

uCosminexus Application Runtime for Apache  
Tomcat ユーザーズガイド

3021-3-K01-20

## 前書き

### ■ 対象製品

●適用 OS : Amazon Linux 2 (AMD/Intel 64), Debian GNU/Linux 10.0 (AMD/Intel 64), Debian GNU/Linux 11.0 (AMD/Intel 64), Red Hat Enterprise Linux 7 (AMD/Intel 64), Red Hat Enterprise Linux 8 (AMD/Intel 64), Red Hat Enterprise Linux 9 (AMD/Intel 64), Ubuntu 20.04 (AMD/Intel 64), Ubuntu 22.04 (AMD/Intel 64)

P-9W43-9N11 uCosminexus Application Runtime for Apache Tomcat 01-10

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

### ■ 輸出時の注意

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

なお、不明な場合は、弊社担当営業にお問い合わせください。

### ■ 商標類

HITACHI, Cosminexus, HiRDB, uCosminexus は、株式会社日立製作所の商標または登録商標です。Amazon Web Services, AWS, Powered by AWS ロゴ, Amazon Elastic Compute Cloud (Amazon EC2)は、Amazon.com, Inc. またはその関連会社の商標です。

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

Apache Tomcat, Tomcat(R)は、Apache Software Foundation の米国およびその他の国における登録商標または商標です。

Azure は、マイクロソフト 企業グループの商標です。

Docker および Docker ロゴは、Docker Inc. の米国およびその他の国における商標もしくは登録商標です。

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

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

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

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

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。



## ■ 発行

2024 年 4 月 3021-3-K01-20

## ■ 著作権

All Rights Reserved. Copyright (C) 2022, 2024, Hitachi, Ltd.

## 変更内容

### 変更内容(3021-3-K01-20) uCosminexus Application Runtime for Apache Tomcat 01-10

追加・変更内容	変更箇所
Java SE 17 に対応する記述を追加しました。	3.1, 6.1
JDK 11 に対応する記述を追加しました。	3.2
Tomcat 10.1.x に対応する記述を追加しました。	3.3, 6.3, 12.4, 12.5, 12.6
Tomcat 8.5.x に対応する記述を削除しました。	3.3, 6.3, 9.6, 12.5, 12.6
「空文字 ("")」と示していた項目を、出力なしであることを示す「-」に変更しました。	9.4, 9.5, 9.7, 9.9.1, 9.9.3, 9.9.4, 9.9.5
トレースログのフォーマットを追加しました。	13.3.1(2)
次のメッセージを追加しました。 KDLR00113-E	14.2
日立 JavaVM の起動オプションを追加・変更しました。	15.1

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

このマニュアルは、次の製品について説明したものです。

- uCosminexus Application Runtime for Apache Tomcat
- uCosminexus Application Runtime with Java for Apache Tomcat

以降、この2つの製品を総称して「本製品」と表記します。各製品の独自の説明については、製品名を表記します。

マニュアル中では、本製品の機能、設計、構築および運用方法について説明しています。また、定義ファイルやメッセージなどのリファレンス情報についても記載しています。

マニュアルを読むことで、ユーザがTomcatに本製品を組み込み、適切にシステムを運用できるようになることを目的としています。

### ■ 対象読者

対象読者は、次のとおりです。

- 本製品の概要を知りたい方
- 本製品を導入してシステムを設計・構築・運用する方

また、次の知識をお持ちの方を前提としています。

- Linux に関する知識
- Tomcat に関する知識
- クラウドまたはコンテナに関する知識
- Java Development Kit に関する知識
- Java EE または Jakarta EE に関する知識
- Web アプリケーションに関する知識

### ■ 読書手順

本製品を使用する環境によって、読む必要のある個所が異なります。対応を次の表に示します。

本製品を使用する環境	読む必要のある個所
<ul style="list-style-type: none"><li>• オンプレミス環境</li><li>• 仮想マシン環境</li></ul>	第1編 概要

本製品を使用する環境	読む必要のある箇所
	第2編 設計・構築・運用（オンプレミス環境または仮想マシン環境の場合）
	第4編 機能
	第5編 リファレンス
	付録
コンテナ仮想化環境	第1編 概要
	第3編 設計・構築・運用（コンテナ仮想化環境の場合）
	第4編 機能
	第5編 リファレンス
	付録

## ■ このマニュアルで使用する記号

このマニュアルで使用している記号を、次のように定義します。

記号	意味
	横に並べられた複数の項目に対する項目間の区切りを示し、「または」を意味します。 (例) A   B AまたはBを指定することを示します。
{ }	この記号で囲まれている複数の項目のうちから1つを選択することを示します。項目が横に並べられ、記号 で区切られている場合は、そのうちの1つを選択します。 (例) {A   B   C} A, BまたはCのどれかを指定することを示します。
[ ]	この記号で囲まれている項目は省略してもよいことを示します。複数の項目が横に並べて記述されている場合には、すべてを省略するか、記号{ }と同じくどれか1つを選択します。 (例 1) [A] 「何も指定しない」か「Aを指定する」ことを示します。 (例 2) [B   C] 「何も指定しない」か「BまたはCを指定する」ことを示します。
...	記述が省略されていることを示します。 (例) ABC… ABCの後ろに記述があり、その記述が省略されていることを示します。

記号	意味
< >	<p>この記号で囲まれている項目は、該当する要素やファイルなどを指定したり、該当する要素が表示されたりすることを示します。</p> <p>(例 1) &lt;プロパティ&gt; プロパティを記述します。またはプロパティが表示されます。</p> <p>(例 2) &lt;ファイル名&gt; ファイル名を指定します。</p>

## ■ Tomcat のインストール先パスおよび環境変数 CATALINA\_BASE が指すパスの表記

Tomcat のインストール先パスを `${CATALINA_HOME}`、環境変数 CATALINA\_BASE が指すパスを `${CATALINA_BASE}` と表記します。環境変数 CATALINA\_BASE を使用していない場合は、`${CATALINA_BASE}` を `${CATALINA_HOME}` に読み替えてください。

# 目次

前書き	2
変更内容	4
はじめに	5

## 第1編 概要

<b>1</b>	<b>概要</b>	<b>14</b>
1.1	本製品の目的	15
1.2	本製品の概要	16
1.3	本製品の前提条件	19
1.3.1	インストールが必要な OS, OSS および Java VM	19
1.3.2	Tomcat サーバプロセスの前提	19
1.3.3	その他の注意事項	19
1.4	本製品のユースケース	21
1.4.1	オンプレミス環境またはオンプレミス相当のクラウド上仮想マシン環境の場合	21
1.4.2	オートスケーリング構成のクラウド上仮想マシン環境の場合	22
1.4.3	コンテナ仮想化環境の場合	24

## 第2編 設計・構築・運用（オンプレミス環境または仮想マシン環境の場合）

<b>2</b>	<b>設計</b>	<b>27</b>
2.1	オンプレミス環境および仮想マシン環境共通の設計ポイント	28
2.1.1	本製品による性能影響を確認する	28
2.1.2	Tomcat プロセスモニタの HTTP 機能に対するセキュリティを確認する	29
2.1.3	本製品によるリソース消費量を確認する	29
2.2	オートスケーリング構成の仮想マシン特有の設計ポイント	30
2.2.1	スナップショットログの出力先を不揮発なストレージに設定する	30
2.2.2	オートスケーリンググループからの閉塞を高速化する	30
2.2.3	スケールイン開始前にスナップショットログ出力処理時間分の猶予を設ける	31
<b>3</b>	<b>構築</b>	<b>34</b>
3.1	セットアップの前提条件を確認する	35
3.2	インストールする	36
3.3	Tomcat に組み込む	38



<b>4</b>	<b>運用 40</b>
4.1	システムを起動する 41
4.1.1	本製品を組み込んだ Tomcat の起動方法 41
4.2	システムを停止する 43
4.2.1	本製品を組み込んだ Tomcat の停止方法 43
4.3	設定を変更する 45
4.4	オートスケーリング環境で運用する 46
4.5	保守資料を収集する 47
4.6	サポートサービスへ問い合わせる 48
4.7	修正パッチを適用する 49
4.8	アンセットアップする 50

### 第3編 設計・構築・運用（コンテナ仮想化環境の場合）

<b>5</b>	<b>設計 52</b>
5.1	コンテナ仮想化環境共通の設計ポイント 53
5.1.1	本製品による性能影響を確認する 53
5.1.2	Tomcat プロセスモニタの HTTP 機能に対するセキュリティを確認する 54
5.1.3	本製品によるリソース消費量を確認する 54
5.1.4	スナップショットログの出力先を不揮発なストレージに設定する 55
5.2	Kubernetes などのコンテナオーケストレーションツール特有の設計ポイント 56
5.2.1	コンテナの閉塞を高速化する 56
5.2.2	スケールイン開始前にスナップショットログ出力処理時間分の猶予を設ける 57
<b>6</b>	<b>構築 58</b>
6.1	セットアップの前提条件を確認する 59
6.2	インストーラを準備する 61
6.3	Dockerfile を作成する 62
6.4	Docker イメージをビルドする 64
<b>7</b>	<b>運用 65</b>
7.1	システムを起動する 66
7.2	システムを停止する 67
7.3	設定を変更する 68
7.4	Kubernetes 環境で運用する 69
7.5	保守資料を収集する 71
7.6	サポートサービスへ問い合わせる 72
7.7	修正パッチを適用した Docker イメージをビルドする 73
7.7.1	修正パッチ適用前の前提条件を確認する 73
7.7.2	修正パッチを適用した Dockerfile を作成する 73

## 第4編 機能

### 8 Tomcat プロセスモニタ機能 75

- 8.1 Tomcat プロセスモニタ機能の概要 76
- 8.2 Tomcat プロセスモニタ機能の適用方法 77
- 8.3 Tomcat プロセスモニタ機能の解除方法 78
- 8.4 Tomcat プロセスモニタ機能適用後の終了ステータス 79
- 8.5 自動でスナップショットログが収集されるタイミング 80
- 8.6 Tomcat サーバプロセスの強制終了 81
- 8.7 Tomcat プロセスモニタの HTTP 機能 82
- 8.8 Tomcat プロセスモニタの一時領域 83

### 9 トレース機能 84

- 9.1 トレース機能の概要 85
- 9.2 トレース機能のセットアップ方法 87
- 9.3 トレース機能のアンセットアップ方法 89
- 9.4 サーバの開始時および終了時のトレース 90
- 9.5 Web アプリケーションの開始時および終了時のトレース 91
- 9.6 HTTP リクエストおよびサーブレット・フィルタのトレース 92
- 9.7 HTTP セッションのトレース 95
- 9.8 JAX-RS クライアントのトレース 96
- 9.9 Tomcat JDBC Connection Pool のトレース 98
  - 9.9.1 javax.sql.DataSource インタフェースのトレース 98
  - 9.9.2 java.sql.Connection インタフェースのトレース 98
  - 9.9.3 java.sql.Statement インタフェースのトレース 100
  - 9.9.4 java.sql.PreparedStatement インタフェースのトレース 101
  - 9.9.5 java.sql.CallableStatement インタフェースのトレース 103
  - 9.9.6 Tomcat JDBC Connection Pool のトレース取得シーケンス 104
- 9.10 ユーザ作成スレッドおよび非同期処理 API 利用上の注意事項 106

### 10 稼働監視機能 107

- 10.1 稼働監視機能の概要 108
  - 10.1.1 稼働監視機能の位置づけ 108
  - 10.1.2 稼働監視機能の概要 109
  - 10.1.3 稼働監視機能の検知内容の出力（イベントプロパティ） 111
- 10.2 稼働監視機能の設定（基本項目） 112
  - 10.2.1 Tomcat プロセスモニタ側の設定 112
  - 10.2.2 Tomcat サーバプロセスへの稼働監視用ライフサイクルリスナーの設定 113

10.3	稼働監視機能の設定 (監視項目)	114
10.3.1	プロセスの起動監視	114
10.3.2	ハートビート監視	118
10.3.3	プロセス生存監視	120
10.3.4	ヘルスチェック	122
10.3.5	リクエスト処理の停滞監視	125
10.4	稼働監視機能の設定 (ユーザコマンドの実行)	129
<b>11</b>	<b>スナップショットログ収集機能</b>	<b>130</b>
11.1	スナップショットログ収集機能の概要	131
11.2	スナップショットログの出力	132
11.2.1	スナップショットログの出力先	132
11.2.2	スナップショットログの出力形式	132
11.3	スナップショットログの収集対象	134
11.3.1	ファイルによる情報の収集	135
11.3.2	コマンドの実行による情報の取得	138
11.4	機密情報のマスキング	144
11.4.1	定義情報のマスキング	144
11.4.2	config.properties (本製品の設定ファイル) のマスキング	145
11.5	スナップショットログの自動収集	146
11.5.1	障害時の保守情報の収集	146
11.6	スナップショットログの手動収集	148
11.6.1	Tomcat プロセスモニタが稼働しているときの収集方法	148
11.6.2	Tomcat プロセスモニタが稼働していないときの取得方法	149
11.6.3	スナップショットログの多重実行に関する注意事項	151
11.6.4	手動収集のユースケース	152
11.7	スナップショットログの出力テスト	153
11.8	ユースケース別の設定 (自動収集・手動収集共通)	154
11.8.1	収集対象を変更したい場合	154
11.8.2	機密情報のマスキングルールを追加したい場合	154
11.8.3	Context 要素の altDDName を設定している場合	155
11.8.4	monitor.tomcat.change.work.directory.enabled を false に変更している場合	155
11.8.5	スレッドダンプの出力先を変更したい場合 (日立 JavaVM 使用時)	155
11.8.6	ログ出力先を <common.base の指定値> 以外に設定している場合	156
11.8.7	同一環境で複数の Tomcat プロセスモニタを動作させる場合	156
11.9	ユースケース別の設定 (自動収集)	157
11.9.1	スナップショットログの出力先を変更したい場合	157
11.9.2	Tomcat サーバプロセスの正常終了時にもスナップショットログを出力したい場合	157
11.9.3	Tomcat サーバプロセスの停止要求時、停止する前に Tomcat 稼働中情報を取得したい場合	157

- 11.9.4 稼働監視で障害を検知したときの Tomcat 稼働中情報取得時の条件をカスタマイズしたい場合 158
- 11.10 ユースケース別の設定 (手動収集) 160
- 11.10.1 オプションの指定を省略したときのデフォルト値を変更したい場合 160

## 第5編 リファレンス

### 12 定義ファイル 161

- 12.1 定義ファイルの種類 162
- 12.2 config.properties (本製品の設定ファイル) 163
  - 12.2.1 形式 163
  - 12.2.2 ファイルの格納先, および格納先の変更方法 165
  - 12.2.3 機能 165
  - 12.2.4 指定可能なプロパティ 166
  - 12.2.5 ログ取得レベル 183
- 12.3 setenv.sh (Tomcat 起動時の環境変数定義ファイル) 185
  - 12.3.1 形式 185
  - 12.3.2 ファイルの格納先 185
  - 12.3.3 機能 186
  - 12.3.4 指定可能な環境変数 186
- 12.4 catalina.properties (Tomcat のプロパティ定義ファイル) 188
- 12.5 server.xml (Tomcat のサーバ設定ファイル) 189
- 12.6 context.xml (Tomcat のコンテキスト設定ファイル) 191

### 13 ログファイル 193

- 13.1 ログファイルの種類 194
- 13.2 Tomcat プロセスモニタのログ 195
  - 13.2.1 メッセージログ 195
  - 13.2.2 保守ログ 195
  - 13.2.3 JavaVM ログ 196
- 13.3 Tomcat サーバプロセスのログ 197
  - 13.3.1 Tomcat トレースログ 197
  - 13.3.2 JavaVM ログ 199

### 14 メッセージ 201

- 14.1 メッセージの形式 202
  - 14.1.1 メッセージの記述形式 202
  - 14.1.2 java.util.logging のレベル 203
- 14.2 メッセージの詳細 205

