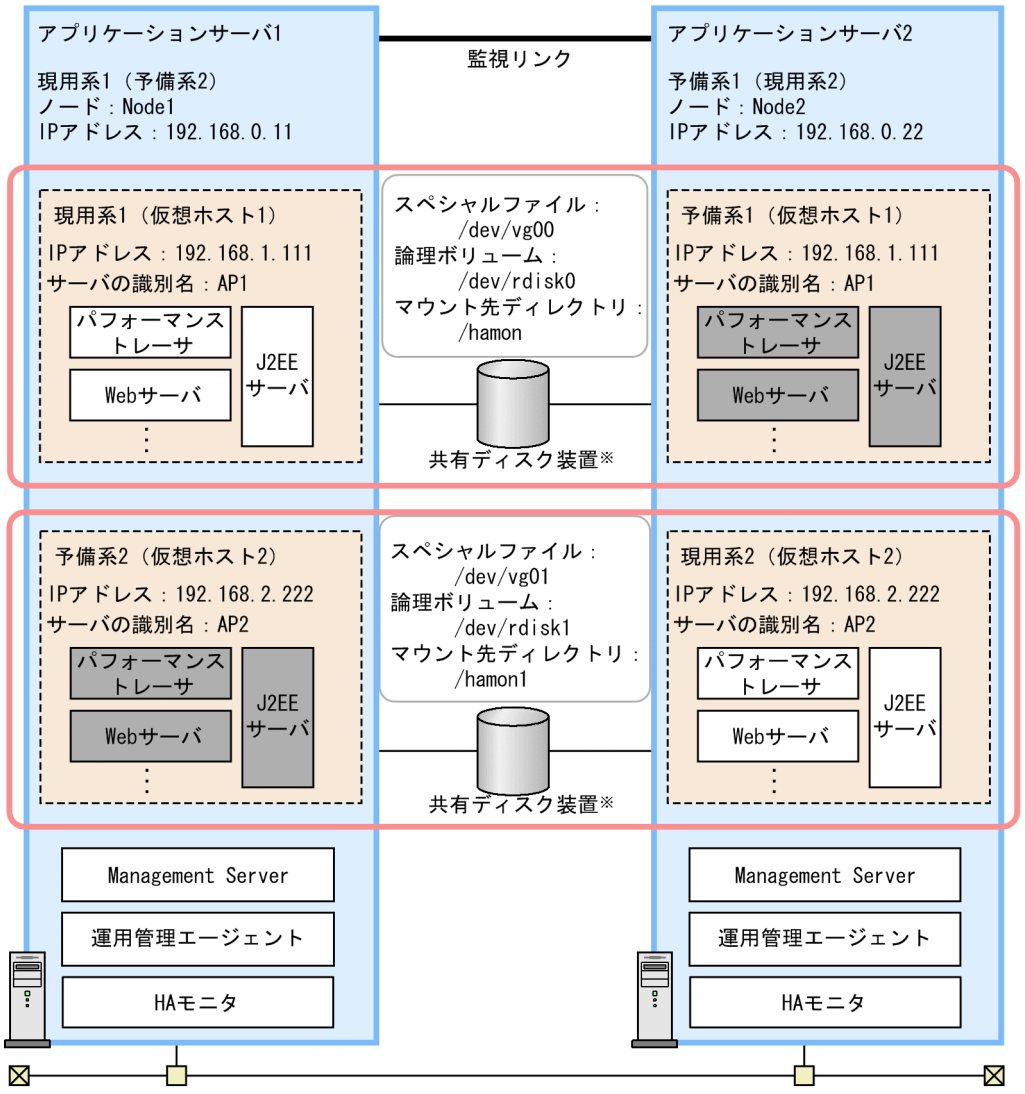
18.5.1　相互系切り替えシステムの設定手順

ここでは，システムの構成例とシステムの設定手順について説明します。

(1)　システムの構成例

相互系切り替えシステムの構成例を次の図に示します。なお，以降の項では，このシステムの構成例を使用したシステムの設定例を示します。

図 18-7　相互系切り替えシステムの構成例（UNIX の場合）



（凡例）

：系切り替えの単位となります。

：起動しているプロセス。

：起動していないプロセス。

注※　グローバルトランザクションを使用するときに必要となります。ローカルトランザクションを使用するときは不要となります。

この例では，2 種類のアプリケーションサーバ（アプリケーションサーバ 1，アプリケーションサーバ 2 とします）を使用しています。アプリケーションサーバ 1 の実行系（現用系 1）と待機系（予備系 2），アプリケーションサーバ 2 の実行系（現用系 2）と待機系（予備系 1）をそれぞれ 1:1 で配置しています。それぞれ別の運用管理ドメインを持つ Management Server を配置し，両方のマシンで Management Server を起動させています。

相互系切り替えシステムでは，一つの運用管理ドメイン内に二つの仮想ホストを定義し，それぞれを現用系アプリケーションサーバのホスト，予備系アプリケーションサーバのホストとして構築します。この例では，次のような組み合わせになります。

- Node1 の現用系 1（仮想ホスト 1）と Node2 の予備系 1（仮想ホスト 1）
- Node2 の現用系 2（仮想ホスト 2）と Node1 の予備系 2（仮想ホスト 2）

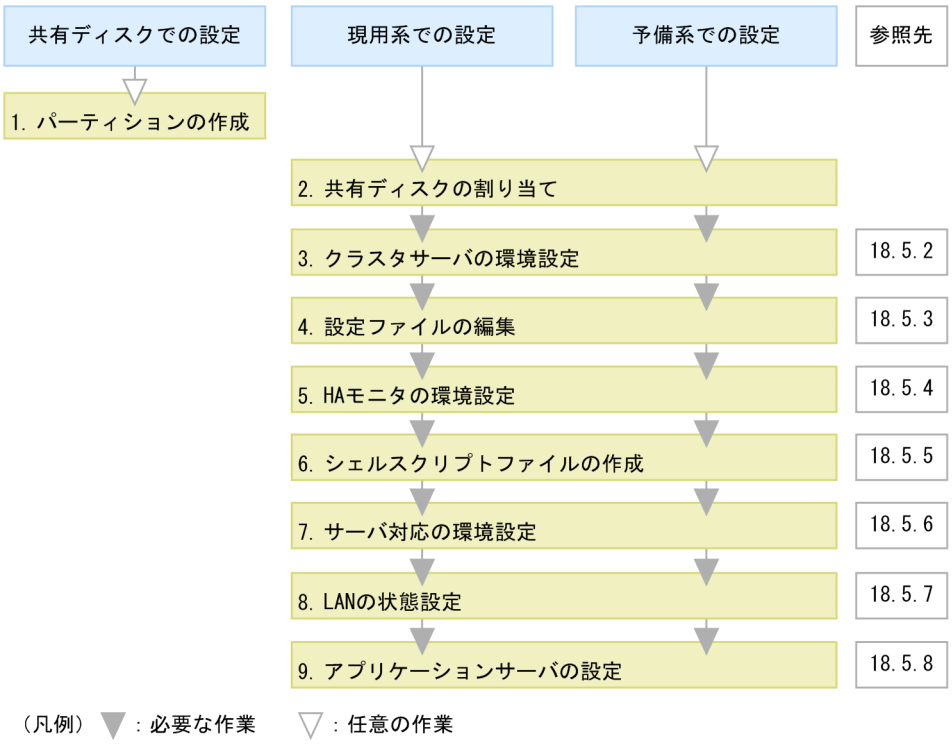
仮想ホストでは，一つの運用管理エージェントによってアプリケーションサーバの起動，停止を制御しますが，運用で使用する IP アドレスには異なる IP アドレスを割り当てて，見かけ上は異なるホストであるように定義しています。

アプリケーションサーバをクラスタ構成に配置するためには，エイリアス IP アドレスを設けて，稼働中のノードがエイリアス IP アドレスを引き継ぐことで，クライアントがクラスタ内のノードを意識しないようにします。相互系切り替えシステムの場合，エイリアス IP アドレスは，HA モニタによって動的に割り当てられるアドレスとなります。アプリケーションサーバの運用に使用する IP アドレスには，エイリアス IP アドレスを使用し，Management Server から運用管理エージェントに対するリクエストの送信には，系切り替えによって他系に移動しない IP アドレス（ステーショナリ IP アドレス）を使用します。

(2)　システムの設定手順

HA モニタと連携する場合には，Management Server や HA モニタのファイルの設定などが必要になります。相互系切り替えシステムの設定手順を次の図に示します。

図 18-8　相互系切り替えシステムの設定手順（UNIX の場合）



図中の 1.～9.について説明します。

1. グローバルトランザクションを使用する場合は，共有ディスクにパーティションを作成して，ファイルシステムを構築します。

   グローバルトランザクションを使用するためには，トランザクション情報の格納場所を作成します。

2. グローバルトランザクションを使用する場合は，システムに共有ディスクを割り当てます。

システムに共有ディスクを割り当てる際は，現用系 1 と予備系 1，現用系 2 と予備系 2 で，同じマウント先ディレクトリにしてください。

3. Management Server でクラスタサーバの環境を設定します。

Management Server の Smart Composer 機能の簡易構築定義ファイルで，HA モニタを使用する場合の設定をします。詳細は，「18.5.2　クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

簡易構築定義ファイルでは設定できない，運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。詳細は，「18.5.3　設定ファイルの編集」を参照してください。

5. HA モニタの環境を設定します。

HA モニタの sysdef ファイルで，HA モニタの環境を定義します。詳細は，「18.5.4　HA モニタの環境設定」を参照してください。

6. シェルスクリプトファイルを作成します。

運用管理エージェントを監視，論理サーバを起動および停止するためのシェルスクリプトファイルを作成します。詳細は，「18.5.5　シェルスクリプトファイルの作成」を参照してください。

7. サーバ対応の環境を設定します。

HA モニタの servers ファイルで，系で稼働させる現用系サーバや予備系サーバの環境を定義します。詳細は，「18.5.6　サーバ対応の環境設定」を参照してください。

8. LAN の状態を設定します。

HA モニタの LAN の状態設定ファイルで，LAN アダプタの IP アドレスなどを指定して HA モニタでの LAN の切り替えについて定義します。詳細は，「18.5.7　LAN の状態設定」を参照してください。

9. Management Server と HA モニタのコマンドで，アプリケーションサーバの設定をします。

Management Server と HA モニタのコマンドを使用して，アプリケーションサーバをクラスタ構成に配置し，J2EE アプリケーションやリソースアダプタを設定します。詳細は，「18.5.8　アプリケーションサーバの設定」を参照してください。

18.5.2　クラスタサーバの環境設定

ここでは，HA モニタと連携する場合の，Management Server での設定時の留意点について説明します。

参考

Management Server を初めて使用するホストでは，Management Server をセットアップする必要があります。Management Server のセットアップについては，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「4.1.14　運用管理機能を構築する」を参照してください。

(1)　簡易構築定義ファイルでの設定

Management Server の Smart Composer 機能の簡易構築定義ファイルで，Web システムおよびホストの定義をします。論理サーバを設置するホストおよび各論理サーバの構成を定義するときの留意点を次に示します。

- **Web システムの定義**

  Smart Composer 機能で用いる Web システムの定義をします。仮想ホストごとに，異なる Web システムを設定します。

- **ホストの定義**

  運用管理ドメイン内のホストを定義します。現用系および予備系の仮想ホストで，<host-name>タグにそれぞれのサーバの運用に使用するエイリアス IP アドレスを設定します。例では，次のように設定します。

  - Node1 の現用系 1 と Node2 の予備系 1 の場合：192.168.1.111
  - Node2 の現用系 2 と Node1 の予備系 2 の場合：192.168.2.222

  また，<agent-host>タグには，自ホストのステーショナリ IP アドレスを設定します。例では，次のように設定します。

  - Node1 の場合：192.168.0.11
  - Node2 の場合：192.168.0.22

また，論理 J2EE サーバ（j2ee-server），論理 Web サーバ（web-server）の<configuration>タグの設定での留意点を次に示します。

(a)　論理 J2EE サーバの設定

- **ホストの固定の設定**

  論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の次のパラメタに，任意の値を設定して，管理用ホストおよび運用用ホストを固定してください。

  - mngagent.connector.host パラメタ※（運用管理エージェントのホスト名または IP アドレス）
  - vbroker.se.iiop_tp.host パラメタ（J2EE サーバ単位で EJB コンテナのホスト名または IP アドレス）
  - webserver.connector.nio_http.bind_host パラメタ（NIO HTTP サーバで使用するホスト名または IP アドレス）

注※　mngagent.connector.host パラメタとあわせて，mngagent.connector.port パラメタを指定する場合は，仮想ホストごとに異なるポート番号を指定してください。

- グローバルトランザクションの設定

  グローバルトランザクションを使用する場合は，論理 J2EE サーバ（j2ee-server）の<configuration>タグ内の次のパラメタに，次の設定をしてください。

  - ejbserver.distributedtx.XATransaction.enabled パラメタ（ライトトランザクション機能の有効または無効の設定）に「true」を設定します。デフォルトでは「false」が設定されています。
  - ejbserver.distributedtx.ots.status.directory1 パラメタ（インプロセス OTS のステータスファイルのバックアップの格納先）に，共有ディスク装置のディレクトリを設定します。デフォルトでは「otsstatus」が設定されています。

    設定例

    /hamon

(b)　論理 Web サーバの設定

論理 Web サーバ（web-server）の<configuration>タグ内に，次の設定が必要です。

- ホストの固定の設定

  Listen パラメタで，リクエストを受け付ける IP アドレスまたはホスト名を設定して，Web サーバのホストを固定してください。

18.5.3　設定ファイルの編集

設定ファイルを編集します。ここでは，HA モニタと連携する場合に注意が必要な設定ファイルについて説明します。

(1)　運用管理エージェントの設定

adminagent.properties（運用管理エージェントプロパティファイル）に設定する項目のうち，HA モニタと連携する場合に留意する設定項目について説明します。なお，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.adapter.bind_host キー

  adminagent.adapter.bind_host キーでステーショナリ IP アドレス（例では，192.168.0.11，または 192.168.0.22）を指定します。管理用のリクエストは，ステーショナリ IP アドレスだけで受け付けます。

(2) Management Server の運用管理コマンド（mngsvrutil）の設定

/opt/Cosminexus/manager/config ディレクトリ下に，Management Server の運用管理コマンド（mngsvrutil）のクライアント側共通定義ファイル（mngsvrutilcl.properties）を用意し，Management Server の管理ユーザのユーザ ID とパスワードを設定してください。また，クライアント側共通定義ファイルには適切なアクセス権限を設定してください。

このファイルは，運用管理エージェントのプロセスを監視するスクリプト，論理サーバを起動および停止するスクリプトで，mngsvrutil コマンドを実行する際に使用します。

なお，mngsvrutil コマンドのクライアント側共通定義ファイル（mngsvrutilcl.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「8.2.16 mngsvrutilcl.properties（mngsvrutil コマンドのクライアント側共通定義ファイル)」を参照してください。

(3) Management Server の設定

mserver.properties（Management Server 環境設定ファイル）に設定する項目のうち，HA モニタと連携する場合に留意する設定項目について説明します。なお，mserver.properties 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「8.2.6　mserver.properties（Management Server 環境設定ファイル)」を参照してください。

- com.cosminexus.mngsvr.logical_server_abnormal_stop.exit

  論理サーバの稼働状況を示すステータスが異常停止状態（自動再起動回数を超えた状態，または自動再起動回数の設定が 0 の場合で障害を検知した状態）になったタイミングで系を切り替えるための設定をします。

  設定例を次に示します。

  com.cosminexus.mngsvr.logical_server_abnormal_stop.exit=true

  注意事項

  Management Server に自動再起動を設定している場合，このタイミングでは系切り替えが実行されません。このタイミングで系切り替えを実行する場合は，Management Server に自動再起動を設定しないでください。

また，次の項目を必要に応じて設定してください。

- webserver.connector.http.bind_host

  webserver.connector.http.bind_host キーに実 IP アドレスを指定します。

- mngsvr.myhost.name

  mngsvr.myhost.name キーに実 IP アドレスを指定します。

- com.cosminexus.mngsvr.management.host

  com.cosminexus.mngsvr.management.host キーに仮想 IP アドレスを指定します。

18.5.4　HA モニタの環境設定

使用しているシステムに従って，sysdef という定義ファイルに HA モニタの環境を定義してください。

HA モニタの環境設定については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

18.5.5　シェルスクリプトファイルの作成

運用管理エージェントのプロセスを監視し，論理サーバを起動および停止するために，次のシェルスクリプトファイルを作成します。

- 運用管理エージェントのプロセスを監視するシェルスクリプトファイル
- 論理サーバを起動するシェルスクリプトファイル
- 論理サーバを停止するシェルスクリプトファイル

なお，論理サーバの稼働状況を示すステータスが異常停止状態（自動再起動回数を超えた状態，または自動再起動回数の設定が 0 の場合で障害を検知した状態）になったタイミングで系を切り替えるための設定のとき，シェルスクリプトファイルに Management Server の監視処理を追加してください。

(1)　運用管理エージェントのプロセスを監視するシェルスクリプトファイル

運用管理エージェントのプロセスを監視するシェルスクリプトファイルの例（manager_adminagent_monitor.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
AA=/opt/Cosminexus/manager/bin/adminagent
MNGDIR=/opt/Cosminexus/manager

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

logg "### $0: started. ###"
while true
do
    $MNGDIR/bin/mngsvrutil -m 192.168.0.11:28080 -t 192.168.1.111 check adminAgent
    if [ $? -ne 0 ]
    then
        logg "### $0: stop. ###"
        exit 0
    fi
    sleep 10
done
```

このシェルスクリプトファイルでは，mngsvrutil の check コマンドで運用管理エージェントの稼働状況を確認します。mngsvrutil コマンドの-m オプションの引数のホスト名には自ホストのステーショナリ IP アドレスを指定し，-t オプションの引数には現用系の仮想ホストのエイリアス IP アドレスを指定します。

なお，このシェルスクリプトファイルは，Node1 の現用系 1（予備系 2）で稼働する運用管理エージェントを監視する場合の例です。Node2 の予備系 1（現用系 2）で使用する場合には，-m オプションの引数のホスト名を Node2 のステーショナリ IP アドレス（192.168.0.22）に，-t オプションの引数を現用系の仮想ホストのエイリアス IP アドレス（192.168.2.222）に置き換えます。

(2) 論理サーバを起動するシェルスクリプトファイル

論理サーバを起動するシェルスクリプトファイルの例（manager_apserver1_start.sh）を次に示します。

```
#!/bin/sh

MNGDIR=/opt/Cosminexus/manager
logg()
{
    echo `date '+[%Y/%m/%d %H:%M:%S]'`"[$$]: $1" ¥
        >> ${LOGDIR}/adminagent.log 2>&1
}
# start logical server
logg "### $0: starting logical servers. ###"
$MNGDIR/bin/mngsvrutil -m 192.168.0.11:28080 -t <パフォーマンストレーサ> -s start server
$MNGDIR/bin/mngsvrutil -m 192.168.0.11:28080 -t <J2EEサーバ1> -s start server
$MNGDIR/bin/mngsvrutil -m 192.168.0.11:28080 -t <Webサーバ1> -s start server
exit 0
```

このシェルスクリプトファイルでは，論理サーバを起動します。mngsvrutil コマンドの-m オプションの引数のホスト名には自ホストのステーショナリ IP アドレスを指定し，-t オプションの引数には起動する論理サーバ名を指定します。

なお，このシェルスクリプトファイルは，Node1 の現用系 1 で論理サーバを起動する場合の例です。

(3) 論理サーバを停止するシェルスクリプトファイル

論理サーバを停止するシェルスクリプトファイルの例（manager_apserver1_stop.sh）を次に示します。

```
#!/bin/sh

MNGDIR=/opt/Cosminexus/manager
logg()
{
    echo `date '+[%Y/%m/%d %H:%M:%S]'`"[$$]: $1" ¥
        >> ${LOGDIR}/adminagent.log 2>&1
}
# stop logical server
logg "### $0: stop logical servers. ###"
$MNGDIR/bin/mngsvrutil -m 192.168.0.11:28080 -t <Webサーバ1> -s stop server graceful:300
```

```
$MNGDIR/bin/mngsvrutil -m 192.168.0.11:28080 -t 〈J2EEサーバ1〉 -s stop server
$MNGDIR/bin/mngsvrutil -m 192.168.0.11:28080 -t 〈パフォーマンストレーサ〉 -s stop server
```

このシェルスクリプトファイルでは，論理サーバを停止します。mngsvrutil コマンドの-m オプションの引数のホスト名には自ホストのステーショナリ IP アドレスを指定し，-t オプションの引数には停止する論理サーバ名を指定します。

なお，このシェルスクリプトファイルは，Node1 の現用系 1 で論理サーバを停止する場合の例です。

18.5.6　サーバ対応の環境設定

HA モニタでのサーバ対応の環境設定では，系で稼働させる実行サーバや待機サーバの環境を定義します。

servers という定義ファイルに，相互系切り替えシステム用のサーバ対応の環境を定義してください。サーバ対応の環境設定での設定内容を次の表に示します。

表 18-4　サーバ対応の環境設定での設定内容（相互系切り替えシステムの場合）

オペランド	設定内容
name	論理サーバを起動するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_apserver1_start.sh
alias	サーバの識別名を指定します。対応する現用系と予備系で同じ名称を指定します。 指定例：AP1（現用系 1 と予備系 1 の場合），AP2（現用系 2 と予備系 2 の場合）
acttype	サーバの起動方法を指定します。ここでは，HA モニタのコマンドでサーバを起動するため，「monitor」を指定します。
termcommand	論理サーバを停止するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_apserver1_stop.sh
initial	サーバ起動時の状態を指定します。 • 現用系の場合 「online」を指定します。 • 予備系の場合 「standby」を指定します。
disk	共有ディスク装置のキャラクタ型スペシャルファイル名を指定します。 指定例：/dev/vg00（現用系 1 と予備系 1 の場合），/dev/vg01（現用系 2 と予備系 2 の場合）
lan_updown	LAN の状態設定ファイルを使用するかどうかを指定します。ここでは，LAN の状態設定ファイルを使用するため，「use」を指定します。
fs_name	切り替えるファイルシステムに対応する論理ボリュームの絶対パス名を指定します。なお，この設定は，$TPFS を UNIX ファイルで使用する場合だけ必要です。 指定例：/dev/rdisk0

オペランド	設定内容
fs_mount_dir	切り替えるファイルシステムのマウント先ディレクトリの絶対パス名を指定します。なお，この設定は，$TPFS を UNIX ファイルで使用する場合だけ必要です。 指定例：/hamon
patrolcommand	運用管理エージェントのプロセスを監視するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/manager_adminagent_monitor.sh
servexec_retry	障害を検出した場合の再起動の回数を指定します。ここでは，障害が発生した場合に再起動しないで系を切り替えるため，「0」を指定します。
waitserv_exec	論理サーバの起動完了処理を実行するときに起動コマンドの実行完了を待つかどうかを指定します。ここでは，実行完了を待つため，「yes」を指定します。

servers ファイルの例を次に示します。

servers ファイルの例（Node1 の現用系 1（予備系 2）の場合）

Node1 の現用系 1（予備系 2）の場合の servers ファイルの例を次に示します。

```
server  name           /home/manager/hamon/bin/manager_apserver1_start.sh,
        alias          AP1,
        acttype        monitor,
        termcommand    /home/manager/hamon/bin/manager_apserver1_stop.sh,
        initial        online,
        disk           /dev/vg00,
        lan_updown     use,
        fs_name        /dev/rdisk0,
        fs_mount_dir   /hamon,
        patrolcommand  /home/manager/hamon/bin/manager_adminagent_monitor.sh,
        servexec_retry 0,
        waitserv_exec  yes;

server  name           /home/manager/hamon/bin/manager_apserver2_start.sh,
        alias          AP2,
        acttype        monitor,
        termcommand    /home/manager/hamon/bin/manager_apserver2_stop.sh,
        initial        standby,
        disk           /dev/vg01,
        lan_updown     use,
        fs_name        /dev/rdisk1,
        fs_mount_dir   /hamon1,
        patrolcommand  /home/manager/hamon/bin/manager_adminagent_monitor.sh,
        servexec_retry 0,
        waitserv_exec  yes;
```

servers ファイルの例（Node2 の予備系 1（現用系 2）の場合）

Node2 の予備系 1（現用系 2）の場合の servers ファイルの例を次に示します。

```
server  name           /home/manager/hamon/bin/manager_apserver1_start.sh,
        alias          AP1,
        acttype        monitor,
        termcommand    /home/manager/hamon/bin/manager_apserver1_stop.sh,
        initial        standby,
        disk           /dev/vg00,
```

```
        lan_updown     use,
        fs_name        /dev/rdisk0,
        fs_mount_dir   /hamon,
        patrolcommand  /home/manager/hamon/bin/manager_adminagent_monitor.sh,
        servexec_retry 0,
        waitserv_exec  yes;

server  name           /home/manager/hamon/bin/manager_apserver2_start.sh,
        alias          AP2,
        acttype        monitor,
        termcommand    /home/manager/hamon/bin/manager_apserver2_stop.sh,
        initial        online,
        disk           /dev/vg01,
        lan_updown     use,
        fs_name        /dev/rdisk1,
        fs_mount_dir   /hamon1,
        patrolcommand  /home/manager/hamon/bin/manager_adminagent_monitor.sh,
        servexec_retry 0,
        waitserv_exec  yes;
```

この設定例では，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格納先に，共有ディスク上のディレクトリを指定しています。なお，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格納先は，Management Server の Smart Composer 機能の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，ejbserver.distributedtx.ots.status.directory1 パラメタで指定します。

18.5.7　LAN の状態設定

HA モニタの LAN の状態設定ファイルで，LAN アダプタの IP アドレスなどを指定して HA モニタでの LAN の切り替えについて定義します。

次のファイルにエイリアス IP アドレスを設定してください。

- <サーバ識別名>.up ファイル
  LAN を接続する場合に使用します。LAN アダプタに追加するエイリアス IP アドレスを指定します。
- <サーバ識別名>.down ファイル
  LAN を切り離す場合に使用します。LAN アダプタから削除するエイリアス IP アドレスを指定します。

<サーバ識別名>には，サーバ対応の環境設定（servers ファイル）の alias の値を指定してください。

LAN の状態設定の詳細については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

18.5.8　アプリケーションサーバの設定

アプリケーションサーバの設定では，Management Server と HA モニタのコマンドを使用してアプリケーションサーバをクラスタ構成に配置します。また，論理サーバをセットアップして設定情報を配布し，J2EE サーバに J2EE アプリケーションとリソースアダプタをインポート，および開始します。アプリケーションサーバの設定手順を次に示します。

1. Node1 および Node2 で，それぞれ運用管理エージェント，および Management Server を起動します。

   運用管理エージェント，および Management Server の起動方法については，「2.6　システムの起動と停止の設定」を参照してください。

2. Node1 および Node2 で HA モニタの monbegin コマンドを実行して，現用系のアプリケーションサーバを実行系として起動し，予備系のアプリケーションサーバを待機状態にします。

   - Node1 でのコマンドの実行例

     「# monbegin AP1」を実行して，Node1 の現用系 1 を起動します。

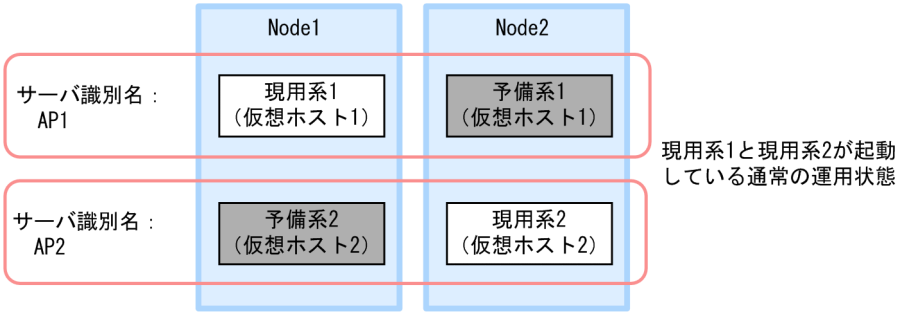
     「# monbegin AP2」を実行して，Node1 の予備系 2 を待機状態にします。

   - Node2 でのコマンドの実行例

     「# monbegin AP1」を実行して，Node2 の予備系 1 を待機状態にします。

     「# monbegin AP2」を実行して，Node2 の現用系 2 を起動します。

   現用系のアプリケーションサーバ（Node1 の現用系 1 と Node2 の現用系 2）が起動されて，予備系のアプリケーションサーバ（Node2 の予備系 1 と Node1 の予備系 2）が待機状態になっている，通常の運用状態になります。



3. 現用系のアプリケーションサーバ（Node1 の現用系 1 と Node2 の現用系 2）に対して，Management Server の Smart Composer 機能のコマンドで，設定済みの簡易構築定義ファイルを基に，Web システムをセットアップして，起動します。また，サーバ管理コマンドを使用して，J2EE アプリケーションとリソースアダプタをインポートして，インポートした J2EE アプリケーションとリソースアダプタを開始します。

   J2EE 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 システム構築・運用ガイド」の「4.1.29　業務アプリケーションを設定して開始する（CUI 利用時）」，リソースアダプタの設定については，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」の次の個所を参照してください。

   - 4.1.26　DB Connector を設定する（CUI 利用時）

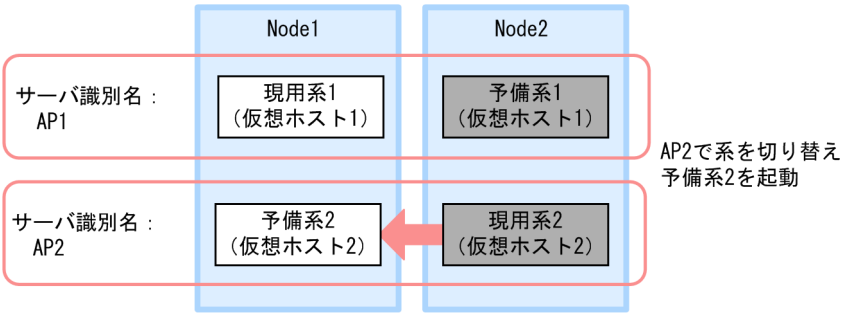
- 4.1.27　DB Connector 以外のリソースアダプタを設定する（CUI 利用時）
- 4.1.28　リソースアダプタを開始する（CUI 利用時）

また，サーバ管理コマンドでの操作については，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。

4. Node2 で HA モニタの monswap コマンドを実行して，系を切り替えます。

- Node2 でのコマンドの実行例

「# monswap AP2」を実行して，サーバ識別名「AP2」で系を切り替えます。Node1 の予備系 2 が起動され，Node2 の現用系 2 が待機状態になります。



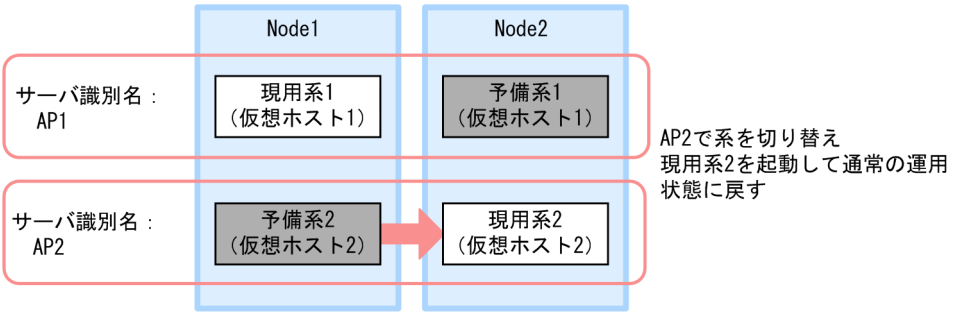
5. Node1 の予備系 2 のアプリケーションサーバに対して，Management Server の Smart Composer 機能のコマンドで，設定済みの簡易構築定義ファイルを基に，Web システムをセットアップして起動します。また，サーバ管理コマンドを使用して，J2EE アプリケーションとリソースアダプタをインポートして，インポートした J2EE アプリケーションとリソースアダプタを開始します。

Node1 の予備系 2 には，Node2 の現用系 2 と同じ J2EE アプリケーションとリソースアダプタをインポートします。

6. Node1 で HA モニタの monswap コマンドを実行して，系を切り替えて通常の運用状態に戻します。

- Node1 でのコマンドの実行例

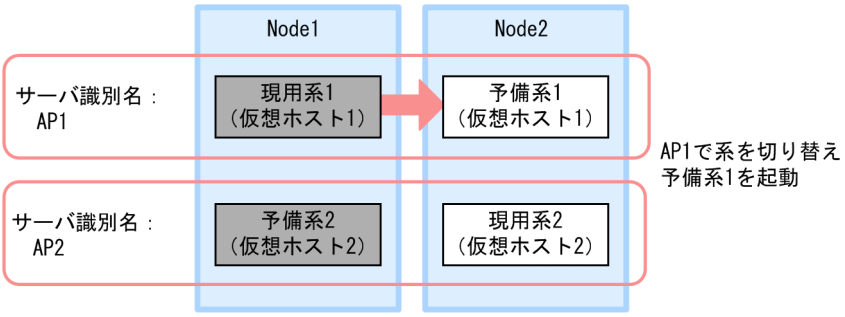
「# monswap AP2」を実行して，サーバ識別名「AP2」で系を切り替えます。Node2 の現用系 2 が起動され，Node1 の予備系 2 が待機状態になります。



7. Node1 で HA モニタの monswap コマンドを実行して，系を切り替えます。

- Node1 でのコマンドの実行例

「# monswap AP1」を実行して，サーバ識別名「AP1」で系を切り替えます。Node2 の予備 1 が起動され，Node1 の現用系 1 が待機状態になります。



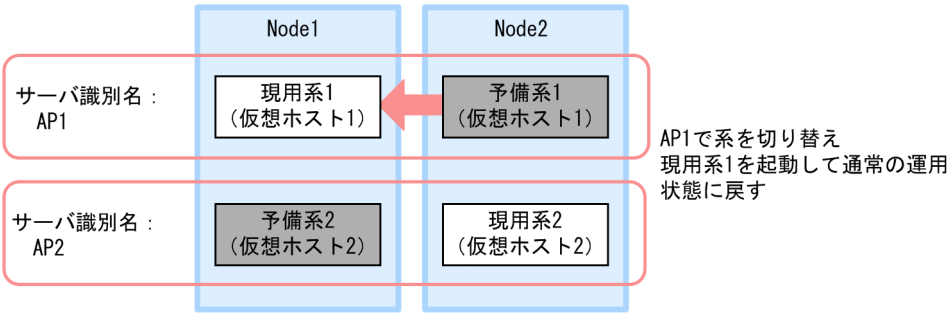
8. Node2 の予備系 1 のアプリケーションサーバに対して，Management Server の Smart Composer 機能のコマンドで，設定済みの簡易構築定義ファイルを基に，Web システムをセットアップして起動します。また，サーバ管理コマンドを使用して，J2EE アプリケーションとリソースアダプタをインポートして，インポートした J2EE アプリケーションとリソースアダプタを開始します。

   Node2 の予備系 1 には，Node1 の現用系 1 と同じ J2EE アプリケーションとリソースアダプタをインポートします。

9. Node2 で HA モニタの monswap コマンドを実行して，系を切り替えて通常の運用状態に戻します。

   - Node2 でのコマンドの実行例

     「# monswap AP1」を実行して，サーバ識別名「AP1」で系を切り替えます。Node1 の現用系 1 が起動され，Node2 の予備 1 が待機状態になります。



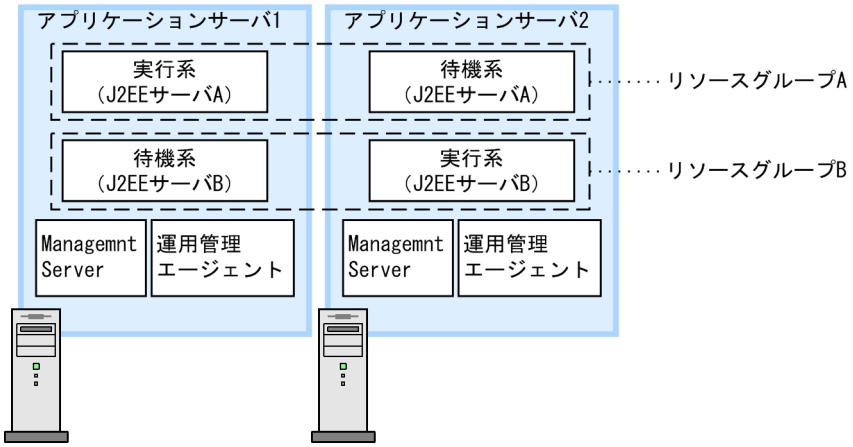
18.6　相互系切り替えシステムの起動と停止（Windows の場合）

この節では，相互系切り替えシステムを利用する場合の，システムの起動と停止方法について説明します。

相互系切り替えシステムは，アプリケーションサーバの実行系と待機系を 1:1 にする構成の一つです。それぞれのアプリケーションサーバを実行系として稼働させながら，同時にお互いの待機系として構成します。実行系と待機系の詳細については，「15.2　クラスタソフトウェアと連携して実現できる運用」を参照してください。

ここでは，それぞれのアプリケーションサーバを，アプリケーションサーバ 1，アプリケーションサーバ 2 として，アプリケーションサーバ 1 には，J2EE サーバ A の実行系と J2EE サーバ B の待機系を，アプリケーションサーバ B には J2EE サーバ A の待機系と J2EE サーバ B の実行系を配置する構成を例にして説明します。例にする構成を次の図に示します。この例では，J2EE サーバ A の実行系と J2EE サーバ A の待機系をリソースグループ A，J2EE サーバ B の実行系と J2EE サーバ B の待機系をリソースグループ B と設定しています。

図 18-9　相互スタンバイシステムの構成例



注意事項

Management Server または運用管理エージェントのダウンによって系切り替えが発生した場合，ダウンしたホストには各論理サーバのプロセスが残っている場合があります。プロセスが残っていると系の切り戻しが発生した時に論理サーバの起動ができなくなってしまうため，系の切り戻しが発生する前にすべてのプロセスを停止させる必要があります。また，そのほかに，業務が正しく開始されるための作業はあらかじめ実施しておいてください。

18.6.1　相互系切り替えシステムの起動

ここでは，相互系切り替えシステムを利用する場合のシステムの起動手順について説明します。

相互系切り替えシステムを起動するには，あらかじめ Management Server，および運用管理エージェントを起動しておくことが必要です。Management Server，および運用管理エージェントを起動していない場合には，これらを先に起動しておいてください。

相互系切り替えシステムの起動手順を次に示します。

1. [スタート] メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。
   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）でアプリケーションサーバ 1 のノードを選択し，［ファイル］メニューの［クラスタサービスの開始］を選択します。
   アプリケーションサーバ 1 のノードのクラスタサービスが開始されます。

3. コンソールツリー（左ペイン）でアプリケーションサーバ 2 のノードを選択し，［ファイル］メニューの［クラスタサービスの開始］を選択します。
   アプリケーションサーバ 2 のノードのクラスタサービスが開始されます。

4. コンソールツリー（左ペイン）でリソースグループ A を選択して，［ファイル］メニューの［オンラインにする］を選択します。
   実行系（アプリケーションサーバ A）がオンラインになります。

5. コンソールツリー（左ペイン）でリソースグループ B を選択して，［ファイル］メニューの［オンラインにする］を選択します。
   実行系（アプリケーションサーバ B）がオンラインになります。

18.6.2 相互系切り替えシステムの停止

ここでは，相互系切り替えシステムの停止方法について説明します。

1. [スタート] メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。
   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）でリソースグループ B を選択して，［ファイル］メニューの［オフラインにする］を選択します。
   リソースグループ B がオフラインになります。

3. コンソールツリー（左ペイン）でリソースグループ A を選択して，［ファイル］メニューの［オフラインにする］を選択します。
   リソースグループ A がオフラインになります。

4. コンソールツリー（左ペイン）でアプリケーションサーバ 2 のノードを選択し，［ファイル］メニューの［クラスタサービスの停止］を選択します。

   アプリケーションサーバ 2 のノードのクラスタサービスが停止します。

5. コンソールツリー（左ペイン）でアプリケーションサーバ 1 のノードを選択し，［ファイル］メニューの［クラスタサービスの停止］を選択します。

   アプリケーションサーバ 1 のノードのクラスタサービスが停止します。

なお，Management Server，および運用管理エージェントは最後に手動で停止します。これらの停止方法については，次のマニュアルを参照してください。

- J2EE アプリケーションを実行するシステムの場合

  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「4.1.3　システムの停止手順」およびマニュアル「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「4.1.4　システムの停止方法」を参照してください。

- バッチアプリケーションを実行するシステムの場合

  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「6.1.3　システムの停止手順」およびマニュアル「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「6.1.4　システムの停止方法」を参照してください。

18.6.3　相互系切り替えシステムで計画的に系を切り替える場合の起動と停止

トラブル発生時以外で，相互系切り替えシステムの実行系と待機系を切り替える手順を次に示します。系を切り替えるときは，待機系のホストでアプリケーションサーバが待機状態になっている必要があります。

ここでは，J2EE サーバ A の系を切り替える場合について説明します。J2EE サーバ B の場合も同じ手順で系を切り替えることができます。

1. ［スタート］メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。

   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）でリソースグループ A を選択して，［ファイル］メニューの［グループの移動］を選択します。

   系が切り替わります。

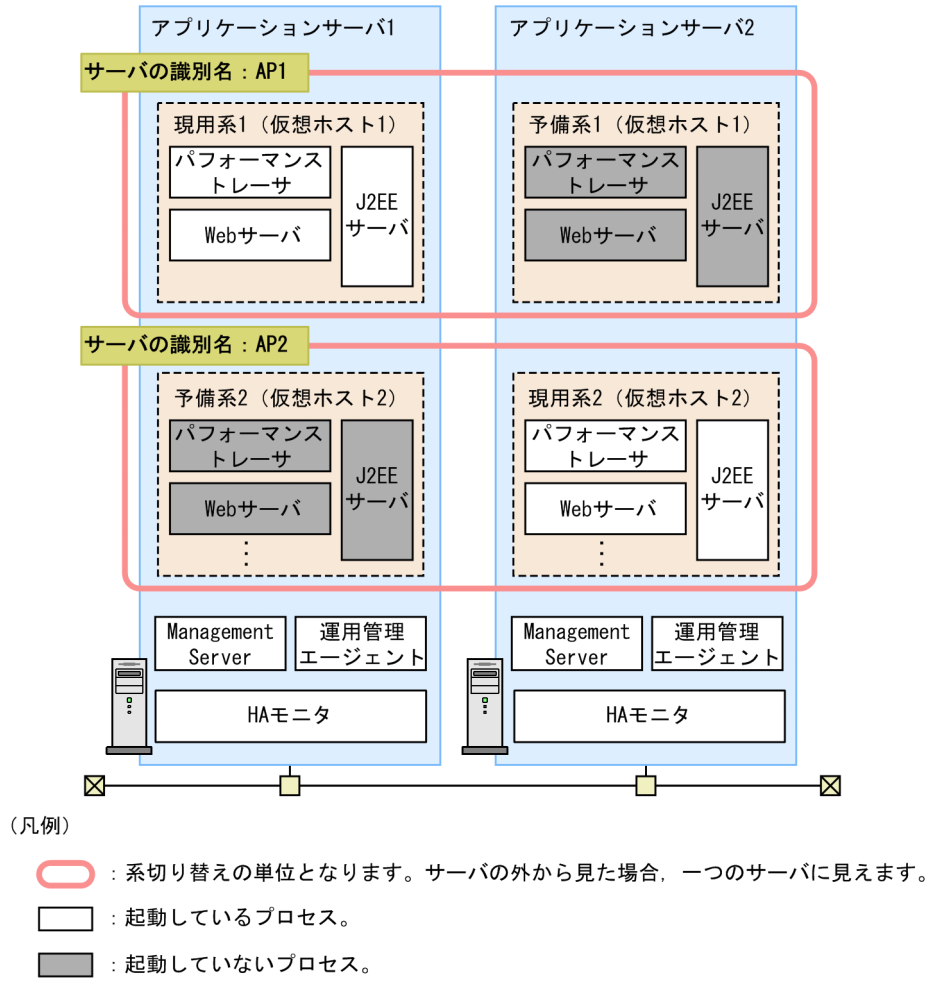
18.7　相互系切り替えシステムの起動と停止（UNIX の場合）

この節では，HA モニタによる相互系切り替えシステムを利用する場合の，システムの起動と停止の手順について説明します。稼働開始後に J2EE サーバ，およびバッチサーバをメンテナンスする場合の起動と停止の手順についても説明します。なお，HA モニタが利用できるのは，AIX または Linux の場合だけです。

相互系切り替えシステムを利用して運用する場合は，あらかじめ現用系と予備系の 2 種類のホストの用意や，HA モニタの監視の対象となる運用管理エージェントを監視・起動・停止するスクリプトの登録など，必要な環境設定をしておく必要があります。設定方法については，「18.5.4　HA モニタの環境設定」の HA モニタの設定に関する説明を参照してください。

相互系切り替えシステムのシステム構成例を次の図に示します。この節では，このシステム構成例を基に，相互系切り替えシステムの起動および停止の手順について説明します。

図 18-10　相互系切り替えシステムのシステム構成例



図中の各アプリケーションサーバでの servers ファイルの定義（HA モニタでのサーバの環境設定）のうち，起動・停止に関連する定義項目を次の表に示します。

表 18–5　アプリケーションサーバ 1 の servers ファイルの定義

オペランド	説明	指定値	
		現用系 1 （仮想ホスト 1）	予備系 2 （仮想ホスト 2）
alias	サーバの識別名	AP1	AP2
initial	HA モニタでのサーバの起動方法	online	standby

表 18–6　アプリケーションサーバ 2 の servers ファイルの定義

オペランド	説明	指定値	
		予備系 1 （仮想ホスト 1）	現用系 2 （仮想ホスト 2）
alias	サーバの識別名	AP1	AP2
initial	HA モニタでのサーバの起動方法	standby	online

なお，servers ファイルの定義の詳細については，「18.5.6　サーバ対応の環境設定」を参照してください。

また，次のコマンドは，HA モニタが提供しているコマンドです。詳細は，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

- monbegin（HA モニタとのインタフェースを持たないサーバを起動）
- monend（HA モニタとのインタフェースを持たない実行サーバの停止連絡）
- monsbystp（待機サーバの停止）
- monswap（計画系切り替え）

注意事項

Management Server または運用管理エージェントのダウンによって系切り替えが発生した場合，ダウンしたホストには各論理サーバのプロセスが残っている場合があります。プロセスが残っていると系の切り戻しが発生した時に論理サーバの起動ができなくなってしまうため，系の切り戻しが発生する前にすべてのプロセスを停止させる必要があります。また，そのほかに，業務が正しく開始されるための作業はあらかじめ実施しておいてください。

18.7.1　相互系切り替えシステムの起動

相互系切り替えシステムを利用する場合のシステム起動時の留意事項，およびシステムの起動手順について説明します。

(1)　システム起動時の留意事項

相互系切り替えシステムを利用する場合のシステム起動時の留意事項を次に示します。

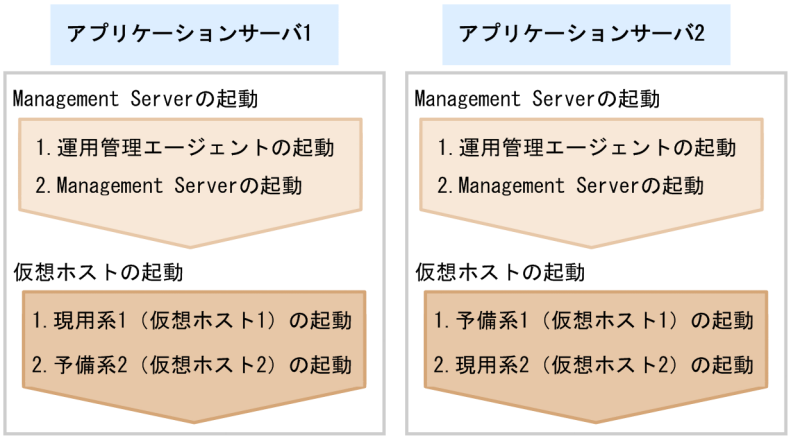
- アプリケーションサーバ 1 とアプリケーションサーバ 2 のホストに配置された HA モニタは，OS の起動と同時に起動されています。
- アプリケーションサーバ 1 およびアプリケーションサーバ 2 の各ホストで仮想ホストを起動する前に，Management Server をあらかじめ起動しておいてください。

(2)　システムの起動手順

相互系切り替えシステムの起動の流れを次の図に示します。

アプリケーションサーバ 1 およびアプリケーションサーバ 2 の各ホストで次の図に示す流れでシステムを起動してください。

図 18–11　相互系切り替えシステムの起動の流れ



参考

アプリケーションサーバ 1 およびアプリケーションサーバ 2 の各ホストでの仮想ホストの起動順序は任意です。現用系仮想ホストまたは予備系仮想ホストのどちらから起動してもかまいません。なお，このマニュアルでは，仮想ホスト 1，仮想ホスト 2 の順序で起動する場合の手順について説明します。

図中の Management Server の起動の手順 1.～手順 2.については，次のマニュアルを参照してください。

- J2EE アプリケーションを実行するシステムの場合

  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「4.1.1　システムの起動手順」を参照してください。
- バッチアプリケーションを実行するシステムの場合

  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「6.1.1　システムの起動手順」を参照してください。

なお，自動起動の設定をしている場合，ホストの起動時に，運用管理エージェント，Management Server も起動されるため，手順 1.～手順 2.は省略できます。システムの起動方法の設定については，「2.6　システムの起動と停止の設定」を参照してください。

ここでは，図中の仮想ホストの起動手順について説明します。アプリケーションサーバ 1 およびアプリケーションサーバ 2 の各ホストでの起動手順を次に示します。

- **アプリケーションサーバ 1 での起動手順**

  1. **現用系 1（仮想ホスト 1）の起動**

     monbegin コマンドを実行して，現用系 1（仮想ホスト 1）を実行系として起動します。

     ```
     # monbegin AP1
     ```

     これによって，現用系の仮想ホストを起動するスクリプトファイルに定義されている処理が実行されて，現用系の仮想ホストが実行系として起動されます。

  2. **予備系 2（仮想ホスト 2）の起動**

     monbegin コマンドを実行して，予備系 2（仮想ホスト 2）を待機系として起動します。

     ```
     # monbegin AP2
     ```

     これによって，予備系の仮想ホストが待機系となり，実行系の障害に備えます。

- **アプリケーションサーバ 2 での起動手順**

  1. **予備系 1（仮想ホスト 1）の起動**

     monbegin コマンドを実行して，予備系 1（仮想ホスト 1）を待機系として起動します。

     ```
     # monbegin AP1
     ```

     これによって，予備系の仮想ホストが待機系となり，実行系の障害に備えます。

  2. **現用系 2（仮想ホスト 2）の起動**

     monbegin コマンドを実行して，現用系 2（仮想ホスト 2）を実行系として起動します。

     ```
     # monbegin AP2
     ```

     これによって，現用系の仮想ホストを起動するスクリプトファイルに定義されている処理が実行されて，現用系の仮想ホストが実行系として起動されます。

18.7.2　相互系切り替えシステムの停止

相互系切り替えシステムの停止手順について説明します。

ここでは，次の二つの場合の停止手順について説明します。

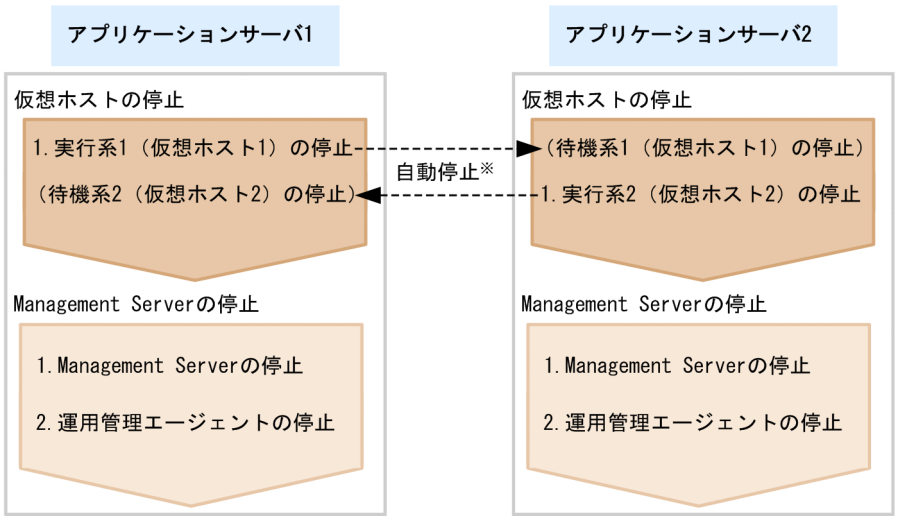
- 実行系と待機系の両方の仮想ホストを停止する場合
- 待機系の仮想ホストだけ停止する場合

(1)　実行系と待機系の両方の仮想ホストを停止する場合

実行系と待機系の両方の仮想ホストを停止する場合の手順について説明します。

実行系と待機系の両方の仮想ホストを停止する場合の，相互系切り替えシステムの停止の流れを次の図に示します。

図 18-12　相互系切り替えシステムの停止の流れ



（凡例）

---▶　：HAモニタによる自動停止の指示の流れ

注※　実行系が停止したあと，HAモニタによって待機系に停止指示が出され，待機系が自動停止します。

> **参考**
>
> 仮想ホスト 1 および仮想ホスト 2 の停止順序は任意です。このマニュアルでは，仮想ホスト 1，仮想ホスト 2 の順序で停止する場合の手順について説明します。

すべての仮想ホストを停止したあと，アプリケーションサーバ 1 およびアプリケーションサーバ 2 の各ホストの Management Server を停止します。

図中の Management Server の停止の手順 1.～手順 2.については，次のマニュアルを参照してください。

- J2EE アプリケーションを実行するシステムの場合

  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「4.1.3　システムの停止手順」を参照してください。
- バッチアプリケーションを実行するシステムの場合

  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「6.1.3　システムの停止手順」を参照してください。

なお，自動停止の設定をしている場合，ホストの停止時に，Management Server，および運用管理エージェントも停止されるため，手順 1.～手順 2.は省略できます。システムの停止方法の設定については，「2.6　システムの起動と停止の設定」を参照してください。

ここでは，図中の仮想ホストの停止手順について説明します。アプリケーションサーバ 1 およびアプリケーションサーバ 2 の各ホストでの停止手順を次に示します。

- アプリケーションサーバ 1 での停止手順

  1. アプリケーションサーバ 1 で monend コマンドを実行して，実行系の仮想ホスト 1 を停止します。

     ```
     # monend AP1
     ```

     これによって，アプリケーションサーバ 1 の実行系 1（仮想ホスト 1）が停止します。また，HA モニタによって待機系に自動的に停止指示が出され，アプリケーションサーバ 2 の待機系 1（仮想ホスト 1）も停止します。

- アプリケーションサーバ 2 での停止手順

  1. アプリケーションサーバ 2 で monend コマンドを実行して，実行系の仮想ホスト 2 を停止します。

     ```
     # monend AP2
     ```

     これによって，アプリケーションサーバ 2 の実行系 2（仮想ホスト 2）が停止します。また，HA モニタによって待機系に自動的に停止指示が出され，アプリケーションサーバ 1 の待機系 2（仮想ホスト 2）も停止します。

(2)　待機系の仮想ホストだけ停止する場合

実行系の仮想ホストを停止しないで待機系の仮想ホストで待機状態をやめる場合の手順について説明します。

- アプリケーションサーバ 1 の待機系 2（仮想ホスト 2）で待機状態をやめる場合

  アプリケーションサーバ 1 のホストで次のコマンドを実行します。