<b>15</b>	<b>JavaVM 起動オプション</b>	<b>248</b>
15.1	日立 JavaVM を使用する場合	249
15.2	他社製 JavaVM を使用する場合	263
<b>16</b>	<b>運用管理用コマンド</b>	<b>266</b>
16.1	運用管理用コマンドの概要	267
16.1.1	運用管理用コマンド文法の記述形式	267
16.1.2	運用管理用コマンドの入力形式	268
16.2	スナップショットログ収集コマンド	270
	collect-snapshot.sh (スナップショットログ収集)	270
<b>17</b>	<b>運用管理用 REST API</b>	<b>273</b>
17.1	運用管理用 REST API の概要	274
17.1.1	運用管理用 REST API の記述形式	274
17.2	スナップショットログ収集 REST API	275
	GET メソッド	275
<b>18</b>	<b>ユーザアプリケーションで使用できる API</b>	<b>278</b>
18.1	ユーザアプリケーションで使用できる API の概要	279
18.1.1	ユーザアプリケーションで使用できる API の記述形式	279
18.2	性能解析トレースで使用する API	280
	getClientApInfo メソッド	280
	getPrfTrace メソッド	281
	getRootApInfo メソッド	281
<b>付録</b>	<b>283</b>	
付録 A	GUI を使用したインストール	284
付録 B	GUI を使用したアンセットアップ	286
付録 C	本製品のアーカイブファイルを用いたインストールおよびアンセットアップ	288
付録 D	このマニュアルの参考情報	289
付録 D.1	関連マニュアル	289
付録 D.2	このマニュアルでの表記	289
付録 D.3	英略語	290
付録 D.4	KB (キロバイト) などの単位表記について	290

## 索引 291

# 1

## 概要

この章では、本製品の目的、概要、前提条件、およびユースケースについて説明します。

## 1.1 本製品の目的

---

本製品は、Apache Tomcat（以降、Tomcat と呼びます）※に対し、主に次のアドイン機能を付加します。

- 本番運用中の障害を早期に検知・通知する機能
- スムーズに問題解決するための保守情報の収集機能
- uCosminexus Application Server や HiRDB などの日立ミドルウェアと保守情報を連携するための情報付加機能

本製品を適用することで、問題解決に必要な保守情報が Tomcat のプロセスダウンと同時に消失することを回避できます。また、Tomcat が動作しているマシンに直接ログインできなくても、遠隔で保守情報を収集できます。そのため、特に次の環境で利用するのに適しています。

- オートスケーリング機能によって仮想マシンの自動増減が発生するクラウド環境
- オーケストレーションツールによってコンテナの起動・停止が頻発するコンテナ仮想化環境

本製品のサポートサービスでは、ユーザにこれらのアドイン機能をご利用いただき、そこで得られた情報を活用して、Tomcat に対する高度なサポートサービスを提供します。

### 注※

Tomcat は、Web コンテナ機能単体のアプリケーションサーバとして、デファクトスタンダードとなっています。

## 1.2 本製品の概要

本製品は、次の2つのコンポーネントで構成されています。

- Tomcat プロセスモニタ

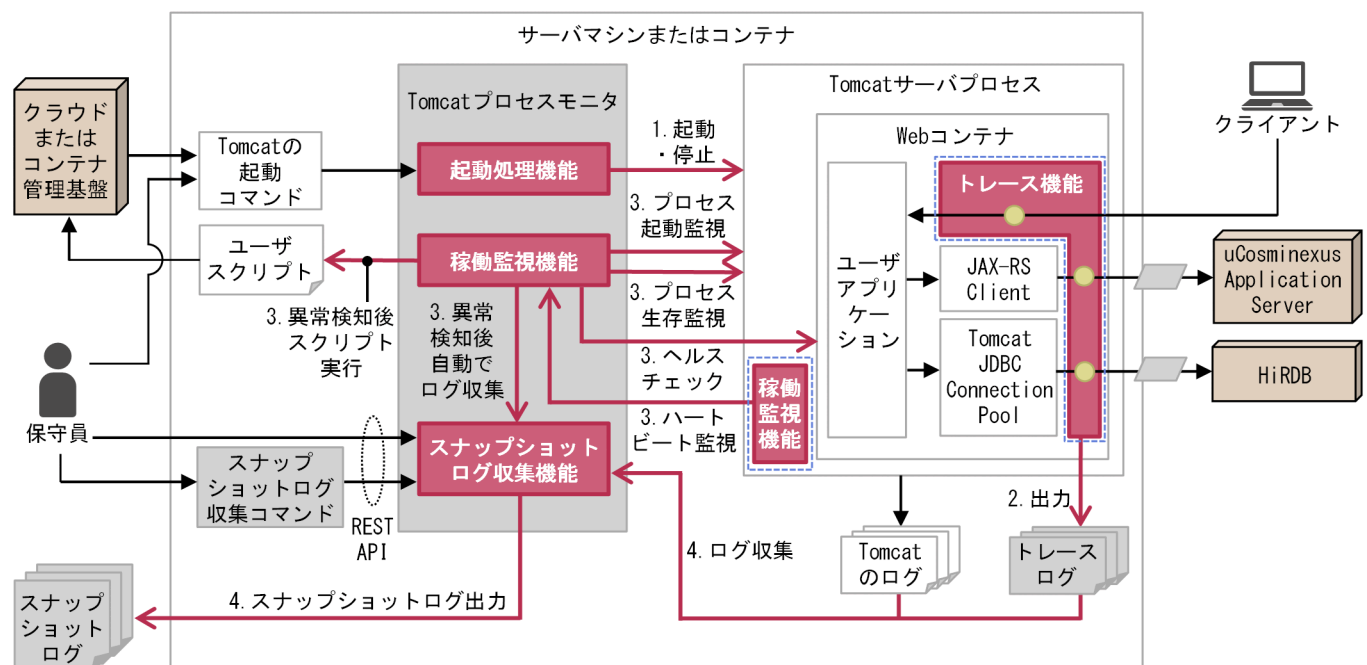
Tomcat プロセスモニタは、Tomcat の起動コマンドと Tomcat サーバプロセスの間に割り込み、Tomcat サーバプロセスの稼働状態監視や保守情報の収集を実行します。

- Tomcat サーバプロセス内のフィルタ

Tomcat サーバプロセス内のフィルタは、Tomcat の Valve や Listener などの実装として提供されます。各フィルタは、Tomcat サーバプロセス内の各種処理やリクエスト処理の間に割り込み、トレース情報や稼働監視情報を出力します。

これら2つのコンポーネントが図 1-1 のように連携することで、本製品は、Tomcat 単体では実現が困難な機能を提供します。これによって、Tomcat の保守性を向上できます。

図 1-1 本製品の機能構成全体図



(凡例)

- : 本製品の機能
- : 本製品のコンポーネント、ログおよびコマンド
- : 本製品のトレース取得ポイント
- : Cosminexus Performance Tracer用アプリケーション情報
- : Tomcatサーバプロセス内のフィルタ
- : この節で説明する本製品の機能の処理
- : そのほかの処理

本製品が提供する機能について説明します。各説明の番号は図中の番号と対応しています。



1. 起動処理機能は、Tomcat の起動処理を代行し、Tomcat の起動と同時に稼働監視や運用管理用 REST API の受付を開始します。

Tomcat が提供する設定変更の仕組みを利用して、Tomcat を起動するタイミングで Tomcat プロセスモニタを起動します。Tomcat プロセスモニタは Tomcat の起動コマンドの延長で自動起動します。詳細は、「[8. Tomcat プロセスモニタ機能](#)」を参照してください。

2. トレース機能は、リクエスト処理の内部状態を把握できる独自のトレース情報を出力します。

リクエストごとに一意の ID を採番し、リクエスト処理がどのような状態でどこまで到達していたかを把握できるトレース情報を常時出力します。uCosminexus Application Server の「[性能解析トレース機能](#)」に相当する機能です。

詳細は、「[9. トレース機能](#)」を参照してください。

3. 稼働監視機能は、プロセス監視やヘルスチェックを代行し、異常検知時には即座に指定したコマンド呼び出しや保守情報の収集をします。

クラウドのマネージドサービスやコンテナオーケストレーションツールが提供する既存のヘルスチェック機能に比べて、より Tomcat に特化した高度なプロセス監視・ヘルスチェックを提供します。異常発生からその検知と閉塞までのタイムラグ短縮に寄与します。

詳細は、「[10. 稼働監視機能](#)」を参照してください。

## ヒント

一般的なクラウドサービスも、オートスケーリング機能の自動スケールインを実現するために、ヘルスチェック機能を提供しています。ただし、それらの多くは、一定時間おきに投入する HTTP リクエストの応答によって正常/異常を判別しています。そのため、プロセスダウンのように、即座に業務が継続できなくなるような障害が発生しても、次の HTTP リクエストが投入されるまでの一定時間はプロセスダウンしているサーバを使い続けます。

一方、本製品の稼働監視機能では、異常発生からその検知と閉塞までのタイムラグが大幅に短縮できます。HTTP リクエストの応答だけでなく、プロセスダウンの即時検知や、高頻度なハートビートによるハングアップも検知するためです。また、本製品にはユーザが指定した任意のシェルスクリプトを呼び出す機能も備わっています。そのため、異常検知時には保守情報を自動収集するだけでなく、クラウドサービスに対して即座に閉塞を指示するコマンドを発行できます。

4. スナップショットログ収集機能は、異常検知直後に自動的に必要な保守資料を収集し、指定したディレクトリにアーカイブを出力します。

異常が検知されたときだけ、その時点で出力されていた保守資料を自動的に収集・アーカイブし、指定した永続化領域に出力する機能を提供します。

詳細は、「[11. スナップショットログ収集機能](#)」を参照してください。

## ヒント

次の環境では、障害発生時の原因解析に必要なログファイルを別のストレージ領域\*に永続化しておく必要があります。

- オートスケーリング機能によって仮想マシンの自動増減が発生するクラウド環境
- オーケストレーションツールによってコンテナの起動・停止が頻発するコンテナ仮想化環境

注※

ログファイルを格納するストレージ領域は、サーバがスケールインされてもログファイルが消失しない領域にする必要があります。

しかし、起動から停止まで正常稼働していた場合を含め、すべてのログファイルを永続化すると、ストレージ領域の容量や書き込み転送量が増大し、従量課金額がかさむリスクが生じます。また、一定時間経過後のログファイルを削除するなど、単調増加を防ぐ仕組みをユーザ側で構築する必要があります。

本製品のスナップショットログ収集機能では、障害発生時にサポートサービスに提供する保守資料（ログファイル）が、障害発生時にだけ自動で出力されます。そして、それを1つのアーカイブファイルとして永続化できます。保守資料がアーカイブされているため、個々の保守資料を永続化するための作業が削減できます。また、永続化先のストレージ容量と書き込み転送量を削減できます。

## 1.3 本製品の前提条件

---

この節では、次について説明します。

- インストールが必要な OS, OSS および Java VM
- Tomcat サーバプロセスの前提
- その他の注意事項

### 1.3.1 インストールが必要な OS, OSS および Java VM

本製品を使用するためには、次をインストールする必要があります。

- リリースノートに記載された OS
- Tomcat
- JavaVM

uCosminexus Application Runtime with Java for Apache Tomcat の場合は製品をインストールすると、JavaVM もインストールされます。

詳細は、リリースノートを参照してください。

### 1.3.2 Tomcat サーバプロセスの前提

本製品を組み込んだ Tomcat を使用する場合は注意事項を示します。

- Tomcat サーバプロセスに対して、Java のセキュリティマネージャは有効化できません。
- Tomcat のドキュメントに記載されている Tomcat 起動方法のうち一部は使用できません。詳細は「[4.1 システムを起動する](#)」を参照してください。

### 1.3.3 その他の注意事項

本製品全体に関わる注意事項を次に示します。

#### ファイルパスに使用できる文字に関する注意事項

本製品で使用するファイルパス・ディレクトリパスに、「"」,「\$」,「:」, および「\」を含めることはできません。含めると、正しく動作しないおそれがあります。

#### ファイルパスに関する前提

本製品で設定値として指定するファイルパス中に「../」が含まれる場合、次の前提条件を満たす必要があります。

- ファイルパス中のディレクトリは、本製品起動前に存在している  
存在しない場合、ファイルの出力やファイルの読み込みに失敗するおそれがあります。

本製品で設定値として指定するファイルパス中にシンボリックリンクが含まれる場合、次の前提条件を満たす必要があります。

- 本製品起動後に、シンボリックリンクが指すパスの変更をしない  
変更した場合、正しく動作しないおそれがあります。

また、本製品のファイル出力先のディレクトリは、本製品起動後に削除できません。ファイルの出力に失敗するおそれがあります。

#### 複数のコンテナ・マシン間によるディレクトリの共有

複数のコンテナや、複数のマシンで本製品を起動する場合、ファイルの出力先となるディレクトリが一意である必要があります。複数のコンテナ・マシン間でディレクトリを共有する場合、次の設定に対しては、同一の共有ディレクトリを指定しないでください。

- common.base (config.properties (本製品の設定ファイル) のプロパティ)  
common.base については「[12.2.4\(1\) 本製品全体に関するプロパティ](#)」を参照してください。
- デフォルトで common.base を参照する製品のプロパティ (デフォルトから変更する場合だけ)
- Tomcat サーバプロセスの一時領域ディレクトリ  
デフォルトは\${CATALINA\_BASE}/temp です。環境変数 CATALINA\_TMPDIR で変更できます。
- JAVACOREDİR 環境変数 (利用する場合だけ)

## 1.4 本製品のユースケース

本製品によって機能を拡張された Tomcat 環境は、OS 上に直接実行環境を構築するオンプレミス環境や仮想マシン環境だけでなく、次のような環境でも容易に運用できます。

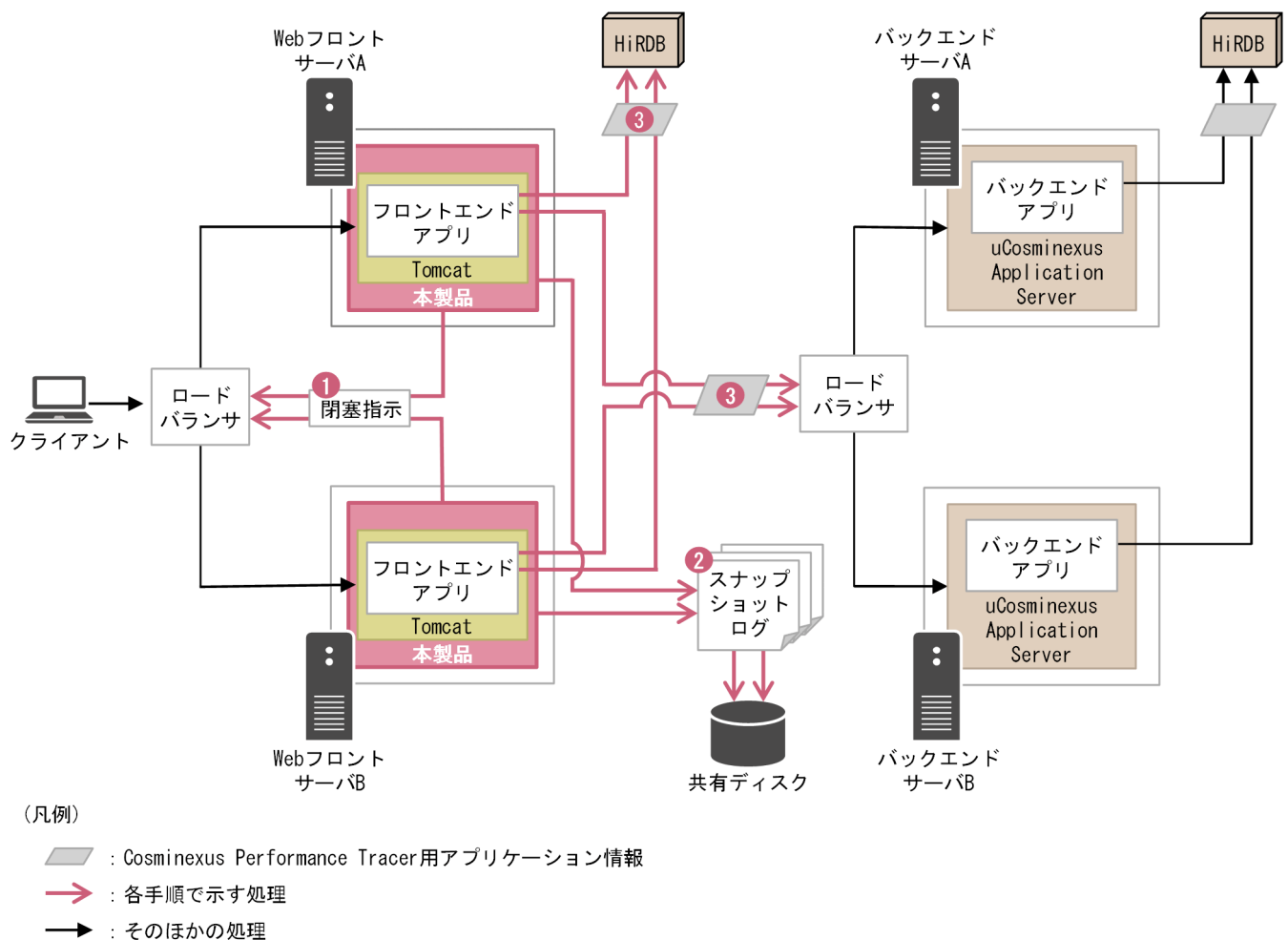
- クラウドのマネージドサービスによってひな型の仮想マシンイメージからオートスケーリングさせる環境
- コンテナ仮想化技術を使い、オーケストレーションツールによって Docker イメージの起動・停止をする環境

環境に応じたユースケースを次に示します。

### 1.4.1 オンプレミス環境またはオンプレミス相当のクラウド上仮想マシン環境の場合

本製品を組み込んだ Tomcat 環境をオンプレミス環境で使用する場合、負荷分散や業務継続性を考慮すると、次の図のようなシステム構成が想定されます。

図 1-2 オンプレミス環境の想定システム構成



オンプレミス環境，またはオンプレミス環境相当のクラウド上仮想マシン環境で本製品を使用した効果を次に示します。各説明の番号は図中の番号と対応しています。

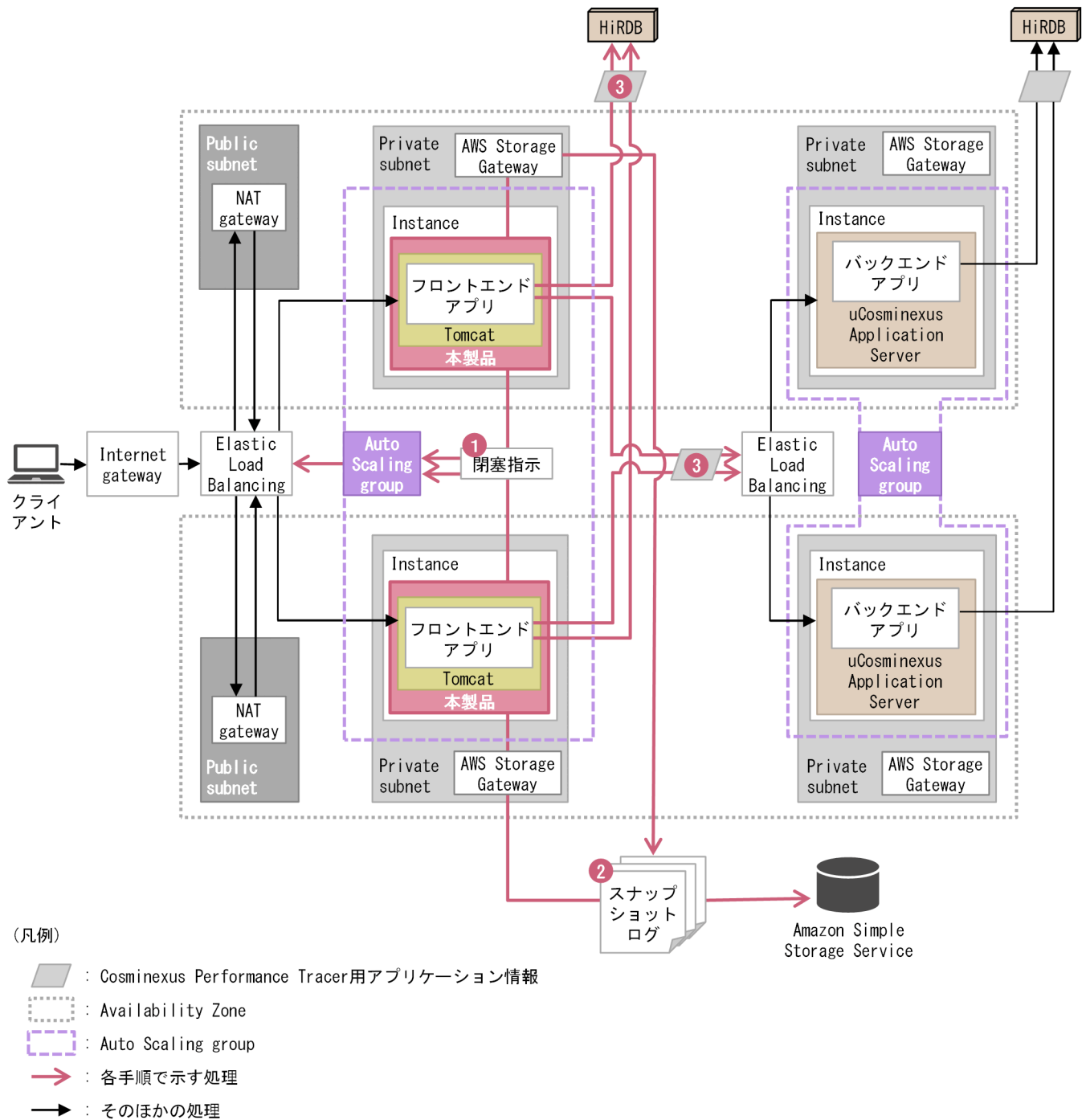
1. 本製品が異常を検知した場合，ユーザスクリプトを自動実行させることができます。  
ロードバランサの閉塞操作をするようユーザスクリプトに定義することで，異常発生からリクエスト閉塞までの時間を極小化できます。
2. 本製品が異常を検知した場合，サポートサービスへの問い合わせに必要な保守資料（ログファイル，定義ファイル，環境情報など）が自動的に収集されます。  
収集された保守資料は，スナップショットログという1つのアーカイブファイルとして出力されます。
3. Tomcat に対する HTTP リクエストには，本製品によってリクエストごとに一意の ID が採番され，トレースログに出力されます。  
採番された ID は，JAX-RS クライアントや HiRDB の JDBC ドライバに渡されるため，uCosminexus Application Server や HiRDB の性能解析トレース機能と連携できます。

オンプレミス環境，またはオンプレミス環境相当のクラウド上仮想マシン環境で本製品を使用する場合の設計・構築・運用については，このマニュアルの第2編を参照してください。

## 1.4.2 オートスケーリング構成のクラウド上仮想マシン環境の場合

オートスケーリング構成のクラウド環境（AWS や Azure など）で，本製品を組み込んだ Tomcat 環境を使用する場合，負荷分散や業務継続性を考慮すると，次の図のようなシステム構成が想定されます。

図 1-3 クラウド環境 (AWS) の想定システム構成



オートスケーリング構成のクラウド上仮想マシン環境で本製品を使用した効果を次に示します。各説明の番号は図中の番号と対応しています。

1. 本製品が異常を検知した場合、ユーザスクリプトを自動実行させることができます。  
ロードバランサの閉塞操作やオートスケーリングのスケールイン操作をユーザスクリプトに実装することで、異常発生からリクエスト閉塞までの時間を極小化することや、閉塞後のスケール回復ができます。
2. 本製品が異常を検知した場合、サポートサービスへの問い合わせに必要な保守資料（ログファイル、定義ファイル、環境情報など）が自動的に収集されます。

収集された保守資料は、スナップショットログという1つのアーカイブファイルとして出力されます。スナップショットログの出力先は、インスタンスがスケールインされても揮発しない外部ストレージに格納してください。

3. Tomcat に対する HTTP リクエストには、本製品によってリクエストごとに一意の ID が採番され、トレースログに出力されます。

採番された ID は、JAX-RS クライアントや HiRDB の JDBC ドライバに渡されるため、uCosminexus Application Server や HiRDB の性能解析トレース機能と連携できます。

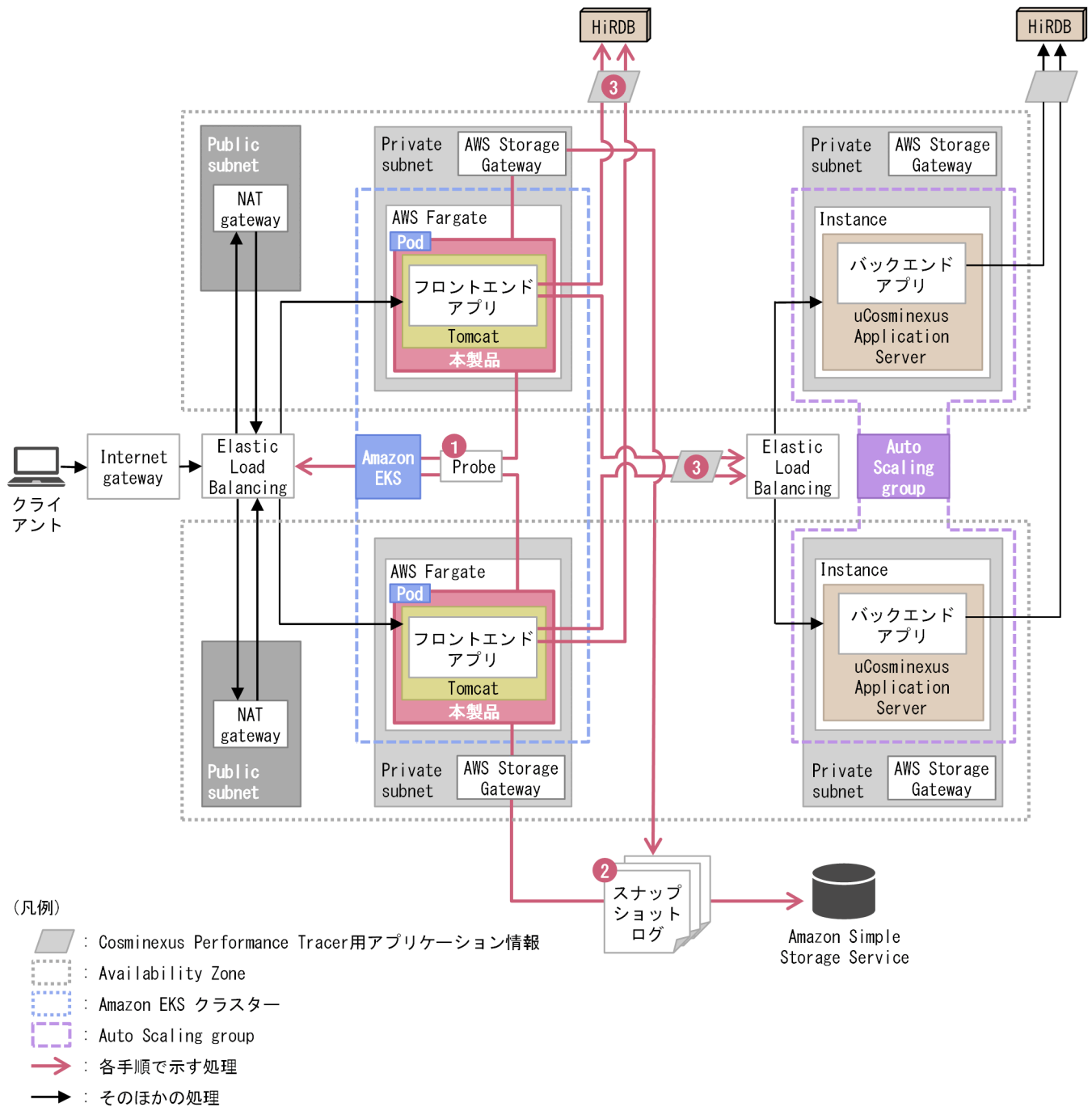
オートスケーリング構成のクラウド上仮想マシン環境で本製品を使用する場合の設計・構築・運用については、このマニュアルの第2編を参照してください。

### 1.4.3 コンテナ仮想化環境の場合

コンテナ仮想化環境のクラウド環境（AWS や Azure など）で、本製品を組み込んだ Tomcat 環境を使用する場合、負荷分散や業務継続性を考慮すると、次の図のようなシステム構成が想定されます。



図 1-4 コンテナ仮想化環境 (Amazon EKS + AWS Fargate) の想定システム構成



コンテナ仮想化環境で本製品を使用した効果を次に示します。各説明の番号は図中の番号と対応しています。

1. 本製品が異常を検知した場合、ユーザスクリプトを自動実行させることができます。  
 オーケストレーションツールに異常を知らせる Probe (Readiness Probe) をユーザスクリプトに実装することで、異常発生からリクエスト閉塞までの時間を極小化することや、閉塞後のスケール回復ができます。
2. 本製品が異常を検知した場合、サポートサービスへの問い合わせに必要な保守資料 (ログファイル, 定義ファイル, 環境情報など) が自動的に収集されます。

収集された保守資料は、スナップショットログという 1 つのアーカイブファイルとして出力されます。スナップショットログの出力先は、インスタンスがスケールインされても揮発しない Persistent Volume に格納してください。

3. Tomcat に対する HTTP リクエストには、本製品によってリクエストごとに一意の ID が採番され、トレースログに出力されます。

採番された ID は、JAX-RS クライアントや HiRDB の JDBC ドライバに渡されるため、uCosminexus Application Server や HiRDB の性能解析トレース機能と連携できます。

コンテナ仮想化環境で本製品を使用する場合の設計・構築・運用については、このマニュアルの第 3 編を参照してください。

# 2

## 設計

この章では、オンプレミス環境や仮想マシン環境で本製品を使用する場合に考慮する必要がある環境設計ポイントについて説明します。

## 2.1 オンプレミス環境および仮想マシン環境共通の設計ポイント

---

ここでは、オンプレミス環境および仮想マシン環境共通の設計ポイントについて説明します。設計のポイントは、次のとおりです。

- 本製品による性能影響を確認する
- Tomcat プロセスモニタの HTTP 機能に対するセキュリティを確認する
- 本製品によるリソース消費量を確認する

それぞれについて説明します。

### 2.1.1 本製品による性能影響を確認する

本製品では保守性を向上させるために、Tomcat サーバプロセスに対するトレース情報を拡充しています。また、Tomcat サーバプロセスに対する稼働監視を強化しています。

そのため、Tomcat の起動および停止性能や、アプリケーションに対するリクエスト処理性能に若干の影響を及ぼします。

本製品を使用する場合は、次に示す性能影響があります。これらの性能影響を考慮して性能設計をしてください。

#### (1) リクエスト処理性能

本製品のトレース機能では、リクエスト処理とは別スレッドでログ出力をします。これによって、リクエスト処理性能への影響を極小化しています。ただし、トレース情報として出力する値を取得するために、リクエスト処理スレッド上で実施する処理があるので、若干の性能影響が生じます。アプリケーションの内部保留時間に関係なく、リクエストごとに処理時間が一定であるため、内部保留時間が長い場合は無視できる範囲の影響です。ただし、内部保留時間が短い場合は無視できない性能影響となるおそれがあります。本製品を適用後の環境で性能評価をすることを強く推奨します。

#### (2) 起動性能

Tomcat サーバプロセスを起動させる前に Tomcat プロセスモニタの起動処理が割り込むこととなります。そのため、アプリケーションの規模や個数に関係なく、起動に掛かる時間が数秒増加します。

この起動時間の増分は、スケールアウトを開始してからアプリケーションが稼働状態になるまでの時間に影響します。

#### (3) 停止性能

Tomcat サーバプロセスを停止させる前に、スナップショットログ収集機能で Tomcat 稼働中情報が必ず取得されます。停止処理に失敗する可能性を考慮し、結果的に正常停止であっても取得されます。そのた

め、Tomcat 稼働中情報の取得処理によって停止時間が増加します。Tomcat 稼働中情報の取得については、「11. スナップショットログ収集機能」を参照してください。

なお、異常停止だけでなく、正常停止でもスナップショットログを収集する設定にしている場合は、スナップショットログ収集処理時間の分、さらに停止時間が増加します。スナップショットログ収集に必要な処理時間は、収集対象のデータ量に比例して増加します。