  ```
  # monsbystp AP2
  ```

- アプリケーションサーバ 2 の待機系 1（仮想ホスト 1）で待機状態をやめる場合

  アプリケーションサーバ 2 のホストで次のコマンドを実行します。

  ```
  # monsbystp AP1
  ```

18.7.3　相互系切り替えシステムで計画的に系を切り替える場合の起動と停止

相互系切り替えシステムでトラブル発生時以外の場合に，実行系と待機系を計画的に切り替えるときの手順について説明します。系を切り替えるときには，待機系の仮想ホストが待機状態になっている必要があります。

ここでは，仮想ホスト 1 および仮想ホスト 2 の実行系と待機系を計画的に切り替える場合の手順について説明します。なお，現用系のホストが実行系として起動されていることを前提にして説明します。

参考

相互系切り替えシステムをメンテナンスする場合の起動と停止の手順については，「18.7.4　相互系切り替えシステムをメンテナンスする場合の起動と停止」を参照してください。

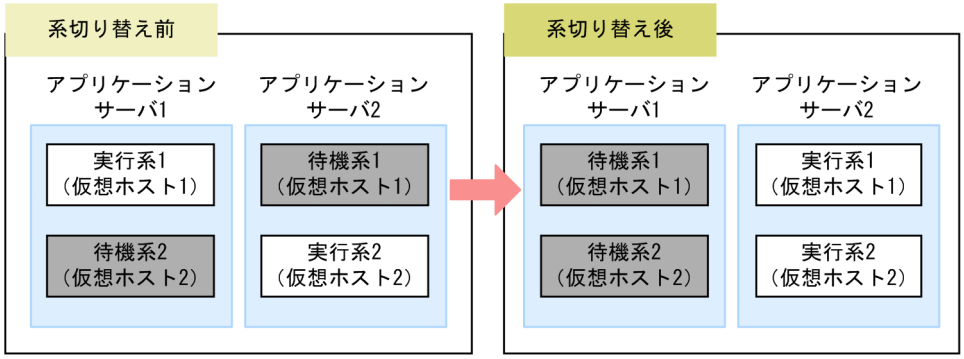
(1)　仮想ホスト 1 を計画的に系切り替えする場合

仮想ホスト 1 を計画的に系切り替えする場合の手順を次に示します。

1. アプリケーションサーバ 1 で monswap コマンドを実行して，系を切り替えます。

```
# monswap AP1
```

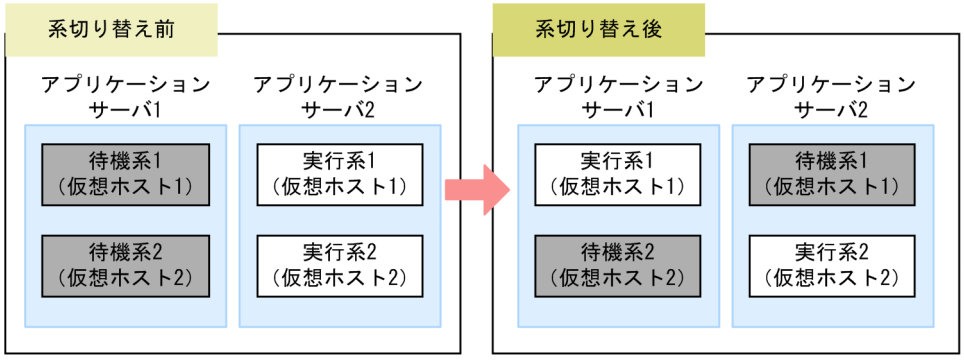
これによって，次の図に示すように，アプリケーションサーバ 1 の実行系 1（仮想ホスト 1）が待機系の仮想ホストに切り替わり，アプリケーションサーバ 2 の待機系 1（仮想ホスト 1）が実行系の仮想ホストに切り替わります。



2. 通常運用の実行系と待機系の状態に戻す場合，アプリケーションサーバ 2 で monswap コマンドを実行して，系を切り替えます。

```
# monswap AP1
```

これによって，次の図に示すように，アプリケーションサーバ 2 の実行系 1（仮想ホスト 1）が待機系の仮想ホストに切り替わり，アプリケーションサーバ 1 の待機系 1（仮想ホスト 1）が実行系の仮想ホストに切り替わります。これで，通常運用の実行系と待機系の状態に戻ります。



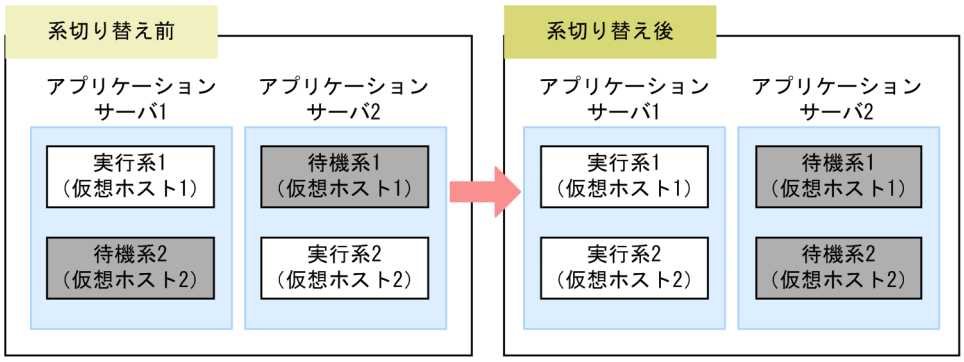
(2)　仮想ホスト 2 を計画的に系切り替えする場合

仮想ホスト 2 を計画的に系切り替えする場合の手順を次に示します。

1. アプリケーションサーバ 2 で monswap コマンドを実行して，系を切り替えます。

```
# monswap AP2
```

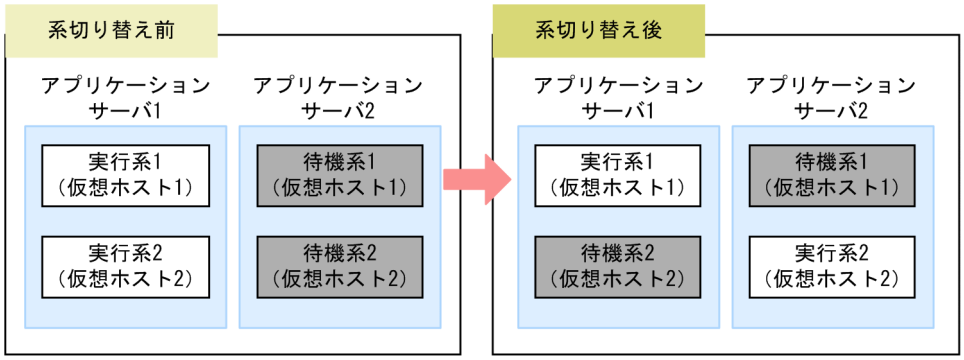
これによって，次の図に示すように，アプリケーションサーバ 2 の実行系 2（仮想ホスト 2）が待機系の仮想ホストに切り替わり，アプリケーションサーバ 1 の待機系 2（仮想ホスト 2）が実行系の仮想ホストに切り替わります。



2. 通常運用の実行系と待機系の状態に戻す場合，アプリケーションサーバ 1 で monswap コマンドを実行して，系を切り替えます。

```
# monswap AP2
```

これによって，次の図に示すように，アプリケーションサーバ 1 の実行系 2（仮想ホスト 2）が待機系の仮想ホストに切り替わり，アプリケーションサーバ 2 の待機系 2（仮想ホスト 2）が実行系の仮想ホストに切り替わります。これで，通常運用の実行系と待機系の状態に戻ります。



18.7.4　相互系切り替えシステムをメンテナンスする場合の起動と停止

相互系切り替えシステムをメンテナンスする場合の，起動と停止の手順を次に示します。

ここでは，次の三つの場合の停止手順について説明します。

- 再起動が不要なメンテナンスの場合

- 再起動が必要なメンテナンスの場合（両方の系を同時に停止する方法）
- 再起動が必要なメンテナンスの場合（両方の系を同時に停止しない方法）

なお，これらの手順は，すでに実行系の仮想ホストが両方とも起動している場合の手順です。

(1) 再起動が不要なメンテナンスの場合

再起動が不要なメンテナンスの場合の起動と停止の手順を次に示します。

1. 実行系の仮想ホスト（現用系の仮想ホスト）で，動作中の J2EE アプリケーションとリソースアダプタを停止します。

   バッチサーバの場合は，バッチアプリケーションが実行中でないことを確認してからリソースアダプタを停止してください。

   停止方法については，次のマニュアルを参照してください。

   - J2EE アプリケーションを実行するシステムの場合

     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「4.1.3　システムの停止手順」およびマニュアル「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「4.1.4　システムの停止方法」を参照してください。

   - バッチアプリケーションを実行するシステムの場合

     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「6.1.3　システムの停止手順」およびマニュアル「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「6.1.4　システムの停止方法」を参照してください。

2. サーバ管理コマンドを使用してメンテナンス処理を実行します。

   サーバ管理コマンドでの操作については，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。

3. 手順 1.で停止した J2EE アプリケーションとリソースアダプタを開始します。

   J2EE サーバの場合は J2EE アプリケーションとリソースアダプタを開始します。また，バッチサーバの場合はリソースアダプタを開始します。

   開始方法については，次のマニュアルを参照してください。

   - J2EE アプリケーションを実行するシステムの場合

     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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

   - バッチアプリケーションを実行するシステムの場合

     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「6.1.1　システムの起動手順」およびマニュアル「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「6.1.2　システムの起動方法」を参照してください。

4. 必要に応じて J2EE アプリケーションとリソースアダプタの動作確認をします。

5. 実行系の仮想ホスト（現用系の仮想ホスト）に対して，monswap コマンドを実行して，系を切り替えます。

```
# monswap サーバの識別名
```

下線部分には，実行系の仮想ホストが配置されているアプリケーションサーバの，servers ファイルのオペランド「alias」に指定されている実行系の仮想ホストのサーバの識別名を指定します。

実行系の仮想ホスト（現用系の仮想ホスト）が待機系の仮想ホストに切り替わり，待機系だった仮想ホスト（予備系の仮想ホスト）が実行系の仮想ホストになります。

6. 切り替え後の実行系の仮想ホスト（予備系の仮想ホスト）で手順 1.～手順 4.を実行します。

7. 切り替え後の実行系のホスト（予備系の仮想ホスト）で，必要に応じて，再度 monswap コマンドを実行します。

```
# monswap サーバの識別名
```

下線部分には，手順 5.で monswap コマンドの引数に指定したサーバの識別名と同じサーバの識別名を指定します。

現用系の仮想ホストを実行系に戻す場合に実行してください。

これで，メンテナンス時の起動と停止の手順は完了です。

(2) 再起動が必要なメンテナンスの場合（両方の系を同時に停止する方法）

再起動が必要なメンテナンスの場合の起動と停止の手順を次に示します。

1. 実行系の仮想ホストに対して monend コマンドを実行して，システムを停止します。

```
# monend サーバの識別名
```

下線部分には，実行系の仮想ホストが配置されているアプリケーションサーバの，servers ファイルのオペランド「alias」に指定されている実行系の仮想ホストのサーバの識別名を指定します。

実行系の仮想ホストが停止します。また，HA モニタによって待機系に自動的に停止指示が出され，待機系の仮想ホストも停止します。

2. 現用系と予備系の両方の仮想ホストに対して，定義ファイルの変更など，メンテナンス作業を実行します。

3. 現用系の仮想ホストに対して，monbegin コマンドを実行します。

```
# monbegin サーバの識別名
```

下線部分には，手順 1.で monend コマンドの引数に指定したサーバの識別名と同じサーバの識別名を指定します。

これによって，現用系の仮想ホストが，実行系として起動します。

4. 予備系の仮想ホストに対して，monbegin コマンドを実行します。

```
# monbegin サーバの識別名
```

下線部分には，手順 1.で monend コマンドの引数に指定したサーバの識別名と同じサーバの識別名を指定します。

これによって，予備系の仮想ホストが，待機系として起動します。

5. 実行系の仮想ホストで，必要に応じて，定義の変更に関連した動作確認を実行します。

参考

servers ファイルの定義（HA モニタでのサーバの環境設定）については，「18.5.6　サーバ対応の環境設定」を参照してください。

(3)　再起動が必要なメンテナンスの場合（両方の系を同時に停止しない方法）

再起動が必要なメンテナンスの場合の起動と停止の手順を次に示します。

1. 待機系の仮想ホストで，定義ファイルの変更など，メンテナンス作業を実行します。

2. 実行系の仮想ホストに対して monswap コマンドを実行して，系を切り替えます。

```
# monswap サーバの識別名
```

下線部分には，実行系の仮想ホストが配置されているアプリケーションサーバの，servers ファイルのオペランド「alias」に指定されている実行系の仮想ホストのサーバの識別名を指定します。

実行系の仮想ホストが停止してから，待機していた系が起動します。これによって，メンテナンス済みの仮想ホストが実行系の仮想ホストになります。

3. 切り替え後の実行系のホスト（予備系の仮想ホスト）に対して，必要に応じて，定義の変更に関連した動作確認を実行します。

4. 切り替え後の待機系のホスト（現用系の仮想ホスト）で定義ファイルの編集など，メンテナンス作業を実行します。

5. 切り替え後の実行系の仮想ホスト（予備系のホスト）に対して，必要に応じて，再度 monswap コマンドを実行します。

```
# monswap サーバの識別名
```

下線部分には，手順 2.で monswap コマンドの引数に指定したサーバの識別名と同じサーバの識別名を指定します。

現用系の仮想ホストを実行系に戻す場合に実行してください。

6. 切り替え後の実行系の仮想ホスト（現用系の仮想ホスト）で，必要に応じて，定義の変更に関連した動作確認を実行します。

これで，メンテナンス時の起動と停止の手順は完了です。

19 N:1 リカバリシステム（クラスタソフトウェアとの連携）

この章では，クラスタソフトウェアと連携した N:1 リカバリシステムで運用するシステムについて説明します。

19.1　この章の構成

この章の構成を次の表に示します。

表 19–1　この章の構成（N:1 リカバリシステム（クラスタソフトウェアとの連携））

分類	タイトル	参照先
解説	N:1 リカバリシステムの概要	19.2
	N:1 リカバリシステムのシステム構成と動作	19.3
設定	N:1 リカバリシステムの設定（Windows の場合）	19.4
	N:1 リカバリシステムの設定（UNIX の場合）	19.5
運用	N:1 リカバリシステムの起動と停止（Windows の場合）	19.6
	N:1 リカバリシステムの起動と停止（UNIX の場合）	19.7

注　「実装」,「注意事項」について，この機能固有の説明はありません。

19.2 N:1 リカバリシステムの概要

N:1 リカバリシステムとは，クラスタ構成になっている実行系の複数（N 台）のアプリケーションサーバに対して，1 台のリカバリ専用サーバを待機系として配置したシステムです。ただし，バッチアプリケーションの実行環境の場合，N:1 リカバリシステムは使用できません。

19.3　N:1 リカバリシステムのシステム構成と動作

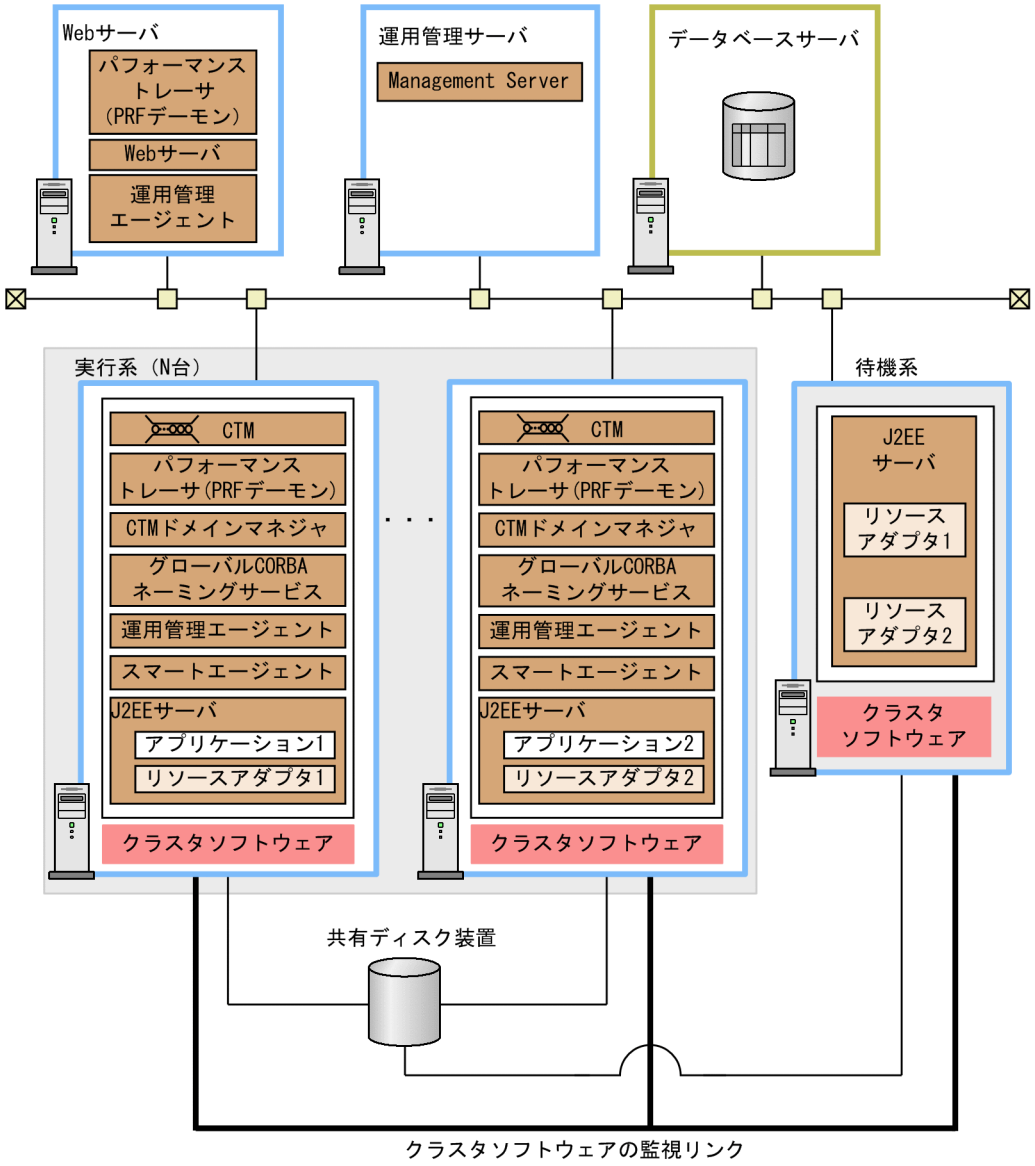
ここでは，システムの構成例，リカバリ処理の流れについて説明します。

19.3.1　N:1 リカバリシステムのシステム構成例

クラスタソフトウェアと連携した N:1 リカバリシステムのシステム構成例を次の図に示します。クラスタ構成で実行中のアプリケーションサーバに障害が発生すると，待機系のリカバリ専用サーバが起動し，障害が発生した実行系のトランザクションを決着します。

待機系であるリカバリ専用サーバの J2EE サーバには，実行系にインポートされているリソースアダプタをすべてインポートします。なお，リカバリ専用サーバは障害が発生した実行系のアプリケーションサーバのリカバリを実施するだけなので，アプリケーションをデプロイする必要はありません。

図 19-1　N:1 リカバリシステムのシステム構成例



N:1 リカバリシステムでは次のような運用ができます。

共有ディスク装置の使用

共有ディスク装置が必要になります。共有ディスク装置は，OTS のステータスなどのトランザクション情報を引き継ぐために使用します。トランザクション情報は，実行系のそれぞれのアプリケーションサーバと，待機系のリカバリ専用サーバとの間で引き継ぎします。

負荷分散機の適用

このシステム構成例では示していませんが，同一構成の Web サーバを複数用意して，負荷分散機を適用することもできます。これによって，Web サーバの信頼性・稼働率を上げることができます。

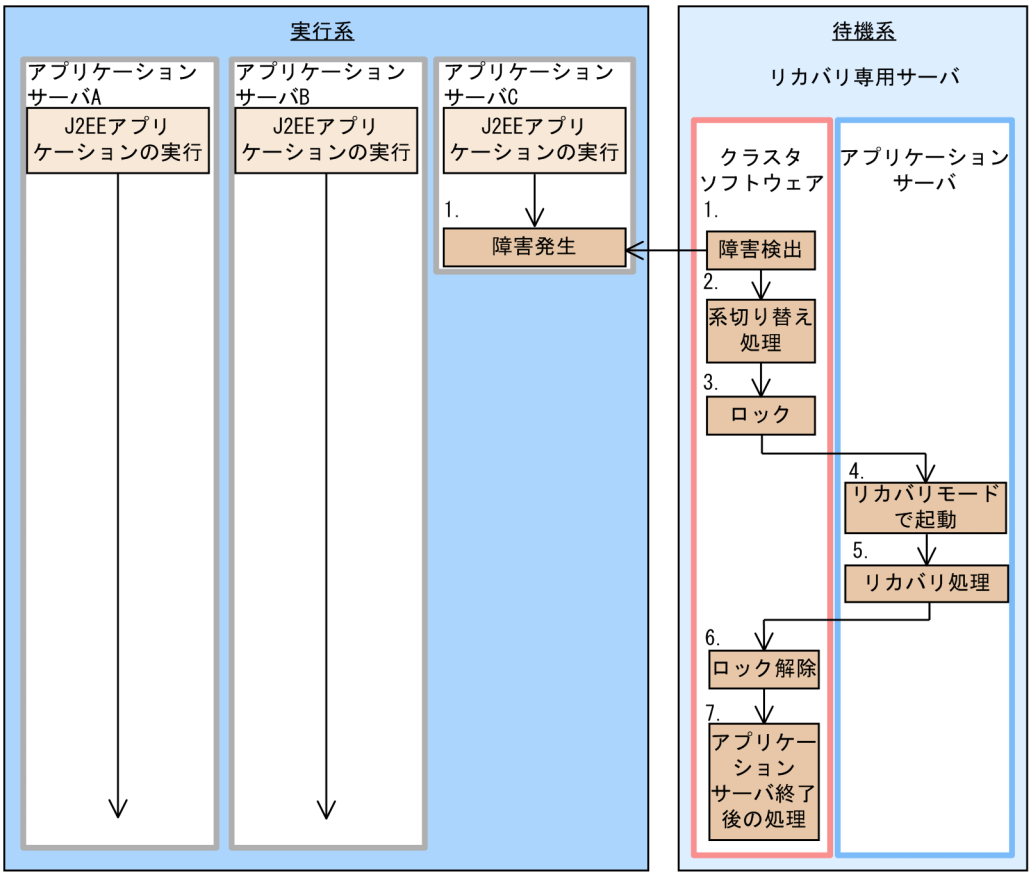
なお，N:1 リカバリシステムのシステム構成の詳細については，マニュアル「アプリケーションサーバ システム設計ガイド」の「3.11.5　リカバリ専用サーバを使用する場合の構成（N:1 リカバリシステム）」を参照してください。

19.3.2　リカバリ処理の流れ

クラスタ構成の実行系と，リカバリ専用サーバである 1 台の待機系で稼働します。実行系の複数のアプリケーションサーバのうち一つでも障害が発生すると，クラスタソフトウェアで障害を検出し，待機系であるリカバリ専用サーバに切り替えられます。待機系では，ダウンした実行系のアプリケーションサーバのトランザクションを決着します。

リカバリ処理の流れについて次の図に示します。

図 19-2　リカバリ処理の流れ



1. リカバリ専用サーバのクラスタソフトウェアが障害を検知します。

   実行系のアプリケーションサーバ C で障害が発生し，ダウンすると，待機系のリカバリ専用サーバのクラスタソフトウェアはアプリケーションサーバ C のダウンを検知します。

2. リカバリ専用サーバのクラスタソフトウェアで系切り替えを実施します。

   待機系のリカバリ専用サーバに切り替えます。このとき，共有ディスク装置の切り替え，共有ディスク装置のマウント，および仮想ホストの IP アドレスの設定をします。

3. リカバリ専用サーバをロックします。

   リカバリ専用サーバでは，順次，リカバリ処理を実施します。このため，リカバリ処理を実施する前に，リカバリ専用サーバをロックします。

4. リカバリ専用サーバのクラスタソフトウェアでリカバリコマンドを実行し，リカバリ専用サーバのアプリケーションサーバをリカバリモードで起動します。

5. ダウンしたアプリケーションサーバ C で仕掛かり中だったトランザクションのリカバリ処理を，リカバリ専用サーバで実施します。

   リカバリ専用サーバでは，トランザクションのリカバリ処理だけが実施されます。リカバリ処理終了後，リカバリ専用サーバのアプリケーションサーバは終了します。

6. リカバリ専用サーバのロックを解除します。

7. リカバリ専用サーバのクラスタソフトウェアでアプリケーションサーバ終了後の処理を実施します。

共有ディスク装置の切り替え，共有ディスク装置のアンマウント，および仮想ホストの IP アドレスの削除をします。

ほかの実行系のアプリケーションサーバに障害が発生している場合は，リカバリ処理をします。なお，実行系のアプリケーションサーバ A および B については，アプリケーションサーバ C のダウンの影響を受けず，アプリケーション処理を実行します。

注意事項

- 複数の実行系で障害が発生し，アプリケーションサーバがダウンした場合は，リカバリ専用サーバ（待機系）では，排他で順次リカバリ処理をします。
- リカバリ専用サーバでのリカバリ処理中は，リカバリ専用サーバのサービスポートが閉塞されるため，処理の受け付けはされません。
- リカバリ専用サーバでは，リカバリ処理が終了しない場合に備えて，タイムアウト監視が実施されます。
- データベースのダウンによってタイムアウトした場合は，手動でリカバリ処理をしてください。詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「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」を参照してください。
- リカバリ専用サーバがダウンした場合などの二重障害については，対応していません。

19.3.3　系切り替え時の情報の引き継ぎ

系切り替え時に，待機系に引き継がれる情報と，引き継がれない情報について説明します。

- 引き継がれる情報
  - OTS トランザクション情報の引き継ぎ（グローバルトランザクションの場合）

    実行系から待機系のへの系切り替え後，OTS トランザクション情報を引き継ぎ，トランザクションの回復処理をします。
- 引き継がれない情報

  系切り替えが発生したとき，次に示す情報は切り替え後の系に引き継がれないので注意してください。
  - Web コンテナが管理するアプリケーションの HTTP セッション情報
  - Web コンテナの ServletContext に設定したアトリビュート情報
  - ホームインタフェースとコンポーネントインタフェースの Bean のセッション情報
  - アプリケーションの稼働状態，JNDI のキャッシュ，コネクションプール，CTM のキューなどの状態に関する情報

19.4　N:1 リカバリシステムの設定（Windows の場合）

N:1 リカバリシステムは，N 台の実行系に対して 1 台の待機系（**リカバリ専用サーバ**）を用意する構成のシステムです。J2EE サーバを冗長化した構成でグローバルトランザクションを使用する場合に，特定の J2EE サーバにトラブルが発生したとき，トランザクションを決着するために使用します。このシステムは，運用管理ポータルを使用して構築します。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。この節では，N:1 リカバリシステムの設定について説明します。

業務は，N 台の実行系のアプリケーションサーバで実行します。業務実行中のアプリケーションサーバに障害が発生すると，クラスタソフトウェアがこれを検知して，リカバリ専用サーバで，障害が発生したアプリケーションサーバの仕掛かり中のトランザクションを決着します。その後，残りの稼働中の実行系のアプリケーションサーバで業務を続行します。

なお，N:1 リカバリシステムの詳細については，「19.2　N:1 リカバリシステムの概要」を参照してください。システム構成については，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「3.11.5 リカバリ専用サーバを使用する場合の構成（N:1 リカバリシステム）」を参照してください。

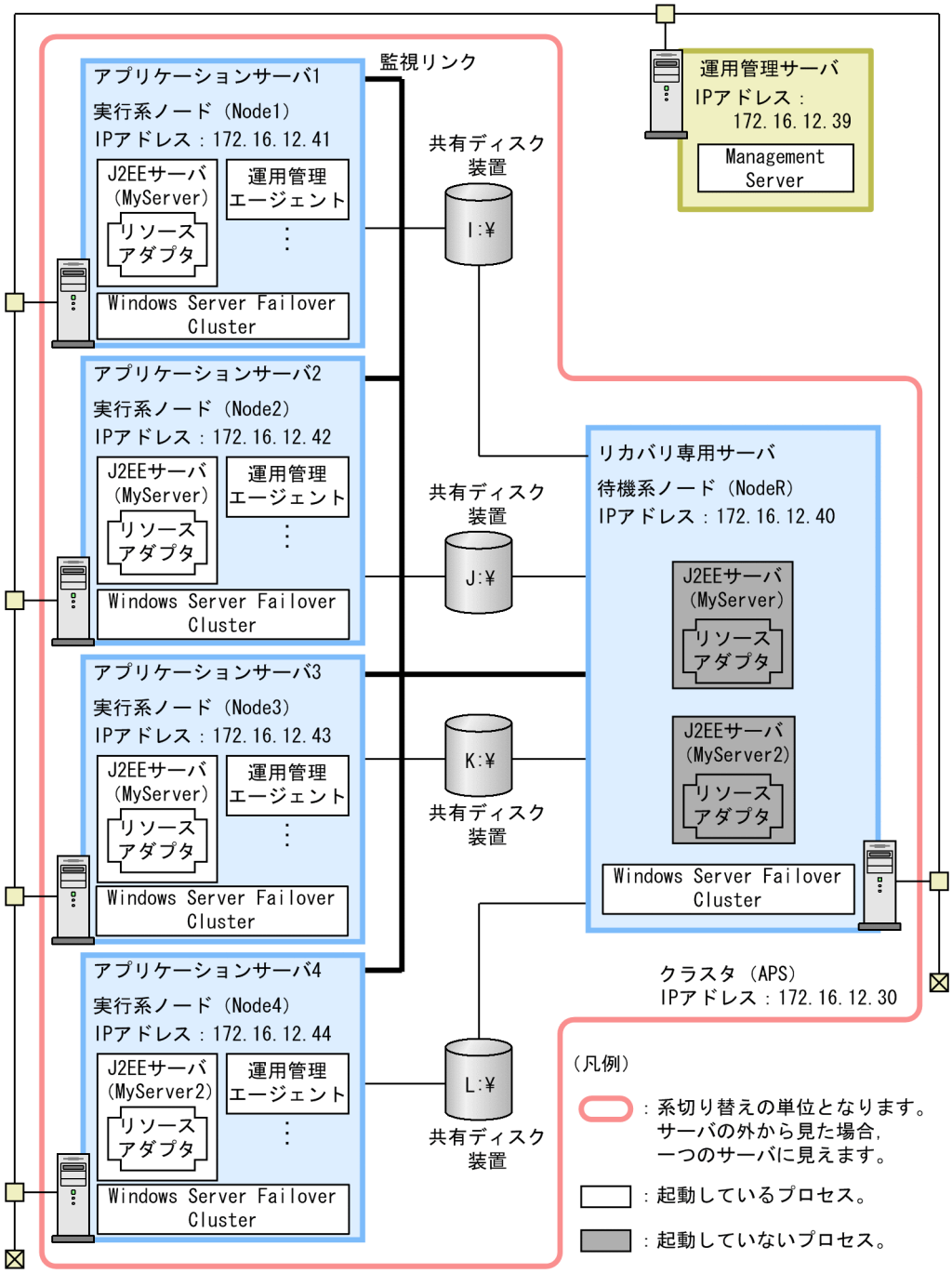
19.4.1　N:1 リカバリシステムの設定手順

ここでは，Windows Server Failover Cluster と連携する場合の，システムの構成例とシステムの設定手順について説明します。

(1)　システムの構成例

N:1 リカバリシステムの構成例を次の図に示します。なお，以降の項では，このシステムの構成例を使用したシステムの構築例を示します。

図 19-3　システムの構成例（N:1 リカバリシステムの場合）

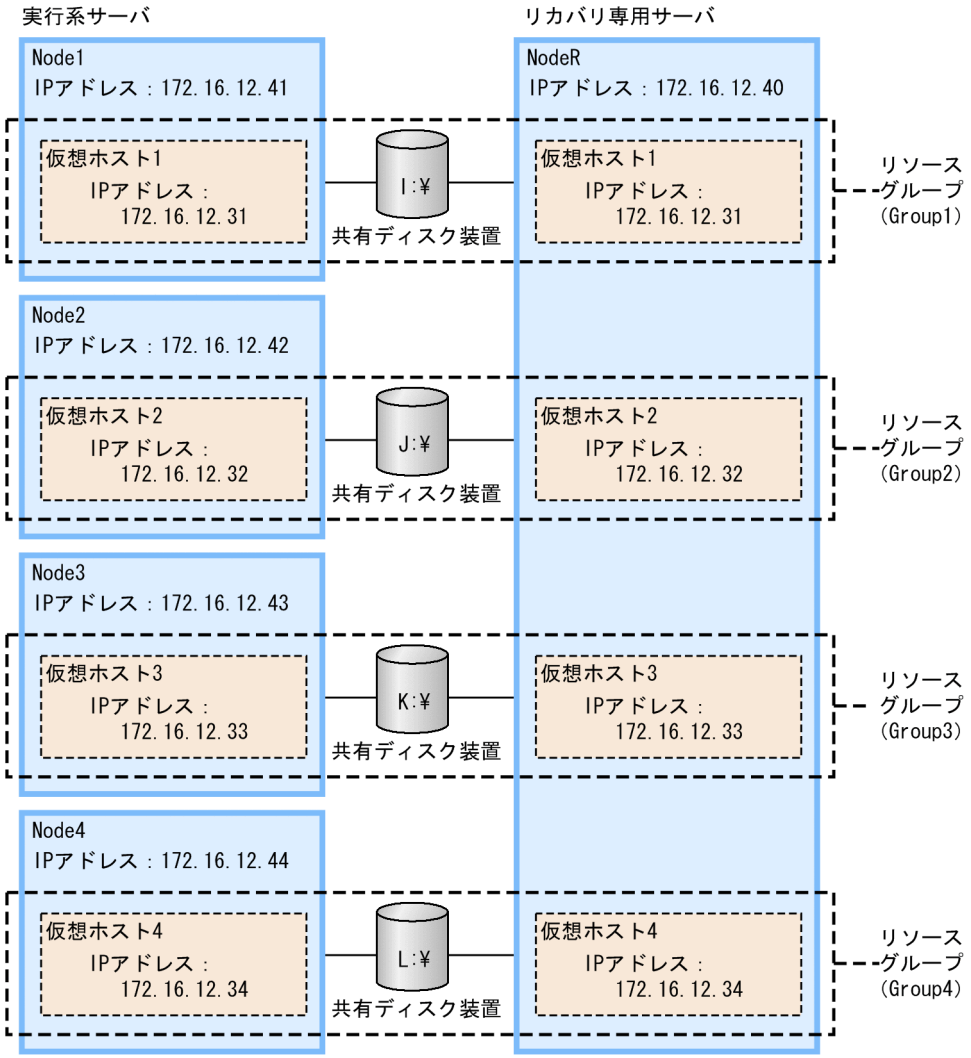


この例では，4 台の実行系サーバと 1 台のリカバリ専用サーバを配置しています。4 台の実行系のうち，3 台の実行系では，同じ J2EE サーバ名で同じ J2EE アプリケーションとリソースアダプタがデプロイされている「MyServer」という J2EE サーバを配置しています。残りの 1 台の実行系では，J2EE サーバ名や，デプロイされている J2EE アプリケーションとリソースアダプタが異なる「MyServer2」という J2EE サーバを配置しています。

N:1 リカバリ構成の場合，一つのクラスタ内に複数の仮想ホストを構築します。仮想ホストは実行系ノードとリカバリ専用ノードの組で構成し，J2EE サーバをクラスタ構成で配置します。N:1 リカバリ構成の場合，J2EE サーバはクラスタ内ではなく，仮想ホスト内で動作します。このため，仮想ホストの IP アドレスおよびホスト名を設定してください。

仮想ホストは一つのリソースグループとして定義され，「物理ディスクリソース」，「IP アドレスリソース」，「ネットワークリソース名」，「汎用スクリプトリソース」が仮想ホストに含まれます。ここでのリソースグループの例を次の図に示します。

図 19-4　リソースグループの例



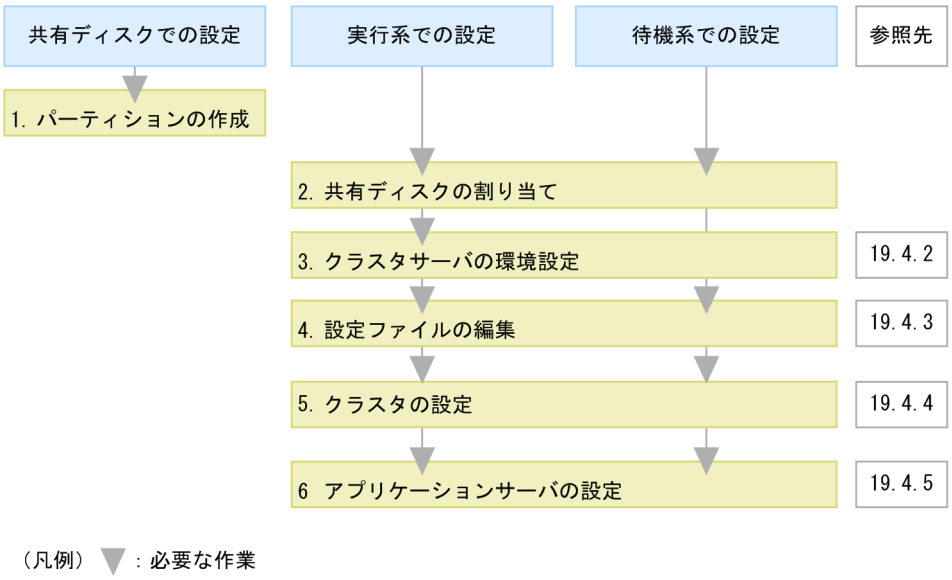
アプリケーションサーバをクラスタ構成に配置するためには，エイリアス IP アドレスを設けて，稼働中のノードがエイリアス IP アドレスを引き継ぐことで，クライアントがクラスタ内のノードを意識しないようにします。N:1 リカバリ構成の場合，エイリアス IP アドレスは，仮想ホストの IP アドレスとなります。

待機系の J2EE サーバには，実行系の J2EE サーバで使用しているすべてのリソースアダプタと同じ設定のリソースアダプタをインポート，およびデプロイしてください。

(2)　システムの設定手順

Windows Server Failover Cluster と連携する場合には，運用管理ポータルやクラスタアドミニストレータの設定などが必要になります。N:1 リカバリシステムの設定手順を次の図に示します。

図 19-5　N:1 リカバリシステムの設定手順（Windows の場合）



図中の 1.～6.について説明します。

1. 共有ディスクに N 台分のパーティションを作成して，ファイルシステムを構築します。

   Windows Server Failover Cluster のクラスタ管理用ファイルの格納場所を作成します。また，N:1 リカバリシステムではグローバルトランザクションを使用するため，トランザクション情報の格納場所を作成します。なお，クラスタ管理用ファイルの格納場所とトランザクション情報の格納場所は，同じパーティションでもかまいません。

2. システムに共有ディスクを割り当てます。

   システムに共有ディスクを割り当てる際は，実行系と待機系で，同じドライブ文字を割り当ててください。

3. 運用管理ポータルでクラスタサーバの環境を設定します。

   運用管理ポータルの「運用管理ドメインの構成定義」，および「論理サーバの環境設定」で，Windows Server Failover Cluster を使用する場合の設定をします。詳細は，「19.4.2　クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

   運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。詳細は，「19.4.3　設定ファイルの編集」を参照してください。

5. クラスタを設定します。

   クラスタソフトウェアの環境を設定します。詳細については，「19.4.4　クラスタの設定」を参照してください。

6. 運用管理ポータル，クラスタアドミニストレータなどで，アプリケーションサーバの設定をします。

運用管理ポータル，クラスタアドミニストレータなどを使用して，アプリケーションサーバをクラスタ構成に配置し，J2EE アプリケーションやリソースアダプタを設定します。詳細は，「19.4.5　アプリケーションサーバの設定」を参照してください。

19.4.2　クラスタサーバの環境設定

Windows Server Failover Cluster と連携する場合の，運用管理ポータルでの設定の留意点については，「17.4.2　クラスタサーバの環境設定」を参照してください。

なお，N:1 リカバリ構成の場合，ホスト名には，クラスタ IP アドレスではなく，仮想ホストの IP アドレスを設定してください。

19.4.3　設定ファイルの編集

運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。ここでは，Windows Server Failover Cluster と連携する場合に注意が必要なファイルの設定について説明します。

(1)　設定が必要なファイル

設定が必要なファイルは実行系と待機系で異なります。

実行系の場合

実行系の場合に注意が必要な設定を次に示します。

- 運用管理エージェントの設定
- HTTP Server の設定
- ワーカの設定
- スマートエージェントの設定

これらのファイルの設定については，「17.4.3　設定ファイルの編集」を参照してください。なお，N:1 リカバリ構成の場合には，クラスタ IP アドレスではなく，仮想ホストの IP アドレスを設定してください。

待機系（リカバリ専用）の場合

待機系（リカバリ専用）の場合に注意が必要な設定を次に示します。

- J2EE サーバのプロパティの設定

この設定について，次に説明します。

(2) 待機系での J2EE サーバのプロパティの設定

待機系の J2EE サーバで usrconf.properties にプロパティを設定するときに注意が必要なキーを次に示します。

- ejbserver.distributedtx.XATransaction.enabled
  ライトトランザクションを無効にするため「true」を設定してください。

19.4.4 クラスタの設定

クラスタに対して，次の設定を実施します。

- スクリプトファイルの作成
- クラスタソフトウェアの環境設定

(1) スクリプトファイルの作成

実行系のアプリケーションサーバを起動，停止するためのスクリプトファイルを VBScript で記述します。実行系の各ノードで同一のスクリプトファイルを使用し，同一のパスに配置してください。

スクリプトファイルは汎用スクリプトをそのまま使用します。汎用スクリプトファイルの例については，「(2) 実行系の汎用スクリプトファイルの例」を参照してください。

ここでは，N:1 リカバリシステムのシステムの監視対象および監視方法について説明します。

(a) 監視対象

アプリケーションサーバの 1:1 系切り替えシステムの監視対象を次に示します。

- 運用管理エージェント

運用管理エージェントが終了したタイミングで系切り替えが実行されます。

(b) 監視方法

運用管理エージェントのプロセスの有無を監視します。監視するプロセスを次に示します。

- adminagent.exe

(2) 実行系の汎用スクリプトファイルの例

N:1 リカバリシステムの実行系で使用する汎用スクリプトファイルの例を次に示します。

```
Dim WshShell, fso, AccessInfoFile, oExec, oRun, oCosmiHome, oAdmin, oMngsvut, oCjsleep
Dim oMngHost, oUser, oPass
Set WshShell = CreateObject("WScript.Shell")
```

```
oCosmiHome = WshShell.ExpandEnvironmentStrings("%COSMINEXUS_HOME%")
oAdminac = """" & oCosmiHome & "¥manager¥bin¥adminagentctl"""
oMngsvut = """" & oCosmiHome & "¥manager¥bin¥mngsvrutil"""
oCjsleep = """" & oCosmiHome & "¥CC¥server¥bin¥cjsleep"""

oMngHost = "172.16.12.39" '--- Management Server
oUser = "admin" ' --- Management Server's userid
oPass = "admin" ' --- Management Server's password

'==== Open ====
Function Open( )
    Resource.LogInformation "======== Entering Open. ========"
    ' --- J2EE Server Name ---
    If Resource.PropertyExists("ServerName") = False Then
        Resource.LogInformation "### Private property ServerName is not set. ###"
    End If
    ' --- Inprocess OTS Status Path ---
    If Resource.PropertyExists("StatusPath") = False Then
        Resource.LogInformation "### Private property StatusPath is not set. ###"
    End If
    ' --- Cluster IP Address ---
    If Resource.PropertyExists("IPAddress") = False Then
        Resource.LogInformation "### Private property IPAddress is not set. ###"
    End If
    ' --- Cluster Host Name ---
    If Resource.PropertyExists("HostName") = False Then
        Resource.LogInformation "### Private property HostName is not set. ###"
    End If
    Open = True
End Function

'==== Online ====
Function Online( )
    Resource.LogInformation "======== Entering Online. ========"

    '--- private properties ---
    If Resource.PropertyExists("ServerName") = False Or _
       Resource.PropertyExists("StatusPath") = False Or _
       Resource.PropertyExists("IPAddress") = False Or _
       Resource.PropertyExists("HostName") = False Then
         Resource.LogInformation "### Private property is not set. ###"
         Online = False
         Exit Function
    End If

    '--- adminagentctl ---
    Set oExec = WshShell.Exec(oAdminac & " start")
    Do While oExec.Status = 0
        WshShell.Run oCjsleep & " 1", 0, True
    Loop
    If oExec.ExitCode <> 0 Then
        Resource.LogInformation "### Administration Agent cannot start. ### rtn=" & oResult
        Resource.LogInformation oExec.StdErr.ReadAll
        Online = False
        Exit Function
    End If
```

```
    '--- mngsvrutil ---
    oRun = oMngsvut & " -m "& oMngHost &":28080 -u " & oUser & " -p " & oPass & _
                      " -t " & Resource.HostName & " -k host -s start server"
    Set oExec = WshShell.Exec(oRun)
    Do While oExec.Status = 0
        WshShell.Run oCjsleep & " 1", 0, True
    Loop
    If oExec.ExitCode <> 0 Then
        Resource.LogInformation "### Logical Servers start failed. ### rtn=" & oResult
        Resource.LogInformation oExec.StdErr.ReadAll
        Online = False
        Exit Function
    End If
    Online = True
End Function

'==== LooksAlive ====
Function LooksAlive( )
    Resource.LogInformation "======== Entering LooksAlive. ========"

    '--- tasklist ---
    Set oExec = WshShell.Exec("tasklist /NH /FI ""IMAGENAME eq adminagent.exe""")
    Do While oExec.Status = 0
        WshShell.Run oCjsleep & " 1", 0, True
    Loop

    If InStr(1, oExec.StdOut.ReadAll, "adminagent.exe", 1) <> 0 Then
        LooksAlive = True
    Else
        LooksAlive = False
    End If
End Function

'==== IsAlive ====
Function IsAlive( )
    Resource.LogInformation "======== Entering IsAlive. ========"

    '--- mngsvrutil ---
    oRun = oMngsvut & " -m " & oMngHost & ":28080 -u " & oUser & " -p " & oPass & _
                      " -t " & Resource.HostName & " -k host check adminAgent"
    Set oExec = WshShell.Exec(oRun)
    Do While oExec.Status = 0
        WshShell.Run oCjsleep & " 1", 0, True
    Loop
    If oExec.ExitCode <> 0 Then
        Resource.LogInformation "### Administration Agent command failed. ### rtn=" & oExec.
ExitCode
        Resource.LogInformation oExec.StdErr.ReadAll
        IsAlive = False
    Else
        IsAlive = True
    End If
End Function

'==== Offline ====
Function Offline( )
    Resource.LogInformation "======== Entering Offline. ========"
```

```
    '--- mngsvrutil ---
    oRun = oMngsvut & " -m " & oMngHost & ":28080 -u " & oUser & " -p " & oPass & _
                      " -t " & Resource.HostName & " -k host -s stop server"
    Set oExec = WshShell.Exec(oRun)
    Do While oExec.Status = 0
        WshShell.Run oCjsleep & " 1", 0, True
    Loop
    If oExec.ExitCode <> 0 Then
        Resource.LogInformation "### Logical Server stopping failed. ### rtn=" & oResult
        Resource.LogInformation oExec.StdErr.ReadAll
        Offline = False
        Exit Function
    End If

    '--- adminagentctl ---
    Set oExec = WshShell.Exec(oAdminac & " stop")
    Do While oExec.Status = 0
        WshShell.Run oCjsleep & " 1", 0, True
    Loop
    If oExec.ExitCode <> 0 Then
        Resource.LogInformation "### Administration Agent stopping failed. ### rtn=" & oResu
lt
        Resource.LogInformation oExec.StdErr.ReadAll
        Offline = False
        Exit Function
    End If
    Offline = True
End Function

'==== Close ====
Function Close( )
    Resource.LogInformation "======== Entering Close. ========"
    Close = True
End Function

'==== Terminate ====
Function Terminate( )
    Resource.LogInformation "======== Entering Terminate. ========"
    Terminate = True
End Function
```

実行系の汎用スクリプトファイルでは，必要に応じて変数の値を変更してください。汎用スクリプトファイルで定義する変数については，「(a)　汎用スクリプトファイルの変数」を参照してください。

また，N:1 リカバリシステムの汎用スクリプトファイルでは，一つのスクリプトファイルを複数の仮想サーバから参照します。このため，汎用スクリプトファイルがサーバごとの設定を参照できるように，プライベートプロパティを使用します。仮想サーバごとに設定するプライベートプロパティについては，「(3) プライベートプロパティの設定」を参照してください。

(a)　汎用スクリプトファイルの変数

実行系の汎用スクリプトファイルでは，必要に応じて変数の値を変更してください。次の表に示す変数の値を変更してください。

表 19-2　実行系の汎用スクリプトファイルの変数（N:1 リカバリシステムの場合）

変数名	内容
oMngHost	Management Server が稼働しているホスト
oUser	Management Server にログインするためのログイン ID
oPass	Management Server にログインするためのパスワード

(3)　プライベートプロパティの設定

仮想サーバごとにプライベートプロパティを設定してください。汎用スクリプトファイルに必要なプライベートプロパティを次の表に示します。

表 19-3　汎用スクリプトファイルに必要なプライベートプロパティ

プライベートプロパティ	内容（例）
ServerName	J2EE サーバ名 指定例：MyServer
StatusPath	インプロセス OTS のステータスパス 指定例：I:¥otsstatus
IPAddress	仮想サーバの IP アドレス 指定例：172.16.12.31
HostName	仮想サーバ名 指定例：Server1

プライベートプロパティは cluster コマンドで設定します。cluster コマンドについては，使用している OS のマニュアルを参照してください。次に cluster コマンドを使用したプライベートプロパティの設定例を示します。

```
cluster res Script1 /priv ServerName=MyServer
cluster res Script1 /priv StatusPath="I:¥otsstatus"
cluster res Script1 /priv IPAddress=172.16.12.31
cluster res Script1 /priv HostName=Server1

cluster res Script2 /priv ServerName=MyServer
cluster res Script2 /priv StatusPath="J:¥otsstatus"
cluster res Script2 /priv IPAddress=172.16.12.32
cluster res Script2 /priv HostName=Server2

cluster res Script3 /priv ServerName=MyServer
cluster res Script3 /priv StatusPath="K:¥otsstatus"
cluster res Script3 /priv IPAddress=172.16.12.33
cluster res Script3 /priv HostName=Server3

cluster res Script4 /priv ServerName=MyServer2
cluster res Script4 /priv StatusPath="L:¥otsstatus"
```

```
cluster res Script4 /priv IPAddress=172.16.12.34
cluster res Script4 /priv HostName=Server4
```

(4) クラスタソフトウェアの環境設定

Windows Server Failover Cluster の環境設定については，使用するクラスタソフトウェアのマニュアルを参照してください。

19.4.5 アプリケーションサーバの設定

アプリケーションサーバの設定では，運用管理ポータル，クラスタアドミニストレータなどを使用してアプリケーションサーバをクラスタ構成に配置します。また，論理サーバをセットアップして設定情報を配布し，J2EE サーバに J2EE アプリケーションとリソースアダプタをインポートします。アプリケーションサーバの設定手順を次に示します。

1. 各実行系ノードで，各リソースグループをオンラインにします。

   Node1 の Group1，Node2 の Group2，Node3 の Group3，および Node4 の Group4 をオンラインにします。

2. 運用管理ポータルで，各実行系の J2EE サーバをセットアップします。

   「運用管理ドメインの構成定義」の［セットアップ］画面で，J2EE サーバをセットアップします。運用管理ポータルでの操作手順および画面の詳細については，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。

3. 運用管理ポータルで，J2EE サーバに設定した情報を実行系ノードに配布します。

   「論理サーバの環境設定」の［設定情報の配布］画面で，J2EE サーバに設定した情報を各実行系に配布します。

4. サーバ管理コマンドを使用して，各実行系のアプリケーションサーバの J2EE サーバに J2EE アプリケーションとリソースアダプタをインポートします。また，インポートした J2EE アプリケーションとリソースアダプタを設定および開始します。

   J2EE 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 システム構築・運用ガイド」の「4.1.29　業務アプリケーションを設定して開始する（CUI 利用時）」，リソースアダプタの設定については，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」の次の個所を参照してください。

   - 4.1.26　DB Connector を設定する（CUI 利用時）
   - 4.1.27　DB Connector 以外のリソースアダプタを設定する（CUI 利用時）
   - 4.1.28　リソースアダプタを開始する（CUI 利用時）

   また，サーバ管理コマンドでの操作については，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。

なお，J2EE サーバおよびその前提プロセスを起動してからサーバ管理コマンドを実行してください。また，サーバ管理コマンドでの操作が終わったら，サーバ管理コマンドを停止してください。

5. クラスタアドミニストレータで，各リソースグループに対して「グループの移動」を実行して系を切り替え，リソースの所有者を待機系ノード（NodeR）に変更します。

6. サーバ管理コマンドを使用して，待機系のアプリケーションサーバにリソースアダプタをインポートします。

   待機系の J2EE サーバには，実行系の各 J2EE サーバで使用しているリソースアダプタと同じ設定のリソースアダプタをインポートしてください。

   ただし，待機系の J2EE サーバに定義するリソースアダプタでは，コネクションプーリング機能を使用する必要はありません。サーバ管理コマンドで設定するコネクションの最小値と最大値には 0 を指定してください。

7. クラスタアドミニストレータで，各リソースグループに対して「グループの移動」を実行して系を切り替え，リソースの所有者を各実行系ノードに戻します。

19.5　N:1 リカバリシステムの設定（UNIX の場合）

N:1 リカバリシステムは，N 台の実行系に対して 1 台の待機系（**リカバリ専用サーバ**）を用意する構成のシステムです。J2EE サーバを冗長化した構成でグローバルトランザクションを使用する場合に，特定の J2EE サーバにトラブルが発生したとき，トランザクションを決着するために使用します。このシステムは，運用管理ポータルを使用して構築します。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。この節では，N:1 リカバリシステムの設定について説明します。

業務は，N 台の実行系のアプリケーションサーバで実行します。業務実行中のアプリケーションサーバに障害が発生すると，HA モニタがこれを検知して，リカバリ専用サーバで，障害が発生したアプリケーションサーバの仕掛かり中のトランザクションを決着します。その後，残りの稼働中の実行系のアプリケーションサーバで業務を続行します。

なお，N:1 リカバリシステムの運用は，AIX または Linux の場合だけ使用できます。

機能の詳細については，「19.2　N:1 リカバリシステムの概要」を参照してください。システム構成については，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「3.11.5　リカバリ専用サーバを使用する場合の構成（N:1 リカバリシステム）」を参照してください。また，HA モニタについては，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

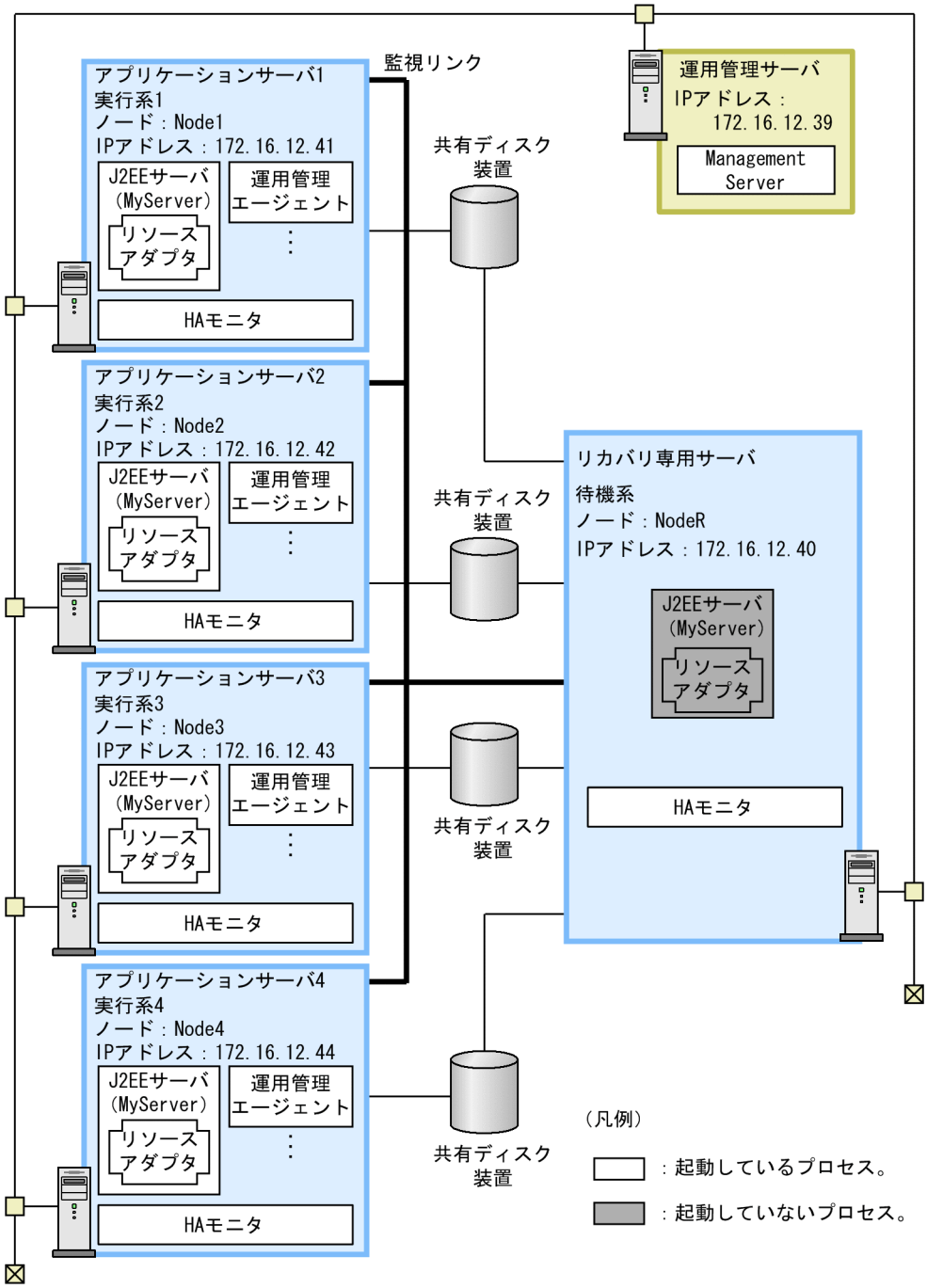
19.5.1　N:1 リカバリシステムの設定手順

ここでは，システムの構成例とシステムの設定手順について説明します。また，実行系を追加または削除する場合の設定手順についても説明します。

(1)　システムの構成例

N:1 リカバリシステムの構成例を次の図に示します。なお，以降の項では，このシステムの構成例を使用したシステムの構築例を示します。

図 19-6　N:1 リカバリシステムの構成例（UNIX の場合）



この例の概要を次に示します。

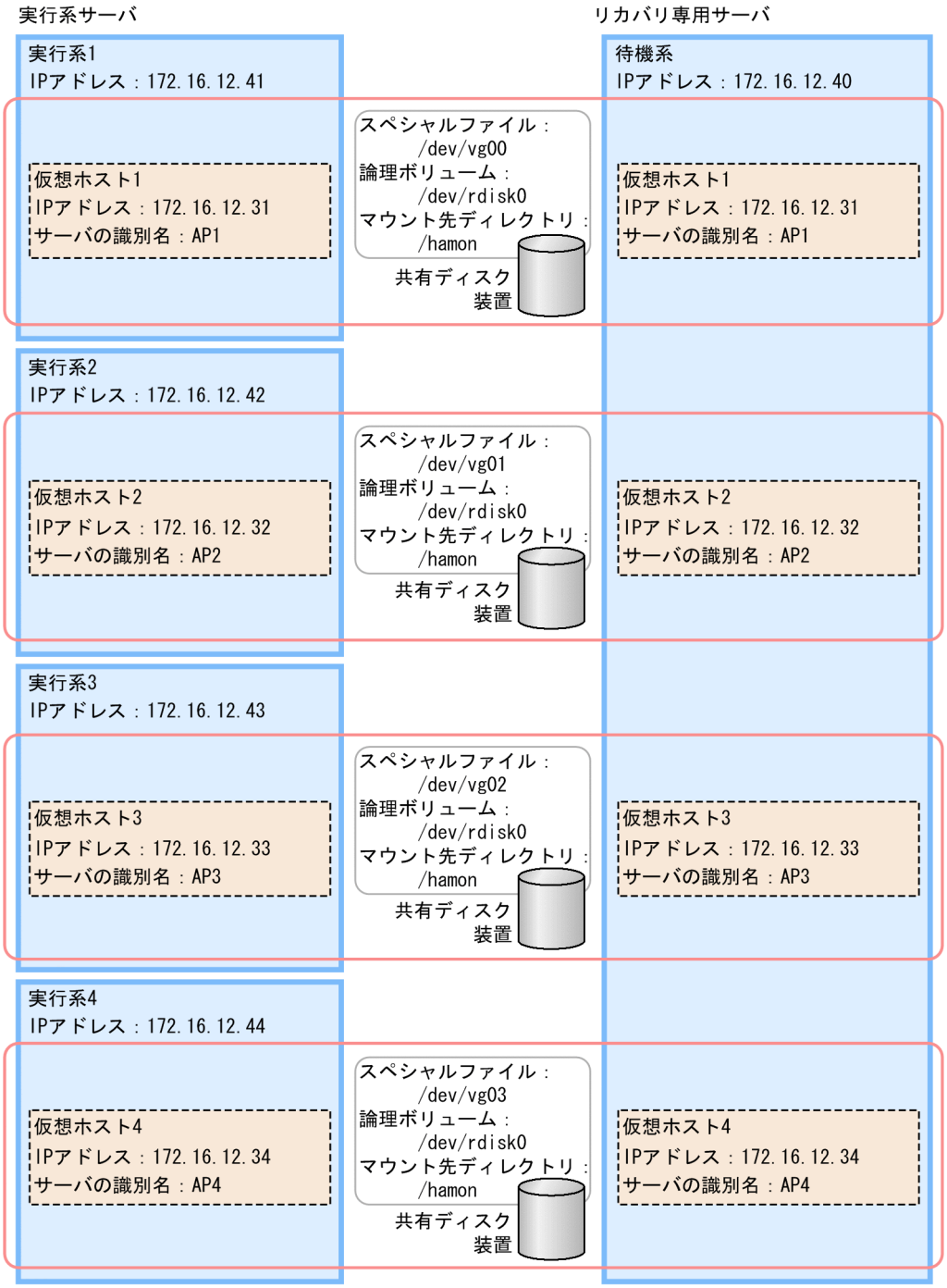
- 4 台の実行系サーバと 1 台のリカバリ専用サーバを配置しています。4 台の実行系の J2EE サーバの構成は J2EE アプリケーションとリソースアダプタの設定が均質な構成とします。違いはエイリアス IP アドレスとマウントする共有ディスクデバイスです。
- 実行系と待機系で，J2EE サーバは同じ名称にします。ここでは，実行系で「MyServer」という J2EE サーバを配置しているため，待機系にも「MyServer」という J2EE サーバを配置します。
- Management Server の運用管理ポータルを利用してシステムを構築，運用管理します。
- 実行系では運用管理エージェントを動作させます。待機系では運用管理エージェントを動作させません。

なお，N:1 リカバリシステムを運用する場合は，次の条件を満たすアプリケーションサーバを構築してください。

- OTS を使用したグローバルトランザクションを使用します。トランザクションのステータスファイルは共有ディスク装置に格納します。
- ネーミングサービスはインプロセスで動作させます。
- 待機系の J2EE サーバには，実行系の J2EE サーバで使用しているすべてのリソースアダプタと同じ設定のリソースアダプタをインポート，およびデプロイします。
- N:1 リカバリ構成の場合，一つのクラスタ内に複数の仮想ホストを構築します。仮想ホストは実行系とリカバリ専用サーバの組で構成し，J2EE サーバをクラスタ構成で配置します。N:1 リカバリ構成の場合，J2EE サーバはクラスタ内ではなく，仮想ホスト内で動作します。このため，仮想ホストの IP アドレスおよびホスト名を設定してください。

  アプリケーションサーバをクラスタ構成に配置するためには，エイリアス IP アドレスを設けて，稼働中のノードがエイリアス IP アドレスを引き継ぐことで，クライアントがクラスタ内のノードを意識しないようにします。N:1 リカバリ構成の場合，エイリアス IP アドレスは，仮想ホストの IP アドレスとなります。N:1 リカバリ構成での仮想ホストの構成例を次に示します。

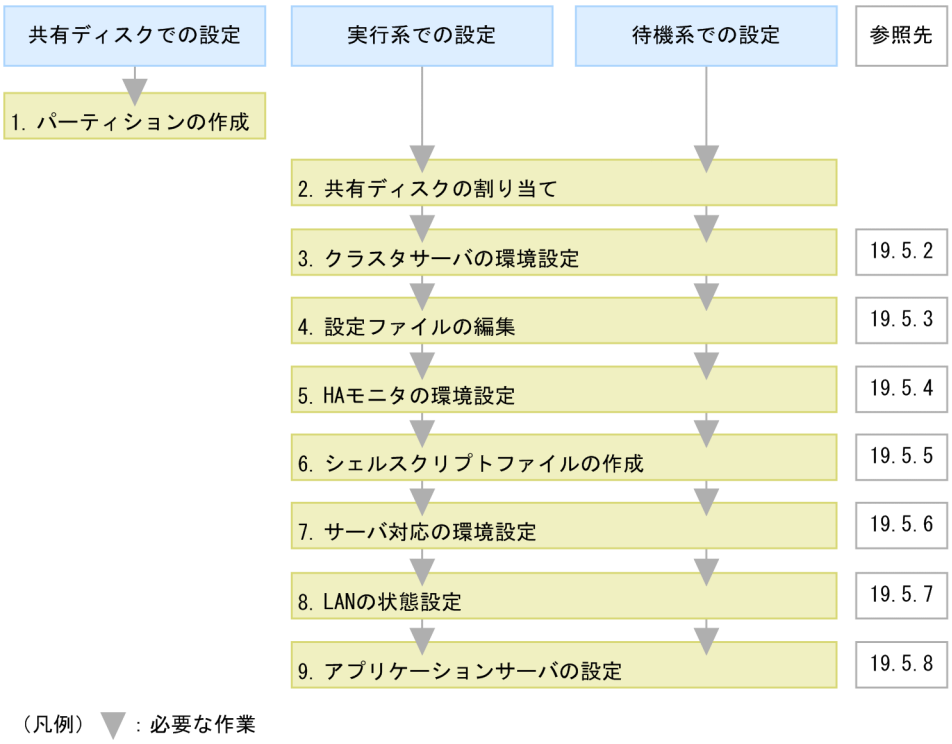
図 19-7　N:1 リカバリ構成での仮想ホストの構成例



(2)　システムの設定手順

HA モニタと連携する場合には，運用管理ポータルや HA モニタのファイルの設定などが必要になります。N:1 リカバリシステムの設定手順を次の図に示します。

図 19-8　N:1 リカバリシステムの設定手順（UNIX の場合）



図中の 1.～9.について説明します。

1. 共有ディスクに N 個のパーティションを作成して，ファイルシステムを構築します。

2. システムに共有ディスクを割り当てます。

   システムに共有ディスクを割り当てる際は，現用系と予備系で，同じマウント先ディレクトリにしてください。

3. 運用管理ポータルでクラスタサーバの環境を設定します。

   運用管理ポータルの「Cosminexus Management Server の設定」，「運用管理ドメインの構成定義」，および「論理サーバの環境設定」で，HA モニタを使用する場合の設定をします。詳細は，「19.5.2 クラスタサーバの環境設定」を参照してください。

4. 設定ファイルを編集します。

   運用管理エージェントや Management Server，HTTP Server などの各種定義ファイルを設定します。詳細は，「19.5.3　設定ファイルの編集」を参照してください。

5. HA モニタの環境を設定します。

   HA モニタの sysdef ファイルで，HA モニタの環境を定義します。詳細は，「19.5.4　HA モニタの環境設定」を参照してください。

6. シェルスクリプトファイルを作成します。

   運用管理エージェントを監視，アプリケーションサーバ上の論理サーバを起動，停止するためのシェルスクリプトファイルを作成します。詳細は，「19.5.5　シェルスクリプトファイルの作成」を参照してください。

7. サーバ対応の環境を設定します。

HA モニタの servers ファイルで，系で稼働させる現用系サーバや予備系サーバの環境を定義します。詳細は，「19.5.6　サーバ対応の環境設定」を参照してください。

8. LAN の状態を設定します。

HA モニタの LAN の状態設定ファイルで，LAN アダプタの IP アドレスなどを指定して HA モニタでの LAN の切り替えについて定義します。詳細は，「19.5.7　LAN の状態設定」を参照してください。

9. 運用管理ポータルと HA モニタのコマンドで，アプリケーションサーバの設定をします。

運用管理ポータル，HA モニタのコマンドを使用して，アプリケーションサーバをクラスタ構成に配置し，J2EE アプリケーションやリソースアダプタを設定します。詳細は，「19.5.8　アプリケーションサーバの設定」を参照してください。

参考

- N:1 リカバリシステムで，待機系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は，手動でリカバリを実行する必要があります。

  タイムアウトの原因を解消したあと，待機系のホストで，停止した実行系のホストに対応する monbegin コマンドを実行し，停止した実行系のホストに対応する monact コマンドを実行します。

  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 保守／移行編」の「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」を参照してください。

N:1 リカバリシステムで設定が必要なファイルは，実行系と待機系で異なります。N:1 リカバリシステムでのファイルの設定の要否を次の表に示します。

表 19-4　N:1 リカバリシステムでのファイルの設定の要否

分類		ファイルの種類	ファイルの設定の要否		参照先
			実行系（N 台）	待機系（リカバリ専用 1 台）	
アプリケーションサーバの設定	運用管理エージェントの設定	adminagent.properties	必要（N 台で個別の設定）	−	19.5.3
	スマートエージェントの設定	localaddr			
	HTTP Server の設定	httpsd.conf※			
	J2EE サーバのプロパティの設定	usrconf.properties※	必要（N 台で共通の設定）	必要（エイリアス IP アドレス以外は実行系とほとんど同じ）	

分類		ファイルの種類	ファイルの設定の要否		参照先
			実行系（N 台）	待機系（リカバリ専用 1 台）	
HA モニタの設定	HA モニタの環境設定	sysdef	必要（各ホストで個別の設定）	必要（N 台分の設定）	19.5.4
	シェルスクリプトファイルの作成	monitor.sh start.sh stop.sh	必要（N 台で共通のコマンドを使用）	－	19.5.5
	サーバ対応の環境設定	servers	必要（N 台で個別の設定）	必要（N 台分の設定）	19.5.6
		recover.sh	－	必要（N 台の実行系に対して共通のコマンドを使用。差異は引数で渡す）	
	LAN の状態設定	<サーバの識別名>.up <サーバ識別名>.down	必要（各ホストで個別の設定）	必要（実行系の設定ファイルのコピー）	19.5.7

（凡例） －：不要。

注※　運用管理ポータルを利用してアプリケーションサーバの実行環境を構築するときに，これらのファイルを設定できます。

(3)　実行系を追加または削除する場合の設定手順

N:1 リカバリシステムで，1 台の実行系を追加または削除（縮退）する場合の設定手順を次に示します。

1 台の実行系を追加する場合

1. 追加する実行系で，マウントディレクトリを作成します。
2. 待機系の servers ファイルに追加する実行系に対応したエントリを追加します。
   なお，追加する実行系に対応したエントリがすでに存在する場合はこの手順は不要です。
3. 待機系で，追加した実行系に対応する monbegin コマンドを実行します。
4. 実行系で，monbegin コマンドを実行します。
5. 運用管理ポータルを利用して，追加する実行系にアプリケーションサーバの実行環境を構築します。
6. 運用管理ポータルを使用して，一括起動で実行系のアプリケーションサーバを起動します。

1 台の実行系を削除する場合

1. 実行系で，monend コマンドを実行します。

19.5.2　クラスタサーバの環境設定

HA モニタと連携する場合の，運用管理ポータルでの設定の留意点については，「17.6.2　クラスタサーバの環境設定」を参照してください。なお，N:1 リカバリ構成の場合，ホスト名には，エイリアス IP アドレスではなく，仮想ホストの IP アドレスを設定してください。

19.5.3　設定ファイルの編集

運用管理エージェントや Management Server，HTTP Server などの各種設定ファイルを編集します。ここでは，HA モニタと連携する場合に注意が必要なファイルの設定について説明します。

(1)　設定が必要なファイル

設定が必要なファイルは実行系と待機系で異なります。

実行系（N 台）の場合

実行系（N 台）の場合に注意が必要な設定ファイルについて説明します。

ファイルを直接編集して設定できるファイル

次のファイルは，ファイルを直接編集して設定できます。

- 運用管理エージェントの設定（adminagent.properties）
- スマートエージェントの設定（localaddr）

運用管理ポータルで設定できるファイル

次のファイルは，運用管理ポータルを利用してアプリケーションサーバの実行環境を構築するときに設定できます。

- HTTP Server の設定（httpsd.conf）
- J2EE サーバのプロパティの設定（usrconf.properties）

これらのファイルの設定については，「17.6.3　設定ファイルの編集」を参照してください。なお，N:1 リカバリ構成の場合，ホスト名には，エイリアス IP アドレスではなく，仮想ホストの IP アドレスを設定してください。

待機系（リカバリ専用 1 台）の場合

待機系（リカバリ専用 1 台）の場合に注意が必要な設定ファイルを次に示します。なお，このファイルは，ファイルを直接編集して設定できます。

- J2EE サーバのプロパティの設定（usrconf.properties）

(2)　待機系での J2EE サーバのプロパティの設定

待機系の J2EE サーバで usrconf.properties にプロパティを設定するときに注意が必要なキーを次に示します。

- ejbserver.distributedtx.XATransaction.enabled
  ライトトランザクションを無効にするため「true」を設定してください。

19.5.4 HA モニタの環境設定

使用しているシステムに従って，sysdef という定義ファイルに HA モニタの環境を定義してください。N 台の実行系とリカバリ専用の 1 台の待機系を含めて各ホストで個別の設定が必要です。

name（HA モニタのホスト名），address（CPU リセット用のユニークな値），lan（LAN のホスト名）をホスト環境に合わせて一意に設定してください。なお，N 台の実行系とリカバリ専用の 1 台の待機系を含めて一意の値を設定する必要があります。また，lanport（lan オペランドで指定したホスト名に対応するサービス名）は，N 台の実行系とリカバリ専用の 1 台の待機系を含めて同じ値を設定する必要があります。

sysdef ファイルの例を次に示します。なお，サンプル中の「N」は，N 番目の値であることを意味します。