この停止時間の増分は、システム全体の稼働時間には影響しませんが、リクエストを閉塞させてからスケールインが完了するまでの時間には影響します。

## 2.1.2 Tomcat プロセスモニタの HTTP 機能に対するセキュリティを確認する

本製品では、スナップショットログの手動取得および Tomcat サーバプロセスとのプロセス間通信のために、HTTP リクエストを受け付ける HTTP 機能を使用されます。システムの運用自動化の妨げになることがあるため、この HTTP 機能にはユーザ認証機能や通信暗号化機能を設けていません。

デフォルトの設定では、HTTP 機能の受付ポートに接続できるクライアントはループバックアドレスからだけに設定されているため、マシン外部からの接続はできません。

スナップショットログの手動取得などのユーザ公開 REST API をマシン外部から実行する場合は、HTTP 機能の受付ポートへ外部から接続できるように設定を変更できます。その際は必ずファイアウォールや NAT/NAPT を適切に設定して、意図しない外部のユーザが不正に HTTP 機能の受付ポートに接続できないように通信をブロックしてください。

## 2.1.3 本製品によるリソース消費量を確認する

本製品を適用した Tomcat サーバプロセスでは、リクエスト処理スレッドとは別のスレッドで稼働監視やトレースログ出力を行うため、本製品を適用していない状態と比べてスレッド数が「2 スレッド」分増加します。

Tomcat サーバプロセスが生成する最大スレッド数については、Tomcat のドキュメントを参照して算出してください。本製品を使用する場合は、その最大スレッド数に対して「2」を加算して見積もってください。

また、本製品では、Tomcat サーバプロセスと同時に Tomcat プロセスモニタのプロセスが起動されるため、OS の仮想メモリの消費量が約 2,400 MB 分増加します。仮想メモリの消費量を見積もる際には、Tomcat サーバプロセスが消費する仮想メモリに対して、2,400 MB を加算して見積もってください。

Tomcat サーバプロセスが消費する仮想メモリについては、使用する JavaVM のドキュメントを参照して算出してください。日立 JavaVM を使用している場合は、マニュアル「uCosminexus Application Runtime - Cosminexus Developer's Kit for Java 機能解説・リファレンス」を参照してください。

## 2.2 オートスケーリング構成の仮想マシン特有の設計ポイント

ここでは、オートスケーリング構成の仮想マシン特有の設計ポイントについて説明します。設計のポイントは、次のとおりです。

- スナップショットログの出力先を不揮発なストレージに設定する
- オートスケーリンググループからの閉塞を高速化する
- スケールイン開始前にスナップショットログ出力処理時間分の猶予を設ける

それぞれについて説明します。

### 2.2.1 スナップショットログの出力先を不揮発なストレージに設定する

Tomcat をオートスケーリング構成のマシン上で使用している場合、マシン内だけに保存していたログファイルおよび環境情報はスケールインとともに削除されます。障害発生を契機にスケールインされると、その障害要因の特定が困難になります。

本製品では、稼働監視機能とスナップショットログ収集機能によって、障害発生が検知されるとすぐに障害解析に必要なログファイルや環境情報が自動収集されます。これらを基に、スナップショットログが生成されます。そのため、このスナップショットログの出力先は、マシン外の不揮発ストレージにマウントされたディレクトリに設定してください。これによって、スケールイン後もスナップショットログを参照できます。詳細は、「[4.4 オートスケーリング環境で運用する](#)」を参照してください。

### 2.2.2 オートスケーリンググループからの閉塞を高速化する

Tomcat をオートスケーリング構成のマシン上で使用している場合、本製品の稼働監視機能によって障害が検知された瞬間に、オートスケーリンググループからの切り離し（閉塞）が実施されます。ロードバランサのヘルスチェック機能を使用する場合に比べて、障害発生から閉塞までの不稼働時間を短縮できます。

稼働監視機能のユーザコマンドによる情報通知を使用することで、障害の検知を契機に任意のスクリプトを実行できます。そのスクリプト内で、クラウドベンダが提供するコマンドおよび API などを用いて自マシンをオートスケーリンググループから切り離す処理を実装してください。

オートスケーリンググループから切り離す手段については、各クラウドベンダが提供するドキュメントを参照してください。また、次の点にも注意してください。

- 実行するコマンドがインストール済みか
- コマンドおよび API の実行に必要な権限があるか

AWS で Amazon EC2 Auto Scaling を使用している場合のユーザスクリプト実装例を次に示します。

```
#!/bin/bash

# Get InstanceID
TOKEN=$(curl -X PUT http://169.254.169.254/latest/api/token \
-H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCEID=$(curl http://169.254.169.254/latest/meta-data/instance-id/)\
-H "X-aws-ec2-metadata-token: ${TOKEN}")
echo InstanceID=${INSTANCEID}

# Get AutoScalingGroupName
ASGNAME=$(aws autoscaling describe-auto-scaling-instances --instance-ids ${INSTANCEID} | \
jq -r '.AutoScalingInstances[0].AutoScalingGroupName')
echo AutoScalingGroupName=${ASGNAME}

# Enter Standby state
aws autoscaling enter-standby --instance-ids ${INSTANCEID} \
--auto-scaling-group-name ${ASGNAME} --no-should-decrement-desired-capacity
```

## 注

このスクリプトは実装例であり、これをそのまま使用した場合の動作は保証しません。必ずクラウドベンダが提供する最新のドキュメントを参照してユーザ側で適切なスクリプトを記述してください。また、次の点にも注意してください。

- 実行するコマンドがインストール済みか
- コマンドの実行に必要な権限があるか

## 2.2.3 スケールイン開始前にスナップショットログ出力処理時間分の猶予を設ける

Tomcat をオートスケーリング構成のマシン上で使用している場合、本製品がスナップショットログの収集処理をしている間は、できる限りマシンがスケールインされないようにする必要があります。

### ユーザスクリプトによる閉塞をすることで自動的にスケールインされないように構築している場合

このケースに該当する場合は、Tomcat の停止後にスケールインを実行するスクリプトが実行されるように設定してください。これによって、スナップショットログ出力処理が終わるまでスケールインされないようにできます。

systemd を用いて OS 起動と同時に Tomcat をサービス起動させ、Tomcat の停止後にスケールインを実行する場合のユニットファイル定義例を次に示します。

```
[Unit]
Description=Apache Tomcat
After=network.target

[Service]
Type=forking
ExecStart=<Tomcatインストール先>/bin/startup.sh
ExecStop=<Tomcatインストール先>/bin/shutdown.sh
ExecStopPost=<スケールインを実行するスクリプトのパス>
```

```
TimeoutStopSec=700
SuccessExitStatus=143
User=tomcat
Group=tomcat

[Install]
WantedBy=multi-user.target
```

AWS で Amazon EC2 Auto Scaling を使用している場合のスケールインを実行するスクリプトの実装例を次に示します。

```
#!/bin/bash

# Get InstanceID
TOKEN=$(curl -X PUT http://169.254.169.254/latest/api/token \
-H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCEID=$(curl http://169.254.169.254/latest/meta-data/instance-id/)

echo InstanceID=${INSTANCEID}

# Polling Instance status
MAX_POLLING_COUNT=10
RETRY_INTERVAL=60
for ((i=1;i<=${MAX_POLLING_COUNT};i++)); do
    INSTSTATUS=$(aws autoscaling describe-auto-scaling-instances --instance-ids ${INSTANCEID} \
    | jq -r '.AutoScalingInstances[].LifecycleState')
    echo InstanceStatus=${INSTSTATUS}
    if [ "${INSTSTATUS}" == "Standby" ]; then
        # Terminate instance
        aws autoscaling terminate-instance-in-auto-scaling-group --instance-id ${INSTANCEID} \
        --no-should-decrement-desired-capacity
        echo "Instance will be terminated."
        exit 0
    fi
    if [ ${i} -eq ${MAX_POLLING_COUNT} ]; then
        echo "Max polling count reached (count = ${i})."
        exit 1
    fi
    echo "InstanceStatus is not STANDBY. Retry after ${RETRY_INTERVAL} seconds (count = ${i})."
    sleep ${RETRY_INTERVAL}s
done
```

## 注

このスクリプトは実装例であり、これをそのまま使用した場合の動作は保証しません。必ずクラウドベンダが提供する最新のドキュメントを参照してユーザ側で適切なスクリプトを記述してください。また、次の点にも注意してください。

- 実行するコマンドがインストール済みか
- コマンドの実行に必要な権限があるか



オートスケーリンググループからの切り離しと同時にスケールインに遷移するように設定している場合（マネージドサービス側で管理）

スナップショットログの出力に掛かる時間を確保できる、十分な猶予時間を設定してください。猶予時間とは、障害を検知されたマシンがオートスケーリンググループから切り離されてから、OSのシャットダウンが開始されるまでの時間を指します。

# 3

## 構築

この章では、オンプレミス環境または仮想マシン環境の場合の本製品の構築手順について説明します。

## 3.1 セットアップの前提条件を確認する

---

ここで説明するセットアップ手順は、次に示す条件を満たしていることを前提としています。

- Tomcat をインストールしている
- 環境変数 CATALINA\_HOME および環境変数 CATALINA\_BASE の値にシンボリックリンクが含まれる場合、シンボリックリンクの後に「..」が含まれていない
- Tomcat を起動していない
- バージョンに関係なく、本製品をインストールしていない
- sudo コマンドを実行して管理者権限でのコマンド実行ができる  
ここでは sudo コマンドを実行して管理者権限で操作することを前提に説明しています。管理者権限でログインして操作する場合は、sudo コマンドは不要です。

uCosminexus Application Runtime for Apache Tomcat を使用する場合は、さらに次の前提条件を満たしていることも確認してください。

- Java SE 8 以降、Java SE 11 以降、または Java SE 17 以降に準拠する Java Runtime Environment (JRE) をインストールしている
- 上記の JRE のインストールディレクトリの絶対パスを、環境変数 JAVA\_HOME または JRE\_HOME に設定している (「`{JAVA_HOME}/bin/java`」または「`{JRE_HOME}/bin/java`」が存在する)

この章では、環境構築の自動化を想定して、GUI などによるオペレータの操作が不要なインストール方法について説明します。

GUI を使ったオペレータの操作でインストールしたい場合は、「[付録 A GUI を使用したインストール](#)」を参照してください。

本製品のアーカイブファイルを用いてインストールすることもできます。アーカイブファイルを用いたインストールを実施することによるメリットおよび手順については、「[付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ](#)」を参照してください。

## 3.2 インストールする

本製品をインストールするための操作手順を次に示します。

### 操作手順

#### 1. インストール CD-ROM のデータを、本製品をインストールするマシンにコピーする。

本製品のインストール CD-ROM をマウントし、CD-ROM に格納されている X64LIN ディレクトリを、インストール先のマシン上の任意のディレクトリに X64LIN ディレクトリごとコピーしてください。これ以降、X64LIN ディレクトリのコピー先を<インストーラのパス>と表記します。

#### 2. setup コマンドに実行権限を付ける。

「<インストーラのパス>/X64LIN/setup」という実行ファイルに対して、管理者権限で実行権限を付与します。

#### 実行例

```
$ sudo chmod +x <インストーラのパス>/X64LIN/setup
```

#### 3. PP インストーラの setup コマンドで、セットアッププログラムを実行する。

次の引数で setup コマンドを実行します。

```
$ sudo <インストーラのパス>/X64LIN/setup -f -k <形名> <インストーラのパス>
```

<形名>はインストールする製品エディションによって異なります。

形名が P-9W43-9N11 の場合は、次のとおりコマンドを実行してください。

```
$ sudo <インストーラのパス>/X64LIN/setup -f -k P-9W43-9N11 <インストーラのパス>
```

これでインストール作業は完了です。本製品は次のパスにインストールされます。

```
/opt/hitachi/ucart
```

uCosminexus Application Runtime with Java for Apache Tomcat をインストールした場合は、本製品に加えて日立 JavaVM がインストールされます。インストールされるパスを次に示します。

#### JDK 11 ベースの日立 JavaVM をインストールする場合

```
/opt/Cosminexus/jdk
```

インストールが正常に完了しているかどうかは、インストール先の install.log で確認できます。次のコマンドを実行してください。

```
$ sudo cat /opt/hitachi/ucart/install.log
```

次の出力例のように「rc=0 msg=I:Installation completed.」と出力されていればインストールは正常に完了しています。

#### 出力例

```
2022/06/30 12:34:56 rc=0 msg=I:Installation completed.
```

なお、ソフトウェアサポートサービスの Web サイトから修正パッチが提供されている場合は、最新の修正パッチを入手して適用してください。ソフトウェアサポートサービスの Web サイトにアクセスできない場合は、本製品に同梱されている修正パッチ CD を利用してください。修正パッチの適用方法については、「[4.7 修正パッチを適用する](#)」を参照してください。

## 3.3 Tomcat に組み込む

本製品を Tomcat に組み込むための方法を次に示します。

ここでは、Tomcat のインストール先パスを `${CATALINA_HOME}` と表記し、環境変数 `CATALINA_BASE` が指すパスを `${CATALINA_BASE}` と表記します。環境変数 `CATALINA_BASE` を使用していない場合は、`${CATALINA_BASE}` を `${CATALINA_HOME}` に読み替えてください。

操作手順を次に示します。

### 操作手順

#### 1. インストールディレクトリとファイルの所有グループを Tomcat 実行ユーザに合わせて変更する。

root 以外のユーザで Tomcat を実行する場合、本製品のインストールディレクトリとその配下のサブディレクトリおよびファイルの所有グループを、Tomcat 実行ユーザが属するグループに変更します (Tomcat 実行ユーザに書き込み権限を与えないように、所有ユーザは root のままにしてください)。Tomcat 実行ユーザが属するグループ名が「tomcat」の場合の実行例を次に示します。

```
$ sudo chgrp -R tomcat /opt/hitachi/ucart
```

#### 2. 各種定義ファイルのバックアップを作成する。

次の 3 つの定義ファイルについて、編集前の状態でバックアップを作成しておきます。

- `${CATALINA_BASE}/conf/catalina.properties` (Tomcat のプロパティ定義ファイル)
- `${CATALINA_BASE}/conf/server.xml` (Tomcat のサーバ設定ファイル)
- `${CATALINA_BASE}/conf/context.xml` (Tomcat のコンテキスト設定ファイル)

また、次の 2 つのファイルについても、すでに存在する場合はバックアップを作成しておきます。

- `${CATALINA_HOME}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)
- `${CATALINA_BASE}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)

#### 3. `${CATALINA_BASE}/bin/setenv.sh` を作成する。

本製品の動作に必要な環境変数を、`${CATALINA_BASE}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル) に定義します。

このファイルのテンプレートをコピーして使うこともできます。次のコマンドでコピーできます。

```
$ sudo cp /opt/hitachi/ucart/template/tomcat/setenv.sh "${CATALINA_BASE}/bin/setenv.sh"
```

ユーザ側でカスタマイズが必要な場合は、コピーしたあとのファイルを修正してください。

`setenv.sh` (Tomcat 起動時の環境変数定義ファイル) の詳細は、「[12.3 setenv.sh \(Tomcat 起動時の環境変数定義ファイル\)](#)」を参照してください。

#### 4. `${CATALINA_BASE}/conf/catalina.properties` を編集する。

本製品の動作に必要なプロパティを、`${CATALINA_BASE}/conf/catalina.properties` (Tomcat のプロパティ定義ファイル) に追記します。

sed コマンドを利用して置換する場合は、次のように実行してください。

```
$ sudo sed -i -e "s/^(common%. loader=[^%r]*%)(%r%?%)$/%1,%"%"%${com.cosminexus.appruntime.lib.path}%/tomcat%/common%/*.jar"%2/g" "%${CATALINA_BASE}/conf/catalina.properties"
```

catalina.properties (Tomcat のプロパティ定義ファイル) の詳細は、「[12.4 catalina.properties \(Tomcat のプロパティ定義ファイル\)](#)」を参照してください。

## 5. \${CATALINA\_BASE}/conf/server.xml を編集する。

本製品の動作に必要な定義を、\${CATALINA\_BASE}/conf/server.xml (Tomcat のサーバ設定ファイル) に追記します。

このファイルのテンプレートをコピーして使うこともできます。ユーザ側で server.xml (Tomcat のサーバ設定ファイル) のカスタマイズが必要な場合は、テンプレートからコピーしたあとでカスタマイズしてください。