```
environment name     $host_N,     #自系のホスト名
            address  $uniq_N,     #CPUリセット用のユニークな値
            patrol   5,           #系障害と判断する時間
            lan      $host_N      #LANのホスト名
            lanport  $service_N   #lanオペランドのホスト名に対応するサービス名
```

HA モニタの環境設定については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

19.5.5 シェルスクリプトファイルの作成

運用管理エージェントのプロセスを監視し，運用管理エージェントと論理サーバを起動および停止するために，次のシェルスクリプトファイルを作成します。

実行系（N 台）の場合

実行系で設定が必要なシェルスクリプトファイルを次に示します。なお，N 台で共通のコマンドを使用します。

- 運用管理エージェントのプロセスを監視するシェルスクリプトファイル
- 運用管理エージェントを起動するシェルスクリプトファイル
- 運用管理エージェントと論理サーバを停止するシェルスクリプトファイル

待機系（リカバリ専用 1 台）の場合

待機系で設定が必要なシェルスクリプトファイルを次に示します。なお，N 台の実行系に対して共通のコマンドを使用し，差異はコマンドの引数で渡すようにします。

- リカバリ処理を実行するシェルスクリプトファイル

各スクリプトについて説明します。

(1) 運用管理エージェントのプロセスを監視するシェルスクリプトファイル

運用管理エージェントのプロセスを監視するシェルスクリプトファイルの例（monitor.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
AA=/opt/Cosminexus/manager/bin/adminagent

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

logg "### $0: started. ###"
while true
do
    CHECK=`ps -ef | grep $AA | grep -v grep`

    if [ "$CHECK" = "" ]
    then
        logg "### $0: stop. ###"
        exit 0
    fi
    sleep 10
done
```

この例では，運用管理エージェントのプロセスが存在しているかどうかを 10 秒おきに監視しています。

(2) 運用管理エージェントを起動するシェルスクリプトファイル

運用管理エージェントを起動するためのシェルスクリプトファイルの例（start.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
MNGDIR=/opt/Cosminexus/manager

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

# start Administration Agent
logg "### $0: starting Administration Agent. ###"
$MNGDIR/bin/adminagentctl start
if [ $? -eq 0 ] ; then
  logg "### $0: Administration Agent start normally. ###"
else
  logg "### $0: Administration Agent cannot start. ###"
```

```
  exit 1
fi

exit 0
```

(3) 運用管理エージェントと論理サーバを停止するシェルスクリプトファイル

運用管理エージェントと論理サーバを停止するためのシェルスクリプトファイルの例（stop.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
MNGDIR=/opt/Cosminexus/manager

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

# stop logical server
logg "### $0: stop logical servers. ###"
$MNGDIR/bin/mngsvrutil -m mnghost:28080 -u admin -p admin ¥
                           -t 172.16.12.31 -k host -s stop server

# stop Administration Agent
logg "### $0: stopping Administration Agent. ###"
$MNGDIR/bin/adminagentctl stop

exit 0
```

シェルスクリプトファイルでの設定内容のポイントを説明します。

- **論理サーバの停止**

  同一ホスト上の論理サーバを一括停止するように設定します。論理サーバの一括停止には mngsvrutil コマンドを使用します。

  ユーザ名とパスワードは引数として渡しています。これらの値は，mngsvrutil コマンドのクライアント側定義ファイル（.mngsvrutilrc）に記述して，適切なアクセス権を設定して管理するようにしてください。

  また，mngsvrutil コマンドの-t オプションにはエイリアス IP アドレスを指定してください。

  なお，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。また，クライアント側定義ファイル（.mngsvrutilrc）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「8.2.14　.mngsvrutilrc（mngsvrutil コマンドのクライアント側定義ファイル)」を参照してください。

- **運用管理エージェントの停止**

運用管理エージェントを停止するように設定します。運用管理エージェントの停止には，adminagentctl stop を使用します。

なお，adminagentctl 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「adminagentctl（運用管理エージェントの起動と停止)」を参照してください。

(4) リカバリ処理を実行するシェルスクリプトファイル

各 J2EE サーバに対してリカバリ処理を実行するためのシェルスクリプトファイルの例（recover.sh）を次に示します。

```
#!/bin/sh

LOGDIR=/home/manager/hamon/log
PATH=/opt/Cosminexus/CC/server/bin:/bin:/usr/bin:/home/manager/hamon/bin
LD_LIBRARY_PATH=/opt/DABroker/lib:/opt/Cosminexus/jdk/lib:/opt/Cosminexus/TPB/lib:/opt/Cosminexus/PRF/lib:/opt/Cosminexus/CTM/lib:/bin:/opt/HiRDB/client/lib:/opt/oracle/app/oracle/product/10.1.0/client_1/lib:/opt/oracle/app/oracle/product/10.1.0/client_1/lib/libclntsh.so.10.1:/opt/oracle/app/oracle/product/10.1.0/client_1/lib/libclntsh.so
export LD_LIBRARY_PATH

LOCKFILE=/var/lock/kosmi_recover.lock
SLEEP_TIME=60
RETRIES=30
STATUS_PATH=$3/otsstatus

cjlockfile()
{
    counter=$2
    TMP_LOCK_FILE=`dirname $3`/$$.lock
    echo $$ > $TMP_LOCK_FILE

    if [ -f $3 ]
    then
        kill -0 `cat $3` 2>/dev/null || rm -f $3
    fi

    until ln $TMP_LOCK_FILE $3 2>/dev/null
    do
        counter=`expr $counter - 1`
        if [ $counter -le 0 ]
        then
            rm -f $TMP_LOCK_FILE
            return 1
        fi
        sleep $1
    done

    rm -f $TMP_LOCK_FILE
    return 0
}

logg()
{
    echo `date '+[%Y/%m/%d %H:%M:%S]'`"[$$]: $1" ¥
```

```
        >> ${LOGDIR}/adminagent.log 2>&1
}

if cjlockfile ${SLEEP_TIME} ${RETRIES} ${LOCKFILE}
then
    logg "### $0: started for $1 ###"
    cjstartrecover MyServer -p vbroker.se.iiop_tp.host=$2 ¥
                   -p ejbserver.distributedtx.ots.status.directory1=$STATUS_PATH ¥
                   -t 600
    logg "### $0: ended. $? ###"
fi

rm -f $LOCKFILE

exit 0
```

シェルスクリプトファイルでの設定内容のポイントを説明します。

- リカバリ処理に必要で，実行系ごとに異なる情報（エイリアス名，エイリアス IP アドレス，マウントディレクトリ）は，servers ファイルの待機系の actcommand で，コマンドの引数として渡します。
- LD_LIBRARY_PATH などの環境変数を明示的に指定する必要があります。環境変数については，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「4.1.10　環境変数とは」を参照してください。
- cjlockfile 関数を利用して，リカバリ処理が排他的に実行されるようにしています。この関数は，ファイルロックによるセマフォを提供しています。第 1 引数にスリープ時間（秒単位），第 2 引数にリトライ回数，第 3 引数にロックファイルを指定します。この関数では，第 3 引数で指定されたファイルがロックされている場合には，第 1 引数で指定した時間だけ待って再度ファイルロックの取得を試みます。第 2 引数に指定した回数分リトライしてもファイルロックが取得できない場合には，リカバリ処理に失敗します。ファイルをロックしているプロセスが存在しない場合は，ロックを横取りします。
- J2EE サーバ（MyServer）に対して cjstartrecover でリカバリ処理を実行します。
  - このサンプルでは，タイムアウト時間は 600 秒にしています。
  - cjstartrecover コマンドの-p オプションで，ejbserver.distributedtx.ots.status.directory1 キーに停止した実行系のステータスファイルディレクトリが指定されるようにしています。
  - cjstartrecover コマンドの-p オプションで，vbroker.se.iiop_tp.host キーに系切り替え後のエイリアス IP アドレスが指定されるようにしています。
  - 停止した実行系ごとに異なるプロパティを cjstartrecover コマンドの-p オプションで上書き指定しています。
- J2EE サーバが複数の構成の場合には，cjstartrecover コマンドを J2EE サーバの数だけ実行します。その際，J2EE サーバ名（MyServer）と OTS のステータスファイルディレクトリのパスは適宜修正してください。

なお，cjstartrecover 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「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。
ejbserver.distributedtx.ots.status.directory1 キーおよび vbroker.se.iiop_tp.host キーについては，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。

19.5.6　サーバ対応の環境設定

HA モニタでのサーバ対応の環境設定では，系で稼働させる実行サーバや待機サーバの環境を定義します。

実行系（N 台）の場合

N 台で個別の設定が必要です。

待機系（リカバリ専用 1 台）の場合

N 台分の実行系の設定が必要です。

servers という定義ファイルに，N:1 リカバリシステム用のサーバ対応の環境を定義してください。サーバ対応の環境設定での設定内容を次の表に示します。

表 19–5　サーバ対応の環境設定での設定内容（N:1 リカバリシステムの場合）

オペランド	設定内容
name	• 実行系の場合 起動コマンドを N 台で共通化するため，name にはユニークな名前を指定し，actcommand に運用管理エージェントを起動するシェルスクリプトファイルを記述します。 • 待機系の場合 対応する実行系と同じ値を指定します。
alias	• 実行系の場合 サーバの識別名を指定します。 指定例：AP1 • 待機系の場合 対応する実行系と同じ値を指定します。
acttype	サーバの起動方法を指定します。ここでは，HA モニタのコマンドでサーバを起動するため，「monitor」を指定します。
termcommand	• 実行系の場合 実行系の運用管理エージェントと論理サーバを停止するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/stop.sh • 待機系の場合 待機系の運用管理エージェントと論理サーバを停止するシェルスクリプトファイルを指定します。
initial	サーバ起動時の状態を指定します。 • 実行系の場合 「online」を指定します。 • 待機系の場合 「standby」を指定します。

オペランド	設定内容
disk	共有ディスク装置のキャラクタ型スペシャルファイル名を指定します。 • 実行系の場合 割り当てられた共有ディスクに合わせて指定します。 • 待機系の場合 対応する実行系と同じ値を指定します。
lan_updown	LAN の状態設定ファイルを使用するかどうかを指定します。ここでは，LAN の状態設定ファイルを使用するため，「use」を指定します。
fs_name	切り替えるファイルシステムに対応する論理ボリュームの絶対パス名を指定します。 • 実行系の場合 割り当てられた共有ディスクに合わせて指定します。 • 待機系の場合 対応する実行系と同じ値を指定します。
fs_mount_dir	切り替えるファイルシステムのマウント先ディレクトリの絶対パス名を指定します。 • 実行系の場合 割り当てられた共有ディスクに合わせて指定します。 • 待機系の場合 対応する実行系ごとに異なるマウントディレクトリとします。存在するディレクトリでなくてはならないので，待機系の構築時にあらかじめ作成しておく必要があります。
patrolcommand	• 実行系の場合 運用管理エージェントのプロセスを監視するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/monitor.sh • 待機系の場合 monend コマンドを指定して，actcommand でリカバリが終了したら HA モニタを終了させるようにします。
servexec_retry	実行系で障害を検出した場合の再起動の回数を指定します。実行系および待機系で同じ値を指定してください。
waitserv_exec	実行系および待機系で，起動完了処理を実行するときに起動コマンドの実行完了を待つようにするため，「yes」を指定します。
actcommand	• 実行系の場合 運用管理エージェントを起動するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/start.sh • 待機系の場合 リカバリ処理を実行するシェルスクリプトファイルを指定します。 指定例：/home/manager/hamon/bin/recover.sh

servers ファイルのサンプルを次に示します。なお，サンプル中の「N」や「N+1」は，N 番目または N+1 番目の値であることを意味します。

servers ファイルのサンプル（実行系の場合）

このサンプル中で下線の引いてある個所は，実行系ごとに内容が異なる個所であることを示します。

```
server name             $name_N,
       alias            $alias_N,
       acttype          monitor,
       initial          online,
       termcommand      /home/manager/hamon/bin/stop.sh,
       disk             $disk_N,
       lan_updown       use,
       fs_name          $fs_name_N,
       fs_mount_dir     /hamon,
       patrolcommand    /home/manager/hamon/bin/monitor.sh,
       servexec_retry   3,
       waitserv_exec    yes,
       actcommand       /home/manager/hamon/bin/start.sh;
```

servers ファイルのサンプル（待機系の場合）

このサンプル中では，リカバリ処理を実行するコマンドを一つにするため，実行系ごとに異なる情報（エイリアス名，エイリアス IP アドレス，マウントディレクトリ）は引数として渡しています。また，このサンプル中で下線の引いてある個所は，実行系の servers ファイルと内容が異なる個所であることを示します。

```
server name             $name_N,
       alias            $alias_N,
       acttype          monitor,
       initial          standby,
       termcommand      /home/manager/hamon/bin/stop.sh,
       disk             $disk_N,
       lan_updown       use,
       fs_name          $fs_name_N,
       fs_mount_dir     $mnt_N,
       patrolcommand    "/opt/hitachi/HAmon/bin/monend $alias_N",
       servexec_retry   3,
       waitserv_exec    yes,
       actcommand       "/home/manager/hamon/bin/recover.sh $alias_N $ip_N $mnt_N";

server name             $name_N+1,
       alias            $alias_N+1,
       acttype          monitor,
       initial          standby,
       termcommand      /home/manager/hamon/bin/stop.sh,
       lan_updown       use,
       disk             $disk_N+1,
       fs_name          $fs_name_N+1,
       fs_mount_dir     $mnt_N+1,
       patrolcommand    "/opt/hitachi/HAmon/bin/monend $alias_N+1",
       servexec_retry   3,
       waitserv_exec    yes,
       actcommand       "/home/manager/hamon/bin/recover.sh $alias_N+1 $ip_N+1 $mnt_N+1";
```

待機系の servers ファイルの設定内容のポイントについて説明します。

- 待機系の servers ファイルには，N 台分の実行系の設定を記述してください。また，実行系に対応したエントリを記述してください。
- リカバリ処理を実行するコマンドを一つにするため，実行系ごとに異なる情報（エイリアス名，エイリアス IP アドレス，マウントディレクトリ）は引数として渡すことをお勧めします。

19.5.7　LAN の状態設定

HA モニタの LAN の状態設定ファイルで，LAN アダプタの IP アドレスなどを指定して HA モニタでの LAN の切り替えについて定義します。

次のファイルにエイリアス IP アドレスを設定してください。

- <サーバ識別名>.up ファイル
  LAN を接続する場合に使用します。LAN アダプタに追加するエイリアス IP アドレスを指定します。
- <サーバ識別名>.down ファイル
  LAN を切り離す場合に使用します。LAN アダプタから削除するエイリアス IP アドレスを指定します。

<サーバ識別名>には，サーバ対応の環境設定（servers ファイル）の alias の値を指定してください。

実行系および待機系で，それぞれ LAN の状態設定ファイルを作成してください。

実行系（N 台）

エイリアス IP アドレスを設定・削除するスクリプトを作成します。ホストごとにエイリアス IP アドレスに合わせて設定してください。

待機系（リカバリ専用 1 台）

N 台の実行系ホストの.up ファイルと.down ファイルを用意します。論理ネットワークデバイスは，N 台の実行系ホストごとにユニークな値を指定してください。

LAN の状態設定の詳細については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

Linux の場合のファイルの作成例を次に示します。このほかの OS の場合の作成例については，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

<サーバの識別名>.up ファイルの例

```
#!/bin/sh
/sbin/ifconfig eth0:1 inet 172.16.12.31 ¥
                          netmask 255.255.255.0 broadcast 10.209.112.255 up
/sbin/arping -U -c 2 -I eth0 172.16.12.31
```

<サーバの識別名>.down ファイルの例

```
#!/bin/sh
/sbin/ifconfig eth0:1 down
```

19.5.8　アプリケーションサーバの設定

アプリケーションサーバの設定では，運用管理ポータルと HA モニタのコマンドを使用してアプリケーションサーバをクラスタ構成に配置します。また，論理サーバをセットアップして設定情報を配布し，J2EE サーバに J2EE アプリケーションとリソースアダプタをインポート，および開始します。アプリケーションサーバの設定手順を次に示します。

1. 待機系ノードで，リカバリ用の J2EE サーバを起動します。

   cjstartsv コマンドを使用して，待機系（NodeR）のリカバリ用の J2EE サーバを起動します。

2. 待機系ノードのリカバリ用の J2EE サーバで，リソースアダプタを設定します。

   待機系では，実行系で使用しているすべてのリソースアダプタと同じ設定のリソースアダプタをインポート，デプロイ，および開始状態にしてください。ただし，待機系の J2EE サーバに定義するリソースアダプタでは，コネクションプーリング機能を使用する必要はありません。サーバ管理コマンドで設定するコネクションの最小値と最大値は 0 を指定してください。

   リソースアダプタのインポート，デプロイ，および開始には，サーバ管理コマンドを使用します。サーバ管理コマンドでの操作については，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。

3. リソースアダプタの設定が完了したら，待機系ノードのリカバリ用の J2EE サーバを停止します。

   cjstopsv コマンドを使用して，待機系（NodeR）のリカバリ用の J2EE サーバを停止します。

4. 待機系ノードで HA モニタの monbegin コマンドを実行して，待機系のアプリケーションサーバを待機状態にします。

   - 待機系ノードでのコマンドの実行例

     待機系（NodeR）で，N 台の実行系（Node1～Node4）に対応する「# monbegin <サーバ識別名>」を実行して，N 台の実行系に対応する待機系を待機状態にします。

5. 運用管理サーバを起動します。

   運用管理サーバを起動して，運用管理サーバ上の運用管理エージェント，Management Server，および論理サーバ（監視用 CORBA ネーミングサービス）を起動します。運用管理サーバの起動方法については，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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

6. 実行系ノード（Node1～Node4）で HA モニタの monbegin コマンドを実行して，実行系のアプリケーションサーバを実行系として起動します。

   - 実行系ノードでのコマンドの実行例

     Node1～Node4 で「# monbegin <サーバ識別名>」を実行して，各ノードの実行系を起動します。この例では，シェルスクリプトファイル（start.sh）が実行されて，Node1～Node4 の各実行系の運用管理エージェントが起動されます。

7. 運用管理ポータルで，Node1～Node4 の各実行系の J2EE サーバをセットアップします。

「運用管理ドメインの構成定義」の［セットアップ］画面で，J2EE サーバをセットアップします。運用管理ポータルでの操作手順および画面の詳細については，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。

8. **運用管理ポータルで，J2EE サーバに設定した情報を Node1～Node4 の各実行系に配布します。**

   「論理サーバの環境設定」の［設定情報の配布］画面で，J2EE サーバに設定した情報を Node1～Node4 の各実行系に配布します。

9. **運用管理ポータルで，各実行系の論理サーバを起動します。**

   この例では，シェルスクリプトファイル（start.sh）の中で実行系の論理サーバを起動しないため，運用管理ポータルを使用して論理サーバを起動してください。

10. **サーバ管理コマンドを使用して，Node1～Node4 の各実行系の J2EE サーバに J2EE アプリケーションとリソースアダプタをインポートします。また，インポートした J2EE アプリケーションとリソースアダプタを設定および開始します。**

    サーバ管理コマンドでの操作については，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。

    なお，J2EE サーバおよびその前提プロセスを起動してからサーバ管理コマンドを実行してください。また，サーバ管理コマンドでの操作が終わったら，サーバ管理コマンドを停止させてください。

19.6　N:1 リカバリシステムの起動と停止（Windows の場合）

この節では，N:1 リカバリシステムを利用する場合の，システムの起動と停止の手順について説明します。

N:1 リカバリシステムを利用して運用する場合は，あらかじめ現用系と予備系の 2 種類のホストの用意や，クラスタサービスの対象となる運用管理エージェントを監視・起動・停止するスクリプトの登録など，必要な環境設定をしておく必要があります。設定方法については，「19.4　N:1 リカバリシステムの設定（Windows の場合）」を参照してください。

ポイント

N:1 リカバリシステムを利用する場合，Management Server は運用管理サーバで起動します。なお，個々のアプリケーションサーバのホストでは，運用管理エージェントを OS 起動と同時に起動する設定にしないでください。

参考

動作中の系を区別する場合，「実行系」と「待機系」の 2 種類の系に分けられます。

- **実行系**
  業務処理を実行する N 台のアプリケーションサーバがある系のことです。
- **待機系**
  1 台のリカバリ専用サーバがある系のことです。

19.6.1　N:1 リカバリシステムの起動

ここでは N:1 リカバリシステムを利用する場合のシステムの起動手順について説明します。

N:1 リカバリシステムを利用するには，あらかじめ運用管理サーバを起動しておくことが必要です。運用管理サーバを起動していない場合には，運用管理サーバを先に起動しておいてください。運用管理サーバの起動方法については，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「4.1.1 システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

また，N:1 リカバリシステムでは，先に起動したノードが実行系ノードとして，あとに起動したノードが待機系ノードとして動作します。必ず現用系ノードを予備系ノードよりも先に起動してください。

N:1 リカバリシステムの起動手順を次に示します。

1. **[スタート] メニューの [コントロールパネル] − [パフォーマンスとメンテナンス] − [管理ツール] から，[クラスタ アドミニストレータ] を選択します。**
   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）で現用系ノードを選択し，［ファイル］メニューの［クラスタサービスの開始］を選択します。

   選択した現用系ノードのクラスタサービスが開始されます。すべての現用行系ノードに対して実行してください。

3. コンソールツリー（左ペイン）で予備系ノードを選択し，［ファイル］メニューの［クラスタサービスの開始］を選択します。

   予備系ノードのクラスタサービスが開始されます。

4. コンソールツリー（左ペイン）で現用系ノードおよび予備系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［オンラインにする］を選択します。

   選択したリソースグループがオンラインになります。すべてのリソースグループに対して実行してください。

5. 実行系のホストの各論理サーバを起動します。

   起動方法については，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「4.1.2 システムの起動方法」を参照してください。

19.6.2　N:1 リカバリシステムの停止

ここでは，N:1 リカバリシステムの停止方法について説明します。

1. ［スタート］メニューの［コントロールパネル］－［パフォーマンスとメンテナンス］－［管理ツール］から，［クラスタ アドミニストレータ］を選択します。

   クラスタアドミニストレータが開始されます。

2. コンソールツリー（左ペイン）で実行系ノードおよび待機系ノードが含まれているリソースグループを選択して，［ファイル］メニューの［オフラインにする］を選択します。

   選択したリソースグループがオフラインになります。すべてのリソースグループに対して実行してください。

3. コンソールツリー（左ペイン）で予備系ノードを選択し，［ファイル］メニューの［クラスタサービスの停止］を選択します。

   予備系ノードのクラスタサービスが停止します。

4. コンソールツリー（左ペイン）で現用系ノードを選択し，［ファイル］メニューの［クラスタサービスの停止］を選択します。

   選択した現用系ノードのクラスタサービスが停止します。すべての現用系ノードに対して実行してください。

なお，運用管理サーバは最後に手動で停止します。運用管理サーバの停止方法については，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「4.1.3　システムの停止手順」およびマニュア

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「4.1.4　システムの停止方法」を参照してください。

19.7　N:1 リカバリシステムの起動と停止（UNIX の場合）

この節では，HA モニタによる N:1 リカバリシステムを利用する場合の，システムの起動と停止の手順について説明します。なお，HA モニタが利用できるのは，AIX または Linux の場合だけです。

HA モニタによる N:1 リカバリシステムを利用して運用する場合は，あらかじめ実行系と待機系の 2 種類のホストの用意や，HA モニタの監視の対象となる運用管理エージェントを監視・起動・停止するスクリプトの登録など，必要な環境設定をしておく必要があります。設定方法については，「19.5　N:1 リカバリシステムの設定（UNIX の場合）」を参照してください。

ポイント

HA モニタによる N:1 リカバリシステムを利用する場合，Management Server は運用管理サーバで起動して，個々のアプリケーションサーバのホストでは，運用管理エージェントだけを起動します。なお，個々のアプリケーションサーバのホストでは，運用管理エージェントを OS 起動と同時に起動する設定にしないでください。

また，次のコマンドは，HA モニタが提供しているコマンドです。詳細は，マニュアル「高信頼化システム監視機能　HA モニタ」を参照してください。

- monbegin（HA モニタとのインタフェースを持たないサーバを起動）
- monend（HA モニタとのインタフェースを持たない実行サーバの停止連絡）

参考

HA モニタの動作中の系を区別する場合，「実行系」と「待機系」の 2 種類の系に分けられます。

- **実行系**
  業務処理を実行する N 台のアプリケーションサーバがある系のことです。
- **待機系**
  1 台のリカバリ専用サーバがある系のことです。

19.7.1　N:1 リカバリシステムの起動

HA モニタによる N:1 リカバリシステムを利用する場合のシステム起動時の留意事項，およびシステムの起動手順について説明します。

(1)　システム起動時の留意事項

HA モニタによる N:1 リカバリシステムを利用する場合のシステム起動時の留意事項を次に示します。

- 実行系と待機系のホストに配置された HA モニタは，OS の起動と同時に起動されています。

- 運用管理サーバを起動していない場合は，運用管理サーバをあらかじめ起動しておいてください。運用管理サーバの起動方法については，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「4.1.1　システムの起動手順」およびマニュアル「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「4.1.2　システムの起動方法」を参照してください。

(2)　システムの起動手順

HA モニタによる N:1 リカバリシステムを利用する場合のシステムの起動手順を次に示します。

1. 待機系のホストで，各実行系に対応する monbegin コマンドを N 回実行します。

```
# 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。

これによって，待機系のリカバリ専用サーバが実行系の監視を開始します。

2. 実行系の各ホスト（N 台）で monbegin コマンドを実行して，実行系の各ホストの運用管理エージェントを起動します。

```
# 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」に指定されているサーバの識別名を指定します。

これによって，実行系の各ホストの運用管理エージェントが起動します。

注意事項

Management Server 起動時に運用管理ドメイン内の論理サーバを一括起動する設定にしていない場合は，実行系のホストの論理サーバを起動してください。なお，Management Server 起動時に運用管理ドメイン内の論理サーバを一括起動するように設定にしている場合，論理サーバの手動起動は不要です。起動方法については，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「4.1.2　システムの起動方法」を参照してください。

参考

servers ファイルの定義（HA モニタでのサーバの環境設定）については，「19.5.6　サーバ対応の環境設定」を参照してください。

19.7.2　N:1 リカバリシステムの停止

HA モニタによる N:1 リカバリシステムを利用する場合のシステムの停止手順を次に示します。

1. 運用管理エージェントを停止するスクリプトで，運用管理ドメイン内の論理サーバを停止する設定にしていない場合は，実行系のホストで各論理サーバを停止します。

運用管理エージェントを停止するスクリプトで論理サーバを停止する設定にしている場合は，不要な手順です。論理サーバの停止方法については，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「4.1.4　システムの停止方法」を参照してください。

2. 実行系の各ホスト（N 台）で monend コマンドを実行して，実行系のホストの運用管理エージェントを停止します。

```
# monend 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」に指定されているサーバの識別名を指定します。

これによって，実行系の各ホストの運用管理エージェントが停止します。

参考

servers ファイルの定義（HA モニタでのサーバの環境設定）については，「19.5.6　サーバ対応の環境設定」を参照してください。

なお，運用管理サーバは最後に手動で停止します。運用管理サーバの停止方法については，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「4.1.3　システムの停止手順」およびマニュアル「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「4.1.4　システムの停止方法」を参照してください。

付録

付録 A　クラスタソフトウェア連携時の TPBroker の設定（Windows の場合）

クラスタソフトウェア（Windows Server Failover Cluster）を使用して，TPBroker をクラスタ構成で運用することができます。

TPBroker では，系切り替えの要因として次のものに対応します。

- ハードウェア障害などクラスタサービスによるシステム障害の検出
- ユーザによる計画的な系切り替え

付録 A.1　システムの起動と停止

システムの要求事項によって，系切り替えの構築に必要な TPBroker の機能と環境設定項目を決定します。

まず，次の機能を使用するかどうかを決定してください。

- ORB 機能
- OTS 機能

また，OTS 機能を使用する場合，共有ディスクを使用します。

(1)　ORB 機能

スマートエージェントやスマートエージェントに接続するプロセスをクラスタソフトウェア（Windows Server Failover Cluster）で運用する場合は，ORB 機能に関する各種 IP アドレスを設定する必要があります。詳細については，「付録 A.2　ORB 機能使用時の設定」を参照してください。

なお，スマートエージェントに接続するプロセスとは，CORBA ネーミングサービス，CTM デーモンおよびスマートエージェントを使用する J2EE サーバ※のことです。

注※

usrconf.properties ファイルの vbroker.agent.enableLocator キーに true を指定して J2EE サーバを起動している場合を指します。

(2)　OTS 機能

トランザクションサービスはインプロセスで起動してください。

付録 A.2　ORB 機能使用時の設定

ここでは，ORB 機能を使用する場合の設定について説明します。

この設定は，Windows Server Failover Cluster を起動するホスト上で次のプロセスを起動する場合に設定してください。

- スマートエージェント
- CORBA ネーミングサービス

また，スマートエージェントをフェイルオーバの対象とする場合は，Windows Server Failover Cluster に「汎用アプリケーション」として登録してください。

(1) スマートエージェントの localaddr ファイルおよび htc.clienthandleraddr ファイルの設定

クラスタソフトウェア上でスマートエージェントを起動する場合，各ノードの localaddr ファイルおよび htc.clienthandleraddr ファイルを設定してください。なお，設定内容は，マルチホームドホスト環境の場合とそれ以外の場合で異なります。

マルチホームドホスト環境，またはマルチホームドホストでないホスト環境上でクラスタ環境を構築する場合の，スマートエージェントの localaddr ファイルの設定内容を次の表に示します。

表 A-1　スマートエージェントの localaddr ファイルの設定内容

クラスタサービスおよびスマートエージェントを起動するホスト環境	スマートエージェントの localaddr ファイルの設定内容	
	スマートエージェントをフェイルオーバの対象にする場合	スマートエージェントをフェイルオーバの対象にしない場合
マルチホームドホスト環境	• ステーショナリ IP アドレス（プライマリ IP アドレス） • スマートエージェントに明示的に指定したい IP アドレス • クラスタ IP アドレス	• ステーショナリ IP アドレス（プライマリ IP アドレス） • スマートエージェントに明示的に指定したい IP アドレス
マルチホームドホストではないホスト環境	• ステーショナリ IP アドレス（プライマリ IP アドレス） • クラスタ IP アドレス	• ステーショナリ IP アドレス（プライマリ IP アドレス）

次に，環境ごとの設定内容の詳細について説明します。また，htc.clienthandleraddr ファイルについても説明します。

(a) マルチホームドホスト環境

- **スマートエージェントをフェイルオーバの対象にする場合**

  localaddr ファイルには，ステーショナリ IP アドレス（プライマリ IP アドレス），スマートエージェントに明示的に指定したい IP アドレス※，およびスマートエージェントが使用するクラスタ IP アドレスを必ず設定してください。

  また，スマートエージェントに接続するプロセスに対して，クラスタ IP アドレスを返却するように htc.clienthandleraddr ファイルを定義してください。

- スマートエージェントをフェイルオーバの対象にしない場合

  localaddr ファイルには，ステーショナリ IP アドレス（プライマリ IP アドレス），およびスマートエージェントに明示的に指定したい IP アドレス※を必ず設定してください。クラスタ IP アドレスは設定しないでください。

  また，スマートエージェントに接続するプロセスに対して，ステーショナリ IP アドレスを返却するように htc.clienthandleraddr ファイルを定義してください。

  注※

  localaddr ファイルを設定すると，スマートエージェントは，localaddr ファイルに設定された IP アドレスだけを認識します。localaddr ファイルを設定しない場合，スマートエージェントはデフォルトで gethostbyname()から取得できる IP アドレスを認識します。

(b) マルチホームドホストではないホスト環境

- スマートエージェントをフェイルオーバの対象にする場合

  localaddr ファイルには，ステーショナリ IP アドレス（プライマリ IP アドレス）およびスマートエージェントが使用するクラスタ IP アドレスを必ず設定してください。

  また，スマートエージェントに接続するプロセスに対して，クラスタ IP アドレスを返却するように htc.clienthandleraddr を定義してください。

- スマートエージェントをフェイルオーバの対象にしない場合

  localaddr ファイルには，ステーショナリ IP アドレス（プライマリ IP アドレス）を必ず設定してください。クラスタ IP アドレスは設定しないでください。

  また，スマートエージェントに接続するプロセスに対して，ステーショナリ IP アドレスを返却するように htc.clienthandleraddr ファイルを定義してください。

(2) スマートエージェントまたはスマートエージェントに接続するプロセスの agentaddr ファイル，vbroker.agent.addr プロパティ，および環境変数「OSAGENT_ADDR」の設定

異なるネットワークドメイン上にあるスマートエージェント間で通信をする場合，または異なるネットワークドメイン上にあるスマートエージェントとスマートエージェントに接続するプロセス間の通信をする場合は，スマートエージェントまたはスマートエージェントに接続するプロセスの設定が必要です。設定は，各ノードの agentaddr ファイル，vbroker.agent.addr プロパティ，環境変数「OSAGENT_ADDR」に設定します。

(a) スマートエージェントへの設定

異なるネットワークドメイン上のスマートエージェント間で通信していて，スマートエージェントをフェイルオーバの対象とする場合，対象のスマートエージェントと通信をするスマートエージェントの agentaddr ファイルに，クラスタ IP アドレスを必ず設定してください。

(b) スマートエージェントに接続するプロセスへの設定

スマートエージェントとスマートエージェントに接続するプロセスとを異なるネットワークドメイン上で起動し，かつスマートエージェントをフェイルオーバの対象とする場合，スマートエージェントに接続するプロセスの agentaddr ファイル，vbroker.agent.addr プロパティ，環境変数「OSAGENT_ADDR」のどれかに，フェイルオーバの対象となるスマートエージェントのクラスタ IP アドレスを必ず設定してください。

(3) CORBA ネーミングサービスの設定

CORBA ネーミングサービスをフェイルオーバの対象とする場合は，CORBA ネーミングサービスの起動プロパティを設定します。

nameserv コマンドに次の起動プロパティを設定して CORBA ネーミングサービスを起動してください。

```
-J-Dvbroker.se.iiop_tp.host=クラスタIPアドレスまたはクラスタのネットワーク名
```

また，このプロパティに設定したクラスタ IP アドレスまたはクラスタのネットワークを，usrconf.properties（J2EE サーバ用ユーザプロパティファイル）の次のキーの設定内容に反映してください。

- ejbserver.naming.host キーに設定する IP アドレスまたはホスト名称
- ejbserver.jndi.namingservice.group.<Specify group name>.providerurls キーに設定するプロバイダ URL

usrconf.properties 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「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。nameserv コマンドを使用した CORBA ネーミングサービスの起動については，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「付録 C.2　システムの起動方法」を参照してください。

(4) J2EE サーバの vbroker.se.iiop_tp.host キー，および vbroker.se.iiop_tp.scm.iiop_tp.listener.port キーの設定

J2EE サーバをフェイルオーバの対象とする場合は，vbroker.se.iiop_tp.host キーには，クラスタ IP アドレスを指定して J2EE サーバを起動してください。また，J2EE サーバが使用するポート番号は，すべての系で同じポート番号が使用できるように，usrconf.properties ファイルの vbroker.se.iiop_tp.scm.iiop_tp.listener.port キーを使用して管理してください。

(5) 環境変数およびオプションの設定

クラスタ環境でスマートエージェントを起動する場合，スマートエージェントに次の環境変数およびオプションの設定が必要です。

オプションは，スマートエージェントをフェイルオーバの対象とする場合だけ指定してください。

- 環境変数名：OSAGENT_CLIENT_HANDLER_PORT
- オプション：-m

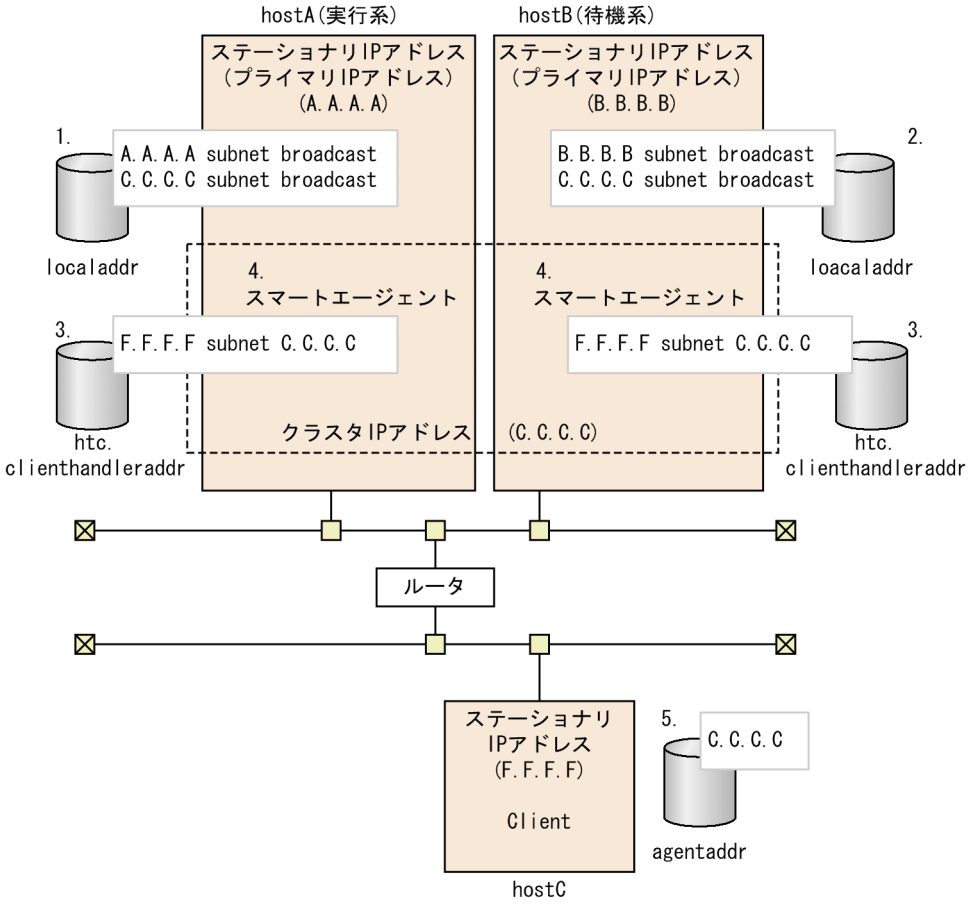
なお，環境変数の設定方法については，マニュアル「Borland(R) Enterprise Server VisiBroker(R) デベロッパーズガイド」の VisiBroker ORB ドメイン内の作業についての説明を参照してください。

(6) 設定例

IP アドレスの設定例を次に示します。

設定例 1

図 A-1 スマートエージェントをフェイルオーバの対象にする場合の IP アドレスの設定例



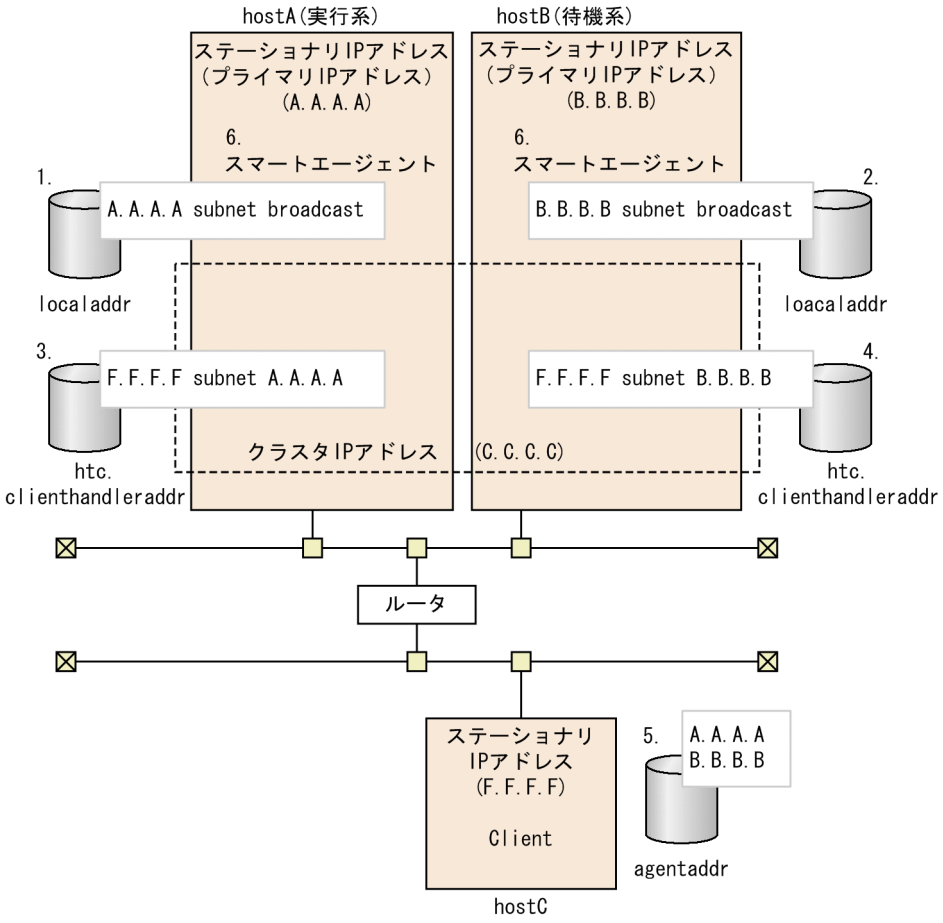
図中の 1.～5.について説明します。

1. hostA の localaddr ファイルに「hostA のステーショナリ IP アドレス（A.A.A.A） subnet broadcast」,「クラスタ IP アドレス（C.C.C.C） subnet broadcast」を記述します。
2. hostB の localaddr ファイルに「hostB のステーショナリ IP アドレス（B.B.B.B） subnet broadcast」,「クラスタ IP アドレス（C.C.C.C） subnet broadcast」を記述します。
3. hostA，hostB の htc.clienthandleraddr ファイルに「クライアントのステーショナリ IP アドレス（F.F.F.F） subnet クラスタ IP アドレス（C.C.C.C)」を記述します。

4. hostA および hostB で起動するスマートエージェントに環境変数「OSAGENT_CLIENT_HANDLER_PORT」および-m オプションを設定します。
5. hostC の agentaddr ファイルにクラスタ IP アドレス（C.C.C.C）を記述します。

設定例 2

図 A-2　スマートエージェントをフェイルオーバの対象にしない場合の IP アドレスの設定例



図中の 1.～6.について説明します。

1. hostA の localaddr ファイルに「hostA のステーショナリ IP アドレス（A.A.A.A） subnet broadcast」を記述します。
2. hostB の localaddr ファイルに「hostB のステーショナリ IP アドレス（B.B.B.B） subnet broadcast」を記述します。
3. hostA の htc.clienthandleraddr ファイルに「クライアントのステーショナリ IP アドレス（F.F.F.F） subnet hostA のステーショナリ IP アドレス（A.A.A.A）」を記述します。
4. hostB の htc.clienthandleraddr ファイルに「クライアントのステーショナリ IP アドレス（F.F.F.F） subnet hostB のステーショナリ IP アドレス（B.B.B.B）」を記述します。
5. hostC の agentaddr ファイルに A.A.A.A および B.B.B.B を記述します。

6. hostA および hostB で起動するスマートエージェントに環境変数
「OSAGENT_CLIENT_HANDLER_PORT」を設定します。

付録 B　クラスタソフトウェア連携時の TPBroker の設定（UNIX の場合）

クラスタソフトウェア（HA モニタ）を使用して，TPBroker をクラスタ構成で運用することができます。

TPBroker では，系切り替えの要因として次のものに対応します。

- ハードウェア障害などクラスタサービスによるシステム障害の検出
- ユーザによる計画的な系切り替え

付録 B.1　システムの起動と停止

システムの要求事項によって，系切り替えの構築に必要な TPBroker の機能と環境設定項目を決定します。

まず，次の機能を使用するかどうかを決定してください。

- ORB 機能
- OTS 機能

また，OTS 機能を使用する場合，共有ディスクを使用します。

(1)　ORB 機能

スマートエージェントやスマートエージェントに接続するプロセスをクラスタソフトウェア（HA モニタ）で運用する場合は，ORB 機能に関する各種 IP アドレスを設定する必要があります。詳細については，「付録 B.2　ORB 機能使用時の設定」を参照してください。

なお，スマートエージェントに接続するプロセスとは，CORBA ネーミングサービス，CTM デーモンおよびスマートエージェントを使用する J2EE サーバ※のことです。

注※

usrconf.properties の vbroker.agent.enableLocator キーに true を指定して J2EE サーバを起動している場合を指します。

(2)　OTS 機能

トランザクションサービスはインプロセスで起動してください。

(3)　クラスタ構成で運用するための留意点

TPBroker をクラスタ構成で運用する場合，次の点に留意してください。

- TPBroker を含むプログラム群は各系のローカルディスクに格納してください。
- すべての系の TPBroker のバージョンを統一してください。

- すべての系の TPBroker のシステム環境定義を統一してください。
- J2EE アプリケーションで使用するポート番号が固定の場合，すべての系で同じポート番号が使用できるように管理してください。

付録 B.2　ORB 機能使用時の設定

ここでは，ORB 機能を使用する場合の設定について説明します。

この設定は，HA モニタを起動するホスト上で次のプロセスを起動する場合に設定してください。

- スマートエージェント
- CORBA ネーミングサービス

(1)　スマートエージェントの localaddr ファイルおよび htc.clienthandleraddr ファイルの設定

クラスタソフトウェア上でスマートエージェントを起動する場合，各ノードの localaddr ファイルおよび htc.clienthandleraddr ファイルを設定してください。なお，設定内容は，マルチホームドホスト環境の場合とそれ以外の場合で異なります。

マルチホームドホスト環境，またはマルチホームドホストでないホスト環境上でクラスタ環境を構築する場合の，スマートエージェントの localaddr ファイルの設定内容を次の表に示します。

表 B-1　スマートエージェントの localaddr ファイルの設定内容

クラスタサービスおよびスマートエージェントを起動するホスト環境	スマートエージェントの localaddr ファイルの設定内容	
	スマートエージェントを系切り替えの対象にする場合	スマートエージェントを系切り替えの対象にしない場合
マルチホームドホスト環境	• ステーショナリ IP アドレス（プライマリ IP アドレス） • スマートエージェントに明示的に指定したい IP アドレス • エイリアス IP アドレス	• ステーショナリ IP アドレス（プライマリ IP アドレス） • スマートエージェントに明示的に指定したい IP アドレス
マルチホームドホストではないホスト環境	• ステーショナリ IP アドレス（プライマリ IP アドレス） • エイリアス IP アドレス	• ステーショナリ IP アドレス（プライマリ IP アドレス）

次に，環境ごとの設定内容の詳細について説明します。また，htc.clienthandleraddr ファイルについても説明します。

(a)　マルチホームドホスト環境

- スマートエージェントを系切り替えの対象にする場合

localaddr ファイルには，ステーショナリ IP アドレス（プライマリ IP アドレス），スマートエージェントに明示的に指定したい IP アドレス※，およびスマートエージェントが使用するエイリアス IP アドレスを必ず設定してください。

また，スマートエージェントに接続するプロセスに対して，エイリアス IP アドレスを返却するように htc.clienthandleraddr ファイルを定義してください。

- スマートエージェントを系切り替えの対象にしない場合

localaddr ファイルには，ステーショナリ IP アドレス（プライマリ IP アドレス），およびスマートエージェントに明示的に指定したい IP アドレス※を必ず設定してください。エイリアス IP アドレスは設定しないでください。

また，スマートエージェントに接続するプロセスに対して，ステーショナリ IP アドレスを返却するように htc.clienthandleraddr ファイルを定義してください。

注※

localaddr ファイルを設定すると，スマートエージェントは，localaddr ファイルに設定された IP アドレスだけを認識します。localaddr ファイルを設定しない場合，スマートエージェントはデフォルトで gethostbyname()から取得できる IP アドレスを認識します。なお，スマートエージェントで認識する IP アドレスは，スマートエージェント起動時に，osagent コマンドに-v オプションを指定して，バーボースモードをオンにして起動することで確認できます。

(b) マルチホームドホストではないホスト環境

- スマートエージェントを系切り替えの対象にする場合

localaddr ファイルには，ステーショナリ IP アドレス（プライマリ IP アドレス）およびスマートエージェントが使用するエイリアス IP アドレスを必ず設定してください。

また，スマートエージェントに接続するプロセスに対して，エイリアス IP アドレスを返却するように htc.clienthandleraddr を定義してください。

- スマートエージェントを系切り替えの対象にしない場合

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

また，スマートエージェントに接続するプロセスに対して，ステーショナリ IP アドレスを返却するように htc.clienthandleraddr ファイルを定義してください。

(2) スマートエージェントまたはスマートエージェントに接続するプロセスの agentaddr ファイル，vbroker.agent.addr プロパティ，および環境変数「OSAGENT_ADDR」の設定

異なるネットワークドメイン上にあるスマートエージェント間で通信をする場合，または異なるネットワークドメイン上にあるスマートエージェントとスマートエージェントに接続するプロセス間の通信をする場合は，スマートエージェントまたはスマートエージェントに接続するプロセスの設定が必要です。設定は，各ノードの agentaddr ファイル，vbroker.agent.addr プロパティ，環境変数「OSAGENT_ADDR」に設定します。

(a) スマートエージェントへの設定

異なるネットワークドメイン上のスマートエージェント間通信で，スマートエージェントを系切り替えの対象とする場合，対象のスマートエージェントと通信をするスマートエージェントの agentaddr ファイルに，エイリアス IP アドレスを必ず設定してください。

(b) スマートエージェントに接続するプロセスへの設定

スマートエージェントとスマートエージェントに接続するプロセスとを異なるネットワークドメイン上で起動し，かつスマートエージェントを系切り替えの対象とする場合，スマートエージェントに接続するプロセスの agentaddr ファイル，vbroker.agent.addr プロパティ，環境変数「OSAGENT_ADDR」のどれかに，系切り替えの対象となるスマートエージェントのエイリアス IP アドレスを必ず設定してください。

(3) CORBA ネーミングサービスの設定

CORBA ネーミングサービスを系切り替えの対象とする場合は，CORBA ネーミングサービスの起動プロパティを設定します。

nameserv コマンドに次の起動プロパティを設定して CORBA ネーミングサービスを起動してください。

```
-J-Dvbroker.se.iiop_tp.host=エイリアスIPアドレスまたはエイリアスホスト名
```

また，このプロパティに設定したエイリアス IP アドレスまたはエイリアスホスト名を，usrconf.properties（J2EE サーバ用ユーザプロパティファイル）の次のキーの設定内容に反映してください。

- ejbserver.naming.host キーに設定する IP アドレスまたはホスト名称
- ejbserver.jndi.namingservice.group.<Specify group name>.providerurls キーに設定するプロバイダ URL

usrconf.properties 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「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。nameserv コマンドを使用した CORBA ネーミングサービスの起動については，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「付録 C.2　システムの起動方法」を参照してください。

(4) J2EE サーバの vbroker.se.iiop_tp.host キー，および vbroker.se.iiop_tp.scm.iiop_tp.listener.port キーの設定

J2EE サーバをフェイルオーバの対象とする場合は，vbroker.se.iiop_tp.host キーには，クラスタ IP アドレスを指定して J2EE サーバを起動してください。また，J2EE サーバが使用するポート番号は，すべての系で同じポート番号が使用できるように，usrconf.properties ファイルの vbroker.se.iiop_tp.scm.iiop_tp.listener.port キーを使用して管理してください。

(5)　環境変数の設定

クラスタ環境でスマートエージェントを起動する場合，スマートエージェントに次の環境変数の設定が必要です。

- 環境変数名：OSAGENT_CLIENT_HANDLER_PORT

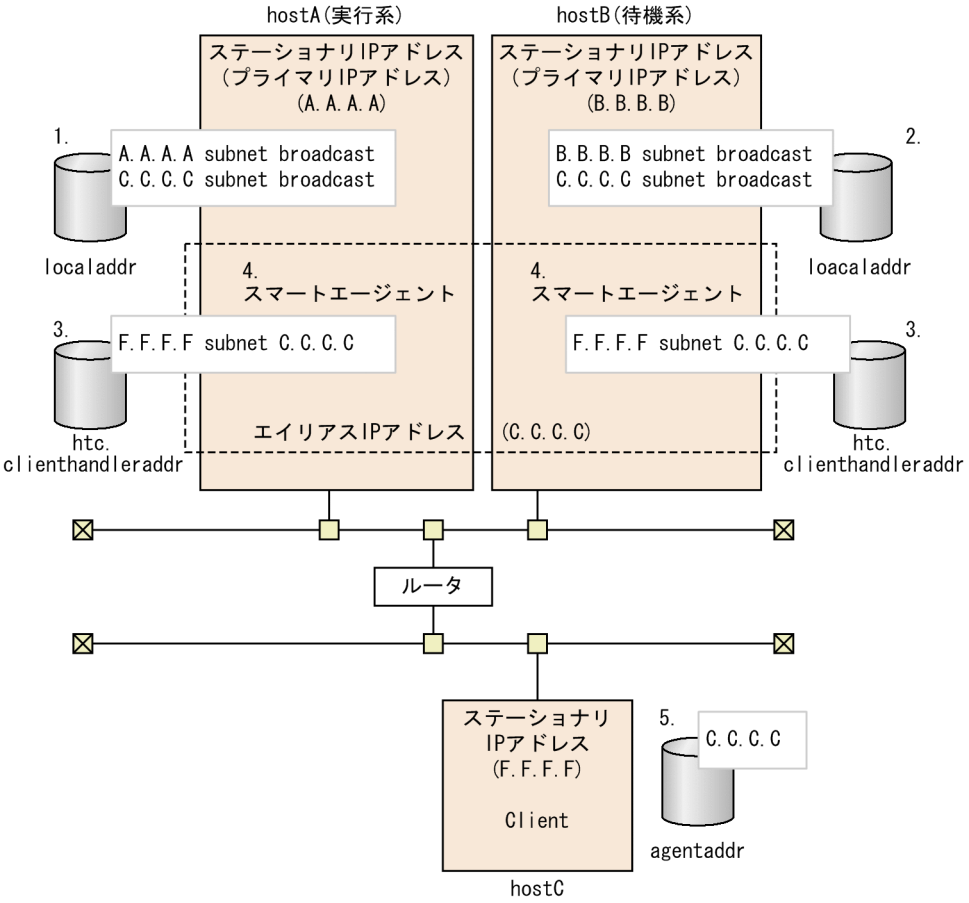
なお，環境変数の設定方法については，マニュアル「Borland(R) Enterprise Server VisiBroker(R) デベロッパーズガイド」の VisiBroker ORB ドメイン内の作業についての説明を参照してください。

(6)　設定例

IP アドレスの設定例を次に示します。

設定例 1

図 B-1　スマートエージェントを系切り替えの対象にする場合の IP アドレスの設定例



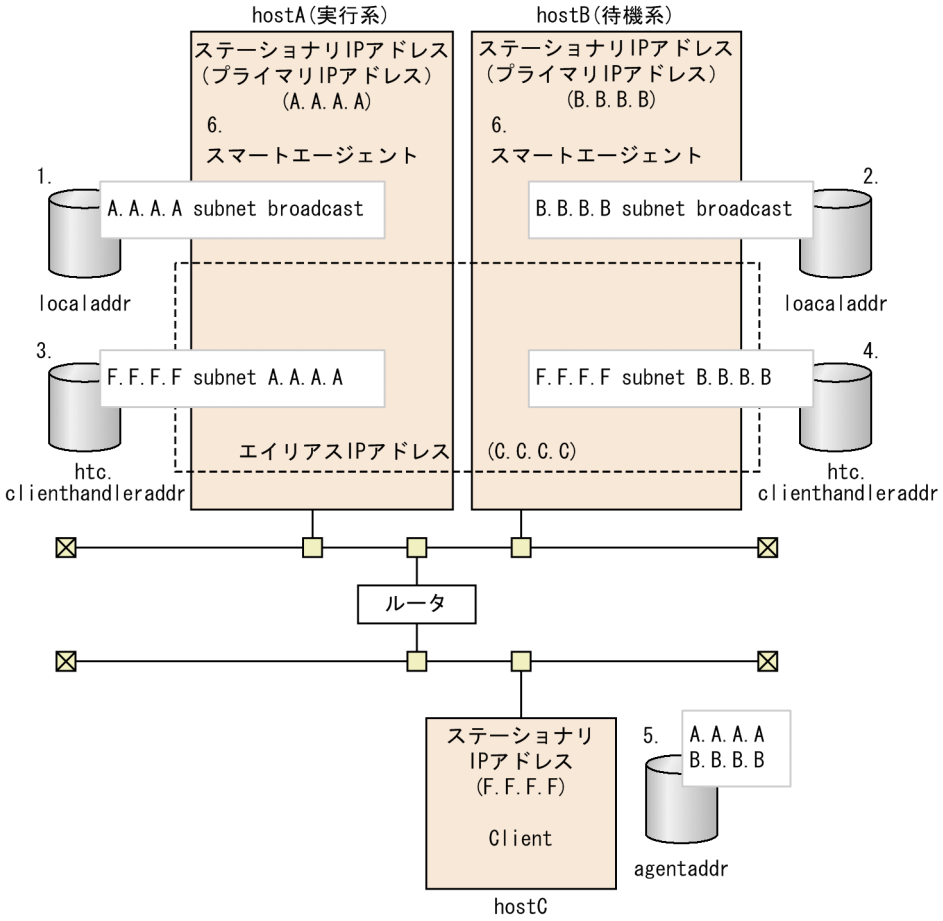
図中の 1.～5.について説明します。

1. hostA の localaddr ファイルに「hostA のステーショナリ IP アドレス（A.A.A.A） subnet broadcast」，「エイリアス IP アドレス（C.C.C.C） subnet broadcast」を記述します。

2. hostB の localaddr ファイルに「hostB のステーショナリ IP アドレス（B.B.B.B） subnet broadcast」,「エイリアス IP アドレス（C.C.C.C） subnet broadcast」を記述します。
3. hostA，hostB の htc.clienthandleraddr ファイルに「クライアントのステーショナリ IP アドレス（F.F.F.F） subnet エイリアス IP アドレス（C.C.C.C)」を記述します。
4. hostA および hostB で起動するスマートエージェントに環境変数「OSAGENT_CLIENT_HANDLER_PORT」を設定します。
5. hostC の agentaddr ファイルにエイリアス IP アドレス（C.C.C.C）を記述します。

設定例 2

図 B-2　スマートエージェントを系切り替えの対象にしない場合の IP アドレスの設定例



図中の 1.～6.について説明します。

1. hostA の localaddr ファイルに「hostA のステーショナリ IP アドレス（A.A.A.A） subnet broadcast」を記述します。
2. hostB の localaddr ファイルに「hostB のステーショナリ IP アドレス（B.B.B.B） subnet broadcast」を記述します。
3. hostA の htc.clienthandleraddr ファイルに「クライアントのステーショナリ IP アドレス（F.F.F.F） subnet hostA のステーショナリ IP アドレス（A.A.A.A)」を記述します。

4. hostB の htc.clienthandleraddr ファイルに「クライアントのステーショナリ IP アドレス（F.F.F.F）subnet hostB のステーショナリ IP アドレス（B.B.B.B)」を記述します。
5. hostC の agentaddr ファイルに A.A.A.A および B.B.B.B を記述します。
6. hostA および hostB で起動するスマートエージェントに環境変数「OSAGENT_CLIENT_HANDLER_PORT」を設定します。

付録 C　保護区リストの内容

保護区リストの内容を次に示します。

```
#
# DO NOT EDIT THIS FILE.
#
# All Rights Reserved. Copyright (C) 2004, 2015, Hitachi, Ltd.

#-----------------------------------------------------------------------
# Cosminexus Component Container
#-----------------------------------------------------------------------
com.hitachi.software.auditlog.*
com.hitachi.software.ejb.*
com.hitachi.software.jpa.*
com.sun.ejb.*
com.sun.enterprise.*
com.sun.web.*
com.sun.activation.*
com.sun.mail.*

com.hitachi.software.javamail.*
com.hitachi.software.web.*
com.hitachi.software.was.logger.*
com.hitachi.software.was.tracer.*
com.hitachi.software.was.web.*
com.hitachi.software.was.sfo.*
org.apache.ajp.*
org.apache.catalina.*
org.apache.coyote.*
org.apache.jasper.*
org.apache.jk.*
org.apache.naming.*
org.apache.tomcat.*

com.cosminexus.cc.lib.*

#-----------------------------------------------------------------------
# HiRDB Type4 JDBC Driver
#-----------------------------------------------------------------------
JP.co.Hitachi.soft.HiRDB.JDBC.*

#-----------------------------------------------------------------------
# Oracle JDBC Thin Driver
#-----------------------------------------------------------------------
oracle.*

#-----------------------------------------------------------------------
# DB2 Universal JDBC Driver
#-----------------------------------------------------------------------
com.ibm.db2.*
COM.ibm.db2.*
COM.ibm.db2os390.*
sqlj.runtime.*

#-----------------------------------------------------------------------
```

```
# Microsoft SQL Server JDBC Driver
#-----------------------------------------------------------------------
com.microsoft.sqlserver.jdbc.*
microsoft.sql.*

#-----------------------------------------------------------------------
# PostgreSQL JDBC Driver
#-----------------------------------------------------------------------
org.postgresql.*

#-----------------------------------------------------------------------
# MySQL Connector/J
#-----------------------------------------------------------------------
com.mysql.*
org.gjt.mm.mysql.*

#-----------------------------------------------------------------------
# HNTRLib
#-----------------------------------------------------------------------
jp.co.hitachi.soft.hntrlib2.*
jp.co.hitachi.soft.hntrlibM.*

#-----------------------------------------------------------------------
# Cosminexus TPBroker(ORB)
#-----------------------------------------------------------------------
com.inprise.vbroker.*
com.borland.security.*
com.borland.vbroker.*
org.omg.BiDirPolicy.*
org.omg.CORBA.*
org.omg.CORBA_2_3.*
org.omg.CosEventChannelAdmin.*
org.omg.CosEventComm.*
org.omg.CosNaming.*
org.omg.CosNotification.*
org.omg.CosNotifyChannelAdmin.*
org.omg.CosNotifyComm.*
org.omg.CosNotifyFilter.*
org.omg.CosTransactions.*
org.omg.CosTypedEventChannelAdmin.*
org.omg.CosTypedEventComm.*
org.omg.CosTypedNotifyChannelAdmin.*
org.omg.CosTypedNotifyComm.*
org.omg.Dynamic.*
org.omg.DynamicAny.*
org.omg.Firewall.*
org.omg.IOP.*
org.omg.MessageRouting.*
org.omg.Messaging.*
org.omg.PortableInterceptor.*
org.omg.PortableServer.*
org.omg.SendingContext.*
org.omg.TimeBase.*
javax.rmi.*

#-----------------------------------------------------------------------
# Cosminexus TPBroker(OTS)
#-----------------------------------------------------------------------
```

```
COM.Hitachi.software.TPBroker.OTS.*
COM.Hitachi.software.TPBroker.otsinprocess.*
org.omg.CosTransactions.*
org.omg.CosTSInteroperation.*

#-------------------------------------------------------------------
# Cosminexus Performance Tracer
#-------------------------------------------------------------------
com.hitachi.software.ejb.prf.*
com.hitachi.software.ejb.ctm.*
JP.co.Hitachi.soft.CPRF.*

#-------------------------------------------------------------------
# Cosminexus Manager
#-------------------------------------------------------------------
com.cosminexus.mngsvr.*
com.cosminexus.admin.*
com.cosminexus.manager.*

#-------------------------------------------------------------------
# Cosminexus DABroker Library
#-------------------------------------------------------------------
JP.co.Hitachi.soft.DBPSV_Driver.*

#-------------------------------------------------------------------
# Cosminexus Component Library
#-------------------------------------------------------------------
com.cosminexus.cwc.*

#-------------------------------------------------------------------
# TP1/Message Queue - Access
#-------------------------------------------------------------------
jp.co.Hitachi.soft.mqadaptor.*

#-------------------------------------------------------------------
# Cosminexus Reliable Messaging
#-------------------------------------------------------------------
jp.co.Hitachi.soft.reliablemessaging.*
jp.co.Hitachi.soft.reliablemessaging_np.*

#-------------------------------------------------------------------
# Cosminexus TP1 Connector
#-------------------------------------------------------------------
jp.co.hitachi_system.tp1connector.*
JP.co.Hitachi.soft.OpenTP1.*

#-------------------------------------------------------------------
# Cosminexus Service Coordinator
#-------------------------------------------------------------------
jp.co.Hitachi.soft.csc.*
jp.co.Hitachi.soft.csciw.*
com.cosminexus.ftp.*
com.cosminexus.csc.monitor.*

#-------------------------------------------------------------------
# Cosminexus SOAP
#-------------------------------------------------------------------
com.cosminexus.cws.*
```

```
com.cosminexus.c4web.*
org.apache.axis.types.*
org.apache.commons.discovery.tools.*
javax.xml.soap.*
javax.xml.rpc.*

#--------------------------------------------------------------------
# Cosminexus Web Services - Security
#--------------------------------------------------------------------
com.cosminexus.wss.*

#--------------------------------------------------------------------
# Cosminexus JAX-WS(CJW)
#--------------------------------------------------------------------
com.cosminexus.xml.ws.*
com.cosminexus.tools.ws.*
com.cosminexus.istack.ws.*
com.cosminexus.org.jvnet.fastinfoset.*
com.cosminexus.xml.fastinfoset.*
com.cosminexus.org.apache.xml.internal.resolver.*
com.cosminexus.xml.messaging.saaj.*
com.cosminexus.xml.stream.buffer.*
org.jvnet.mimepull.*
com.cosminexus.org.glassfish.external.*
com.cosminexus.org.glassfish.gmbal.*
javax.xml.ws.*
com.sun.xml.ws.*
com.cosminexus.wsrm.*
com.cosminexus.wsit.*

#--------------------------------------------------------------------
# Cosminexus JAX-RS(CJR)
#--------------------------------------------------------------------
com.cosminexus.jersey.*
com.cosminexus.ws.rs.*
com.sun.research.ws.wadl.*
com.cosminexus.org.codehaus.jackson.*
com.cosminexus.org.codehaus.jettison.*
javax.ws.rs.*

#--------------------------------------------------------------------
# uCosminexus Hitachi Code Conversion
#--------------------------------------------------------------------
JP.co.Hitachi.soft.codeconv.*

#--------------------------------------------------------------------
# Cosminexus JMS Provider
#--------------------------------------------------------------------
com.cosminexus.jmsprovider.*

#--------------------------------------------------------------------
# Cosminexus JSF/JSTL/BV
#--------------------------------------------------------------------
javax.faces.*
com.sun.faces.*
com.hitachi.software.faces.*
org.apache.taglibs.*
org.hibernate.validator.*
```

```
#----------------------------------------------------------------------
# uCosminexus Elastic Application Data Store
#----------------------------------------------------------------------
com.hitachi.software.eads.*

#----------------------------------------------------------------------
# Cosminexus CDI
#----------------------------------------------------------------------
com.hitachi.software.cdi.*

#----------------------------------------------------------------------
# J2SE
#----------------------------------------------------------------------
JP.co.Hitachi.soft.jvm.*
com.sun.corba.se.*
com.sun.crypto.provider.*
com.sun.image.*
com.sun.imageio.*
com.sun.inputmethods.internal.indicim.DevanagariInputMethodDescriptor
com.sun.inputmethods.internal.thaiim.ThaiInputMethodDescriptor
com.sun.java.browser.dom.DOMService
com.sun.java.swing.*
com.sun.java.util.jar.pack.Attribute
com.sun.java.util.jar.pack.Coding
com.sun.java.util.jar.pack.ConstantPool
com.sun.java.util.jar.pack.Constants
com.sun.java.util.jar.pack.Instruction
com.sun.java.util.jar.pack.NativeUnpack
com.sun.jdi.Bootstrap
com.sun.jlex.internal.Main
com.sun.jmx.interceptor.DefaultMBeanServerInterceptor
com.sun.jmx.mbeanserver.BaseMetaDataImpl
com.sun.jmx.mbeanserver.ClassLoaderRepositorySupport
com.sun.jmx.mbeanserver.DynamicMetaDataImpl
com.sun.jmx.mbeanserver.JmxMBeanServer
com.sun.jmx.mbeanserver.MBeanInstantiatorImpl
com.sun.jmx.mbeanserver.MetaDataImpl
com.sun.jmx.mbeanserver.RepositorySupport
com.sun.jmx.mbeanserver.StandardMBeanMetaDataImpl
com.sun.jmx.mbeanserver.StandardMetaDataImpl
com.sun.jmx.remote.internal.ArrayNotificationBuffer
com.sun.jmx.remote.internal.ClientCommunicatorAdmin
com.sun.jmx.remote.internal.ClientCommunicatorAdmin$Checker
com.sun.jmx.remote.internal.ClientNotifForwarder
com.sun.jmx.remote.internal.ClientNotifForwarder$LinearExecutor
com.sun.jmx.remote.internal.ClientNotifForwarder$LinearExecutor$1
com.sun.jmx.remote.internal.ClientNotifForwarder$NotifFetcher
com.sun.jmx.remote.internal.ServerCommunicatorAdmin
com.sun.jmx.remote.internal.ServerCommunicatorAdmin$Timeout
com.sun.jmx.remote.internal.ServerNotifForwarder
com.sun.jmx.remote.security.MBeanServerFileAccessController
com.sun.jmx.remote.security.SubjectDelegator
com.sun.jmx.remote.util.ClassLogger
com.sun.jmx.remote.util.EnvHelp
com.sun.jmx.snmp.IPAcl.ASCII_CharStream
com.sun.jmx.snmp.IPAcl.Parser
com.sun.jmx.snmp.IPAcl.SnmpAcl
```

```
com.sun.jmx.snmp.SnmpCounter64
com.sun.jmx.snmp.SnmpInt
com.sun.jmx.snmp.SnmpNull
com.sun.jmx.snmp.SnmpOid
com.sun.jmx.snmp.SnmpOidTableSupport
com.sun.jmx.snmp.SnmpParameters
com.sun.jmx.snmp.SnmpPeer
com.sun.jmx.snmp.SnmpString
com.sun.jmx.snmp.SnmpVarBind
com.sun.jmx.snmp.SnmpVarBindList
com.sun.jmx.snmp.Timestamp
com.sun.jmx.snmp.agent.SnmpMibTable
com.sun.jmx.snmp.agent.SnmpTableSupport
com.sun.jmx.snmp.daemon.CommunicatorServer
com.sun.jmx.snmp.daemon.SendQ
com.sun.jmx.snmp.daemon.SnmpAdaptorServer
com.sun.jmx.snmp.daemon.SnmpInformRequest
com.sun.jmx.snmp.daemon.SnmpRequestCounter
com.sun.jmx.snmp.daemon.SnmpResponseHandler
com.sun.jmx.snmp.daemon.SnmpSendServer
com.sun.jmx.snmp.daemon.SnmpSession
com.sun.jmx.snmp.daemon.SnmpSocket
com.sun.jmx.snmp.daemon.SnmpTimerServer
com.sun.jmx.snmp.daemon.WaitQ
com.sun.jmx.snmp.internal.SnmpEngineImpl
com.sun.jmx.snmp.tasks.ThreadService
com.sun.jmx.snmp.tasks.ThreadService$ExecutorThread
com.sun.jmx.trace.Trace
com.sun.jndi.dns.DnsClient
com.sun.jndi.dns.DnsContext
com.sun.jndi.dns.ZoneNode
com.sun.jndi.ldap.Connection
com.sun.jndi.ldap.EventQueue
com.sun.jndi.ldap.EventSupport
com.sun.jndi.ldap.LdapClient
com.sun.jndi.ldap.LdapCtx
com.sun.jndi.ldap.LdapRequest
com.sun.jndi.ldap.LdapSchemaCtx$SchemaInfo
com.sun.jndi.ldap.pool.ConnectionDesc
com.sun.jndi.ldap.pool.Connections
com.sun.jndi.ldap.pool.Pool
com.sun.jndi.ldap.pool.PoolCleaner
com.sun.jndi.toolkit.corba.CorbaUtils
com.sun.management.OSMBeanFactory
com.sun.management.OperatingSystem
com.sun.management.jmx.Introspector
com.sun.media.*
com.sun.naming.internal.FactoryEnumeration
com.sun.naming.internal.ResourceManager
com.sun.org.apache.bcel.internal.Constants
com.sun.org.apache.bcel.internal.util.ClassLoader
com.sun.org.apache.bcel.internal.util.InstructionFinder
com.sun.org.apache.bcel.internal.verifier.NativeVerifier
com.sun.org.apache.bcel.internal.verifier.VerifierAppFrame
com.sun.org.apache.bcel.internal.verifier.VerifierFactoryListModel
com.sun.org.apache.bcel.internal.verifier.VerifyDialog
com.sun.org.apache.bcel.internal.verifier.statics.Pass3aVerifier$InstOperandConstraintVisito
r
```

```
com.sun.org.apache.html.internal.dom.HTMLBuilder
com.sun.org.apache.html.internal.dom.HTMLCollectionImpl
com.sun.org.apache.html.internal.dom.HTMLDocumentImpl
com.sun.org.apache.html.internal.dom.HTMLTableElementImpl
com.sun.org.apache.html.internal.dom.ObjectFactory
com.sun.org.apache.regexp.internal.REDemo
com.sun.org.apache.regexp.internal.RETest
com.sun.org.apache.regexp.internal.recompile
com.sun.org.apache.xalan.internal.client.XSLTProcessorApplet
com.sun.org.apache.xalan.internal.client.XSLTProcessorApplet$TrustedAgent
com.sun.org.apache.xalan.internal.lib.ExsltStrings
com.sun.org.apache.xalan.internal.lib.Extensions
com.sun.org.apache.xalan.internal.lib.ObjectFactory
com.sun.org.apache.xalan.internal.xslt.ObjectFactory
com.sun.org.apache.xalan.internal.xslt.Process
com.sun.org.apache.xalan.internal.xsltc.cmdline.ObjectFactory
com.sun.org.apache.xalan.internal.xsltc.compiler.ObjectFactory
com.sun.org.apache.xalan.internal.xsltc.compiler.XSLTC
com.sun.org.apache.xalan.internal.xsltc.compiler.util.ObjectFactory
com.sun.org.apache.xalan.internal.xsltc.dom.DocumentCache
com.sun.org.apache.xalan.internal.xsltc.dom.ObjectFactory
com.sun.org.apache.xalan.internal.xsltc.runtime.ObjectFactory
com.sun.org.apache.xalan.internal.xsltc.trax.ObjectFactory
com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesHandlerImpl
com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl
com.sun.org.apache.xerces.internal.dom.CoreDOMImplementationImpl
com.sun.org.apache.xerces.internal.dom.ObjectFactory
com.sun.org.apache.xerces.internal.impl.XMLEntityManager
com.sun.org.apache.xerces.internal.impl.dv.DTDDVFactory
com.sun.org.apache.xerces.internal.impl.dv.ObjectFactory
com.sun.org.apache.xerces.internal.impl.dv.SchemaDVFactory
com.sun.org.apache.xerces.internal.impl.dv.xs.AbstractDateTimeDV$DateTimeData
com.sun.org.apache.xerces.internal.impl.dv.xs.Base64BinaryDV$XBase64
com.sun.org.apache.xerces.internal.impl.dv.xs.DecimalDV$XDecimal
com.sun.org.apache.xerces.internal.impl.dv.xs.DoubleDV$XDouble
com.sun.org.apache.xerces.internal.impl.dv.xs.FloatDV$XFloat
com.sun.org.apache.xerces.internal.impl.dv.xs.HexBinaryDV$XHex
com.sun.org.apache.xerces.internal.impl.dv.xs.ListDV$ListData
com.sun.org.apache.xerces.internal.impl.dv.xs.QNameDV$XQName
com.sun.org.apache.xerces.internal.impl.dv.xs.XSSimpleTypeDecl
com.sun.org.apache.xerces.internal.impl.xpath.regex.Match
com.sun.org.apache.xerces.internal.impl.xpath.regex.ParserForXMLSchema
com.sun.org.apache.xerces.internal.impl.xpath.regex.REUtil
com.sun.org.apache.xerces.internal.impl.xpath.regex.RangeToken
com.sun.org.apache.xerces.internal.impl.xpath.regex.RegexParser
com.sun.org.apache.xerces.internal.impl.xpath.regex.RegularExpression
com.sun.org.apache.xerces.internal.impl.xpath.regex.Token
com.sun.org.apache.xerces.internal.impl.xs.SchemaGrammar
com.sun.org.apache.xerces.internal.impl.xs.SchemaGrammar$BuiltinSchemaGrammar
com.sun.org.apache.xerces.internal.impl.xs.XSAnnotationImpl
com.sun.org.apache.xerces.internal.impl.xs.XSComplexTypeDecl
com.sun.org.apache.xerces.internal.impl.xs.XSModelImpl
com.sun.org.apache.xerces.internal.impl.xs.util.XSNamedMap4Types
com.sun.org.apache.xerces.internal.impl.xs.util.XSNamedMapImpl
com.sun.org.apache.xerces.internal.parsers.CachingParserPool$SynchronizedGrammarPool
com.sun.org.apache.xerces.internal.parsers.JAXPConfiguration
com.sun.org.apache.xerces.internal.parsers.ObjectFactory
com.sun.org.apache.xerces.internal.util.SynchronizedSymbolTable
```

```
com.sun.org.apache.xerces.internal.util.XMLGrammarPoolImpl
com.sun.org.apache.xerces.internal.xinclude.ObjectFactory
com.sun.org.apache.xml.internal.dtm.DTMException
com.sun.org.apache.xml.internal.dtm.DTMManager
com.sun.org.apache.xml.internal.dtm.FactoryFinder
com.sun.org.apache.xml.internal.dtm.ObjectFactory
com.sun.org.apache.xml.internal.dtm.ref.CoroutineManager
com.sun.org.apache.xml.internal.dtm.ref.DTMManagerDefault
com.sun.org.apache.xml.internal.dtm.ref.DTMSafeStringPool
com.sun.org.apache.xml.internal.dtm.ref.ObjectFactory
com.sun.org.apache.xml.internal.serialize.ObjectFactory
com.sun.org.apache.xml.internal.serialize.OutputFormat
com.sun.org.apache.xml.internal.serialize.SerializerFactory
com.sun.org.apache.xml.internal.serializer.CharInfo
com.sun.org.apache.xml.internal.serializer.ObjectFactory
com.sun.org.apache.xml.internal.serializer.OutputPropertiesFactory
com.sun.org.apache.xml.internal.serializer.ToHTMLStream
com.sun.org.apache.xml.internal.serializer.ToStream
com.sun.org.apache.xml.internal.utils.ObjectFactory
com.sun.org.apache.xml.internal.utils.ObjectPool
com.sun.org.apache.xml.internal.utils.StringBufferPool
com.sun.org.apache.xml.internal.utils.XMLReaderManager
com.sun.org.apache.xpath.internal.VariableStack
com.sun.org.apache.xpath.internal.axes.IteratorPool
com.sun.org.apache.xpath.internal.compiler.Compiler
com.sun.org.apache.xpath.internal.compiler.ObjectFactory
com.sun.org.apache.xpath.internal.functions.ObjectFactory
com.sun.org.omg.CORBA.AttrDescriptionSeqHelper
com.sun.org.omg.CORBA.AttributeDescriptionHelper
com.sun.org.omg.CORBA.AttributeModeHelper
com.sun.org.omg.CORBA.ContextIdSeqHelper
com.sun.org.omg.CORBA.ContextIdentifierHelper
com.sun.org.omg.CORBA.DefinitionKindHelper
com.sun.org.omg.CORBA.ExcDescriptionSeqHelper
com.sun.org.omg.CORBA.ExceptionDescriptionHelper
com.sun.org.omg.CORBA.IDLTypeHelper
com.sun.org.omg.CORBA.IdentifierHelper
com.sun.org.omg.CORBA.InitializerHelper
com.sun.org.omg.CORBA.InitializerSeqHelper
com.sun.org.omg.CORBA.OpDescriptionSeqHelper
com.sun.org.omg.CORBA.OperationDescriptionHelper
com.sun.org.omg.CORBA.OperationModeHelper
com.sun.org.omg.CORBA.ParDescriptionSeqHelper
com.sun.org.omg.CORBA.ParameterDescriptionHelper
com.sun.org.omg.CORBA.ParameterModeHelper
com.sun.org.omg.CORBA.RepositoryHelper
com.sun.org.omg.CORBA.RepositoryIdHelper
com.sun.org.omg.CORBA.RepositoryIdSeqHelper
com.sun.org.omg.CORBA.StructMemberHelper
com.sun.org.omg.CORBA.StructMemberSeqHelper
com.sun.org.omg.CORBA.ValueDefPackage.FullValueDescriptionHelper
com.sun.org.omg.CORBA.ValueMemberHelper
com.sun.org.omg.CORBA.ValueMemberSeqHelper
com.sun.org.omg.CORBA.VersionSpecHelper
com.sun.org.omg.CORBA.VisibilityHelper
com.sun.org.omg.SendingContext.CodeBaseHelper
com.sun.org.omg.SendingContext.CodeBasePackage.URLHelper
com.sun.org.omg.SendingContext.CodeBasePackage.URLSeqHelper
```