### Tomcat 9.x の場合

```
$ sudo cp /opt/hitachi/ucart/template/tomcat9/server.xml "%${CATALINA_BASE}/conf/server.xml"
```

### Tomcat 10.x の場合

```
$ sudo cp /opt/hitachi/ucart/template/tomcat10.1/server.xml "%${CATALINA_BASE}/conf/server.xml"
```

server.xml (Tomcat のサーバ設定ファイル) の詳細は、「[12.5 server.xml \(Tomcat のサーバ設定ファイル\)](#)」を参照してください。

## 6. \${CATALINA\_BASE}/conf/context.xml を編集する。

本製品の動作に必要な定義を、\${CATALINA\_BASE}/conf/context.xml (Tomcat のコンテキスト設定ファイル) に追記します。

このファイルのテンプレートをコピーして使うこともできます。ユーザ側で context.xml (Tomcat のコンテキスト設定ファイル) のカスタマイズが必要な場合は、テンプレートからコピーしたあとでカスタマイズしてください。

### Tomcat 9.x の場合

```
$ sudo cp /opt/hitachi/ucart/template/tomcat9/context.xml "%${CATALINA_BASE}/conf/context.xml"
```

### Tomcat 10.x の場合

```
$ sudo cp /opt/hitachi/ucart/template/tomcat10.1/context.xml "%${CATALINA_BASE}/conf/context.xml"
```

context.xml (Tomcat のコンテキスト設定ファイル) の詳細は、「[12.6 context.xml \(Tomcat のコンテキスト設定ファイル\)](#)」を参照してください。

これで Tomcat への組み込み作業は完了です。

# 4

## 運用

この章では、オンプレミス環境または仮想マシン環境の場合のシステムの起動、停止、設定変更、アンセットアップ方法などについて説明します。



## 4.1 システムを起動する

本製品を組み込んだ Tomcat の起動方法を説明します。

オートスケーリング環境下で本製品と Tomcat を使用する場合は、先に「[4.4 オートスケーリング環境で運用する](#)」を参照してください。

### 4.1.1 本製品を組み込んだ Tomcat の起動方法

本製品を組み込んだ Tomcat は、用途に応じて次のどれかのコマンドで起動できます。

- バックグラウンドで起動する場合

次のどちらかのコマンドで起動できます。

```
$ sudo "${CATALINA_HOME}/bin/startup.sh"
```

または

```
$ sudo "${CATALINA_HOME}/bin/catalina.sh" start
```

- リモートデバッグを有効にしてバックグラウンドで起動する場合

```
$ sudo "${CATALINA_HOME}/bin/catalina.sh" jpda start
```

- フォアグラウンドで起動する場合

```
$ sudo "${CATALINA_HOME}/bin/catalina.sh" run
```

root 以外のユーザで Tomcat を実行する場合は、sudo コマンドに「-u <Tomcat 実行ユーザ名またはユーザ ID>」オプションを付けて起動するか、または Tomcat 実行ユーザでログインしているシェル上で sudo コマンドを介さないで起動してください。

なお、同一マシン内に同時に起動する Tomcat のインスタンスが複数ある場合は、Tomcat プロセスモニタの HTTP 機能の受付ポート番号が同一マシン内で重複しないように設定する必要があります。起動する前に config.properties（本製品の設定ファイル）の monitor.rest.port プロパティを変更してください。

#### ❗ 重要

- どの起動方法でも、-security オプションは使用できません。
- 本製品を使用する場合、Tomcat のドキュメント「Tomcat Setup」に記載されている「jsvc」を使用したデーモン起動はできません。また、\${CATALINA\_HOME}/bin/daemon.sh を使用した起動もできません。必ず、\${CATALINA\_HOME}/bin/catalina.sh または \${CATALINA\_HOME}/bin/startup.sh を使用して起動してください。

- catalina.sh debug で Tomcat を起動した場合、本製品が適用されていない (Tomcat プロセスモニタが起動していない) 状態となります。デバッグのために catalina.sh debug を使用したい場合は、catalina.sh jpda start の使用を検討してください。

## 4.2 システムを停止する

本製品を組み込んだ Tomcat の停止方法を説明します。

### 4.2.1 本製品を組み込んだ Tomcat の停止方法

本製品を組み込んだ Tomcat は、用途に応じて次のどれかの方法で正常に停止できます。

- 次のどちらかのコマンドを実行する。

```
$ sudo "${CATALINA_HOME}/bin/shutdown.sh"※
```

または

```
$ sudo "${CATALINA_HOME}/bin/catalina.sh" stop※
```

注※ -force オプションはサポートしていません。

- フォアグラウンドで起動していた場合は、catalina.sh run を実行したコンソール上で [Ctrl] + [C] キーを入力する。
- Tomcat プロセスモニタの PID に対して SIGTERM シグナルを送信する。

#### ❗ 重要

本製品を使用する場合、それぞれのプロセスの PID は親子関係になっています。次の表に、その親子関係を示します。システムの停止をするために SIGTERM シグナルを送信する場合は、表の (A) または (B) の PID に対して送信してください。

プロセス	PID	親 PID	コマンドライン
ラッパー スクリ プト	(A)	catalina.sh または startup.sh の呼び出し 元プロセス の PID で す。 シェルや systemd から 起動した 場合、通常 は「1」で す。	bash /opt/hitachi/ucart/script/wrapper.sh …(略)… org.apache.catalina.startup.Bootstrap start
Tomcat プロセス モニタ	(B)	(A)	<使用している java のパス> …(略)… com.cosminexus.appruntime.tomcat.monitor.ProcessMonitor …(略)… org.apache.catalina.startup.Bootstrap start

プロセス	PID	親 PID	コマンドライン
Tomcat サーバプ ロセス	(C)	(B)	<使用している java のパス※> …(略)… org.apache.catalina.startup.Bootstrap start

注※

uCosminexus Application Runtime with Java for Apache Tomcat を使用している場合は「/opt/Cosminexus/jdk17/bin/java」, 「/opt/Cosminexus/jdk/bin/java」または「/opt/Cosminexus/jdk/jre/bin/java」です。

## 4.3 設定を変更する

本製品の設定変更は、次に示す config.properties（本製品の設定ファイル）で実施します。

```
/opt/hitachi/ucart/conf/config.properties
```

ファイルの編集には管理者権限が必要です。管理者権限を持たないユーザが設定変更する場合は、次の手順を実施してください。

### 操作手順

1. [/opt/hitachi/ucart/template/config.properties] を任意の場所にコピーする。
2. 手順 1 でコピーしたファイルを編集する。
3. 環境変数 PROCESS\_MONITOR\_CONFIG\_PATH に、手順 1 のコピー先のファイルパスを設定する。

#### ❗ 重要

環境変数 PROCESS\_MONITOR\_CONFIG\_PATH が未設定の場合、デフォルトのファイル [/opt/hitachi/ucart/conf/config.properties] で動作します。環境変数 PROCESS\_MONITOR\_CONFIG\_PATH が必ず設定されるようにするため、setenv.sh（Tomcat 起動時の環境変数定義ファイル）への記載を推奨します。

config.properties（本製品の設定ファイル）の詳細は、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

Tomcat がすでに起動している状態で定義内容を変更した場合、変更した定義内容を反映するためには、Tomcat プロセスモニタを一度停止し、再起動する必要があります。

## 4.4 オートスケーリング環境で運用する

クラウドサービスから提供されるオートスケーリング機能（Amazon EC2 Auto Scaling など）を使用して、オートスケーリング環境下のインスタンスで本製品と Tomcat を使用する場合について説明します。オートスケーリング環境で運用するには、スナップショットログの出力先をインスタンスの外部にあるストレージ（インスタンスのスケールインとともに削除されないストレージ）に切り替えてください。これは、異常停止時によるスケールイン時に必要な保守資料を永続化しておくために必要です。

スナップショットログの出力先をインスタンスの外部にあるストレージに切り替える手順を次に示します。システムを起動する前に実施してください。

### 操作手順

#### 1. スナップショットログの出力先となる外部ストレージを NFS でマウントする。

本製品と Tomcat を使用するマシン上の任意のディレクトリに対し、Tomcat 実行ユーザの権限で読み書き可能な状態で、外部ストレージを NFS でマウントします。これ以降、マウントポイントのディレクトリを<スナップショットログ出力先ディレクトリ>と表記します。

#### 2. config.properties（本製品の設定ファイル）を編集してスナップショットログの出力先を変更する。

config.properties に対し、次のプロパティを追加してください。

```
snapshot.log.filepath=<スナップショットログ出力先ディレクトリ>/${INSTANCEID}/snapshot
```

上記の「INSTANCEID」の部分は例です。このあとの手順 3. で設定する環境変数名と一致していれば任意の名称を使用できます。

#### 3. 環境変数 INSTANCEID を設定する。

\${CATALINA\_BASE}/bin/setenv.sh（Tomcat 起動時の環境変数定義ファイル）のスクリプト上で、クラウドサービスから提供されている手段を使用して、オートスケーリング環境下の個々のインスタンスを一意に識別できる ID を取得し、環境変数 INSTANCEID に設定します。

インスタンスを一意に識別できる ID の取得方法は、各クラウドサービスのドキュメントを参照してください。

「INSTANCEID」という環境変数名は例です。snapshot.log.filepath プロパティに使用した環境変数名と一致していれば任意の名称を使用できます。

#### 4. システムを起動する。

システムの起動方法は、「[4.1.1 本製品を組み込んだ Tomcat の起動方法](#)」を参照してください。

これで設定は完了です。

### 設定の確認方法

設定が正しく反映されているかどうかは、Tomcat プロセスモニタを起動した直後に、「<スナップショットログ出力先ディレクトリ>/\${INSTANCEID}」が指す外部ストレージ上のディレクトリが作成されているかどうかで確認できます。

## 4.5 保守資料を収集する

本製品を使用している場合、スナップショットログ収集機能によって、サポートサービスに提供する必要がある保守資料が収集されます。自動で保守情報が収集される場合（自動収集）と手動で保守情報を収集する場合（手動収集）があります。それぞれについて説明します。

### 自動収集

次の契機で保守情報が収集されます。

- Tomcat プロセスモニタによって、Tomcat サーバプロセスの異常が検知されたとき  
Tomcat プロセスモニタの稼働監視機能によって、Tomcat サーバプロセスのプロセスダウン、スローダウン、およびハングアップなどの状態異常を検知した場合に自動的に収集されます。スナップショットログが自動で収集されるタイミングの詳細については、「[8.5 自動でスナップショットログが収集されるタイミング](#)」を参照してください。  
収集されたログは、config.properties（本製品の設定ファイル）の snapshot.log.filepath プロパティに指定したパスに出力されます。config.properties（本製品の設定ファイル）の詳細は、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

### 手動収集

次のどちらかの操作で保守情報を収集します。

- スナップショットログ収集コマンド（collect-snapshot.sh）の実行  
Tomcat を実行しているマシンのコンソールに直接アクセスすることが可能な場合は、次のコマンドを実行して任意のタイミングで収集できます。

```
$ sudo /opt/hitachi/ucart/bin/collect-snapshot.sh <コマンド引数>
```

コマンドの詳細は、「[16.2 スナップショットログ収集コマンド](#)」を参照してください。

- スナップショットログ収集 REST API の実行  
スナップショットログ収集 REST API に対して HTTP で GET リクエストを発行すると、レスポンスボディとしてスナップショットログを収集できます。  
REST API の詳細は、「[17.2 スナップショットログ収集 REST API](#)」を参照してください。  
ただし、リモートマシンから REST API を受け付けられるようにするには、config.properties（本製品の設定ファイル）の monitor.rest.bindaddress プロパティに「接続先の IP アドレス」または「0.0.0.0」を指定する必要があります。

上記の手動収集の方法は、障害時以外でも使用します。Tomcat サーバプロセスの通常稼働時に保守情報を収集する場合や、スナップショットログの出力テストを実施する場合などです。

スナップショットログ収集機能の詳細については、「[11. スナップショットログ収集機能](#)」を参照してください。

## 4.6 サポートサービスへ問い合わせる

---

サポートサービスへ問い合わせる際は、「[4.5 保守資料を収集する](#)」を参照して、スナップショットログを収集し、サポートサービス窓口に送付してください。スナップショットログの送付がない場合、追加の資料の収集が必要になったり、より多くの調査時間が掛かったりすることがあります。

スナップショットログの中に含まれる情報や設定項目については、「[11. スナップショットログ収集機能](#)」を参照してください。



## 4.7 修正パッチを適用する

修正パッチを適用する方法について説明します。

ここで説明する修正パッチの適用手順は、次の環境を前提としています。

- Tomcat および Tomcat プロセスモニタを停止済み（または一度も起動していない）
- 修正パッチの適用対象バージョンである本製品をインストール済み

構築時に「付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ」の内容に従ってインストールを実施した場合、「付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ」の内容に従って修正パッチを適用してください。

### 操作手順

#### 1. 修正パッチのアーカイブ (PACK\_TAR.Z) を取得する。

ソフトウェアサポートサービスの Web サイト、または本製品に同梱された修正パッチ CD から、修正パッチのアーカイブ (PACK\_TAR.Z) を取得し、修正パッチ適用対象のマシン上に配置します。

#### 2. 修正パッチのアーカイブを任意のディレクトリに展開する。

次のコマンドを実行し、修正パッチのアーカイブを任意のディレクトリに展開します。

```
$ cd <PACK_TAR.Zを配置したディレクトリ>
$ uncompress ./PACK_TAR.Z
$ tar -xf ./PACK_TAR
```

#### 3. UPDATE コマンドを実行して修正パッチを適用する。

次のコマンドを管理者権限で実行し、修正パッチを適用します。

```
$ sudo ./UPDATE -f
```

#### 4. インストールディレクトリとファイルの所有グループを Tomcat 実行ユーザに合わせて変更する。

root 以外のユーザで Tomcat を実行するために、構築時に本製品のインストール先ディレクトリとその配下のサブディレクトリおよびファイルの所有グループを Tomcat 実行ユーザが属するグループに変更していた場合は、修正パッチ適用後にも再度変更してください (Tomcat 実行ユーザに書き込み権限を与えないようにするため、所有ユーザは root のまま変更しないでください)。

Tomcat 実行ユーザが属するグループ名が「tomcat」の場合の実行例を次に示します。

```
$ sudo chgrp -R tomcat /opt/hitachi/ucart
```

これで修正パッチの適用は完了です。

## 4.8 アンセットアップする

---

本製品のアンセットアップ方法について説明します。

ここで説明するアンセットアップ手順は、次に示す条件を満たしていることを前提としています。

- Tomcat および Tomcat プロセスモニタを停止している
- 「[3.3 Tomcat に組み込む](#)」のセットアップ手順に従って、次の3つの定義ファイルについて、編集前の状態でバックアップを作成している
  - `${CATALINA_BASE}/conf/catalina.properties` (Tomcat のプロパティ定義ファイル)
  - `${CATALINA_BASE}/conf/server.xml` (Tomcat のサーバ設定ファイル)
  - `${CATALINA_BASE}/conf/context.xml` (Tomcat のコンテキスト設定ファイル)
- 次のファイルがセットアップ前にすでに存在していた場合は、バックアップを作成している
  - `${CATALINA_HOME}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)
  - `${CATALINA_BASE}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)

この章では、環境構築の自動化を想定して、GUI などによるオペレータの操作が不要なアンセットアップ方法について説明します。GUI を使ったオペレータの操作によってアンセットアップしたい場合は、「[付録 B GUI を使用したアンセットアップ](#)」を参照してください。

構築時に「[付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ](#)」の内容に従ってインストールを実施した場合、「[付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ](#)」の内容に従ってアンセットアップを実施してください。

### 操作手順

#### 1. Tomcat の定義ファイルを編集前の状態に戻す。

次の3つの定義ファイルについて、セットアップ時に取得したバックアップを使用して、編集前の状態に戻します。

- `${CATALINA_BASE}/conf/catalina.properties` (Tomcat のプロパティ定義ファイル)
- `${CATALINA_BASE}/conf/server.xml` (Tomcat のサーバ設定ファイル)
- `${CATALINA_BASE}/conf/context.xml` (Tomcat のコンテキスト設定ファイル)