```
com.sun.org.omg.SendingContext.CodeBasePackage.ValueDescSeqHelper
com.sun.rmi.rmid.ExecOptionPermission
com.sun.rmi.rmid.ExecPermission
com.sun.rowset.JdbcRowSetImpl
com.sun.rowset.JdbcRowSetResourceBundle
com.sun.rowset.internal.WebRowSetXmlReader
com.sun.security.*
com.sun.swing.*
com.sun.tools.apt.comp.Apt
com.sun.tools.apt.main.Main
com.sun.tools.corba.se.idl.som.cff.Messages
com.sun.tools.corba.se.idl.toJavaPortable.Factories
com.sun.tools.corba.se.idl.toJavaPortable.Helper
com.sun.tools.doclets.internal.toolkit.builders.LayoutParser
com.sun.tools.doclets.internal.toolkit.util.Util
com.sun.tools.example.debug.expr.ASCII_UCodeESC_CharStream
com.sun.tools.example.debug.expr.ExpressionParserConstants
com.sun.tools.example.debug.expr.ExpressionParserTokenManager
com.sun.tools.example.debug.expr.LValue
com.sun.tools.example.debug.tty.Commands
com.sun.tools.example.debug.tty.Env
com.sun.tools.example.debug.tty.EventHandler
com.sun.tools.example.debug.tty.EventRequestSpec
com.sun.tools.example.debug.tty.EventRequestSpecList
com.sun.tools.example.debug.tty.MessageOutput
com.sun.tools.example.debug.tty.TTYResources
com.sun.tools.example.debug.tty.TTYResources_ja
com.sun.tools.example.debug.tty.TTYResources_zh_CN
com.sun.tools.example.debug.tty.ThreadInfo
com.sun.tools.example.debug.tty.VMConnection
com.sun.tools.javac.code.Flags
com.sun.tools.javac.jvm.ByteCodes
com.sun.tools.javac.jvm.Code$Mneumonics
com.sun.tools.javac.main.Main
com.sun.tools.javac.parser.Tokens
com.sun.tools.javac.resources.compiler
com.sun.tools.javac.resources.compiler_ja
com.sun.tools.javac.tree.Pretty
com.sun.tools.javac.util.Name$Table
com.sun.tools.javadoc.Start
com.sun.tools.jdi.AbstractLauncher$Helper
com.sun.tools.jdi.EventQueueImpl
com.sun.tools.jdi.EventQueueImpl$TimerThread
com.sun.tools.jdi.EventRequestManagerImpl
com.sun.tools.jdi.EventRequestManagerImpl$ClassVisibleEventRequestImpl
com.sun.tools.jdi.EventRequestManagerImpl$EventRequestImpl
com.sun.tools.jdi.EventRequestManagerImpl$ThreadVisibleEventRequestImpl
com.sun.tools.jdi.EventSetImpl
com.sun.tools.jdi.JNITypeParser
com.sun.tools.jdi.ObjectReferenceImpl
com.sun.tools.jdi.Packet
com.sun.tools.jdi.ReferenceTypeImpl
com.sun.tools.jdi.SharedMemoryConnection
com.sun.tools.jdi.SharedMemoryTransportService
com.sun.tools.jdi.SocketConnection
com.sun.tools.jdi.SocketTransportService
com.sun.tools.jdi.StackFrameImpl
com.sun.tools.jdi.TargetVM
```

```
com.sun.tools.jdi.TargetVM$EventController
com.sun.tools.jdi.ThreadReferenceImpl
com.sun.tools.jdi.VMState
com.sun.tools.jdi.VirtualMachineImpl
com.sun.tools.jdi.VirtualMachineManagerImpl
java.applet.*
java.awt.*
java.beans.BeansAppletContext
java.beans.DefaultPersistenceDelegate
java.beans.EventHandler
java.beans.EventSetDescriptor
java.beans.IndexedPropertyDescriptor
java.beans.Introspector
java.beans.MetaData
java.beans.MethodDescriptor
java.beans.PropertyChangeSupport
java.beans.PropertyDescriptor
java.beans.PropertyEditorManager
java.beans.PropertyEditorSupport
java.beans.ReflectionUtils
java.beans.VetoableChangeSupport
java.beans.beancontext.BeanContextChildSupport
java.beans.beancontext.BeanContextEvent
java.beans.beancontext.BeanContextServicesSupport
java.beans.beancontext.BeanContextServicesSupport$BCSSChild
java.beans.beancontext.BeanContextSupport
java.io.BufferedInputStream
java.io.BufferedOutputStream
java.io.BufferedReader
java.io.BufferedWriter
java.io.ByteArrayInputStream
java.io.ByteArrayOutputStream
java.io.CharArrayReader
java.io.CharArrayWriter
java.io.DataOutputStream
java.io.ExpiringCache
java.io.File
java.io.FileDescriptor
java.io.FileInputStream
java.io.FileOutputStream
java.io.FilePermissionCollection
java.io.FileSystem
java.io.FilterInputStream
java.io.IOException
java.io.InputStream
java.io.LineNumberReader
java.io.ObjectInputStream
java.io.ObjectInputStream$Caches
java.io.ObjectInputStream$CallbackContext
java.io.ObjectOutputStream
java.io.ObjectOutputStream$Caches
java.io.ObjectStreamClass
java.io.ObjectStreamClass$Caches
java.io.ObjectStreamClass$EntryFuture
java.io.OutputStream
java.io.PipedInputStream
java.io.PipedOutputStream
java.io.PipedReader
```

```
java.io.PipedWriter
java.io.PrintStream
java.io.PrintWriter
java.io.PushbackInputStream
java.io.PushbackReader
java.io.RandomAccessFile
java.io.Reader
java.io.Serializable
java.io.StringBufferInputStream
java.io.StringReader
java.io.Win32FileSystem
java.io.WinNTFileSystem
java.io.Writer
java.lang.AbstractMethodError
java.lang.ArithmeticException
java.lang.ArrayIndexOutOfBoundsException
java.lang.ArrayStoreException
java.lang.AssertionStatusDirectives
java.lang.Boolean
java.lang.Byte
java.lang.CharSequence
java.lang.Character
java.lang.Class
java.lang.ClassCastException
java.lang.ClassCircularityError
java.lang.ClassFormatError
java.lang.ClassLoader
java.lang.ClassLoader$NativeLibrary
java.lang.ClassNotFoundException
java.lang.CloneNotSupportedException
java.lang.Cloneable
java.lang.Compiler
java.lang.Double
java.lang.Error
java.lang.Exception
java.lang.ExceptionInInitializerError
java.lang.Float
java.lang.IllegalAccessError
java.lang.IllegalAccessException
java.lang.IllegalArgumentException
java.lang.IllegalMonitorStateException
java.lang.IllegalThreadStateException
java.lang.IncompatibleClassChangeError
java.lang.IndexOutOfBoundsException
java.lang.InstantiationError
java.lang.InstantiationException
java.lang.Integer
java.lang.InternalError
java.lang.InterruptedException
java.lang.InvalidClassException
java.lang.LinkageError
java.lang.Long
java.lang.Math
java.lang.NegativeArraySizeException
java.lang.NoClassDefFoundError
java.lang.NoSuchFieldError
java.lang.NoSuchFieldException
java.lang.NoSuchMethodError
```

```
java.lang.NoSuchMethodException
java.lang.NullPointerException
java.lang.Object
java.lang.OutOfMemoryError
java.lang.Package
java.lang.ProcessEnvironment
java.lang.ProcessImpl
java.lang.Runtime
java.lang.RuntimeException
java.lang.SecurityManager
java.lang.Short
java.lang.Shutdown
java.lang.StackOverflowError
java.lang.StackTraceElement
java.lang.StrictMath
java.lang.String
java.lang.StringBuffer
java.lang.StringIndexOutOfBoundsException
java.lang.System
java.lang.Thread
java.lang.ThreadDeath
java.lang.ThreadGroup
java.lang.ThreadLocal
java.lang.Throwable
java.lang.UnsatisfiedLinkError
java.lang.UnsupportedClassVersionError
java.lang.VerifyError
java.lang.management.ManagementFactory
java.lang.ref.FinalReference
java.lang.ref.Finalizer
java.lang.ref.Finalizer$3
java.lang.ref.PhantomReference
java.lang.ref.Reference
java.lang.ref.Reference$ReferenceHandler
java.lang.ref.ReferenceQueue
java.lang.ref.SoftReference
java.lang.ref.WeakReference
java.lang.reflect.AccessibleObject
java.lang.reflect.Array
java.lang.reflect.Constructor
java.lang.reflect.Field
java.lang.reflect.InvocationTargetException
java.lang.reflect.Method
java.lang.reflect.Modifier
java.lang.reflect.Proxy
java.math.BigDecimal
java.math.MathContext
java.net.Authenticator
java.net.CookieHandler
java.net.DatagramPacket
java.net.DatagramSocket
java.net.FactoryURLClassLoader
java.net.Inet4Address
java.net.Inet4AddressImpl
java.net.Inet6Address
java.net.Inet6AddressImpl
java.net.InetAddress
java.net.InetAddressImplFactory
```

```
java.net.MulticastSocket
java.net.NetworkInterface
java.net.PlainDatagramSocketImpl
java.net.PlainSocketImpl
java.net.ResponseCache
java.net.ServerSocket
java.net.Socket
java.net.SocketInputStream
java.net.SocketOutputStream
java.net.SocketPermission
java.net.SocketPermissionCollection
java.net.SocksSocketImpl
java.net.URL
java.net.URLConnection
java.net.URLStreamHandler
java.nio.Bits
java.nio.MappedByteBuffer
java.nio.channels.Channels
java.nio.channels.Channels$1
java.nio.channels.Channels$ReadableByteChannelImpl
java.nio.channels.Channels$WritableByteChannelImpl
java.nio.channels.spi.AbstractInterruptibleChannel
java.nio.channels.spi.AbstractInterruptibleChannel$1
java.nio.channels.spi.AbstractSelectableChannel
java.nio.channels.spi.AbstractSelectionKey
java.nio.channels.spi.AbstractSelector
java.nio.channels.spi.SelectorProvider
java.nio.charset.Charset
java.nio.charset.CoderResult$Cache
java.rmi.activation.ActivationGroup
java.rmi.server.LogStream
java.rmi.server.ObjID$InsecureRandom
java.rmi.server.RMISocketFactory
java.rmi.server.RemoteObjectInvocationHandler$MethodToHash_Maps$1
java.rmi.server.UID
java.security.*
java.sql.DriverManager
java.sql.SQLException
java.sql.Date
java.sql.Time
java.sql.Timestamp
java.sql.BatchUpdateException
java.sql.SQLWarning
java.text.AttributedString
java.text.AttributedString$AttributeMap
java.text.AttributedString$AttributedStringIterator
java.text.Bidi
java.text.BreakIterator
java.text.Collator
java.text.DecimalFormat
java.text.RuleBasedCollator
java.util.AbstractList
java.util.Calendar
java.util.Collections
java.util.Collections$SynchronizedCollection
java.util.Collections$SynchronizedList
java.util.Collections$SynchronizedMap
java.util.Collections$SynchronizedRandomAccessList
```

```
java.util.Collections$SynchronizedSet
java.util.Collections$SynchronizedSortedMap
java.util.Collections$SynchronizedSortedSet
java.util.Currency
java.util.Date
java.util.GregorianCalendar
java.util.Hashtable
java.util.Hashtable$Enumerator
java.util.ListResourceBundle
java.util.Locale
java.util.Observable
java.util.Properties
java.util.PropertyPermission
java.util.PropertyPermissionCollection
java.util.Random
java.util.ResourceBundle
java.util.SimpleTimeZone
java.util.Stack
java.util.TimeZone
java.util.TimeZone$DisplayNames
java.util.Timer
java.util.Timer$1
java.util.TimerTask
java.util.TimerThread
java.util.Vector
java.util.Vector$1
java.util.concurrent.*
java.util.jar.JarFile
java.util.jar.JarVerifier
java.util.jar.Pack200
java.util.logging.ErrorManager
java.util.logging.FileHandler
java.util.logging.Formatter
java.util.logging.Handler
java.util.logging.Level
java.util.logging.LogManager
java.util.logging.LogManager$Cleaner
java.util.logging.LogRecord
java.util.logging.Logger
java.util.logging.MemoryHandler
java.util.logging.SimpleFormatter
java.util.logging.SocketHandler
java.util.logging.StreamHandler
java.util.prefs.AbstractPreferences
java.util.prefs.AbstractPreferences$EventDispatchThread
java.util.prefs.WindowsPreferences
java.util.prefs.XmlSupport
java.util.regex.Pattern
java.util.zip.*
javax.imageio.*
javax.management.AttributeChangeNotificationFilter
javax.management.MBeanInfo
javax.management.MBeanServerDelegate
javax.management.MBeanServerFactory
javax.management.MBeanServerPermission
javax.management.MBeanServerPermissionCollection
javax.management.NotificationBroadcasterSupport
javax.management.NotificationFilterSupport
```

```
javax.management.ObjectName
javax.management.StandardMBean
javax.management.loading.MLet
javax.management.modelmbean.DescriptorSupport
javax.management.modelmbean.RequiredModelMBean
javax.management.monitor.CounterMonitor
javax.management.monitor.GaugeMonitor
javax.management.monitor.Monitor
javax.management.monitor.StringMonitor
javax.management.relation.MBeanServerNotificationFilter
javax.management.relation.RelationService
javax.management.relation.RelationSupport
javax.management.remote.JMXConnectorServer
javax.management.remote.rmi.RMIConnectionImpl
javax.management.remote.rmi.RMIConnector
javax.management.remote.rmi.RMIConnector$RMIClientCommunicatorAdmin
javax.management.remote.rmi.RMIConnectorServer
javax.management.remote.rmi.RMIServerImpl
javax.management.timer.Timer
javax.naming.spi.NamingManager
javax.print.*
javax.rmi.ssl.SslRMIClientSocketFactory
javax.rmi.ssl.SslRMIServerSocketFactory
javax.security.*
javax.sound.*
javax.sql.ConnectionEvent
javax.sql.rowset.spi.SyncFactory
javax.swing.*
javax.xml.datatype.FactoryFinder
javax.xml.parsers.FactoryFinder
javax.xml.transform.FactoryFinder
javax.xml.transform.TransformerException
javax.xml.validation.SchemaFactoryFinder
javax.xml.xpath.XPathFactoryFinder
org.omg.CORBA.AnySeqHelper
org.omg.CORBA.BooleanSeqHelper
org.omg.CORBA.CharSeqHelper
org.omg.CORBA.CompletionStatusHelper
org.omg.CORBA.CurrentHelper
org.omg.CORBA.DefinitionKindHelper
org.omg.CORBA.DoubleSeqHelper
org.omg.CORBA.FieldNameHelper
org.omg.CORBA.FloatSeqHelper
org.omg.CORBA.IDLTypeHelper
org.omg.CORBA.IdentifierHelper
org.omg.CORBA.LongLongSeqHelper
org.omg.CORBA.LongSeqHelper
org.omg.CORBA.NameValuePairHelper
org.omg.CORBA.ORB
org.omg.CORBA.ObjectHelper
org.omg.CORBA.OctetSeqHelper
org.omg.CORBA.ParameterModeHelper
org.omg.CORBA.PolicyErrorCodeHelper
org.omg.CORBA.PolicyErrorHelper
org.omg.CORBA.PolicyHelper
org.omg.CORBA.PolicyListHelper
org.omg.CORBA.PolicyTypeHelper
org.omg.CORBA.RepositoryIdHelper
```

```
org.omg.CORBA.ServiceDetailHelper
org.omg.CORBA.ServiceInformationHelper
org.omg.CORBA.SetOverrideTypeHelper
org.omg.CORBA.ShortSeqHelper
org.omg.CORBA.StringSeqHelper
org.omg.CORBA.StringValueHelper
org.omg.CORBA.StructMemberHelper
org.omg.CORBA.ULongLongSeqHelper
org.omg.CORBA.ULongSeqHelper
org.omg.CORBA.UShortSeqHelper
org.omg.CORBA.UnionMemberHelper
org.omg.CORBA.UnknownUserExceptionHelper
org.omg.CORBA.ValueBaseHelper
org.omg.CORBA.ValueMemberHelper
org.omg.CORBA.VersionSpecHelper
org.omg.CORBA.VisibilityHelper
org.omg.CORBA.WCharSeqHelper
org.omg.CORBA.WStringSeqHelper
org.omg.CORBA.WStringValueHelper
org.omg.CORBA.WrongTransactionHelper
org.omg.CosNaming.BindingHelper
org.omg.CosNaming.BindingIteratorHelper
org.omg.CosNaming.BindingListHelper
org.omg.CosNaming.BindingTypeHelper
org.omg.CosNaming.IstringHelper
org.omg.CosNaming.NameComponentHelper
org.omg.CosNaming.NameHelper
org.omg.CosNaming.NamingContextExtHelper
org.omg.CosNaming.NamingContextExtPackage.AddressHelper
org.omg.CosNaming.NamingContextExtPackage.InvalidAddressHelper
org.omg.CosNaming.NamingContextExtPackage.StringNameHelper
org.omg.CosNaming.NamingContextExtPackage.URLStringHelper
org.omg.CosNaming.NamingContextHelper
org.omg.CosNaming.NamingContextPackage.AlreadyBoundHelper
org.omg.CosNaming.NamingContextPackage.CannotProceedHelper
org.omg.CosNaming.NamingContextPackage.InvalidNameHelper
org.omg.CosNaming.NamingContextPackage.NotEmptyHelper
org.omg.CosNaming.NamingContextPackage.NotFoundHelper
org.omg.CosNaming.NamingContextPackage.NotFoundReasonHelper
org.omg.DynamicAny.AnySeqHelper
org.omg.DynamicAny.DynAnyFactoryHelper
org.omg.DynamicAny.DynAnyFactoryPackage.InconsistentTypeCodeHelper
org.omg.DynamicAny.DynAnyHelper
org.omg.DynamicAny.DynAnyPackage.InvalidValueHelper
org.omg.DynamicAny.DynAnyPackage.TypeMismatchHelper
org.omg.DynamicAny.DynAnySeqHelper
org.omg.DynamicAny.DynArrayHelper
org.omg.DynamicAny.DynEnumHelper
org.omg.DynamicAny.DynFixedHelper
org.omg.DynamicAny.DynSequenceHelper
org.omg.DynamicAny.DynStructHelper
org.omg.DynamicAny.DynUnionHelper
org.omg.DynamicAny.DynValueHelper
org.omg.DynamicAny.FieldNameHelper
org.omg.DynamicAny.NameDynAnyPairHelper
org.omg.DynamicAny.NameDynAnyPairSeqHelper
org.omg.DynamicAny.NameValuePairHelper
org.omg.DynamicAny.NameValuePairSeqHelper
```

```
org.omg.IOP.CodecFactoryHelper
org.omg.IOP.CodecFactoryPackage.UnknownEncodingHelper
org.omg.IOP.CodecPackage.FormatMismatchHelper
org.omg.IOP.CodecPackage.InvalidTypeForEncodingHelper
org.omg.IOP.CodecPackage.TypeMismatchHelper
org.omg.IOP.ComponentIdHelper
org.omg.IOP.IORHelper
org.omg.IOP.MultipleComponentProfileHelper
org.omg.IOP.ProfileIdHelper
org.omg.IOP.ServiceContextHelper
org.omg.IOP.ServiceContextListHelper
org.omg.IOP.ServiceIdHelper
org.omg.IOP.TaggedComponentHelper
org.omg.IOP.TaggedProfileHelper
org.omg.Messaging.SyncScopeHelper
org.omg.PortableInterceptor.AdapterManagerIdHelper
org.omg.PortableInterceptor.AdapterNameHelper
org.omg.PortableInterceptor.AdapterStateHelper
org.omg.PortableInterceptor.CurrentHelper
org.omg.PortableInterceptor.ForwardRequestHelper
org.omg.PortableInterceptor.IORInterceptor_3_0Helper
org.omg.PortableInterceptor.InvalidSlotHelper
org.omg.PortableInterceptor.ORBIdHelper
org.omg.PortableInterceptor.ORBInitInfoPackage.DuplicateNameHelper
org.omg.PortableInterceptor.ORBInitInfoPackage.InvalidNameHelper
org.omg.PortableInterceptor.ORBInitInfoPackage.ObjectIdHelper
org.omg.PortableInterceptor.ObjectIdHelper
org.omg.PortableInterceptor.ObjectReferenceFactoryHelper
org.omg.PortableInterceptor.ObjectReferenceTemplateHelper
org.omg.PortableInterceptor.ObjectReferenceTemplateSeqHelper
org.omg.PortableInterceptor.ServerIdHelper
org.omg.PortableServer.CurrentHelper
org.omg.PortableServer.CurrentPackage.NoContextHelper
org.omg.PortableServer.ForwardRequestHelper
org.omg.PortableServer.POAHelper
org.omg.PortableServer.POAManagerPackage.AdapterInactiveHelper
org.omg.PortableServer.POAPackage.AdapterAlreadyExistsHelper
org.omg.PortableServer.POAPackage.AdapterNonExistentHelper
org.omg.PortableServer.POAPackage.InvalidPolicyHelper
org.omg.PortableServer.POAPackage.NoServantHelper
org.omg.PortableServer.POAPackage.ObjectAlreadyActiveHelper
org.omg.PortableServer.POAPackage.ObjectNotActiveHelper
org.omg.PortableServer.POAPackage.ServantAlreadyActiveHelper
org.omg.PortableServer.POAPackage.ServantNotActiveHelper
org.omg.PortableServer.POAPackage.WrongAdapterHelper
org.omg.PortableServer.POAPackage.WrongPolicyHelper
org.omg.PortableServer.ServantActivatorHelper
org.omg.PortableServer.ServantLocatorHelper
org.omg.stub.javax.management.remote.rmi._RMIConnectionImpl_Tie
org.omg.stub.javax.management.remote.rmi._RMIConnection_Stub
org.omg.stub.javax.management.remote.rmi._RMIServerImpl_Tie
org.omg.stub.javax.management.remote.rmi._RMIServer_Stub
sun.applet.*
sun.audio.*
sun.awt.*
sun.corba.Bridge
sun.dc.pr.*
sun.font.AdvanceCache
```

```
sun.font.CompositeFont
sun.font.FileFont
sun.font.FileFont$FileFontDisposer
sun.font.FileFontStrike
sun.font.Font2D
sun.font.FontFamily
sun.font.FontManager
sun.font.FontStrikeDisposer
sun.font.GlyphLayout
sun.font.GlyphList
sun.font.PhysicalStrike
sun.font.StrikeCache
sun.font.SunLayoutEngine
sun.font.TrueTypeFont
sun.font.TrueTypeFont$TTDisposerRecord
sun.font.Type1Font
sun.instrument.InstrumentationImpl
sun.instrument.TransformerManager
sun.io.CharacterEncoding
sun.io.Converters
sun.java2d.*
sun.jdbc.*
sun.jvmstat.monitor.MonitoredHost
sun.jvmstat.perfdata.monitor.PerfDataBufferImpl
sun.jvmstat.perfdata.monitor.protocol.local.LocalEventTimer
sun.jvmstat.perfdata.monitor.protocol.local.LocalMonitoredVm
sun.jvmstat.perfdata.monitor.protocol.local.LocalVmManager
sun.jvmstat.perfdata.monitor.protocol.local.MonitoredHostProvider
sun.jvmstat.perfdata.monitor.protocol.rmi.MonitoredHostProvider
sun.jvmstat.perfdata.monitor.protocol.rmi.PerfDataBuffer
sun.jvmstat.perfdata.monitor.protocol.rmi.RemoteMonitoredVm
sun.management.Agent
sun.management.ClassLoadingImpl
sun.management.FileSystem
sun.management.FileSystemImpl
sun.management.GarbageCollectorImpl
sun.management.GcInfoBuilder
sun.management.GcInfoCompositeData
sun.management.HotspotRuntime
sun.management.HotspotThread
sun.management.LazyCompositeData
sun.management.MXBeanSupport
sun.management.ManagementFactory
sun.management.MappedMXBeanType
sun.management.MemoryImpl
sun.management.MemoryManagerImpl
sun.management.MemoryPoolImpl
sun.management.NotificationEmitterSupport
sun.management.Sensor
sun.management.ThreadImpl
sun.management.VMManagementImpl
sun.management.counter.perf.PerfInstrumentation
sun.management.jmxremote.ConnectorBootstrap
sun.management.jmxremote.ConnectorBootstrap$PermanentExporter
sun.management.snmp.AdaptorBootstrap
sun.management.snmp.jvminstr.JVM_MANAGEMENT_MIB_IMPL
sun.management.snmp.jvminstr.JvmMemPoolEntryImpl
sun.management.snmp.jvminstr.JvmThreadingImpl
```

```
sun.management.snmp.jvmmib.JvmMemGCTableMeta
sun.management.snmp.jvmmib.JvmMemManagerTableMeta
sun.management.snmp.jvmmib.JvmMemMgrPoolRelTableMeta
sun.management.snmp.jvmmib.JvmMemPoolTableMeta
sun.management.snmp.jvmmib.JvmRTBootClassPathTableMeta
sun.management.snmp.jvmmib.JvmRTClassPathTableMeta
sun.management.snmp.jvmmib.JvmRTInputArgsTableMeta
sun.management.snmp.jvmmib.JvmRTLibraryPathTableMeta
sun.management.snmp.jvmmib.JvmThreadInstanceTableMeta
sun.management.snmp.util.JvmContextFactory
sun.management.snmp.util.SnmpTableCache
sun.misc.AtomicLong
sun.misc.AtomicLongCSImpl
sun.misc.AtomicLongLockImpl
sun.misc.Cache
sun.misc.ClassFileTransformer
sun.misc.Cleaner
sun.misc.ConditionLock
sun.misc.ExtensionDependency
sun.misc.FIFOQueueEnumerator
sun.misc.FloatingDecimal
sun.misc.FloatingDecimal$1
sun.misc.FormattedFloatingDecimal
sun.misc.FormattedFloatingDecimal$1
sun.misc.GC
sun.misc.GC$Daemon
sun.misc.GC$LatencyRequest
sun.misc.LIFOQueueEnumerator
sun.misc.Launcher$AppClassLoader
sun.misc.Lock
sun.misc.MessageUtils
sun.misc.NativeSignalHandler
sun.misc.PathPermissions
sun.misc.Perf
sun.misc.Perf$1
sun.misc.PerformanceLogger
sun.misc.Queue
sun.misc.Ref
sun.misc.RequestProcessor
sun.misc.Resource
sun.misc.Signal
sun.misc.Timer
sun.misc.TimerThread
sun.misc.TimerTickThread
sun.misc.URLClassPath
sun.misc.Unsafe
sun.misc.VM
sun.net.InetAddressCachePolicy
sun.net.ProgressMonitor
sun.net.dns.ResolverConfiguration
sun.net.dns.ResolverConfigurationImpl
sun.net.dns.ResolverConfigurationImpl$AddressChangeListener
sun.net.ftp.FtpClient
sun.net.spi.DefaultProxySelector
sun.net.spi.DefaultProxySelector$2
sun.net.www.HeaderParser
sun.net.www.MessageHeader
sun.net.www.MessageHeader$HeaderIterator
```

```
sun.net.www.MeteredStream
sun.net.www.MimeEntry
sun.net.www.MimeTable
sun.net.www.http.ChunkedInputStream
sun.net.www.http.ChunkedOutputStream
sun.net.www.http.ClientVector
sun.net.www.http.HttpClient
sun.net.www.http.KeepAliveCache
sun.net.www.http.KeepAliveStream
sun.net.www.http.PosterOutputStream
sun.net.www.protocol.doc.DocURLConnection
sun.net.www.protocol.doc.Handler
sun.net.www.protocol.file.FileURLConnection
sun.net.www.protocol.file.Handler
sun.net.www.protocol.ftp.FtpURLConnection
sun.net.www.protocol.http.AuthCacheImpl
sun.net.www.protocol.http.AuthenticationInfo
sun.net.www.protocol.http.DigestAuthentication$Parameters
sun.net.www.protocol.http.HttpURLConnection
sun.net.www.protocol.http.NTLMAuthSequence
sun.net.www.protocol.http.NTLMAuthentication
sun.net.www.protocol.jar.JarFileFactory
sun.net.www.protocol.jar.URLJarFile
sun.net.www.protocol.mailto.Handler
sun.net.www.protocol.mailto.MailToURLConnection
sun.net.www.protocol.netdoc.Handler
sun.nio.ch.AllocatedNativeObject
sun.nio.ch.ChannelInputStream
sun.nio.ch.DatagramChannelImpl
sun.nio.ch.DatagramDispatcher
sun.nio.ch.DatagramSocketAdaptor
sun.nio.ch.FileChannelImpl
sun.nio.ch.FileDispatcher
sun.nio.ch.FileLockImpl
sun.nio.ch.IOUtil
sun.nio.ch.NativeThreadSet
sun.nio.ch.Net
sun.nio.ch.SelectorImpl
sun.nio.ch.ServerSocketAdaptor
sun.nio.ch.ServerSocketChannelImpl
sun.nio.ch.SocketAdaptor
sun.nio.ch.SocketAdaptor$SocketInputStream
sun.nio.ch.SocketChannelImpl
sun.nio.ch.SocketDispatcher
sun.nio.ch.Util
sun.nio.ch.Util$SelectorWrapper$Closer
sun.nio.ch.WindowsSelectorImpl
sun.nio.ch.WindowsSelectorImpl$FinishLock
sun.nio.ch.WindowsSelectorImpl$StartLock
sun.nio.ch.WindowsSelectorImpl$SubSelector
sun.nio.cs.AbstractCharsetProvider
sun.nio.cs.FastCharsetProvider
sun.nio.cs.StreamDecoder
sun.nio.cs.StreamEncoder
sun.print.*
sun.reflect.ConstantPool
sun.reflect.MethodAccessorGenerator
sun.reflect.NativeConstructorAccessorImpl
```

```
sun.reflect.NativeMethodAccessorImpl
sun.reflect.Reflection
sun.reflect.annotation.AnnotationType
sun.reflect.misc.MethodUtil
sun.rmi.log.ReliableLog
sun.rmi.registry.RegistryImpl
sun.rmi.rmic.BatchEnvironment
sun.rmi.rmic.Main
sun.rmi.rmic.iiop.DirectoryLoader
sun.rmi.rmic.iiop.NameContext
sun.rmi.rmic.newrmic.Main
sun.rmi.runtime.Log$LogStreamLog
sun.rmi.runtime.Log$LoggerLog
sun.rmi.runtime.ThreadPool
sun.rmi.runtime.ThreadPool$Worker
sun.rmi.server.ActivatableRef
sun.rmi.server.Activation
sun.rmi.server.Activation$ActivationSystemImpl
sun.rmi.server.Activation$DelayedAcceptServerSocket
sun.rmi.server.Activation$GroupEntry
sun.rmi.server.Activation$GroupEntry$Watchdog
sun.rmi.server.Activation$ObjectEntry
sun.rmi.server.Activation$Shutdown
sun.rmi.server.Activation$ShutdownHook
sun.rmi.server.ActivationGroupImpl
sun.rmi.server.LoaderHandler
sun.rmi.server.MarshalInputStream
sun.rmi.server.PipeWriter
sun.rmi.server.UnicastRef
sun.rmi.server.UnicastServerRef
sun.rmi.server.Util
sun.rmi.server.WeakClassHashMap
sun.rmi.transport.DGCAckHandler
sun.rmi.transport.DGCClient
sun.rmi.transport.DGCClient$EndpointEntry
sun.rmi.transport.DGCClient$EndpointEntry$RenewCleanThread
sun.rmi.transport.DGCImpl
sun.rmi.transport.DGCImpl$LeaseInfo
sun.rmi.transport.ObjectTable
sun.rmi.transport.ObjectTable$Reaper
sun.rmi.transport.Target
sun.rmi.transport.WeakRef
sun.rmi.transport.proxy.HttpOutputStream
sun.rmi.transport.proxy.HttpReceiveSocket
sun.rmi.transport.proxy.HttpSendInputStream
sun.rmi.transport.proxy.HttpSendSocket
sun.rmi.transport.proxy.RMIMasterSocketFactory
sun.rmi.transport.proxy.RMIMasterSocketFactory$AsyncConnector
sun.rmi.transport.proxy.WrappedSocket
sun.rmi.transport.tcp.ConnectionAcceptor
sun.rmi.transport.tcp.ConnectionMultiplexer
sun.rmi.transport.tcp.MultiplexInputStream
sun.rmi.transport.tcp.MultiplexOutputStream
sun.rmi.transport.tcp.TCPChannel
sun.rmi.transport.tcp.TCPEndpoint
sun.rmi.transport.tcp.TCPEndpoint$FQDN
sun.rmi.transport.tcp.TCPTransport
sun.rmi.transport.tcp.TCPTransport$ConnectionHandler
```

```
sun.security.*
sun.swing.*
sun.text.resources.DateFormatZoneData
sun.text.resources.LocaleData
sun.tools.asm.SwitchData
sun.tools.hprof.Tracker
sun.tools.jar.JarVerifierStream
sun.tools.jar.Main
sun.tools.java.Constants
sun.tools.java.Identifier
sun.tools.java.Imports
sun.tools.java.MemberDefinition
sun.tools.java.RuntimeConstants
sun.tools.java.Type
sun.tools.javac.Main
sun.tools.javac.SourceClass
sun.tools.javap.ClassData
sun.tools.javap.Constants
sun.tools.javap.MethodData
sun.tools.javap.RuntimeConstants
sun.tools.javap.Tables
sun.tools.jstat.JStatLogger
sun.tools.jstat.Jstat$2
sun.tools.jstatd.RemoteHostImpl
sun.tools.native2ascii.Main
sun.tools.serialver.SerialVer
sun.tools.tree.SynchronizedStatement
sun.util.calendar.CalendarSystem
sun.util.calendar.ZoneInfo
sun.util.calendar.ZoneInfoFile
javax.crypto.*
javax.net.ServerSocketFactory
javax.net.DefaultServerSocketFactory
javax.net.SocketFactory
javax.net.DefaultSocketFactory
javax.net.ssl.*
sun.net.www.protocol.https.*
com.sun.net.ssl.*
com.sun.management.UnixOperatingSystem
java.io.UnixFileSystem
java.lang.UNIXProcess
java.lang.UNIXProcess$1$1
java.lang.UNIXProcess$2$1
java.lang.UNIXProcess$DeferredCloseInputStream
java.lang.UNIXProcess$Gate
java.util.prefs.FileSystemPreferences
sun.font.NativeStrikeDisposer
sun.font.XMap
sun.io.CharToByteCOMPOUND_TEXT
sun.nio.ch.DevPollArrayWrapper
sun.nio.ch.DevPollSelectorImpl
sun.nio.ch.InheritedChannel
sun.nio.ch.PollSelectorImpl
com.sun.java.util.jar.pack.PropMap
com.sun.java.util.jar.pack.UnpackerImpl
com.sun.org.apache.bcel.internal.verifier.statics.Pass2Verifier$CPESSC_Visitor
com.sun.tools.corba.se.idl.Token
com.sun.tools.javac.comp.Attr
```

```
com.sun.tools.javac.resources.compiler_zh_CN
com.sun.tools.javah.oldjavah.OldHeaders
sun.font.NativeFont
sun.font.NativeStrike
sun.font.X11TextRenderer
sun.jvm.hotspot.HotSpotAgent
sun.jvm.hotspot.HotSpotAgent$1
sun.jvm.hotspot.bugspot.BugSpotAgent
sun.jvm.hotspot.bugspot.BugSpotAgent$1
sun.jvm.hotspot.debugger.PageCache
sun.jvm.hotspot.debugger.dbx.DbxDebuggerLocal
sun.jvm.hotspot.debugger.linux.LinuxDebuggerLocal
sun.jvm.hotspot.debugger.linux.LinuxDebuggerLocal$LinuxDebuggerLocalWorkerThread
sun.jvm.hotspot.debugger.proc.ProcDebuggerLocal
sun.jvm.hotspot.debugger.win32.Win32DebuggerLocal
sun.jvm.hotspot.debugger.windbg.WindbgDebuggerLocal
sun.jvm.hotspot.interpreter.Bytecodes
sun.jvm.hotspot.interpreter.OopMapCacheEntry
sun.jvm.hotspot.interpreter.OopMapForCacheEntry
sun.jvm.hotspot.jdi.SAJDIClassLoader
sun.jvm.hotspot.jdi.VirtualMachineImpl
sun.jvm.hotspot.oops.AccessFlags
sun.jvm.hotspot.tools.JStack
sun.jvm.hotspot.utilities.HeapGXLWriter
sun.jvm.hotspot.utilities.HeapHprofBinWriter
sun.jvm.hotspot.utilities.MessageQueueBackend$MessageQueueImpl
sun.jvm.hotspot.utilities.StreamMonitor
sun.jvm.hotspot.utilities.SystemDictionaryHelper
sun.nio.ch.NativeThread
sun.nio.ch.SinkChannelImpl
sun.nio.ch.SourceChannelImpl
sun.rmi.rmic.iiop.Constants
sun.tools.jconsole.ConnectDialog$ManagedVmTableModel
sun.tools.jconsole.ProxyClient
sun.tools.jconsole.Resources
sun.tools.jconsole.Tab
sun.tools.jconsole.VMPanel
sun.tools.jconsole.Worker
sun.tools.jconsole.inspector.Utils
sun.tools.jconsole.inspector.XMBeanAttributes
sun.tools.jconsole.inspector.XMBeanAttributes$AttributesListener$1
sun.tools.jconsole.inspector.XMBeanNotifications
sun.tools.jconsole.inspector.XMBeanNotifications$XMBeanNotificationsListener
sun.tools.jconsole.inspector.XMBeanTree
sun.tools.jconsole.inspector.XObject
sun.tools.jconsole.inspector.XOperations
sun.tools.jconsole.inspector.XOperations$1
sun.tools.jconsole.inspector.XSheet
sun.tools.jconsole.inspector.XSheet$1
sun.tools.jconsole.inspector.XSheet$3
sun.tools.jconsole.inspector.XSheet$4
sun.tools.jconsole.inspector.XSheet$XMBeanPane
sun.tools.jconsole.inspector.XTree
com.sun.jndi.cosnaming.CNCtx
com.sun.jndi.cosnaming.OrbReuseTracker
com.sun.org.apache.xml.internal.utils.ThreadControllerWrapper$ThreadController$SafeThread
com.sun.servicetag.Installer
com.sun.servicetag.RegistrationData
```

```
com.sun.servicetag.SystemEnvironment
com.sun.servicetag.Util
sun.font.CreatedFontTracker
sun.font.Type1Font$T1DisposerRecord
sun.management.ConnectorAddressLink
sun.net.www.URLConnection
sun.nio.ch.EPollArrayWrapper
sun.nio.ch.EPollSelectorImpl
sun.net.ResourceManager

#-------------------------------------------------------------------
# J2SE (JDK6 以降)
#-------------------------------------------------------------------
com.sun.activation.registries.LogSupport
com.sun.activation.registries.MailcapFile
com.sun.codemodel.internal.CodeWriter
com.sun.codemodel.internal.util.EncoderFactory
com.sun.istack.internal.Pool$Impl
com.sun.istack.internal.tools.MaskingClassLoader
com.sun.jmx.mbeanserver.Introspector
com.sun.jmx.mbeanserver.MBeanInstantiator
com.sun.jmx.mbeanserver.MBeanIntrospector
com.sun.jmx.mbeanserver.MXBeanLookup
com.sun.jmx.mbeanserver.MXBeanSupport
com.sun.jmx.mbeanserver.OpenConverter
com.sun.jmx.mbeanserver.Repository
com.sun.jmx.mbeanserver.StandardMBeanIntrospector
com.sun.jmx.mbeanserver.Util
com.sun.jmx.remote.internal.ServerNotifForwarder$2
com.sun.net.httpserver.spi.HttpServerProvider
com.sun.org.apache.regexp.internal.RECompiler
com.sun.org.apache.regexp.internal.RETestCase
com.sun.org.apache.xalan.internal.xsltc.trax.SAX2DOM
com.sun.org.apache.xerces.internal.impl.dv.xs.PrecisionDecimalDV$XPrecisionDecimal
com.sun.org.apache.xerces.internal.impl.xs.ElementPSVImpl
com.sun.org.apache.xerces.internal.impl.xs.SchemaGrammar$Schema4Annotations
com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
com.sun.org.apache.xerces.internal.jaxp.validation.SoftReferenceGrammarPool
com.sun.org.apache.xerces.internal.jaxp.validation.WeakReferenceXMLSchema
com.sun.org.apache.xerces.internal.util.XMLCatalogResolver
com.sun.org.apache.xml.internal.resolver.Catalog
com.sun.org.apache.xml.internal.security.Init
com.sun.org.apache.xml.internal.security.exceptions.XMLSecurityException
com.sun.org.apache.xml.internal.security.exceptions.XMLSecurityRuntimeException
com.sun.org.apache.xml.internal.security.keys.content.x509.XMLX509SKI
com.sun.org.apache.xml.internal.security.utils.I18n
com.sun.org.apache.xml.internal.security.utils.IdResolver
com.sun.tools.attach.spi.AttachProvider
com.sun.tools.hat.internal.model.Snapshot
com.sun.tools.hat.internal.oql.OQLEngine
com.sun.tools.hat.internal.parser.FileReadBuffer
com.sun.tools.hat.internal.parser.MappedReadBuffer
com.sun.tools.hat.internal.server.PlatformClasses
com.sun.tools.internal.ws.processor.model.AsyncOperation
com.sun.tools.internal.ws.processor.util.IndentingWriter
com.sun.tools.internal.ws.wscompile.Options
com.sun.tools.internal.ws.wsdl.framework.TWSDLParserContextImpl
com.sun.tools.internal.xjc.Driver$1
```

```
com.sun.tools.internal.xjc.SchemaCache
com.sun.tools.internal.xjc.addon.sync.SynchronizedMethodAddOn
com.sun.tools.internal.xjc.api.impl.s2j.SchemaCompilerImpl
com.sun.tools.internal.xjc.reader.xmlschema.bindinfo.BindInfo
com.sun.tools.javac.api.JavacTaskImpl
com.sun.tools.javac.api.JavacTaskImpl$1
com.sun.tools.javac.code.Symbol$VarSymbol
com.sun.tools.javac.processing.JavacFiler
com.sun.tools.javac.processing.JavacFiler$FilerOutputFileObject
com.sun.tools.javac.processing.JavacFiler$FilerOutputStream
com.sun.tools.javac.processing.JavacFiler$FilerWriter
com.sun.tools.javac.processing.JavacProcessingEnvironment$DiscoveredProcessors$ProcessorStat
eIterator
com.sun.tools.javac.processing.JavacProcessingEnvironment$NameProcessIterator
com.sun.tools.javac.processing.JavacProcessingEnvironment$ProcessorState
com.sun.tools.javac.sym.CreateSymbols
com.sun.tools.javac.util.DefaultFileManager
com.sun.tools.javac.util.Log
com.sun.tools.javac.util.Paths
com.sun.tools.javac.zip.ZipFileIndex
com.sun.tools.jconsole.JConsolePlugin
com.sun.tools.jdi.EventRequestManagerImpl$ClassPrepareRequestImpl
com.sun.tools.jdi.MonitorInfoImpl
com.sun.xml.internal.bind.AccessorFactoryImpl
com.sun.xml.internal.bind.v2.runtime.JAXBContextImpl
com.sun.xml.internal.bind.v2.runtime.MarshallerImpl
com.sun.xml.internal.bind.v2.runtime.reflect.Lister
com.sun.xml.internal.bind.v2.runtime.reflect.Lister$IDREFS$Pack
com.sun.xml.internal.bind.v2.runtime.reflect.TransducedAccessor$IDREFTransducedAccessorImpl
com.sun.xml.internal.bind.v2.runtime.reflect.TransducedAccessor$IDREFTransducedAccessorImpl$
1
com.sun.xml.internal.bind.v2.runtime.reflect.opt.Injector
com.sun.xml.internal.bind.v2.runtime.unmarshaller.UnmarshallingContext
com.sun.xml.internal.fastinfoset.AbstractResourceBundle
com.sun.xml.internal.fastinfoset.CommonResourceBundle
com.sun.xml.internal.messaging.saaj.SOAPExceptionImpl
com.sun.xml.internal.messaging.saaj.packaging.mime.MessagingException
com.sun.xml.internal.messaging.saaj.packaging.mime.internet.MimeMultipart
com.sun.xml.internal.messaging.saaj.soap.AttachmentPartImpl
com.sun.xml.internal.messaging.saaj.soap.MessageImpl
com.sun.xml.internal.messaging.saaj.util.ParserPool
com.sun.xml.internal.messaging.saaj.util.TeeInputStream
com.sun.xml.internal.rngom.parse.compact.JavaCharStream
com.sun.xml.internal.ws.api.pipe.Engine
com.sun.xml.internal.ws.api.pipe.Engine$DaemonThreadFactory
com.sun.xml.internal.ws.api.pipe.Fiber
com.sun.xml.internal.ws.api.streaming.XMLStreamReaderFactory$Default
com.sun.xml.internal.ws.api.streaming.XMLStreamWriterFactory$Default
com.sun.xml.internal.ws.client.AsyncResponseImpl
com.sun.xml.internal.ws.client.ResponseImpl
com.sun.xml.internal.ws.client.Stub
com.sun.xml.internal.ws.client.WSServiceDelegate
com.sun.xml.internal.ws.client.dispatch.DispatchImpl
com.sun.xml.internal.ws.client.dispatch.DispatchImpl$DispatchAsyncInvoker$1
com.sun.xml.internal.ws.client.sei.AsyncMethodHandler
com.sun.xml.internal.ws.client.sei.AsyncMethodHandler$Invoker
com.sun.xml.internal.ws.client.sei.CallbackMethodHandler
com.sun.xml.internal.ws.server.StatefulInstanceResolver
```

```
com.sun.xml.internal.ws.server.StatefulInstanceResolver$Instance
com.sun.xml.internal.ws.server.StatefulInstanceResolver$Instance$1
com.sun.xml.internal.ws.server.WSEndpointImpl
com.sun.xml.internal.ws.transport.http.HttpAdapter
com.sun.xml.internal.ws.transport.http.client.CookieJar
com.sun.xml.internal.ws.transport.http.client.HttpTransportPipe
com.sun.xml.internal.ws.transport.http.server.EndpointImpl
com.sun.xml.internal.ws.transport.http.server.HttpEndpoint
com.sun.xml.internal.ws.transport.http.server.ServerMgr
com.sun.xml.internal.ws.transport.http.server.WSHttpHandler
com.sun.xml.internal.ws.transport.http.server.WSHttpHandler$HttpHandlerRunnable
com.sun.xml.internal.ws.util.CompletedFuture
com.sun.xml.internal.ws.util.DOMUtil
com.sun.xml.internal.ws.util.Pool
com.sun.xml.internal.ws.util.pipe.StandalonePipeAssembler
com.sun.xml.internal.ws.util.pipe.StandaloneTubeAssembler
com.sun.xml.internal.xsom.impl.scd.SimpleCharStream
java.beans.java_util_Collections$SynchronizedCollection_PersistenceDelegate
java.beans.java_util_Collections$SynchronizedList_PersistenceDelegate
java.beans.java_util_Collections$SynchronizedMap_PersistenceDelegate
java.beans.java_util_Collections$SynchronizedRandomAccessList_PersistenceDelegate
java.beans.java_util_Collections$SynchronizedSet_PersistenceDelegate
java.beans.java_util_Collections$SynchronizedSortedMap_PersistenceDelegate
java.beans.java_util_Collections$SynchronizedSortedSet_PersistenceDelegate
java.io.Console
java.io.Console$LineReader
java.io.DeleteOnExitHook
java.lang.ApplicationShutdownHooks
java.lang.Byte$ByteCache
java.lang.Character$CharacterCache
java.lang.Integer$IntegerCache
java.lang.Long$LongCache
java.lang.SecurityException
java.lang.Short$ShortCache
java.lang.VirtualMachineError
java.lang.management.GarbageCollectorMXBean
java.lang.management.MemoryManagerMXBean
java.lang.management.MemoryPoolMXBean
java.rmi.server.ObjID
java.sql.DriverService
java.text.NumberFormat
java.util.Arrays
java.util.ServiceLoader$LazyIterator
javax.activation.DataHandler
javax.activation.MailcapCommandMap
javax.activation.MimetypesFileTypeMap
javax.activation.ObjectDataContentHandler
javax.annotation.processing.AbstractProcessor
javax.management.monitor.CounterMonitor$CounterMonitorObservedObject
javax.management.monitor.GaugeMonitor$GaugeMonitorObservedObject
javax.management.monitor.Monitor$DaemonThreadFactory
javax.management.monitor.Monitor$MonitorTask
javax.management.monitor.Monitor$ObservedObject
javax.management.monitor.Monitor$SchedulerTask
javax.management.monitor.StringMonitor$StringMonitorObservedObject
javax.management.openmbean.OpenType
javax.management.remote.rmi._RMIConnectionImpl_Tie
javax.management.remote.rmi._RMIConnection_Stub
```

```
javax.management.remote.rmi._RMIServerImpl_Tie
javax.management.remote.rmi._RMIServer_Stub
javax.tools.DiagnosticCollector
javax.tools.StandardLocation
javax.tools.ToolProvider
javax.tools.ToolProvider$Lazy
javax.xml.bind.ContextFinder
javax.xml.bind.JAXBException
javax.xml.bind.TypeConstraintException
javax.xml.soap.SOAPException
javax.xml.stream.FactoryFinder
javax.xml.ws.Service
sun.font.FontDesignMetrics
sun.font.FontDesignMetrics$KeyReference
sun.font.GlyphLayout$SDCache
sun.jkernel.BackgroundDownloader
sun.jkernel.Bundle
sun.jkernel.DownloadManager
sun.jkernel.Mutex
sun.management.Flag
sun.management.HotSpotDiagnostic
sun.misc.ClassLoaderUtil
sun.misc.Launcher
sun.misc.MetaIndex
sun.misc.VMSupport
sun.misc.Version
sun.net.httpserver.ContextList
sun.net.httpserver.HttpConnection
sun.net.httpserver.HttpServerImpl
sun.net.httpserver.HttpsServerImpl
sun.net.httpserver.LeftOverInputStream
sun.net.httpserver.Request$ReadStream
sun.net.httpserver.Request$WriteStream
sun.net.httpserver.SSLStreams
sun.net.httpserver.SSLStreams$EngineWrapper
sun.net.httpserver.SelectorCache
sun.net.httpserver.SelectorCache$CacheCleaner
sun.net.httpserver.ServerImpl
sun.net.httpserver.ServerImpl$Dispatcher
sun.net.httpserver.ServerImpl$ServerTimerTask
sun.net.www.http.KeepAliveStreamCleaner
sun.net.www.protocol.http.HttpURLConnection$HttpInputStream
sun.net.www.protocol.http.InMemoryCookieStore
sun.net.www.protocol.http.NegotiateAuthentication
sun.nio.ch.FileChannelImpl$SharedFileLockTable
sun.nio.ch.FileChannelImpl$SimpleFileLockTable
sun.rmi.runtime.RuntimeUtil
sun.rmi.runtime.RuntimeUtil$1
sun.tools.attach.HotSpotAttachProvider
sun.tools.attach.HotSpotVirtualMachine
sun.tools.attach.WindowsVirtualMachine
sun.tools.attach.WindowsVirtualMachine$PipedInputStream
sun.tools.jconsole.ClassTab$2
sun.tools.jconsole.HTMLPane
sun.tools.jconsole.InternalDialog$MastheadIcon
sun.tools.jconsole.JConsole
sun.tools.jconsole.MemoryTab$4
sun.tools.jconsole.Plotter
```

```
sun.tools.jconsole.ProxyClient$SnapshotInvocationHandler
sun.tools.jconsole.SummaryTab
sun.tools.jconsole.SummaryTab$1
sun.tools.jconsole.ThreadTab$1
sun.tools.jconsole.inspector.XMBean
sun.tools.jconsole.inspector.XMBeanNotifications$XMBeanNotificationsListener$1
sun.tools.jconsole.resources.JConsoleResources
sun.tools.jconsole.resources.JConsoleResources_ja
sun.tools.jconsole.resources.JConsoleResources_zh_CN
sun.util.LocaleServiceProviderPool
sun.util.TimeZoneNameUtility
com.sun.codemodel.internal.JJavaName
com.sun.codemodel.internal.JMods
com.sun.tools.hat.internal.model.StackFrame
com.sun.tools.internal.ws.processor.generator.Names
com.sun.tools.javac.parser.Token
com.sun.xml.internal.bind.api.impl.NameUtil
java.lang.management.ThreadInfo
javax.lang.model.SourceVersion
sun.jvm.hotspot.memory.SystemDictionary
sun.nio.cs.ext.COMPOUND_TEXT_Encoder
sun.nio.cs.ext.CompoundTextSupport
sun.tools.attach.LinuxVirtualMachine
sun.tools.attach.LinuxVirtualMachine$SocketInputStream
com.sun.org.apache.xalan.internal.xsltc.runtime.BasisLibrary
com.sun.xml.internal.org.jvnet.mimepull.DataFile
com.sun.xml.internal.org.jvnet.mimepull.DataHead
com.sun.xml.internal.org.jvnet.mimepull.DataHead$ReadOnceStream
com.sun.xml.internal.org.jvnet.mimepull.MIMEMessage
com.sun.xml.internal.org.jvnet.mimepull.WeakDataFile
com.sun.xml.internal.ws.api.server.HttpEndpoint
com.sun.xml.internal.ws.client.AsyncInvoker
com.sun.xml.internal.ws.client.sei.AsyncMethodHandler$SEIAsyncInvoker$1
com.sun.xml.internal.ws.encoding.MimeCodec
com.sun.xml.internal.ws.model.WrapperBeanGenerator
com.sun.xml.internal.ws.server.JMXAgent
sun.jvm.hotspot.CommandProcessor$9
sun.jvm.hotspot.CommandProcessor$9$1
sun.jvm.hotspot.SALauncherLoader
sun.jvm.hotspot.ui.AnnotatedMemoryPanel
sun.jvm.hotspot.ui.classbrowser.HTMLGenerator
sun.jvm.hotspot.ui.FindInCodeCachePanel$Visitor$1
sun.jvm.hotspot.ui.ProcessListPanel
sun.jvm.hotspot.utilities.soql.SOQLEngine
sun.net.spi.DefaultProxySelector$3

#-----------------------------------------------------------------------
# "J2SE"に追加するパッケージ, クラス
# ( JDK5またはJDK6の0870に存在しなかったが, 0900に含まれるようになったもの)
#-----------------------------------------------------------------------
com.rsa.*
com.sun.tools.javac.util.CloseableURLClassLoader
java.text.DateFormatSymbols
java.text.SimpleDateFormat
sun.nio.cs.ext.Big5
sun.nio.cs.ext.Big5_Solaris
sun.nio.cs.ext.MS950
```

```
#--------------------------------------------------------------------
# "J2SE"に追加するパッケージ, クラス
# ( 0900に存在しなかったが, 0950に含まれるようになったもの)
#--------------------------------------------------------------------
com.sun.beans.AppContext
com.sun.org.apache.xalan.internal.utils.ObjectFactory

#--------------------------------------------------------------------
# J2SE (JDK7 以降)
#--------------------------------------------------------------------
com.sun.beans.editors.EnumEditor
com.sun.beans.finder.PersistenceDelegateFinder
com.sun.beans.finder.PropertyEditorFinder
com.sun.beans.TypeResolver
com.sun.istack.internal.tools.ParallelWorldClassLoader
com.sun.java.accessibility.AccessBridge
com.sun.java.accessibility.util.AccessibilityListenerList
com.sun.java.accessibility.util.AWTEventMonitor
com.sun.java.accessibility.util.AWTEventMonitor$AWTEventsListener
com.sun.java.accessibility.util.ComponentEvtDispatchThread
com.sun.java.accessibility.util.EventQueueMonitor
com.sun.java.accessibility.util.GUIInitializedListener
com.sun.java.accessibility.util.GUIInitializedMulticaster
com.sun.java.accessibility.util.java.awt.ButtonTranslator
com.sun.java.accessibility.util.java.awt.CheckboxTranslator
com.sun.java.accessibility.util.java.awt.LabelTranslator
com.sun.java.accessibility.util.java.awt.ListTranslator
com.sun.java.accessibility.util.java.awt.TextComponentTranslator
com.sun.java.accessibility.util.SwingEventMonitor
com.sun.java.accessibility.util.SwingEventMonitor$SwingEventListener
com.sun.java.accessibility.util.TopLevelWindowListener
com.sun.java.accessibility.util.TopLevelWindowMulticaster
com.sun.java.accessibility.util.Translator
com.sun.java.util.jar.pack.Driver
com.sun.java.util.jar.pack.PackageReader
com.sun.java.util.jar.pack.PackageWriter
com.sun.java.util.jar.pack.PackerImpl
com.sun.java.util.jar.pack.Utils
com.sun.jmx.mbeanserver.DefaultMXBeanMappingFactory
com.sun.jmx.remote.internal.ServerNotifForwarder$NotifForwarderBufferFilter
com.sun.naming.internal.ResourceManager$AppletParameter
com.sun.nio.zipfs.ZipCoder
com.sun.nio.zipfs.ZipDirectoryStream
com.sun.nio.zipfs.ZipDirectoryStream$1
com.sun.nio.zipfs.ZipFileSystem
com.sun.nio.zipfs.ZipFileSystem$EntryInputStream
com.sun.nio.zipfs.ZipFileSystemProvider
com.sun.nio.zipfs.ZipUtils
com.sun.org.apache.xalan.internal.utils.SecuritySupport
com.sun.org.apache.xalan.internal.utils.XMLSecurityPropertyManager
com.sun.org.apache.xalan.internal.utils.XMLSecurityPropertyManager$Property
com.sun.org.apache.xalan.internal.utils.XMLSecurityPropertyManager$State
com.sun.org.apache.xerces.internal.impl.xpath.regex.CaseInsensitiveMap
com.sun.org.apache.xerces.internal.impl.xpath.regex.RegularExpression$Context
com.sun.org.apache.xerces.internal.utils.SecuritySupport
com.sun.org.apache.xerces.internal.utils.XMLSecurityPropertyManager
com.sun.org.apache.xerces.internal.utils.XMLSecurityPropertyManager$Property
com.sun.org.apache.xerces.internal.utils.XMLSecurityPropertyManager$State
```

```
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport$1
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport$2
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport$3
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport$4
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport$5
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport$6
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport$7
com.sun.org.apache.xerces.internal.xinclude.SecuritySupport$8
com.sun.org.apache.xml.internal.security.algorithms.JCEMapper
com.sun.org.apache.xml.internal.security.algorithms.SignatureAlgorithm
com.sun.org.apache.xml.internal.security.c14n.Canonicalizer
com.sun.org.apache.xml.internal.security.keys.keyresolver.KeyResolver
com.sun.org.apache.xml.internal.security.transforms.Transform
com.sun.org.apache.xml.internal.security.utils.ElementProxy
com.sun.org.apache.xml.internal.security.utils.resolver.ResourceResolver
com.sun.org.apache.xml.internal.security.utils.UnsyncBufferedOutputStream$1
com.sun.org.apache.xml.internal.security.utils.UnsyncByteArrayOutputStream$1
com.sun.org.apache.xml.internal.serialize.SecuritySupport
com.sun.org.apache.xml.internal.serialize.SecuritySupport$1
com.sun.org.apache.xml.internal.serialize.SecuritySupport$2
com.sun.org.apache.xml.internal.serialize.SecuritySupport$3
com.sun.org.apache.xml.internal.serialize.SecuritySupport$4
com.sun.org.apache.xml.internal.serialize.SecuritySupport$5
com.sun.org.apache.xml.internal.serialize.SecuritySupport$6
com.sun.org.apache.xml.internal.serialize.SecuritySupport$7
com.sun.org.apache.xml.internal.serialize.SecuritySupport$8
com.sun.org.glassfish.external.amx.AMXGlassfish$BootAMXCallback
com.sun.org.glassfish.external.amx.AMXGlassfish$WaitForDomainRootListenerCallback
com.sun.org.glassfish.external.amx.MBeanListener$CallbackImpl
com.sun.org.glassfish.external.statistics.impl.AverageRangeStatisticImpl
com.sun.org.glassfish.external.statistics.impl.BoundaryStatisticImpl
com.sun.org.glassfish.external.statistics.impl.BoundedRangeStatisticImpl
com.sun.org.glassfish.external.statistics.impl.CountStatisticImpl
com.sun.org.glassfish.external.statistics.impl.RangeStatisticImpl
com.sun.org.glassfish.external.statistics.impl.StatisticImpl
com.sun.org.glassfish.external.statistics.impl.StatsImpl
com.sun.org.glassfish.external.statistics.impl.StringStatisticImpl
com.sun.org.glassfish.external.statistics.impl.TimeStatisticImpl
com.sun.org.glassfish.gmbal.util.GenericConstructor
com.sun.org.glassfish.gmbal.util.GenericConstructor$1
com.sun.tools.classfile.Attribute$Factory
com.sun.tools.internal.xjc.reader.Ring
com.sun.tools.javac.api.ClientCodeWrapper$WrappedDiagnosticListener
com.sun.tools.javac.api.ClientCodeWrapper$WrappedFileObject
com.sun.tools.javac.api.ClientCodeWrapper$WrappedJavaFileManager
com.sun.tools.javac.api.ClientCodeWrapper$WrappedJavaFileObject
com.sun.tools.javac.api.ClientCodeWrapper$WrappedTaskListener
com.sun.tools.javac.code.Flags$Flag
com.sun.tools.javac.file.CacheFSInfo
com.sun.tools.javac.file.ZipFileIndex
com.sun.tools.javac.file.ZipFileIndexCache
com.sun.tools.javac.processing.JavacProcessingEnvironment$ServiceIterator
com.sun.tools.javac.Server
com.sun.tools.javac.util.BaseFileManager
com.sun.tools.javac.util.SharedNameTable
com.sun.tools.javap.JavapTask
com.sun.xml.internal.bind.v2.runtime.reflect.opt.AccessorInjector
```

```
com.sun.xml.internal.ws.model.AbstractWrapperBeanGenerator
com.sun.xml.internal.ws.model.Injector
com.sun.xml.internal.ws.policy.sourcemodel.AssertionData
com.sun.xml.internal.ws.policy.sourcemodel.PolicySourceModel
com.sun.xml.internal.ws.transport.http.server.PortableHttpHandler
com.sun.xml.internal.ws.transport.http.server.PortableHttpHandler$HttpHandlerRunnable
java.beans.ChangeListenerMap
java.beans.MetaData$java_awt_AWTKeyStroke_PersistenceDelegate
java.beans.MetaData$java_awt_BorderLayout_PersistenceDelegate
java.beans.MetaData$java_awt_CardLayout_PersistenceDelegate
java.beans.MetaData$java_awt_Choice_PersistenceDelegate
java.beans.MetaData$java_awt_Component_PersistenceDelegate
java.beans.MetaData$java_awt_Container_PersistenceDelegate
java.beans.MetaData$java_awt_Font_PersistenceDelegate
java.beans.MetaData$java_awt_font_TextAttribute_PersistenceDelegate
java.beans.MetaData$java_awt_GridBagLayout_PersistenceDelegate
java.beans.MetaData$java_awt_Insets_PersistenceDelegate
java.beans.MetaData$java_awt_List_PersistenceDelegate
java.beans.MetaData$java_awt_Menu_PersistenceDelegate
java.beans.MetaData$java_awt_MenuBar_PersistenceDelegate
java.beans.MetaData$java_awt_MenuShortcut_PersistenceDelegate
java.beans.MetaData$java_awt_SystemColor_PersistenceDelegate
java.beans.MetaData$javax_swing_border_MatteBorder_PersistenceDelegate
java.beans.MetaData$javax_swing_Box_PersistenceDelegate
java.beans.MetaData$javax_swing_DefaultComboBoxModel_PersistenceDelegate
java.beans.MetaData$javax_swing_DefaultListModel_PersistenceDelegate
java.beans.MetaData$javax_swing_JFrame_PersistenceDelegate
java.beans.MetaData$javax_swing_JMenu_PersistenceDelegate
java.beans.MetaData$javax_swing_JTabbedPane_PersistenceDelegate
java.beans.MetaData$javax_swing_ToolTipManager_PersistenceDelegate
java.beans.MetaData$javax_swing_tree_DefaultMutableTreeNode_PersistenceDelegate
java.beans.MetaData$sun_swing_PrintColorUIResource_PersistenceDelegate
java.beans.ThreadGroupContext
java.lang.BootstrapMethodError
java.lang.ClassLoader$ParallelLoaders
java.lang.ClassValue
java.lang.ClassValue$ClassValueMap
java.lang.IllegalStateException
java.lang.invoke.BoundMethodHandle
java.lang.invoke.BoundMethodHandle$SpeciesData
java.lang.invoke.CallSite
java.lang.invoke.LambdaForm
java.lang.invoke.LambdaForm$NamedFunction
java.lang.invoke.MethodHandle
java.lang.invoke.MethodHandleImpl$BindCaller
java.lang.invoke.MethodHandleImpl$BindCaller$2
java.lang.invoke.MethodHandleNatives
java.lang.invoke.MethodType$WeakInternSet
java.lang.invoke.MutableCallSite
java.lang.StringBuilder
java.lang.StringValue
java.lang.StringValue$StringCache
java.lang.Thread$Caches
java.lang.UNIXProcess$1
java.lang.UNIXProcess$ProcessPipeInputStream
java.lang.UNIXProcess$ProcessPipeOutputStream
java.lang.UnsupportedOperationException
java.net.AbstractPlainDatagramSocketImpl
```

```
java.net.AbstractPlainSocketImpl
java.net.DualStackPlainDatagramSocketImpl
java.net.DualStackPlainSocketImpl
java.net.InMemoryCookieStore
java.net.TwoStacksPlainDatagramSocketImpl
java.net.TwoStacksPlainSocketImpl
java.net.URLClassLoader
java.nio.channels.AsynchronousChannelGroup
java.nio.channels.AsynchronousFileChannel
java.nio.channels.AsynchronousSocketChannel
java.nio.channels.Channels$2
java.nio.channels.Channels$3
java.nio.channels.SelectionKey
java.nio.channels.SocketChannel
java.nio.file.attribute.AclEntry
java.nio.file.attribute.AclEntry$1
java.nio.file.attribute.AclEntry$Builder
java.nio.file.attribute.AclEntryFlag
java.nio.file.attribute.AclEntryPermission
java.nio.file.attribute.AclEntryType
java.nio.file.attribute.AclFileAttributeView
java.nio.file.attribute.AttributeView
java.nio.file.attribute.BasicFileAttributes
java.nio.file.attribute.BasicFileAttributeView
java.nio.file.attribute.DosFileAttributes
java.nio.file.attribute.DosFileAttributeView
java.nio.file.attribute.FileAttribute
java.nio.file.attribute.FileAttributeView
java.nio.file.attribute.FileOwnerAttributeView
java.nio.file.attribute.FileStoreAttributeView
java.nio.file.attribute.FileTime
java.nio.file.attribute.FileTime$1
java.nio.file.attribute.FileTime$DaysAndNanos
java.nio.file.attribute.GroupPrincipal
java.nio.file.attribute.PosixFileAttributes
java.nio.file.attribute.PosixFileAttributeView
java.nio.file.attribute.PosixFilePermission
java.nio.file.attribute.PosixFilePermissions
java.nio.file.attribute.PosixFilePermissions$1
java.nio.file.attribute.UserDefinedFileAttributeView
java.nio.file.attribute.UserPrincipal
java.nio.file.attribute.UserPrincipalLookupService
java.nio.file.attribute.UserPrincipalNotFoundException
java.nio.file.CopyMoveHelper
java.nio.file.Files
java.nio.file.SecureDirectoryStream
java.nio.file.spi.FileSystemProvider
java.sql.DriverManager$2
java.text.DecimalFormatSymbols
java.util.Currency$1
java.util.HashMap
java.util.logging.Level$KnownLevel
java.util.logging.LogManager$LoggerContext
java.util.prefs.FileSystemPreferences$6
java.util.prefs.FileSystemPreferences$7
java.util.Vector$Itr
java.util.Vector$ListItr
java.util.XMLUtils
```

```
javax.sql.rowset.serial.SerialClob
sun.dc.DuctusRenderingEngine
sun.dc.DuctusRenderingEngine$FillAdapter
sun.font.CreatedFontTracker$TempFileDeletionHook
sun.font.FcFontConfiguration
sun.font.FontAccess
sun.font.FontConfigManager
sun.font.FontConfigManager$FcCompFont
sun.font.FontConfigManager$FontConfigFont
sun.font.FontConfigManager$FontConfigInfo
sun.font.FontManagerFactory
sun.font.FontManagerFactory$1
sun.font.FontManagerForSGE
sun.font.FontManagerNativeLibrary
sun.font.FontManagerNativeLibrary$1
sun.font.FontScaler
sun.font.FontScalerException
sun.font.FontUtilities
sun.font.FontUtilities$1
sun.font.FreetypeFontScaler
sun.font.GlyphDisposedListener
sun.font.NullFontScaler
sun.font.SunFontManager
sun.font.SunFontManager$1
sun.font.SunFontManager$10
sun.font.SunFontManager$11
sun.font.SunFontManager$12
sun.font.SunFontManager$13
sun.font.SunFontManager$14
sun.font.SunFontManager$2
sun.font.SunFontManager$3
sun.font.SunFontManager$4
sun.font.SunFontManager$5
sun.font.SunFontManager$6
sun.font.SunFontManager$7
sun.font.SunFontManager$8
sun.font.SunFontManager$8$1
sun.font.SunFontManager$9
sun.font.SunFontManager$FamilyDescription
sun.font.SunFontManager$FontRegistrationInfo
sun.font.SunFontManager$T1Filter
sun.font.SunFontManager$TTFilter
sun.font.SunFontManager$TTorT1Filter
sun.font.T2KFontScaler
sun.font.T2KFontScaler$1
sun.font.Underline
sun.font.XRGlyphCache
sun.font.XRGlyphCache$1
sun.font.XRGlyphCacheEntry
sun.font.XRTextRenderer
sun.invoke.util.ValueConversions
sun.jvm.hotspot.CommandProcessor$12
sun.jvm.hotspot.CommandProcessor$12$1
sun.jvm.hotspot.debugger.bsd.BsdDebuggerLocal
sun.jvm.hotspot.debugger.bsd.BsdDebuggerLocal$BsdDebuggerLocalWorkerThread
sun.jvm.hotspot.runtime.CompilerThread
sun.launcher.LauncherHelper
sun.launcher.resources.launcher
```

```
sun.management.jdp.JdpController
sun.management.ManagementFactoryHelper
sun.management.ManagementFactoryHelper$1
sun.management.ManagementFactoryHelper$PlatformLoggingImpl
sun.misc.Hashing
sun.misc.JavaxSecurityAuthKerberosAccess
sun.misc.Launcher$ExtClassLoader
sun.misc.PerfCounter
sun.misc.PostVMInitHook
sun.misc.Service$LazyIterator
sun.net.ftp.FtpClientProvider
sun.net.httpserver.ServerImpl$ServerTimerTask1
sun.net.www.http.HttpCapture
sun.net.www.protocol.http.ntlm.NTLMAuthentication
sun.net.www.protocol.jar.URLJarFile$1
sun.nio.ch.AbstractPollSelectorImpl
sun.nio.ch.AsynchronousChannelGroupImpl
sun.nio.ch.AsynchronousChannelGroupImpl$2
sun.nio.ch.AsynchronousChannelGroupImpl$3
sun.nio.ch.AsynchronousFileChannelImpl
sun.nio.ch.AsynchronousServerSocketChannelImpl
sun.nio.ch.AsynchronousSocketChannelImpl
sun.nio.ch.BsdAsynchronousChannelProvider
sun.nio.ch.CompletedFuture
sun.nio.ch.EPoll
sun.nio.ch.EPollPort
sun.nio.ch.EPollPort$EventHandlerTask
sun.nio.ch.FileChannelImpl$Unmapper
sun.nio.ch.FileDispatcherImpl
sun.nio.ch.Invoker
sun.nio.ch.Iocp
sun.nio.ch.Iocp$EventHandlerTask
sun.nio.ch.KQueue
sun.nio.ch.KQueuePort
sun.nio.ch.KQueuePort$EventHandlerTask
sun.nio.ch.LinuxAsynchronousChannelProvider
sun.nio.ch.MembershipKeyImpl
sun.nio.ch.PendingFuture
sun.nio.ch.PendingIoCache
sun.nio.ch.Port
sun.nio.ch.SctpChannelImpl
sun.nio.ch.SctpMultiChannelImpl
sun.nio.ch.SctpNet
sun.nio.ch.SctpServerChannelImpl
sun.nio.ch.SharedFileLockTable
sun.nio.ch.SimpleAsynchronousFileChannelImpl
sun.nio.ch.SimpleAsynchronousFileChannelImpl$DefaultExecutorHolder
sun.nio.ch.SolarisAsynchronousChannelProvider
sun.nio.ch.SolarisEventPort
sun.nio.ch.SolarisEventPort$EventHandlerTask
sun.nio.ch.ThreadPool
sun.nio.ch.UnixAsynchronousServerSocketChannelImpl
sun.nio.ch.UnixAsynchronousSocketChannelImpl
sun.nio.ch.UnixAsynchronousSocketChannelImpl$1
sun.nio.ch.UnixAsynchronousSocketChannelImpl$2
sun.nio.ch.WindowsAsynchronousChannelProvider
sun.nio.ch.WindowsAsynchronousFileChannelImpl
sun.nio.ch.WindowsAsynchronousFileChannelImpl$LockTask
```

```
sun.nio.ch.WindowsAsynchronousFileChannelImpl$ReadTask
sun.nio.ch.WindowsAsynchronousFileChannelImpl$WriteTask
sun.nio.ch.WindowsAsynchronousServerSocketChannelImpl
sun.nio.ch.WindowsAsynchronousServerSocketChannelImpl$AcceptTask
sun.nio.ch.WindowsAsynchronousSocketChannelImpl
sun.nio.ch.WindowsAsynchronousSocketChannelImpl$ConnectTask
sun.nio.ch.WindowsAsynchronousSocketChannelImpl$ReadTask
sun.nio.ch.WindowsAsynchronousSocketChannelImpl$WriteTask
sun.nio.cs.AbstractCharsetProvider$1
sun.nio.cs.ext.EUC_CN
sun.nio.cs.ext.EUC_KR
sun.nio.cs.ext.GBK
sun.nio.cs.ext.IBM1364
sun.nio.cs.ext.IBM1381
sun.nio.cs.ext.IBM1383
sun.nio.cs.ext.IBM930
sun.nio.cs.ext.IBM933
sun.nio.cs.ext.IBM935
sun.nio.cs.ext.IBM937
sun.nio.cs.ext.IBM939
sun.nio.cs.ext.IBM942
sun.nio.cs.ext.IBM943
sun.nio.cs.ext.IBM948
sun.nio.cs.ext.IBM949
sun.nio.cs.ext.IBM950
sun.nio.cs.ext.IBM970
sun.nio.cs.ext.Johab
sun.nio.cs.ext.MS932
sun.nio.cs.ext.MS936
sun.nio.cs.ext.MS949
sun.nio.fs.AbstractPoller
sun.nio.fs.AbstractPoller$Request
sun.nio.fs.AbstractWatchKey
sun.nio.fs.AbstractWatchService
sun.nio.fs.Cancellable
sun.nio.fs.LinuxNativeDispatcher
sun.nio.fs.PollingWatchService
sun.nio.fs.PollingWatchService$3
sun.nio.fs.PollingWatchService$PollingWatchKey
sun.nio.fs.UnixCopyFile
sun.nio.fs.UnixDirectoryStream
sun.nio.fs.UnixDirectoryStream$UnixDirectoryIterator
sun.nio.fs.UnixFileAttributes
sun.nio.fs.UnixFileAttributeViews$Basic
sun.nio.fs.UnixFileStore
sun.nio.fs.UnixFileSystem$FileStoreIterator
sun.nio.fs.UnixFileSystemProvider
sun.nio.fs.UnixNativeDispatcher
sun.nio.fs.UnixSecureDirectoryStream
sun.nio.fs.UnixSecureDirectoryStream$BasicFileAttributeViewImpl
sun.nio.fs.UnixSecureDirectoryStream$PosixFileAttributeViewImpl
sun.nio.fs.WindowsDirectoryStream
sun.nio.fs.WindowsDirectoryStream$WindowsDirectoryIterator
sun.nio.fs.WindowsFileAttributes
sun.nio.fs.WindowsFileCopy
sun.nio.fs.WindowsFileSystem$FileStoreIterator
sun.nio.fs.WindowsFileSystemProvider
sun.nio.fs.WindowsNativeDispatcher
```

```
sun.nio.fs.WindowsPath
sun.nio.fs.WindowsSecurity
sun.nio.fs.WindowsSecurity$1
sun.nio.fs.WindowsSecurity$Privilege
sun.nio.fs.WindowsSecurityDescriptor
sun.text.normalizer.UBiDiProps
sun.tools.jconsole.inspector.XMBeanAttributes$1
sun.tools.jconsole.inspector.XMBeanAttributes$2
sun.tools.jconsole.inspector.XMBeanAttributes$AttributesListener
sun.util.calendar.LocalGregorianCalendar$1
sun.util.locale.LocaleObjectCache
sun.util.logging.PlatformLogger

#--------------------------------------------------------------------
# J2SE (JDK8 以降)
#--------------------------------------------------------------------
com.sun.beans.util.Cache
com.sun.istack.internal.tools.DefaultAuthenticator
com.sun.java.accessibility.AccessBridge$102
com.sun.java.accessibility.AccessBridge$152
com.sun.java.accessibility.AccessBridge$DefaultNativeWindowHandler
com.sun.java.accessibility.AccessBridge$InvocationUtils
com.sun.java.accessibility.AccessBridge$InvocationUtils$1
com.sun.java.accessibility.AccessBridge$InvocationUtils$CallableWrapper
com.sun.java.accessibility.AccessBridge$ObjectReferences
com.sun.org.apache.xalan.internal.utils.XMLSecurityManager
com.sun.org.apache.xalan.internal.utils.XMLSecurityManager$Limit
com.sun.org.apache.xalan.internal.utils.XMLSecurityManager$NameMap
com.sun.org.apache.xalan.internal.utils.XMLSecurityManager$State
com.sun.org.apache.xerces.internal.utils.XMLSecurityManager
com.sun.org.apache.xerces.internal.utils.XMLSecurityManager$Limit
com.sun.org.apache.xerces.internal.utils.XMLSecurityManager$NameMap
com.sun.org.apache.xerces.internal.utils.XMLSecurityManager$State
com.sun.org.apache.xml.internal.security.algorithms.implementations.SignatureECDSA$Signature
ECDSASHA256
com.sun.org.apache.xml.internal.security.algorithms.implementations.SignatureECDSA$Signature
ECDSASHA384
com.sun.org.apache.xml.internal.security.algorithms.implementations.SignatureECDSA$Signature
ECDSASHA512
com.sun.org.apache.xml.internal.security.c14n.implementations.Canonicalizer11$1
com.sun.org.apache.xml.internal.security.c14n.implementations.Canonicalizer20010315$1
com.sun.org.apache.xml.internal.security.c14n.implementations.CanonicalizerPhysical
com.sun.org.apache.xml.internal.security.encryption.AbstractSerializer
com.sun.org.apache.xml.internal.security.encryption.DocumentSerializer
com.sun.org.apache.xml.internal.security.encryption.Serializer
com.sun.org.apache.xml.internal.security.keys.content.DEREncodedKeyValue
com.sun.org.apache.xml.internal.security.keys.content.KeyInfoReference
com.sun.org.apache.xml.internal.security.keys.content.x509.XMLX509Digest
com.sun.org.apache.xml.internal.security.keys.keyresolver.implementations.DEREncodedKeyValue
Resolver
com.sun.org.apache.xml.internal.security.keys.keyresolver.implementations.KeyInfoReferenceRe
solver
com.sun.org.apache.xml.internal.security.keys.keyresolver.implementations.PrivateKeyResolver
com.sun.org.apache.xml.internal.security.keys.keyresolver.implementations.SecretKeyResolver
com.sun.org.apache.xml.internal.security.keys.keyresolver.implementations.SingleKeyResolver
com.sun.org.apache.xml.internal.security.keys.keyresolver.implementations.X509DigestResolver
com.sun.org.apache.xml.internal.security.keys.storage.implementations.KeyStoreResolver$KeySt
oreIterator$1
```

```
com.sun.org.apache.xml.internal.security.signature.Reference$2
com.sun.org.apache.xml.internal.security.signature.Reference$2$1
com.sun.org.apache.xml.internal.security.signature.reference.ReferenceData
com.sun.org.apache.xml.internal.security.signature.reference.ReferenceNodeSetData
com.sun.org.apache.xml.internal.security.signature.reference.ReferenceOctetStreamData
com.sun.org.apache.xml.internal.security.signature.reference.ReferenceSubTreeData
com.sun.org.apache.xml.internal.security.signature.reference.ReferenceSubTreeData$DelayedNod
eIterator
com.sun.org.apache.xml.internal.security.utils.ClassLoaderUtils
com.sun.org.apache.xml.internal.security.utils.ClassLoaderUtils$1
com.sun.org.apache.xml.internal.security.utils.DOMNamespaceContext
com.sun.org.apache.xml.internal.security.utils.JDKXPathAPI
com.sun.org.apache.xml.internal.security.utils.JDKXPathFactory
com.sun.org.apache.xml.internal.security.utils.resolver.ResourceResolverContext
com.sun.org.apache.xml.internal.security.utils.Signature11ElementProxy
com.sun.org.apache.xml.internal.security.utils.XalanXPathAPI
com.sun.org.apache.xml.internal.security.utils.XalanXPathFactory
com.sun.org.apache.xml.internal.security.utils.XPathAPI
com.sun.org.apache.xml.internal.security.utils.XPathFactory
com.sun.org.apache.xml.internal.serializer.utils.SerializerMessages_pt_BR
com.sun.source.util.Trees
com.sun.tools.classfile.ClassFile
com.sun.tools.doclets.internal.toolkit.util.DocFileFactory
com.sun.tools.doclets.internal.toolkit.util.PathDocFileFactory$StandardDocFile
com.sun.tools.internal.ws.processor.modeler.annotation.WebServiceAp
com.sun.tools.internal.ws.wscompile.WsimportOptions$RereadInputStream
com.sun.tools.javac.code.Lint
com.sun.tools.javac.jvm.Gen
com.sun.tools.javac.parser.Tokens$TokenKind
com.sun.tools.javac.processing.JavacProcessingEnvironment$Round
com.sun.tools.javac.resources.javac
com.sun.tools.javac.util.ServiceLoader$LazyIterator
com.sun.tools.javadoc.api.JavadocTaskImpl
com.sun.xml.internal.bind.DatatypeConverterImpl
com.sun.xml.internal.ws.api.message.Packet
com.sun.xml.internal.ws.api.server.Container
com.sun.xml.internal.ws.api.server.MethodUtil
com.sun.xml.internal.ws.api.server.ThreadLocalContainerResolver
com.sun.xml.internal.ws.api.server.ThreadLocalContainerResolver$2
com.sun.xml.internal.ws.api.WSService
com.sun.xml.internal.ws.binding.BindingImpl
com.sun.xml.internal.ws.client.sei.MethodUtil
com.sun.xml.internal.ws.commons.xmlutil.Converter
com.sun.xml.internal.ws.db.DatabindingImpl
com.sun.xml.internal.ws.dump.LoggingDumpTube
com.sun.xml.internal.ws.dump.MessageDumpingFeature
com.sun.xml.internal.ws.dump.MessageDumpingTube
com.sun.xml.internal.ws.policy.privateutil.MethodUtil
com.sun.xml.internal.ws.server.provider.AsyncProviderInvokerTube
com.sun.xml.internal.ws.server.provider.AsyncProviderInvokerTube$AsyncProviderCallbackImpl
com.sun.xml.internal.ws.server.WSEndpointMOMProxy
com.sun.xml.internal.ws.transport.http.HttpAdapter$HttpToolkit
com.sun.xml.internal.ws.util.InjectionPlan
com.sun.xml.internal.ws.util.InjectionPlan$Compositor
com.sun.xml.internal.ws.util.ServiceFinder$LazyIterator
java.beans.WeakIdentityMap
java.io.FilterOutputStream
java.lang.invoke.InnerClassLambdaMetafactory
```