#### 2. setenv.sh (Tomcat 起動時の環境変数定義ファイル) を編集前の状態に戻す。

次のファイルがセットアップ前にすでに存在していた場合は、バックアップを使用して、編集前の状態に戻します。

- `${CATALINA_HOME}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)
- `${CATALINA_BASE}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)

セットアップ前には存在しなかった場合は、setenv.sh (Tomcat 起動時の環境変数定義ファイル) 自体を削除してください。

### 3. PP インストーラを使用して本製品をアンインストールする。

次のコマンドを管理者権限で実行します。

```
$ sudo /etc/hitachi_x64setup -f -u -k <形名>
```

<形名>はインストールする製品エディションによって異なります。

形名が P-9W43-9N11 の場合は、次のとおりコマンドを実行してください。

```
$ sudo /etc/hitachi_x64setup -f -u -k P-9W43-9N11
```

これでアンセットアップは完了です。

# 5

## 設計

この章では、コンテナ仮想化環境で本製品を使用する場合に考慮する必要がある環境設計ポイントについて説明します。

## 5.1 コンテナ仮想化環境共通の設計ポイント

---

ここでは、コンテナ仮想化環境共通の設計ポイントについて説明します。設計のポイントは、次のとおりです。

- 本製品による性能影響を確認する
- Tomcat プロセスモニタの HTTP 機能に対するセキュリティを確認する
- 本製品によるリソース消費量を確認する
- スナップショットログの出力先を不揮発なストレージに設定する

それぞれについて説明します。

### 5.1.1 本製品による性能影響を確認する

本製品では保守性を向上させるために、Tomcat サーバプロセスに対するトレース情報を拡充しています。また、Tomcat サーバプロセスに対する稼働監視を強化しています。

そのため、Tomcat の起動および停止性能や、アプリケーションに対するリクエスト処理性能に若干の影響を及ぼします。

本製品を使用する場合は、次に示す性能影響があります。これらの性能影響を考慮して性能設計をしてください。

#### (1) リクエスト処理性能

本製品のトレース機能では、リクエスト処理とは別スレッドでログ出力をします。これによって、リクエスト処理性能への影響を極小化しています。ただし、トレース情報として出力する値を取得するために、リクエスト処理スレッド上で実施する処理があるので、若干の性能影響が生じます。アプリケーションの内部保留時間に関係なく、リクエストごとに処理時間が一定であるため、内部保留時間が長い場合は無視できる範囲の影響です。ただし、内部保留時間が短い場合は無視できない性能影響となるおそれがあります。本製品を適用後の環境で性能評価をすることを強く推奨します。

#### (2) 起動性能

Tomcat サーバプロセスを起動させる前に Tomcat プロセスモニタの起動処理が割り込むこととなります。そのため、アプリケーションの規模や個数に関係なく、起動に掛かる時間が数秒増加します。

この起動時間の増分は、コンテナを起動してからアプリケーションが稼働状態になるまでの時間に影響します。

### (3) 停止性能

Tomcat サーバプロセスを停止させる前に、スナップショットログ収集機能で Tomcat 稼働中情報が必ず取得されます。停止処理に失敗する可能性を考慮し、結果的に正常停止であっても取得されます。そのため、Tomcat 稼働中情報の取得処理によって停止時間が増加します。Tomcat 稼働中情報の取得については、「11. スナップショットログ収集機能」を参照してください。

なお、異常停止だけでなく、正常停止でもスナップショットログを取得する設定にしている場合は、スナップショットログ収集処理時間の分、さらに停止時間が増加します。スナップショットログ収集に必要な処理時間は、収集対象のデータ量に比例して増加します。

この停止時間の増分は、システム全体の稼働時間には影響しませんが、リクエストを閉塞させてからコンテナが停止するまでの時間には影響します。

#### 5.1.2 Tomcat プロセスモニタの HTTP 機能に対するセキュリティを確認する

本製品では、スナップショットログの手動取得および Tomcat サーバプロセスとのプロセス間通信のために、HTTP リクエストを受け付ける HTTP 機能を使用されます。システムの運用自動化の妨げになることがあるため、この HTTP 機能にはユーザ認証機能や通信暗号化機能を設けていません。

デフォルトの設定では、HTTP 機能の受付ポートに接続できるクライアントはループバックアドレスからだけに設定されているため、コンテナ外部からの接続はできません。

スナップショットログの手動収集などのユーザ公開 REST API をコンテナ外部から実行する場合は、HTTP 機能の受付ポートへ外部から接続できるように設定を変更することを推奨します。そのため、HTTP 機能の受付ポートに対しては、必ずファイアウォールや NAT/NAPT を適切に設定して、意図しない外部のユーザが不正に HTTP 機能の受付ポートに接続できないように通信をブロックしてください。

#### 5.1.3 本製品によるリソース消費量を確認する

本製品を適用した Tomcat サーバプロセスでは、リクエスト処理スレッドとは別のスレッドで稼働監視やトレースログ出力を行うため、本製品を適用していない状態と比べてスレッド数が「2 スレッド」分増加します。

Tomcat サーバプロセスが生成する最大スレッド数については、Tomcat のドキュメントを参照して算出してください。本製品を使用する場合は、その最大スレッド数に対して「2」を加算して見積もってください。

また、本製品では、Tomcat サーバプロセスと同時に Tomcat プロセスモニタのプロセスが起動されるため、OS の仮想メモリの消費量が約 2,400 MB 分増加します。仮想メモリの消費量を見積もる際には、Tomcat サーバプロセスが消費する仮想メモリに対して、2,400 MB を加算して見積もってください。

Tomcat サーバプロセスが消費する仮想メモリについては、使用する JavaVM のドキュメントを参照して算出してください。日立 JavaVM を使用している場合は、マニュアル「uCosminexus Application Runtime - Cosminexus Developer's Kit for Java 機能解説・リファレンス」を参照してください。

#### 5.1.4 スナップショットログの出力先を不揮発なストレージに設定する

Tomcat をオートスケーリング構成のコンテナ上で使用している場合、コンテナ内だけに保存していたログファイルや環境情報はスケールインとともに削除されるため、障害発生を契機にスケールインされたときにその障害要因の特定が困難になります。

本製品では、稼働監視機能とスナップショットログ収集機能によって、障害発生が検知されるとすぐに障害解析に必要なログファイルや環境情報が自動収集されてスナップショットログが生成されます。そのため、このスナップショットログの出力先をコンテナ外の永続化ストレージにマウントされたディレクトリに設定して、スケールイン後もスナップショットログを参照できるようにしてください。詳細は、「[6.3 Dockerfile を作成する](#)」を参照してください。

## 5.2 Kubernetes などのコンテナオーケストレーションツール特有の設計ポイント

ここでは、Kubernetes などのコンテナオーケストレーションツール特有の設計ポイントについて説明します。設計のポイントは、次のとおりです。

- コンテナの閉塞を高速化する
- スケールイン開始前にスナップショットログ出力処理時間分の猶予を設ける

それぞれについて説明します。

### 5.2.1 コンテナの閉塞を高速化する

Kubernetes などのオーケストレーションツールによって管理されたコンテナ上で Tomcat を使用している場合、本製品の稼働監視機能によって障害が検知された瞬間に、コンテナの切り離し（閉塞）をします。これによって、Tomcat サーバプロセスへのヘルスチェック機能を使用する場合に比べて、障害発生から閉塞までの不稼働時間を短縮できます。

稼働監視機能のユーザコマンドによる情報通知を使用することで、障害の検知を契機に任意のスクリプトを実行できます。そのスクリプト内で、特定のファイルを書き出すなど、Readiness Probe に検知させることができる処理を実装してください。

Readiness Probe の定義方法については、Kubernetes のドキュメントを参照してください。

障害検知時に特定の空ファイルを書き出すユーザスクリプトと、その空ファイルの有無を 1 秒間隔で監視する\*Readiness Probe の定義例を次に示します。

注※

「test ! -e /tmp/tomcat-failure-detected」というコマンドの成否で空ファイルの有無を判定します。

ユーザスクリプトの例

```
#!/bin/bash
touch /tmp/tomcat-failure-detected
```

Kubernetes のマニフェスト定義の例

```
readinessProbe:
  exec:
    command:
      - /bin/sh
      - -c
      - test ! -e /tmp/tomcat-failure-detected
  initialDelaySeconds: 1
  periodSeconds: 1
```



## ❗ 重要

ここに示した定義は例であり、これをそのまま使用した場合の動作は保証しません。必ず Kubernetes が提供する最新のドキュメントを参照して適切なスクリプトを記述してください。

### 5.2.2 スケールイン開始前にスナップショットログ出力処理時間分の猶予を設ける

Kubernetes などのオーケストレーションツールによって管理されたコンテナ上で Tomcat を使用している場合、本製品がスナップショットログの収集処理中は、できる限りコンテナが破棄されないように設計する必要があります。なお、本製品の Tomcat プロセスモニタが停止するまでは、Liveness Probe が正常に判定されます。そのため、稼働監視機能による異常検知を契機にスナップショットログが出力されるケースについては、考慮する必要はありません。

Kubernetes からのスケールイン操作によってコンテナが停止される場合は、停止シグナルを送出してから強制停止に遷移するまでの猶予時間を十分な長さに設定してください。Tomcat の正常停止に失敗したときに、スナップショットログの出力が完全に終わるまで、コンテナが破棄されないようにするためです。

詳細は、「[7.2 システムを停止する](#)」または「[7.4 Kubernetes 環境で運用する](#)」を参照してください。

# 6

## 構築

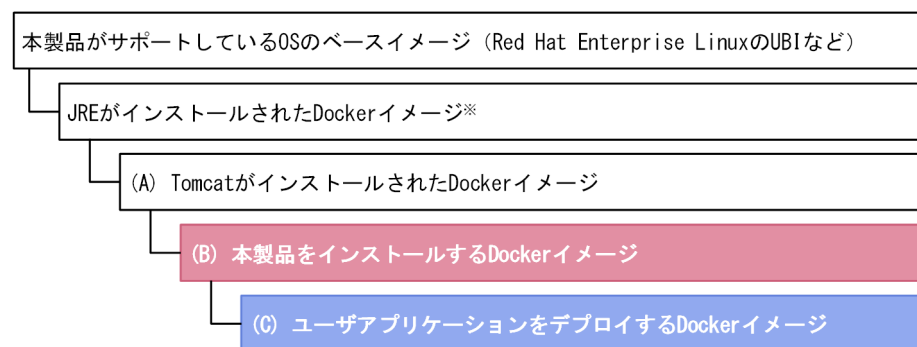
この章では、コンテナ仮想化環境の場合の本製品の構築手順を説明します。なお、Tomcat がインストールされた Docker イメージを作成済みであることを前提としています。

## 6.1 セットアップの前提条件を確認する

ここで説明するセットアップ手順は、本製品の前提 OS をベースイメージとして、Tomcat がインストールされた Docker イメージを作成済みであることを前提としています。Tomcat から提供されている公式の Docker イメージも利用できます。

このセットアップ手順で作成、変更する Docker イメージは、次の図に示す階層になることを想定しています。

図 6-1 作成、変更する Docker イメージ



(凡例)

- : この手順で作成するDockerイメージ
- : この手順で変更するDockerイメージ

### 注※

uCosminexus Application Runtime with Java for Apache Tomcat を使用する場合は、他社製の Java Runtime Environment (JRE) は使用しません。そのため、JRE がインストールされた Docker イメージがなくても問題はありません。

図中の「(A) Tomcat がインストールされた Docker イメージ」は、次の状態となっていることを前提としています。なお、Tomcat から提供されている公式の Docker イメージは、これらの条件を満たしています。

- Tomcat をインストールしている
- Tomcat を起動していない
- 環境変数 CATALINA\_HOME が Tomcat のインストール先の絶対パスを指すように設定している
- 環境変数 CATALINA\_HOME の値にシンボリックリンクが含まれる場合、シンボリックリンクの後に「..」が含まれていない
- 環境変数 CATALINA\_BASE を定義していない
- 環境変数 PATH に、Tomcat のインストール先直下の bin ディレクトリを指す絶対パスが含まれている
- バージョンに関係なく、本製品をインストールしていない

本製品を使用する場合は、さらに次の前提条件も含まれます。なお、Tomcat から提供されている公式の Docker イメージを使用する場合、次の条件を満たしているかどうかは、使用する JRE によって異なります。Tomcat から提供されている公式の Docker イメージの配布サイトでご確認ください。

- Tomcat がインストールされた Docker イメージに、Java SE 8 以降、Java SE 11 以降、または Java SE 17 以降に準拠する Java Runtime Environment (JRE) がインストール済みであること
- 上記の JRE のインストールディレクトリの絶対パスが、環境変数 JAVA\_HOME または JRE\_HOME にセット済みであること (「`{JAVA_HOME}/bin/java`」または「`{JRE_HOME}/bin/java`」が存在すること)

「6.2 インストーラを準備する」および「6.3 Dockerfile を作成する」の手順は、本製品のアーカイブファイルを用いて実施することもできます。アーカイブファイルを用いることによるメリットおよび手順については、「付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ」を参照してください。

## 6.2 インストーラを準備する

---

Docker イメージのビルド時に必要となる本製品のインストーラを準備します。

操作手順を次に示します。

### 操作手順

#### 1. インストール CD-ROM のデータを、 Docker イメージをビルドするマシンにコピーする。

本製品のインストール CD-ROM をマウントし、 インストール CD-ROM に格納されている X64LIN ディレクトリを、 Docker イメージをビルドするマシン上の任意のディレクトリに X64LIN ディレクトリごとコピーしてください。

これ以降、 X64LIN ディレクトリのコピー先を<インストーラのパス>と表記します。

#### 2. インストーラを TAR ファイル形式にアーカイブする。

tar コマンドを実行して X64LIN ディレクトリのコピー先を TAR ファイル形式にアーカイブします。以降、作成したアーカイブファイルのパスを<インストーラアーカイブのパス>と表示します。

tar コマンドの実行例を次に示します。

```
$ tar -cf <インストーラアーカイブのパス> <インストーラのパス>
```

## 6.3 Dockerfile を作成する

「[図 6-1 作成, 変更する Docker イメージ](#)」に示した「(A) Tomcat がインストールされた Docker イメージ」をベースイメージとして、「(B) 本製品をインストールする Docker イメージ」の Dockerfile を作成します。

「(B) 本製品をインストールする Docker イメージ」用の Dockerfile の例を次に示します。下線部分については、使用する環境やベースイメージに合わせて変更してください。

```
# Base image [1.]
FROM tomcat:9.0.60-jre11-openjdk

# Install prerequisite packages [2.]
<前提パッケージのインストール>

# Install uCART [3.]
ADD <インストーラアーカイブのパス> /tmp
RUN chmod +x /tmp/X64LIN/setup
RUN /tmp/X64LIN/setup -f -k <形名> /tmp
RUN rm -rf /tmp/X64LIN

# Overwrite configuration files of Apache Tomcat [4.]
RUN cp /opt/hitachi/ucart/template/tomcat/setenv.sh "${CATALINA_HOME}/bin/"
RUN cp /opt/hitachi/ucart/template/tomcat9/server.xml "${CATALINA_HOME}/conf/"
RUN cp /opt/hitachi/ucart/template/tomcat9/context.xml "${CATALINA_HOME}/conf/"
RUN sed -i -e "s/^(common¥. loader=[^¥r]*¥)(¥r¥?¥)¥/¥1,¥"¥${com.cosminexus.appruntime.lib.p
ath}¥/tomcat¥/common¥/*.jar¥"¥2/g" "${CATALINA_HOME}/conf/catalina.properties"

# Edit configuration file of uCART [5.]
RUN echo monitor.rest.bindaddress=0.0.0.0 >> /opt/hitachi/ucart/conf/config.properties
RUN echo snapshot.log.filepath=/ucart-snapshots/¥${HOSTNAME}/snapshot >> /opt/hitachi/ucart/
conf/config.properties

# Expose HTTP port of uCART Process Monitor [6.]
EXPOSE 28081

# Start Tomcat [7.]
CMD [ "catalina.sh", "run" ]
```

### [説明]

1. ベースとなる「(A) Tomcat がインストールされた Docker イメージ」を指定します。  
Tomcat から提供されている公式の Docker イメージも使用できます。また、ユーザが作成した Docker イメージも使用できます。ユーザが作成した Docker イメージを使用する場合は、必ず環境変数 CATALINA\_HOME が Tomcat のインストール先の絶対パスを指すように設定してください。また、環境変数 CATALINA\_BASE は定義しないでください。
2. 前提パッケージをインストールします。本製品に必要な前提パッケージについては、製品のリリースノートを参照してください。
3. 本製品のインストーラをコンテナ上に展開してインストールを実行します。

4. 本製品の動作に必要な設定を定義します。各定義ファイルの詳細は、「12. 定義ファイル」を参照してください。

この定義例では、本製品の動作に必要な設定があらかじめ記載されている Tomcat の各種定義ファイルを、本製品のテンプレートからコピーして上書きしています。なお、Tomcat 10.1.x を使用する場合は、template ディレクトリ直下の Tomcat 9.x ディレクトリを Tomcat 10.1.x ディレクトリに書き換えてください。

5. config.properties (本製品の設定ファイル) に次の 2 つの設定を追記します。

**monitor.rest.bindaddress=0.0.0.0**

Tomcat プロセスモニタで使用する HTTP 機能の受付ポートを、localhost だけでなくコンテナ外からもアクセスを可能にします。

**snapshot.log.filepath=/ucart-snapshots/\${HOSTNAME}/snapshot**

異常検知時のスナップショットログ出力先を、Docker ホスト上の記憶域にマウントするパスに変更します。親ディレクトリ名に「\${HOSTNAME}」を採用してコンテナごとに異なるディレクトリに出力されるようにすることで、同時刻に複数のコンテナから同じファイル名で出力することを防止できます。

6. 本製品の Tomcat プロセスモニタで使用する HTTP 機能の受付ポートである 28081 番ポートへ、コンテナ外からのアクセスを可能にします。
7. デフォルトの実行コマンドとして、Tomcat の起動コマンドを指定します。「"catalina.sh", "run"」を指定してください。

## 6.4 Docker イメージをビルドする

---

本製品をインストールする Docker イメージ、およびユーザアプリケーションをデプロイする Docker イメージをビルドする手順を説明します。なお、ビルドする前に、次の条件をすべて満たしている必要があります。

- 「[図 6-1 作成, 変更する Docker イメージ](#)」に示した「(A) Tomcat がインストールされた Docker イメージ」のベースイメージが pull 可能であること
- インストーラアーカイブのパスに本製品のインストーラをアーカイブした TAR ファイルが存在すること

### 操作手順

#### 1. [\[6.3 Dockerfile を作成する\]](#) で作成した Docker イメージをビルドする。

docker build コマンドを実行します。

実行例：

```
$ docker build ./ -t tomcat:9.0.60-jre11-openjdk-ucart1.0
```

下線部分のイメージ名とタグは任意です。この例では、イメージ名を「tomcat」、タグ名を「9.0.60-jre11-openjdk-ucart1.0」としています。

#### 2. 「(C) ユーザアプリケーションをデプロイする Docker イメージ」の Dockerfile のイメージ名・タグ名を書き換える。

Dockerfile のベースイメージのイメージ名・タグ名を、手順 1. でビルドした Docker イメージのイメージ名・タグ名に変更します。ここでの例に従った場合は「tomcat:9.0.60-jre11-openjdk-ucart1.0」にします。

#### 3. 手順 2. でイメージ名・タグ名を書き換えた「(C) ユーザアプリケーションをデプロイする Docker イメージ」をビルドする。



# 7

## 運用

この章では、コンテナ仮想化環境の場合のシステムの起動、停止、設定方法、運用方法などについて説明します。

## 7.1 システムを起動する

「6.4 Docker イメージをビルドする」でビルドしたユーザアプリケーションをデプロイした Docker イメージ※に対して `docker run` コマンドを実行します。これによって、Tomcat プロセスモニタがコンテナ上で起動します（本製品が適用された状態）。さらに、Tomcat プロセスモニタが Tomcat サーバプロセスを子プロセスとして起動します。

注※

「[図 6-1 作成, 変更する Docker イメージ](#)」に示した「(C) ユーザアプリケーションをデプロイする Docker イメージ」を指します。

`docker run` コマンドを実行する際、次を指定してください。

- スナップショットログ出力先として指定した「`/ucart-snapshots`」ディレクトリが Data Volume にマウントされるように、「`-v`」オプションで Docker ホスト上に存在する任意のディレクトリを指定する。
- Tomcat のポート番号に加えて、本製品の Tomcat プロセスモニタで使用する HTTP 機能の受付ポートである 28081 番ポートを「`-p`」オプションで公開する。

`docker run` コマンドの指定例を次に示します。

```
$ docker run -p 8080:8080 -p 28081:28081 -v /var/snapshots:/ucart-snapshots <イメージ名>
```

- 下線部の値は任意です。実行環境に合わせて指定してください。
- <イメージ名>には、「(B) 本製品をインストールする Docker イメージ」をベースとしてビルドした「(C) ユーザアプリケーションをデプロイする Docker イメージ」のイメージ名・タグ名を指定してください。

### スナップショットログ出力先の設定が正しく反映されているかどうか確認する方法

設定が正しく反映されているかどうかは、コンテナを起動した直後に、「<スナップショットログ出力先ディレクトリ>/<コンテナごとの環境変数 HOSTNAME の値>」が指す Data Volume 上のディレクトリが作成されているかどうかで確認できます。ここで示した例では、「`/var/snapshots/<コンテナごとの環境変数 HOSTNAME の値>`」というディレクトリが作成されていることを確認します。

## 7.2 システムを停止する

---

起動したコンテナのコンテナ ID を指定して `docker stop` コマンドを実行します。これによって、Tomcat サーバプロセスおよび Tomcat プロセスモニタが停止します。

一定時間内に Tomcat サーバプロセスが正常停止できずに異常停止した場合、スナップショットログの収集および出力をするための猶予時間が必要です。そのため、「-t」オプションで、SIGKILL が送出されるまでの待機時間を延長することを推奨します（デフォルトは 10 秒です）。

`docker stop` コマンドの指定例を次に示します。

```
$ docker stop -t 60 <コンテナID>
```

下線部の SIGKILL 待機秒数は任意です。ここでは例として 60 秒を指定しています。スナップショットの収集と出力を完了させるために必要な時間を指定してください。

## 7.3 設定を変更する

---

本製品の設定変更は、次に示す config.properties (本製品の設定ファイル) で実施します。

```
/opt/hitachi/ucart/conf/config.properties
```

本製品をインストールする Docker イメージ用の Dockerfile の設定項目を追加することで、ほかのプロパティと同じように設定を追加できます。Dockerfile の設定項目の追加については、「[6.3 Dockerfile を作成する](#)」参照してください。config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

## 7.4 Kubernetes 環境で運用する

本製品を組み込んだ Docker イメージを、Kubernetes 環境で運用する場合のマニフェストの定義例を次に示します。下線部分は使用する環境に合わせて変更してください。

```
apiVersion: v1
kind: Service
metadata:
  name: tomcat-ucart
spec:
  selector:
    app: tomcat-ucart
  ports:
    - name: "tomcat-http-port"
      port: 8080
      targetPort: 8080
    - name: "ucart-http-port" [1.]
      port: 28081
      targetPort: 28081
  type: NodePort
---
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat-ucart
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat-ucart
  template:
    metadata:
      labels:
        app: tomcat-ucart
    spec:
      containers:
        - name: tomcat-ucart
          image: <イメージ名> [2.]
          ports:
            - containerPort: 8080
            - containerPort: 28081 [3.]
          volumeMounts: [4.]
            - name: snapshot-volume
              mountPath: /ucart-snapshots
          startupProbe: [5.]
            httpGet:
              path: /path-to-healthcheck
              port: 8080
          readinessProbe: [6.]
            exec:
              command:
                - /bin/sh
                - -c
                - test ! -e /tmp/tomcat-failure-detected
          initialDelaySeconds: 1
          periodSeconds: 1
```

```
terminationGracePeriodSeconds: 60           [7.]
volumes:                                       [4.]
- name: snapshot-volume
  persistentVolumeClaim:
    claimName: ucart-snapshot-pvc
```

#### [説明]

1. コンテナ外に公開するマッピング先ポート番号を指定します。マッピング先ポート番号は、Tomcat プロセスモニタが使用する HTTP 機能の受付ポートのマッピング先ポート番号を指します。
2. 「[6.4 Docker イメージをビルドする](#)」でビルドした、「(C) ユーザアプリケーションをデプロイする Docker イメージ」のイメージ名を指定します。
3. Tomcat プロセスモニタが使用する HTTP 機能の受付ポートである 28081 番ポートを公開ポートとして指定します。
4. スナップショットログの出力先として指定したパスを Persistent Volume にマウントします。マウント先の Persistent Volume と Persistence Volume Claim は、事前に作成するか、またはこのマニフェストファイルに定義を追記してください。
5. Tomcat の起動完了を検知するための Startup Probe を定義します。GET リクエストによってユーザアプリケーションの起動完了を確認できる URL を指定してください。
6. Tomcat の正常稼働を監視するための Readiness Probe を定義します。詳細は、「[5.2.1 コンテナの閉塞を高速化する](#)」を参照してください。
7. コンテナに正常停止シグナルを送出してから、強制停止に遷移するまでの猶予時間を指定します。ここでは例として 60 秒を指定していますが、スナップショットの収集と出力を完了させるのに十分な時間を指定してください。

## 7.5 保守資料を収集する

---

本製品を使用している場合、スナップショットログ収集機能によって、サポートサービスに提供する必要がある保守資料が自動的に収集されます。自動で保守情報が収集される場合（自動収集）と手動で保守情報を収集する場合（手動収集）があります。それぞれについて説明します。

### 自動収集

次の契機で保守情報が収集されます。

- Tomcat プロセスモニタによって Tomcat サーバプロセスの異常が検知されたとき  
Tomcat プロセスモニタの稼働監視機能によって、Tomcat サーバプロセスのプロセスダウン、スローダウン、およびハングアップなどの状態異常を検知した場合に自動的に収集されます。スナップショットログが自動で収集されるタイミングの詳細については、「[8.5 自動でスナップショットログが収集されるタイミング](#)」を参照してください。

収集されたログは、`config.properties`（本製品の設定ファイル）の `snapshot.log.filepath` プロパティに指定したパスに出力されます。`config.properties`（本製品の設定ファイル）の詳細は、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

### 手動収集

次の操作で保守情報を収集します。

- スナップショットログ収集 REST API の実行  
スナップショットログ収集 REST API に対して HTTP で GET リクエストを発行すると、レスポンスボディとしてスナップショットログを収集できます。  
スナップショットログ収集 REST API の詳細は、「[17.2 スナップショットログ収集 REST API](#)」を参照してください。

スナップショットログ収集機能の詳細については、「[11. スナップショットログ収集機能](#)」を参照してください。

## 7.6 サポートサービスへ問い合わせる

---

本製品のサポートサービスへ問い合わせる際は、「[7.5 保守資料を収集する](#)」を参照して、スナップショットログを収集し、サポートサービス窓口に送付してください。スナップショットログの送付がない場合、追加の資料の収集が必要になったり、より多くの調査時間が掛かったりすることがあります。

スナップショットログの中に含まれる情報や設定項目については、「[11. スナップショットログ収集機能](#)」を参照してください。



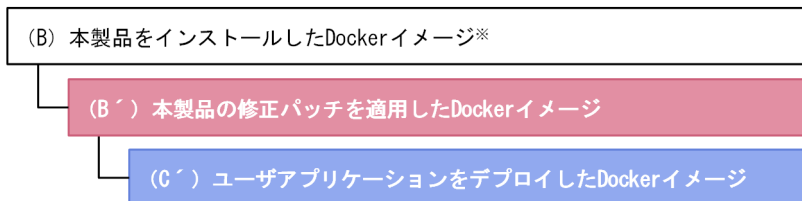
## 7.7 修正パッチを適用した Docker イメージをビルドする

修正パッチを適用した Docker イメージをビルドする場合の前提条件、作成方法、およびビルド方法を説明します。

### 7.7.1 修正パッチ適用前の前提条件を確認する

ここで説明する手順で作成する Docker イメージは、次の図に示す階層になることを想定しています。

図 7-1 作成, 変更する Docker イメージ



(凡例)

- : この手順で作成するDockerイメージ
- : この手順で変更するDockerイメージ

注※

「6.4 Docker イメージをビルドする」で作成した Docker イメージ。

構築時に「付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ」の内容に従って Docker イメージをビルドした場合、「付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ」の内容に従って修正パッチを適用した Docker イメージのビルドを実施してください。

### 7.7.2 修正パッチを適用した Dockerfile を作成する

「6.4 Docker イメージをビルドする」で作成した「(B) 本製品をインストールする Docker イメージ」をベースイメージとして、「(B') 本製品の修正パッチを適用した Docker イメージ」を作成します。

本製品の修正パッチを適用した Docker イメージ用の Dockerfile の例を次に示します。下線部分については、使用する環境やベースイメージに合わせて変更してください。

```
# Base image [1.]  
FROM tomcat:9.0.60-jre11-openjdk-ucart1.0  
  
# Update uCART [2.]  
RUN mkdir /tmp/ucart-patch  
WORKDIR /tmp/ucart-patch  
COPY <修正パッチのアーカイブ(PACK_TAR.Z)のパス> ./
```

```
RUN uncompress <修正パッチのアーカイブ(PACK TAR.Z)のパス> -c | tar xvf -
RUN ./UPDATE -f; [ $? -eq 1 ] && true
WORKDIR /
RUN rm -rf /tmp/ucart-patch
```

#### [説明]

1. ベースとなる「(B) 本製品の修正パッチを適用した Docker イメージ」を指定します。
2. 本製品の修正パッチをコンテナ上に展開して UPDATE プログラムを実行します。  
[-f] オプションを付けて実行してください。なお、UPDATE プログラムの実行結果（戻り値）が”1”であれば、正常にパッチ適用が成功したことを意味します。

### 7.7.3 修正パッチを適用した Docker イメージをビルドする

ここでは、本製品の修正パッチを適用した Docker イメージ、およびユーザアプリケーションをデプロイする Docker イメージをビルドする手順を説明します。なお、ビルドする前に、修正パッチのアーカイブ (PACK\_TAR.Z) のパスにファイルが存在することを確認してください。

#### 操作手順

1. [\[7.7.2 修正パッチを適用した Dockerfile を作成する\]](#) で作成した Docker イメージをビルドする。  
docker build を実行します。

実行例：

```
$ docker build ./ -t tomcat:9.0.60-jre11-openjdk-ucart1.0.1
```

下線部分のイメージ名とタグは任意です。この例では、イメージ名を「tomcat」、タグ名を「9.0.60-jre11-openjdk-ucart1.0.1」としています。

2. [\[6.4 Docker イメージをビルドする\]](#) で作成した「(C) ユーザアプリケーションをデプロイする Docker イメージ」の Dockerfile のイメージ名・タグ名を書き換える。  
Dockerfile のベースイメージのイメージ名・タグ名を、手順 1. でビルドした Docker イメージのイメージ名・タグ名に変更します。ここでの例に従った場合は「tomcat:9.0.60-jre11-openjdk-ucart1.0.1」にします。
3. 手順 2. でイメージ名・タグ名を書き換えた「(C) ユーザアプリケーションをデプロイする Docker イメージ」をビルドする。