```
java.lang.invoke.MemberName
java.lang.invoke.MethodHandleImpl
java.lang.invoke.MethodHandles$Lookup
java.lang.invoke.MethodType$ConcurrentWeakInternSet
java.lang.invoke.MethodTypeForm
java.lang.Process
java.lang.reflect.Executable
java.lang.reflect.Parameter
java.lang.reflect.Proxy$ProxyClassFactory
java.lang.reflect.WeakCache
java.lang.reflect.WeakCache$CacheKey
java.lang.reflect.WeakCache$Factory
java.lang.UNIXProcess$2
java.net.URLPermission
java.net.URLPermission$Authority
java.sql.DriverAction
java.sql.JDBCType
java.sql.SQLType
java.time.chrono.AbstractChronology
java.time.format.DateTimeFormatterBuilder$LocalizedPrinterParser
java.time.format.DateTimeFormatterBuilder$ZoneIdPrinterParser
java.time.format.DateTimeFormatterBuilder$ZoneTextPrinterParser
java.time.format.DateTimeTextProvider
java.time.format.DecimalStyle
java.time.temporal.WeekFields
java.time.zone.TzdbZoneRulesProvider
java.time.zone.ZoneRules
java.time.zone.ZoneRulesProvider
java.time.ZoneOffset
java.util.ArrayPrefixHelpers$CumulateTask
java.util.ArrayPrefixHelpers$DoubleCumulateTask
java.util.ArrayPrefixHelpers$IntCumulateTask
java.util.ArrayPrefixHelpers$LongCumulateTask
java.util.ArraysParallelSortHelpers$EmptyCompleter
java.util.ArraysParallelSortHelpers$FJByte$Merger
java.util.ArraysParallelSortHelpers$FJByte$Sorter
java.util.ArraysParallelSortHelpers$FJChar$Merger
java.util.ArraysParallelSortHelpers$FJChar$Sorter
java.util.ArraysParallelSortHelpers$FJDouble$Merger
java.util.ArraysParallelSortHelpers$FJDouble$Sorter
java.util.ArraysParallelSortHelpers$FJFloat$Merger
java.util.ArraysParallelSortHelpers$FJFloat$Sorter
java.util.ArraysParallelSortHelpers$FJInt$Merger
java.util.ArraysParallelSortHelpers$FJInt$Sorter
java.util.ArraysParallelSortHelpers$FJLong$Merger
java.util.ArraysParallelSortHelpers$FJLong$Sorter
java.util.ArraysParallelSortHelpers$FJObject$Merger
java.util.ArraysParallelSortHelpers$FJObject$Sorter
java.util.ArraysParallelSortHelpers$FJShort$Merger
java.util.ArraysParallelSortHelpers$FJShort$Sorter
java.util.ArraysParallelSortHelpers$Relay
java.util.Collections$SynchronizedNavigableMap
java.util.Collections$SynchronizedNavigableSet
java.util.logging.LogManager$LoggerWeakRef
java.util.SplittableRandom
java.util.stream.AbstractShortCircuitTask
java.util.stream.AbstractTask
java.util.stream.DistinctOps$1
```

```
java.util.stream.DoublePipeline$5$1
java.util.stream.FindOps$FindTask
java.util.stream.ForEachOps$ForEachOrderedTask
java.util.stream.ForEachOps$ForEachTask
java.util.stream.IntPipeline$7$1
java.util.stream.LongPipeline$6$1
java.util.stream.Nodes$CollectorTask
java.util.stream.Nodes$SizedCollectorTask
java.util.stream.Nodes$ToArrayTask
java.util.stream.ReduceOps$ReduceTask
java.util.stream.ReferencePipeline$10$1
java.util.stream.ReferencePipeline$7$1
java.util.stream.ReferencePipeline$8$1
java.util.stream.ReferencePipeline$9$1
java.util.stream.SliceOps$SliceTask
java.util.stream.Streams$1
java.util.stream.Streams$2
java.util.stream.StreamSpliterators$DistinctSpliterator
java.util.stream.StreamSpliterators$UnorderedSliceSpliterator
java.util.Vector$VectorSpliterator
javax.activation.CommandMap
javax.activation.FileTypeMap
javax.sql.rowset.spi.SyncFactory$2
jdk.internal.dynalink.beans.OverloadedMethod
jdk.internal.dynalink.ChainedCallSite
jdk.internal.dynalink.support.CallSiteDescriptorFactory
jdk.internal.dynalink.support.ClassMap
jdk.internal.org.objectweb.asm.commons.InstructionAdapter
jdk.internal.org.objectweb.asm.commons.SerialVersionUIDAdder
jdk.internal.org.objectweb.asm.util.Textifier
jdk.internal.util.xml.PropertiesDefaultHandler
jdk.nashorn.api.scripting.ScriptObjectMirror
jdk.nashorn.api.scripting.ScriptUtils
jdk.nashorn.internal.codegen.ClassEmitter
jdk.nashorn.internal.codegen.CodeGenerator
jdk.nashorn.internal.codegen.CompilationPhase$8
jdk.nashorn.internal.codegen.types.Type
jdk.nashorn.internal.ir.debug.ObjectSizeCalculator
jdk.nashorn.internal.objects.Global
jdk.nashorn.internal.objects.NativeArray
jdk.nashorn.internal.objects.NativeArray$6
jdk.nashorn.internal.objects.NativeDate
jdk.nashorn.internal.objects.NativeObject
jdk.nashorn.internal.runtime.AccessorProperty
jdk.nashorn.internal.runtime.arrays.ArrayData
jdk.nashorn.internal.runtime.arrays.IteratorAction
jdk.nashorn.internal.runtime.CodeStore$2
jdk.nashorn.internal.runtime.CodeStore$3
jdk.nashorn.internal.runtime.Context
jdk.nashorn.internal.runtime.DebuggerSupport
jdk.nashorn.internal.runtime.JSType
jdk.nashorn.internal.runtime.linker.JavaAdapterFactory$AdapterInfo
jdk.nashorn.internal.runtime.linker.LinkerCallSite
jdk.nashorn.internal.runtime.linker.LinkerCallSite$1
jdk.nashorn.internal.runtime.linker.NashornCallSiteDescriptor$1
jdk.nashorn.internal.runtime.ListAdapter
jdk.nashorn.internal.runtime.PropertyListeners
jdk.nashorn.internal.runtime.PropertyMap
```

```
jdk.nashorn.internal.runtime.RecompilableScriptFunctionData
jdk.nashorn.internal.runtime.regexp.joni.Regex
jdk.nashorn.internal.runtime.ScriptFunctionData
jdk.nashorn.internal.runtime.ScriptingFunctions
jdk.nashorn.internal.runtime.ScriptingFunctions$1
jdk.nashorn.internal.runtime.ScriptingFunctions$2
jdk.nashorn.internal.runtime.ScriptLoader
jdk.nashorn.internal.runtime.ScriptObject
jdk.nashorn.internal.runtime.ScriptRuntime
jdk.nashorn.internal.runtime.Source
jdk.nashorn.internal.runtime.Source$URLData
jdk.nashorn.internal.runtime.UserAccessorProperty
sun.jvm.hotspot.CommandProcessor$17
sun.jvm.hotspot.CommandProcessor$17$1
sun.management.DiagnosticCommandImpl
sun.management.OperatingSystemImpl
sun.misc.GThreadHelper
sun.net.ExtendedOptionsImpl
sun.net.PortConfig
sun.nio.ch.sctp.SctpChannelImpl
sun.nio.ch.sctp.SctpMultiChannelImpl
sun.nio.ch.sctp.SctpNet
sun.nio.ch.sctp.SctpServerChannelImpl
sun.nio.cs.ext.IBM300
sun.nio.cs.ext.JIS_X_0208
sun.nio.cs.ext.JIS_X_0208_MS5022X
sun.nio.cs.ext.JIS_X_0208_MS932
sun.nio.cs.ext.JIS_X_0208_Solaris
sun.nio.cs.ext.JIS_X_0212
sun.nio.cs.ext.JIS_X_0212_MS5022X
sun.nio.cs.ext.JIS_X_0212_Solaris
sun.nio.cs.ext.PCK
sun.nio.cs.ext.SJIS
sun.util.calendar.CalendarSystem$1
sun.util.calendar.ZoneInfoFile$1
sun.util.locale.provider.AuxLocaleProviderAdapter
sun.util.locale.provider.HostLocaleProviderAdapterImpl$2
sun.util.locale.provider.HostLocaleProviderAdapterImpl$3
sun.util.locale.provider.HostLocaleProviderAdapterImpl$4
sun.util.locale.provider.JRELocaleProviderAdapter
sun.util.locale.provider.LocaleProviderAdapter
sun.util.locale.provider.LocaleResources
sun.util.locale.provider.LocaleServiceProviderPool
sun.util.locale.provider.SPILocaleProviderAdapter$BreakIteratorProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$CalendarDataProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$CalendarNameProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$CollatorProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$CurrencyNameProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$DateFormatProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$DateFormatSymbolsProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$DecimalFormatSymbolsProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$LocaleNameProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$NumberFormatProviderDelegate
sun.util.locale.provider.SPILocaleProviderAdapter$TimeZoneNameProviderDelegate
sun.util.locale.provider.TimeZoneNameUtility
sun.util.resources.OpenListResourceBundle
sun.util.resources.ParallelListResourceBundle
sun.util.xml.PlatformXmlPropertiesProvider
```

```
#-------------------------------------------------------------------------
# J2EE
#-------------------------------------------------------------------------
javax.activation.*
javax.annotation.*
javax.ejb.*
javax.el.*
javax.jms.*
javax.mail.*
javax.management.*
javax.persistence.*
javax.resource.*
javax.servlet.*
javax.transaction.*
```

付録 D　各バージョンでの主な機能変更

ここでは，11-20 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。11-20 での主な機能変更については，「1.5　アプリケーションサーバ 11-20 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 11-20 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　11-10 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–1　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–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
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–3　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.2　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–4　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。	－	－

（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–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
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

項目	変更の概要	参照先マニュアル	参照個所
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-6 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 H

(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-7 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。	機能解説 保守／移行編	10.3.4

付録 D.3　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–8　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。	機能解説 保守／移行編	9 章

付録 D.4　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–9　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。	機能解説 保守／移行編	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–10　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 G
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 G
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 G

付録 D.5　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–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
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, 8.2.3
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。	このマニュアル	2.6.3, 2.6.4, 2.6.5
		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–12　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。	機能解説 保守／移行編	10.3.1, 10.3.2, 10.3.5

(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–13　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。	機能解説 保守／移行編	付録 A.2

付録 D.6　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–14　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。	機能解説 保守／移行編	9.16

(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–15　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。	機能解説 保守／移行編	9.17

(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–16　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.7　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–17　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–18　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。 • 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	仮想化システム構築・運用ガイド	1.1.2

(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–19　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–20　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。	機能解説 保守／移行編	5.7.2, 5.7.3
		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	機能解説 拡張編	7.2.2, 7.6.5, 7.10, 7.13.1, 7.13.3
		機能解説 保守／移行編	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。	機能解説 保守／移行編	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-21 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。	このマニュアル	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)

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-22 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。	機能解説 保守／移行編	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
		機能解説 保守／移行編	4.20
		リファレンス コマンド編	cjenvsetup（Component 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。	機能解説 保守／移行編	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。	機能解説 保守／移行編	7.5.2，7.5.3，8.25.1

項目	変更の概要	参照先マニュアル	参照個所
	• 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。		
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　09-00 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–23　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
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–24　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, 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,

項目	変更の概要	参照先マニュアル	参照個所
			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–25　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。	－	－

（凡例）－：09-70 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した項目を次の表に示します。

表 D–26　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。	このマニュアル	3 章
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。	機能解説 保守／移行編	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。	このマニュアル	2.4.1, 2.4.2, 2.6.3, 2.6.4

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	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–27 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。	機能解説 保守／移行編	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
		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.9 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–28 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	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
	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。	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
		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–29 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	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。	機能解説 基本・開発編 (コンテナ共通機能)	3.16.3

項目	変更の概要	参照先マニュアル	参照個所
タのグローバルトランザクションへの対応	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。		
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–30 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
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。	機能解説 保守／移行編	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–31 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–32　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。	機能解説 保守／移行編	付録 A

付録 D.10　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した項目を次の表に示します。

表 D–33　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した項目を次の表に示します。

表 D–34　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

項目	変更の概要	参照先マニュアル	参照個所
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した項目を次の表に示します。

表 D-35 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。	機能解説 保守／移行編	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した項目を次の表に示します。

表 D-36 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した項目を次の表に示します。

表 D-37 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	機能解説 基本・開発編 (コンテナ共通機能)	3 章

項目	変更の概要	参照先マニュアル	参照個所
	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。		

（凡例）－：該当なし。

付録 D.11　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した項目を次の表に示します。

表 D–38　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。	このマニュアル	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
		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した項目を次の表に示します。

表 D–39　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

項目	変更の概要	参照先マニュアル	参照個所
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。	機能解説 保守／移行編	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した項目を次の表に示します。

表 D-40 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
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した項目を次の表に示します。

表 D-41 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

項目	変更の概要	参照先マニュアル	参照個所
			(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した項目を次の表に示します。

表 D-42 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。	機能解説 保守／移行編	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
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。	このマニュアル	18.4.3, 18.5.3, 16.2.2, 16.3.3, 16.3.4
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, 4.7.1

項目	変更の概要	参照先マニュアル	参照個所
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。	機能解説 保守／移行編	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.12　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した項目を次の表に示します。

表 D–43　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した項目を次の表に示します。

表 D–44　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
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した項目を次の表に示します。

表 D–45　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

項目	変更の概要	参照先マニュアル	参照個所
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した項目を次の表に示します。

表 D-46 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した項目を次の表に示します。

表 D-47 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。	機能解説 保守／移行編	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。	このマニュアル	付録 C

項目	変更の概要	参照先マニュアル	参照個所
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。	機能解説 保守／移行編	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。	機能解説 保守／移行編	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

.mngsvrmonitorrc 348, 352
.mngsvrutilrc 387, 460

数字

09-70 での主な機能変更 651
09-80 での主な機能変更 650
09-87 での主な機能変更 650
1:1 系切り替えシステム 403, 429
1:1 系切り替えシステムで実行できる運用操作 487
1:1 系切り替えシステムでの系切り替え処理の流れ433
1:1 系切り替えシステムでの系切り替えの動作 433
1:1 系切り替えシステムの起動と停止（UNIX の場合） 487
1:1 系切り替えシステムのシステム構成例 430
11-00 での主な機能変更 648
11-10 での主な機能変更 647

A

actcommand 578
adminagent.adapter.bind_host 443, 459, 513, 523
adminagent.cluster.localaddress.check 443, 459
adminagent.j2ee.process.console_event.enabled 328
adminagent.j2ee.process.console_log.enabled 328
adminagent.process.consolelog.enabled 328
adminagent.process.consolelog.filenum 328
adminagent.process.consolelog.filesize 328
adminagent.properties 328, 443, 523
adminagent.userserver.process.console_event.enabled 328
adminagent.userserver.process.console_log.enabled 328
adminagentcheck 282
agentaddr 594, 601
auditsetup 266

C

cjjspc 190
cjlistthread 148
cjstopthread 150
cluster 563
cmx_stop_target コマンド 163
com.cosminexus.mngsvr.externals.jp1event.JP1EventHandler 361
com.cosminexus.mngsvr.jp1event.enabled 359
Component Container 管理者 31
Connector 属性ファイルの設定 112
cosminexus.xml でのイベントの発行機能の定義 100
cosminexus.xml でのリソース枯渇監視機能の定義 108
CTM の稼働統計情報の収集方法 317
CTM の稼働統計情報の出力先と出力情報 318
CTM の稼働統計情報の出力例 321
CTM のスケジュールキューの閉塞処理 168

D

DB Connector の稼働情報ファイルに出力される情報 87

E

ejbserver.distributedtx.ots.status.directory1 512, 523, 578
ejbserver.distributedtx.XATransaction.enabled 512, 523, 559
ejbserver.instrumentation.enabled 388
ejbserver.manager.agent.JP1EventAgent.enabled 359
ejbserver.manager.agent.MEventAgent.enabled 302
EJB クライアントアプリケーションで使用するコマンドの監査ログの出力ポイント 256
Enterprise Bean のメソッド呼び出し処理でのタイムアウト値の設定方法 144

Enterprise Bean を実行するための機能（EJB コンテナ） 20

F

FullGC 回数 96

H

HA モニタ 401
HA モニタによる 1:1 系切り替えシステム 487
HA モニタの環境設定 473
htc.clienthandleraddr 593, 600
HTTP Server で使用するコマンド，および HTTP Server に対する操作の監査ログの出力ポイント（UNIX の場合） 259
HTTP Server で使用するコマンド，および HTTP Server に対する操作の監査ログの出力ポイント（Windows の場合） 257
HTTP リクエスト実行待ちキュー監視 107
HTTP リクエスト実行待ちキュー枯渇監視情報 119

I

IM 構成ホスト 346

J

J2EE アプリケーションからネットワークリソースにアクセスする 125
J2EE アプリケーションからのネットワークリソースへのアクセス 194
J2EE アプリケーション起動／停止ジョブ定義用の定義プログラム 389
J2EE アプリケーションごとのメソッドタイムアウト時間 132
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 151
J2EE アプリケーション実行時間の監視 174
J2EE アプリケーション実行時間の監視とは 129
J2EE アプリケーション実行時間の監視の設定 305
J2EE アプリケーション停止処理の流れ 167
J2EE アプリケーション停止の種類 165
J2EE アプリケーションの入れ替え 181, 186
J2EE アプリケーションの入れ替えと保守 126, 186
J2EE アプリケーションの入れ替え方法 182
J2EE アプリケーションの運用 124
J2EE アプリケーションの運用機能 26
J2EE アプリケーションの運用機能に必要な定義 173
J2EE アプリケーションの強制停止 173
J2EE アプリケーションの構成パターンと閉塞方法 169
J2EE アプリケーションの実行時間の監視とキャンセル 125, 128
J2EE アプリケーションの実行時間の監視とキャンセルの流れ 148
J2EE アプリケーションの実行状態の確認 148
J2EE アプリケーションの状態 176
J2EE アプリケーションの世代管理 192
J2EE アプリケーションの停止 126, 165
J2EE アプリケーションの閉塞 126
J2EE アプリケーションの名称変更 192
J2EE アプリケーションを入れ替える 125
J2EE アプリケーションを開始するときの JSP の事前コンパイル 190
J2EE アプリケーションを強制停止する 124
J2EE アプリケーションを停止する 124
J2EE サーバ起動時に出力されるログ情報 151
J2EE サーバ共通用メッセージマッピングファイル 360
J2EE サーバ個別用メッセージマッピングファイル 360
J2EE サーバで使用するコマンドの監査ログの出力ポイント 223
J2EE サーバの運用監視の設定 388
J2EE サーバの機能 20
J2EE サーバ用 JP1 イベントの設定 359
J2EE ユーザ用 JP1 イベントの設定 361
JavaVM の稼働情報ファイルに出力される情報 81
jbs_route.conf 364
jbs_route.conf ファイル 356
jbsrt_distrib 356, 364
JCA リソースの稼働情報ファイルに出力される情報 89
JP1 331
JP1/AJS と連携してシステムの自動運転をする場合のシステム構築例（J2EE サーバの場合） 382
JP1/AJS を使用したシステムの自動運転 379

JP1/AJS を使用したバッチアプリケーションの自動実行 380
JP1/AJS - Agent 379
JP1/AJS - Manager 379
JP1/AJS - View 379
JP1/Base のイベントサーバ名の設定 363
JP1/Base の構成定義の作成 356, 364
JP1/Base の構成定義ファイル 356
JP1/IM 339
JP1/IM と連携した障害監視処理の流れ 350
JP1/IM 連携用モニタ起動設定ファイル 348
JP1/IM - View 339
JP1 イベント 339
JP1 イベントとして発行されるアプリケーションサーバシステムの障害情報 351
JP1 イベントのフィルタリング設定 349
JP1 連携による運用管理機能 27
JSP 事前コンパイル機能 183, 190
jstartrecover 578

K

KDJE54000-I 274
KDJE54001-I 274
KDJE54051-E 279

L

LAN の状態設定 464
localaddr 444, 459, 593, 600

M

maction.properties 305
Management Server で使用するコマンドの監査ログの出力ポイント 246
Management Server の自動再起動 40
Management Server の設定情報のコピー手順 451
Management Server 用 JP1 イベントの設定 359
Management アクション 44, 105, 296
Management アクション実行コマンド 302
Management アクション実行コマンドのコマンドファイルで使用できる環境変数 308
Management アクション実行コマンドの設定 308
Management アクション実行コマンドの動作 313
Management アクション実行コマンドのワーキングディレクトリ 313
Management アクション実行用プロパティファイル 302, 305
Management アクション実行用プロパティファイルの設定 305
Management アクションの実行制御 298
Management アクションの実行制御方式 298
Management イベント 44, 105, 296
Management イベントによる処理の自動実行の設定 301
Management イベントによる処理の自動実行の設定手順 301
Management イベント発行機能を利用する各機能の設定 305
Management イベント発行サーバを再起動するサンプル 309
Management イベント発行時の動作の設定 303
Management イベント発行の有効化 302
Management イベント発行用メッセージ ID リストファイル 302
Management イベント発行用メッセージ ID リストファイルの設定 303
manager.jp1event.system.filtering.severity.alert 360
manager.jp1event.system.filtering.severity.critical 360
manager.jp1event.system.filtering.severity.emergency 360
manager.jp1event.system.filtering.severity.error 360
manager.jp1event.system.filtering.severity.information 360
manager.jp1event.system.filtering.severity.notice 360
manager.jp1event.system.filtering.severity.warning 360
manager.jp1event.user.filtering.enabled 361
manager.jp1event.user.filtering.filter 361

manager.jp1event.user.mapping.level.config 361
manager.jp1event.user.mapping.level.fine 361
manager.jp1event.user.mapping.level.finer 361
manager.jp1event.user.mapping.level.finest 361
manager.jp1event.user.mapping.level.info 361
manager.jp1event.user.mapping.level.severe 361
manager.jp1event.user.mapping.level.warning 361
manager.mevent.message_id.list 303
manager.mevent.retry.interval 303
manager.mevent.retry.limit 303
manager.mevent.send.max 303
manager.mevent.send.timeout 303
manager.mevent.sender.bind.host 303
Message-driven Bean の稼働情報ファイルに出力される情報 86
MIB オブジェクト 282
mngagent.connector.host 512, 522
mngsvr_adapter_setup 354
mngsvr.jp1event.event_server_name 363
mngsvr.myhost.name 451, 471
mngsvrctl setup 355, 388
mngsvrutil.connect.host 355, 388
mngsvrutil.connect.password 355, 388
mngsvrutil.connect.userid 355, 388
mngsvrutil.properties 388
mngsvrutilcl.properties 388
mngsvrutil コマンドの実行環境の設定 387
monbegin 487, 537, 588
monend 487, 537, 588
monsbystp 487, 537
monswap 487, 537
mserver.jp1event.system.mapping.properties 359
mserver.properties 359, 450
mstrexport 451, 452, 472
mstrimport 452

N

N:1 リカバリ構成での仮想ホストの構成例 569
N:1 リカバリシステム 404, 549
N:1 リカバリシステムでのファイルの設定の要否 571
N:1 リカバリシステムの起動と停止（UNIX の場合） 588
N:1 リカバリシステムの起動と停止（Windows の場合） 585
N:1 リカバリシステムの構成例（UNIX の場合） 567
N:1 リカバリシステムのシステム構成例 550
N:1 リカバリシステムの設定（UNIX の場合） 566
N:1 リカバリシステムの設定（Windows の場合） 554
N 秒 293

O

ORB 機能 592, 599
ORB 機能使用時の設定 592, 600
OTS 機能 592, 599

S

SNMP 連携用形式 282
Stateful Session Bean の稼働情報ファイルに出力される情報 84
Stateless Session Bean の稼働情報ファイルに出力される情報 85
sysdef 460, 525, 574

U

URL グループ単位の実行待ちリクエスト数 97
URL グループの稼働情報ファイルに出力される情報 94

V

vbroker.agent.addr 594, 601
vbroker.agent.enableLocator 599
vbroker.se.iiop_tp.host 512, 522, 578

W

webserver.connector.nio_http.bind_host 512, 522

Web アプリケーション単位の実行待ちリクエスト数 97
Web アプリケーション単位の全体実行待ちリクエスト数 97
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） 20
Web アプリケーションの稼働情報ファイルに出力される情報 91
Web アプリケーションのサービスの部分閉塞による入れ替え 185
Web アプリケーションのサービスの閉塞と閉塞解除とは 156
Web アプリケーションの閉塞処理 168
Web アプリケーションのリクエスト処理でのタイムアウト値の設定方法 143
Web アプリケーションを実行するための機能（Web コンテナ） 20
Web コンテナ単位の実行待ちリクエスト数 97
Web コンテナ単位の全体実行待ちリクエスト数 96
Web コンテナの稼働情報ファイルに出力される情報 92
Web フロントシステムで実行できるサービス閉塞161
Web フロントシステムでのサービス閉塞 161
Windows Server Failover Cluster 401

あ

アーカイブ形式 183
アダプタコマンド 347
アダプタコマンドの実行環境の設定 354
アダプタコマンドのセットアップ 354
アプリケーション強制停止コマンド 130
アプリケーションサーバ 18
アプリケーションサーバ 11-20 での主な機能変更 33
アプリケーションサーバの 1:1 系切り替えシステムでの計画系切り替えの処理の流れ 435
アプリケーションサーバの 1:1 系切り替えシステムでの自動系切り替えの処理の流れ 434
アプリケーションサーバの JP1 イベント発行の設定 358
アプリケーションサーバのカスタムジョブの登録（Windows の場合） 389
アプリケーションサーバの機能 17
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 18
アプリケーションサーバを 1:1 系切り替えシステムで運用する場合のシステム構成例 430
アプリケーションサーバを対象にした 1:1 系切り替えシステム 478
アプリケーションサーバを対象にした 1:1 系切り替えシステムの起動と停止（Windows の場合） 478
アプリケーションサーバを対象にした 1:1 系切り替えシステムの構成例（UNIX の場合） 455
アプリケーションサーバを対象にした 1:1 系切り替えシステムの構成例（Windows の場合） 440
アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合） 454
アプリケーションサーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合） 439
アプリケーション制御用カスタムジョブ 385
アプリケーションの監査ログを出力するための実装 264
アプリケーションのユーザログの取得 276
アンデプロイ処理 172

い

一覧表示コマンド 130
イベントの発行機能 96
イベントの発行方法 98

う

運用管理エージェントの自動再起動 40
運用管理サーバの 1:1 系切り替えシステムでの計画系切り替えの処理の流れ 435
運用管理サーバの 1:1 系切り替えシステムでの自動系切り替えの処理の流れ 434
運用管理サーバを 1:1 系切り替えシステムで運用する場合のシステム構成例 431
運用管理サーバを対象にした 1:1 系切り替えシステム 484
運用管理サーバを対象にした 1:1 系切り替えシステムの起動と停止（Windows の場合） 484
運用管理サーバを対象にした 1:1 系切り替えシステムの構成例（UNIX の場合） 468

運用管理サーバを対象にした 1:1 系切り替えシステムの構成例（Windows の場合） 448
運用管理サーバを対象にした 1:1 系切り替えシステムの設定（UNIX の場合） 467
運用管理サーバを対象にした 1:1 系切り替えシステムの設定（Windows の場合） 447
運用時のトラブルについて 197

え

エイリアス IP アドレス 437, 440, 504

か

拡張 MIB オブジェクト定義ファイル 282
カスタムジョブ 333, 384
カスタムジョブによるジョブの定義（Windows の場合） 384
稼働状況のステータスの種類と意味 47
稼働情報監視で表示できる項目（J2EE サーバの場合） 285
稼働情報監視で表示できる項目（バッチサーバの場合） 290
稼働情報収集機能 67
稼働情報として監視できる項目 285
稼働情報の種類と稼働情報を出力する J2EE サーバの機能 68
稼働情報ファイル 67
稼働情報ファイルで収集できる情報 74
稼働情報ファイルで収集できる情報の種類 71
稼働情報ファイルの出力機能 68
稼働情報ファイルの出力形式と出力内容 77
稼働情報ファイルの出力先 75
稼働情報ファイルの出力先とファイル面数 75
稼働情報ファイルのファイル名 76
稼働情報ファイルのファイル面数と面の切り替え間隔 75
稼働統計出力ファイル 318
稼働統計出力ファイルに出力される情報 319
稼働統計出力ファイルの出力先 318
稼働統計出力ファイル名 319
稼働統計情報の出力情報 319
稼働統計情報ファイル 316
稼働統計情報ファイルの利用方法 323
簡易構築定義ファイルでの Management イベントの発行の定義（J2EE サーバ） 302
簡易構築定義ファイルでのイベントの発行の定義 101
簡易構築定義ファイルでの稼働情報ファイルの収集に必要な定義 77
簡易構築定義ファイルでのデータベース監査証跡との連携機能の定義 274
簡易構築定義ファイルでのリソース枯渇監視機能の定義 110
監査事象 203
監査事象の定義 202
監査で使用するメッセージの出力項目 206
監査ログ 202
監査ログが出力される流れと主な出力情報 202
監査ログ出力機能を使用するシステムで必要な作業と参照先 203
監査ログ出力の設定 266
監査ログ出力用の API の実装例 264
監査ログとは 202
監査ログの出力形式 205
監査ログの出力項目 206
監査ログの出力先 205
監査ログの出力に失敗した場合にシステムを自動で停止する流れ 212
監査ログの出力方式 205
監査ログの保存 212
監査ログを自動でアーカイブする流れ 213
監査ログを出力するコマンドの一覧（EJB クライアントアプリケーションで使用するコマンド） 221
監査ログを出力するコマンドの一覧（J2EE サーバで使用するコマンド） 214
監査ログを出力するコマンドの一覧（Management Server で使用するコマンド） 218
監査ログを出力するコマンドの一覧（性能解析トレース・CTM で使用するコマンド） 218
監査ログを出力するコマンドの一覧（バッチサーバで使用するコマンド） 216
監査ログを出力するコマンド・操作の一覧（UNIX の場合の HTTP Server で使用するコマンド・操作） 222

監査ログを出力するコマンド・操作の一覧（Windows の場合の HTTP Server で使用するコマンド・操作） 221
監視基盤 129
監視ツリー自動生成の処理の流れ 347
監視ツリーの自動生成 342, 356
監視ツリーの自動生成に必要な製品または構成ソフトウェア 353
監視ツリーの自動生成の設定 353
監視できるリソースの種類 106

き

機能ごとに取得できる稼働情報の種類 71
機能の分類と機能解説のマニュアルの対応 22
旧バージョンの製品から移行するための機能（移行機能） 22
旧バージョンの製品との互換のための機能（互換機能） 22
強制停止 166
強制停止処理 172
業務指向ツリー 343
業務指向ツリーの構成 344
業務指向ツリーのノード 344

く

クライアント側共通設定ファイル 388
クライアント側定義ファイル 387, 460
クラスタ IP アドレス 440
クラスタソフトウェアとの連携による系切り替え機能 27, 403
クラスタソフトウェア連携時の TPBroker の設定（UNIX の場合） 599
クラスタソフトウェア連携時の TPBroker の設定（Windows の場合） 592

け

系 402
計画系切り替え 434, 506
計画系切り替えの処理の流れ 435, 507
系切り替えシステムの設定内容の確認方法 497
系切り替え時の情報の引き継ぎ 437, 553
系切り替えの管理対象となるサーバ 406
系切り替えの処理の流れ 506
系切り替えのタイミング 432
系切り替えの動作 505
現用系 402
現用系から予備系への Management Server の設定情報のコピー 451, 471

こ

コールドスタンバイ 408
コネクションプール監視 108
コネクションプール枯渇監視情報 120
コマンドによってメソッドキャンセルを実行した場合に出力されるログ情報 154
コンソール出力情報 326
コンソールログ 326
コンソールログの出力対象 326
コンソールログの出力対象となる操作 326
コンソールログの出力対象となるプロセス 326
コンソールログの取得の設定 328

さ

サーバ側定義ファイル 388
サーバ指向ツリー 346
サーバ指向ツリーの構成 347
サーバ対応の環境設定での設定内容（N:1 リカバリシステムの場合） 579
サーバ対応の環境設定での設定内容（アプリケーションサーバの 1:1 系切り替えの場合） 463
サーバ対応の環境設定での設定内容（運用管理サーバの 1:1 系切り替えの場合） 475
サーバ対応の環境設定での設定内容（相互系切り替えシステムの場合） 527
サーバの稼働情報の出力方法 284
サービスの部分閉塞ができるシステム構成 185
サービスの部分閉塞による入れ替え 182
サービスの部分閉塞の方法 185
サービス部分閉塞 163
サービス閉塞 124, 166
サービス閉塞の解除 160

サービス閉塞の方法　156
サービス閉塞の方法と表示されるエラーページ　159
サービスユニット　36
サービスユニットのステータス監視　45
サンプリング時間　293

し

しきい値イベント　67
しきい値を設定できる監視対象　96
システムの運用監視　43
システムの運用状況の監視　43
システムの稼働監視　43
システムの監査を支援する機能　27
システムの起動　48
システムの起動と停止の設定　48
システムの起動と停止の手順　48
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）　21
システムの構成例（N:1 リカバリシステムの場合）　555
システムの自動運転　333
システムの自動運転をするために使用する JP1 プログラムの機能（J2EE サーバの場合）　382
システムの自動運転をするために使用する JP1 プログラムの機能（バッチサーバの場合）　383
システムの集中監視　332
システムの集中監視に必要なプログラム　341
システムの集中監視の仕組み　340
システムの集中監視をするために使用する JP1 プログラムの機能　342
システムの停止　50
システムの日常運用を支援する機能　25
システムの保守を支援する機能　26
システムの利用状況を監視するための機能（監視機能）　21
実行系　402, 585, 588
実行系の前提条件　407
自動起動の設定　52
自動系切り替え　434, 506
自動系切り替えの処理の流れ　434, 507
自動再起動の設定　55
自動停止の設定　58
出力される稼働情報　69
障害監視に必要な製品または構成ソフトウェア　358
障害監視の設定　358
障害の監視の仕組み　348
障害発生時の自動再起　40
ジョブ　379
ジョブによるシステムの自動運転に必要な製品または構成ソフトウェア　387
ジョブによるシステムの自動運転に必要なプログラム　381
ジョブによるシステムの自動運転の仕組み　379
ジョブによるシステムの自動運転の設定　387
ジョブネット　379
ジョブの定義　390

す

スケジュールの定義　397
ステーショナリ IP アドレス　437, 504
ステータス監視で確認できる項目　46
ステータスの監視方法　46
ステートメントキャンセル　166
スマートエージェントの localaddr ファイルの設定内容　593, 600
スマートエージェントを系切り替えの対象にしない場合の IP アドレスの設定例　604
スマートエージェントを系切り替えの対象にする場合の IP アドレスの設定例　603
スマートエージェントをフェイルオーバの対象にしない場合の IP アドレスの設定例　597
スマートエージェントをフェイルオーバの対象にする場合の IP アドレスの設定例　596
スレッド監視　107
スレッド枯渇監視情報　118
スレッドダンプ監視　107
スレッドダンプ枯渇監視情報　119
スレッドの状態とメソッドキャンセルの実行可否　150

せ

性能解析トレースの取得　277
性能解析トレース・CTM で使用するコマンドの監査ログの出力ポイント　242
セッション数監視　108
セッション数枯渇監視情報　120

そ

相互系切り替えシステム　404
相互系切り替えシステムでの計画系切り替えの処理の流れ　507
相互系切り替えシステムでの系切り替え処理の流れ　505
相互系切り替えシステムでの系切り替えの動作　506
相互系切り替えシステムでの自動系切り替えの処理の流れ　507
相互系切り替えシステムの起動と停止（UNIX の場合）　536
相互系切り替えシステムの起動と停止（Windows の場合）　533
相互系切り替えシステムの起動の流れ　538
相互系切り替えシステムの構成例　503
相互系切り替えシステムの構成例（UNIX の場合）　519
相互系切り替えシステムの構成例（Windows の場合）　509
相互系切り替えシステムのシステム構成例　502, 536
相互系切り替えシステムの設定（UNIX の場合）　518
相互系切り替えシステムの設定（Windows の場合）　508
相互系切り替えシステムの停止の流れ　540
相互系切り替えシステムをメンテナンスする場合の起動と停止　543
相互スタンバイ構成　508, 518
相互スタンバイシステムの構成例　533

た

待機系　402, 585, 588
待機系の前提条件　407
タイムアウトが発生したリクエストのキャンセル　150
タイムアウト検知間隔とメソッドタイムアウト時間　131
タイムアウトしたリクエストをキャンセルする　124
タイムアウト値の設定例と設定値の有効範囲　136
タイムアウトの延長でメソッドキャンセルを実行した場合に出力されるログ情報　152
タイムアウトの判定とタイムアウト後の動作　131
タイムアウトを検知する時間の間隔　131

つ

通常停止　165
通常の J2EE アプリケーションの入れ替えの手順　181
通常のサービス閉塞　163

て

停止処理　170
データベース監査証跡　270
データベース監査証跡と連携した運用例　276
データベース監査証跡に出力する情報　272
データベース監査証跡の出力個所　273
データベース監査証跡への出力の流れ　271
データベース監査証跡への情報の出力　271
データベース監査証跡連携機能　270
データベース監査証跡連携機能を使用できるデータベースと DBConnector　272
データベースの監査証跡情報の取得　277
デフォルトのタイムアウト時間で通常停止を実行して，停止しなかった場合に強制停止を実行する　177
デフォルトのタイムアウト時間で通常停止を実行する　175
展開ディレクトリ形式　184

と

統合機能メニュー　348
統合コンソール　342
統合スコープ　342
トラブル発生時などに対処するための機能（保守機能）　22
トランザクションサービスの稼働情報ファイルに出力される情報　90

に

日常運用での起動と停止　36

任意のタイムアウト時間を設定して通常停止を実行して，停止しなかった場合に強制停止を実行する　178

任意のタイムアウト時間を設定して通常停止を実行して，停止しなかった場合に自動的に強制停止を実行する　179

任意のタイムアウト時間を設定して通常停止を実行する　176

ね

ネットワークリソースにアクセスするための設定 195

ネットワークリソースへのアクセスの概要　194

は

バックシステム（CTM を使用していない場合）で実行できるサービス閉塞　162

バックシステム（CTM を使用している場合）で実行できるサービス閉塞　162

バックシステムでのサービス閉塞（CTM を使用しているシステムの場合）　161

バッチサーバで使用するコマンドの監査ログの出力ポイント　237

ひ

非保護区　132

ビューアー　339, 342

ふ

ファイルディスクリプタ監視　106

ファイルディスクリプタ枯渇監視情報　118

負荷分散機を使用したシステム構成でのサービス閉塞　157

負荷分散機を利用したサービス部分閉塞　163

負荷分散機を利用したサービス閉塞　162

プライベートプロパティの設定　563

プロセスリソースの稼働情報ファイルに出力される情報　84

フロント　160

フロント EJB　162, 173

フロント EJB の閉塞処理　168

へ

閉塞処理　168

ヘッダファイル　74

ヘッダファイルと稼働情報ファイルの対応　74

ほ

ほかの製品と連携して運用するための機能（連携機能）　21

保護区　128, 132, 150

保護区リストファイル　132

め

メソッドキャンセル機能　129

メソッドキャンセルコマンド　130

メソッドキャンセル実行のタイミング　135

メソッドキャンセル時の動作　135

メソッドキャンセルとは　132

メソッドキャンセルの処理　133

メソッドキャンセルを利用するアプリケーション開発時の注意事項　146

メソッドタイムアウト機能　129

メソッドタイムアウト機能の対象となる処理　131

メソッドタイムアウトとは　131

メソッドタイムアウトの適用対象（Enterprise Bean 呼び出し側）　144

メソッドタイムアウトの適用対象（Web アプリケーションのリクエスト処理）　143

メソッドタイムアウトの適用対象（コールバックメソッド側）　145

メソッドタイムアウト発生時のオプションメッセージの設定　146

メモリ監視　106

メモリ枯渇監視情報　115

も

モニタ起動コマンド　348

ゆ

ユーザ JP1 イベントの推奨設定　362

ユーザログ出力機能　352

よ

予備系　403

ら

ライトトランザクション機能の利用　406

ランタイム属性　188

り

リカバリ処理の流れ　551

リカバリ専用サーバ　554, 566

リソース枯渇監視機能とリソース枯渇監視情報の出力　106

リソース枯渇監視機能の有効化　110

リソース枯渇監視情報　105

リソース枯渇監視情報に出力される情報　113

リソース枯渇監視情報の出力形式　114

リソース枯渇監視の設定　305

リソース枯渇監視ログファイル　105

リソース種別ごとの監視の設定　110

リデプロイ　183, 187

リデプロイ機能による入れ替え　183

リデプロイ機能による入れ替え手順　183

リデプロイによって J2EE アプリケーションを入れ替えるときの JSP の事前コンパイル　191

リデプロイによる J2EE アプリケーションの入れ替え　187

リロード　184, 189

リロード機能による入れ替え　184

リロードによって J2EE アプリケーションを入れ替えるときの JSP の事前コンパイル　191

リロードによる J2EE アプリケーションの入れ替え　189

る

ルートアプリケーション情報　270, 272

ろ

ログイン方法の設定〔JP1/IM 連携〕　352

論理サーバおよび J2EE アプリケーションの起動と停止の自動化の仕組み　385

論理サーバ起動／停止ジョブ定義用の定義プログラム　389

論理サーバ制御用カスタムジョブ　384

論理サーバの稼働確認方法　38

論理サーバの起動と稼働確認　37

論理サーバの起動・停止の仕組み　37

論理サーバの自動再起動　40

論理サーバのステータス監視　45

論理サーバの停止　39