## 8

## Tomcat プロセスモニタ機能

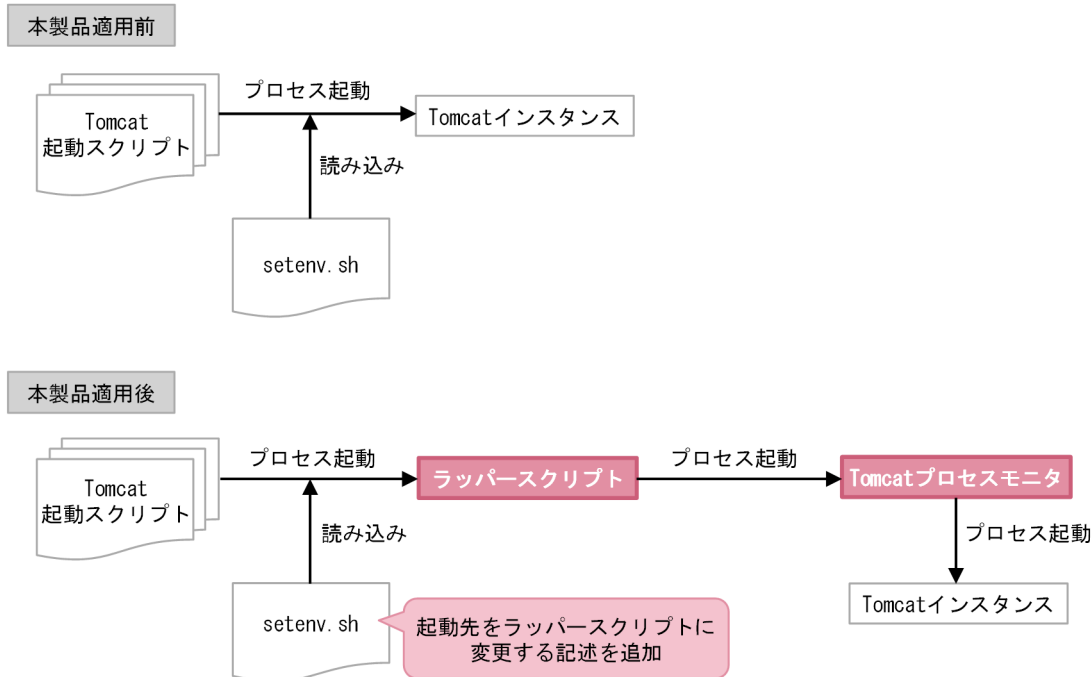
この章では、Tomcat プロセスモニタ機能の概要、適用方法、解除方法、Tomcat プロセスモニタを使用する上で知っておいていただきたいことなどを説明します。

## 8.1 Tomcat プロセスモニタ機能の概要

Tomcat プロセスモニタは、Tomcat サーバプロセスを管理・監視するプロセスです。Tomcat の高信頼化を実現するための、各種機能（トレース機能、稼働監視機能、およびスナップショットログ収集機能）を管理します。

Tomcat プロセスモニタは、ラッパースクリプトを介して、Tomcat が提供する起動オペレーションに従って起動されます。次の図に示すとおり、Tomcat プロセスモニタは Tomcat と同時に起動されるため、通常の運用で Tomcat プロセスモニタを意識する必要はありません。

図 8-1 本製品適用前と適用後の起動オペレーション



## 8.2 Tomcat プロセスモニタ機能の適用方法

---

Tomcat プロセスモニタ機能の適用方法を説明します。Tomcat プロセスモニタ機能を含め、本製品の設定をデフォルトのまま使用する場合は次の手順を実施してください。これによって、Tomcat プロセスモニタが起動されます。

- オンプレミス環境または仮想マシン環境の場合  
「3. 構築」および「4. 運用」
- コンテナ仮想化環境の場合  
「6. 構築」および「7. 運用」

Tomcat プロセスモニタの起動時の設定をデフォルトから変更したい場合は、「12.3 [setenv.sh \(Tomcat 起動時の環境変数定義ファイル\)](#)」を参照して、[setenv.sh \(Tomcat 起動時の環境変数定義ファイル\)](#) の内容を変更してください。

Tomcat プロセスモニタは、「12.2 [config.properties \(本製品の設定ファイル\)](#)」に示すファイルの設定に従い動作します。設定を変更したい場合は、「12.2 [config.properties \(本製品の設定ファイル\)](#)」を参照して変更してください。

## 8.3 Tomcat プロセスモニタ機能の解除方法

---

Tomcat プロセスモニタ機能の設定を解除したい場合は、Tomcat プロセスモニタ機能のために編集した `setenv.sh` (Tomcat 起動時の環境変数定義ファイル) の内容を、編集前の状態に戻してください。

## 8.4 Tomcat プロセスモニタ機能適用後の終了ステータス

Tomcat プロセスモニタ機能適用後の、Tomcat 起動失敗時と停止時の終了ステータスについて説明します。

### Tomcat の起動失敗時の終了ステータス

本製品を組み込んだ Tomcat の起動に失敗した原因とその終了ステータスを次の表に示します。

表 8-1 Tomcat の起動に失敗した原因と終了ステータス

原因	終了ステータス
環境変数 JRE_HOME の値不正	101※
config.properties (本製品の設定ファイル) の I/O エラー	101
config.properties (本製品の設定ファイル) のバリデーションエラー	101
Tomcat プロセスモニタの一時領域へのアクセスエラー	101
セキュリティマネージャが設定されている	101
その他の Tomcat 起動前のエラー	102
稼働監視機能による Tomcat の停止要求発生	110
Tomcat プロセスモニタ終了時点で、Tomcat が停止していない	103
上記以外の原因	Tomcat プロセスモニタ機能を適用していないときの終了ステータスと同じ値

注※ Tomcat のバージョンによっては、Tomcat プロセスモニタ機能を適用していないときの終了ステータスと同じ値になります。

### Tomcat 停止時の終了ステータス

Tomcat の停止時の終了ステータスは、原則、Tomcat プロセスモニタ機能を適用していないときの終了ステータスと同じ値となります。ただし、SIGTERM 以外の受信可能なシグナルを Tomcat プロセスモニタが受信した場合は、SIGTERM シグナルを送信したときと同じ値になります。

## 8.5 自動でスナップショットログが収集されるタイミング

---

デフォルトでは、次に示すどれかの条件が成立した場合に、自動でスナップショットログが収集されます。

- Tomcat サーバプロセス開始前に Tomcat プロセスモニタが終了する場合  
ただし、Tomcat プロセスモニタのログファイルのセットアップが完了する前は収集されません。
- 稼働監視機能が異常を検知した場合
- Tomcat サーバプロセスが、次に示す値以外の終了ステータスで終了した場合
  - 0
  - 143 (SIGTERM シグナルで終了した場合)

ここに示すスナップショットログの収集に関する条件は、`config.properties` (本製品の設定ファイル) で変更できます。稼働監視機能については、「[10.3 稼働監視機能の設定 \(監視項目\)](#)」を参照してください。終了ステータスの条件については、「[12.2.4\(4\) スナップショットログ収集機能に関するプロパティ](#)」の `snapshot.onshutdownrequest.collect.condition` を参照してください。



## 8.6 Tomcat サーバプロセスの強制終了

---

稼働監視機能が検知した障害または異常な設定によって、Tomcat プロセスモニタが停止する場合、Tomcat サーバプロセスが強制終了されます。

Tomcat サーバプロセスの強制終了は、次の 2 段階で実行されます。

1. Tomcat サーバプロセスに対する SIGTERM の送信
2. Tomcat サーバプロセスに対する SIGKILL の送信

1.の Tomcat サーバプロセスが終了したかどうかの確認は、config.properties (本製品の設定ファイル) の「monitor.tomcat.forcestop.timeout」で設定するタイムアウト時間内で実施されます。そのため、Tomcat サーバプロセスが確実に終了するタイムアウト時間を設定する必要があります。

「monitor.tomcat.forcestop.timeout」については、「[12.2.4\(2\) Tomcat プロセスモニタに関するプロパティ](#)」を参照してください。

「monitor.tomcat.forcestop.timeout」で設定するタイムアウト時間内に Tomcat サーバプロセスが終了しなかった場合は、2.が実行されます。

## 8.7 Tomcat プロセスモニタの HTTP 機能

---

Tomcat プロセスモニタは、次の処理を実行するために、HTTP 機能を持っています。

- 運用管理用 REST API の受付処理
- 稼働監視コンポーネントと Tomcat サーバプロセスとのプロセス間 HTTP 通信処理

Tomcat プロセスモニタの HTTP 機能の各設定については、「[12.2.4\(2\) Tomcat プロセスモニタに関するプロパティ](#)」の `monitor.rest` から始まるプロパティを参照してください。また、稼働監視コンポーネント専用の HTTP 通信の設定については、「[10. 稼働監視機能](#)」を参照してください。

## 8.8 Tomcat プロセスモニタの一時領域

---

Tomcat プロセスモニタが利用する一時領域ディレクトリは、次のとおりです。

<Tomcatサーバプロセスの一時領域※1>/<TomcatプロセスモニタのHTTP機能の受付ポート番号※2>

注※1

Tomcat の環境変数 CATALINA\_TMPDIR で変更できます。

注※2

config.properties (本製品の設定ファイル) の monitor.rest.port に指定した値です。

# 9

## トレース機能

この章では、トレース機能の概要、セットアップ方法、アンセットアップ方法、および取得されるトレース情報について説明します。

## 9.1 トレース機能の概要

本製品のトレース機能とは、Tomcat を利用したシステムで、リクエスト実行中の性能影響個所や障害発生個所の特定を手助けする機能です。Tomcat の処理の入口と出口には、多くのトレース取得ポイントが設定されています。これによって、障害発生個所などを特定しやすくなります。また、uCosminexus Application Server と組み合わせて使用する場合、PRF トレース機能と連携できます。

### メモ

Tomcat の内部処理の調査はスレッドダンプなどを活用してください。

トレース機能を使用するための前提条件、トレース取得ポイント、トレースの種類、および uCosminexus Application Server との連携について説明します。

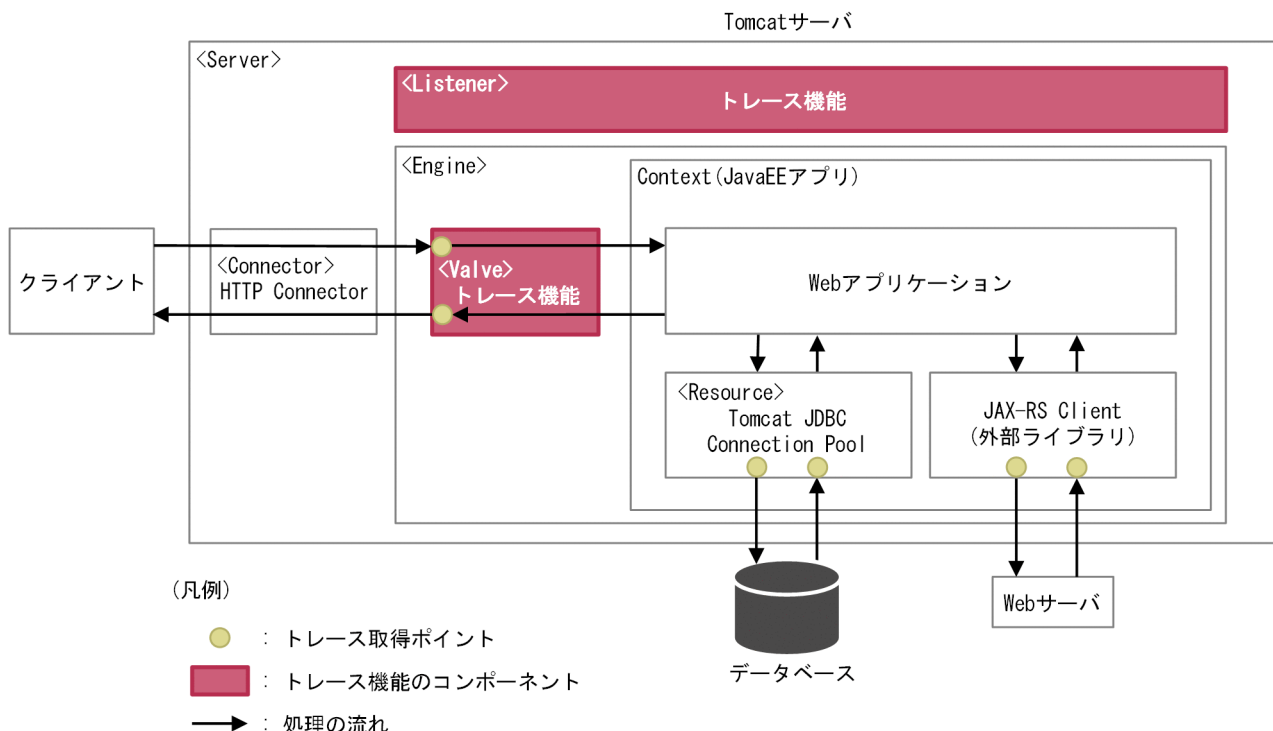
### トレース機能使用時の前提条件

トレース機能を使用する際の前提条件は、ホストアドレスに IPv4 のアドレスが割り当てられていることです。IPv4 アドレスおよび IPv6 アドレスの両方を持つホストで稼働させる場合、`java.net.preferIPv6Addresses` はデフォルト値 (false) のままにします。

### トレース取得ポイント

本製品のトレース取得ポイントを次の図に示します。

図 9-1 本製品のトレース取得ポイント



### トレースの種類

本製品のトレース機能では、次の種類のトレースを取得します。

- サーバの開始時および終了時のトレース

詳細は、「9.4 サーバの開始時および終了時のトレース」を参照してください。

- Web アプリケーションの開始時および終了時のトレース

詳細は、「9.5 Web アプリケーションの開始時および終了時のトレース」を参照してください。

- HTTP リクエストおよびサーブレット・フィルタのトレース

詳細は、「9.6 HTTP リクエストおよびサーブレット・フィルタのトレース」を参照してください。

- HTTP セッションのトレース

詳細は、「9.7 HTTP セッションのトレース」を参照してください。

- JAX-RS クライアントのトレース

詳細は、「9.8 JAX-RS クライアントのトレース」を参照してください。

- Tomcat JDBC Connection Pool のトレース

詳細は、「9.9 Tomcat JDBC Connection Pool のトレース」を参照してください。

### ❗ 重要

- データベースへのアクセスのトレースは、Tomcat JDBC Connection Pool 経由でアクセスした場合に取得します。
- JAX-RS クライアントのトレースは、JAX-RS Client 仕様のサードパーティライブラリ経由でアクセスした場合に取得します。java.net.URLConnection クラスを直接利用したアクセスなど、JavaAPI を直接呼び出した場合にはトレースを取得しません。
- セキュリティマネージャが有効な環境での動作は保証しません。

## uCosminexus Application Server との連携

Tomcat と uCosminexus Application Server を組み合わせてシステムを構築する場合に限り、PRF トレースと連携できます。連携すると、Tomcat から uCosminexus Application Server への HTTP の呼び出しを 1 つの処理単位として、uCosminexus Application Server のパフォーマンストレーサで認識できます。これによって、Tomcat から uCosminexus Application Server までの処理をまとめて調査できます。PRF トレースおよびパフォーマンストレーサについては、uCosminexus Application Server のマニュアルを参照してください。

## 9.2 トレース機能のセットアップ方法

本製品をインストールしたあと、トレース機能を有効にするために次の手順でセットアップをしてください。

### 1. Server タグの子要素に設定を追記する。

`#{CATALINA_BASE}/conf/server.xml` (Tomcat のサーバ設定ファイル) に記載されている Server タグの子要素に、次の内容を追記してください。Server タグの子要素のほかのタグよりも前に記載することを推奨します。

```
<Listener className="com.cosminexus.appruntime.tomcat.tracer.ServerComponentHandler" />
```

### 2. Engine タグの子要素に設定を追記する。

`#{CATALINA_BASE}/conf/server.xml` (Tomcat のサーバ設定ファイル) に記載されている Engine タグの子要素に、次の内容を追記してください。Engine タグの子要素のほかのタグよりも前に記載することを推奨します。

```
<Valve className="com.cosminexus.appruntime.tomcat.tracer.RequestTraceValve" />
```

### 3. Resource タグの jdbcInterceptors 属性に設定を追記する。

次のファイルに記載されている Resource タグの `jdbcInterceptors` 属性に `com.cosminexus.appruntime.tomcat.tracer.JdbcTraceHandler` を追記してください。ただし、Resource タグの `factory` 属性が `org.apache.tomcat.jdbc.pool.DataSourceFactory` の場合に限りません。

- `#{CATALINA_BASE}/conf/server.xml` (Tomcat のサーバ設定ファイル)
- `#{CATALINA_BASE}/conf/context.xml` (Tomcat のコンテキスト設定ファイル)

なお、設定を追記しない場合は、Tomcat JDBC Connection Pool のトレースを取得できません。設定を追記する例を次に示します。太字が追記する箇所です。

```
<Resource
  . . . 省略 . . .
  factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
  . . . 省略 . . .
  jdbcInterceptors="com.cosminexus.appruntime.tomcat.tracer.JdbcTraceHandler"
  . . . 省略 . . .
/>
```

### 4. Context タグの子要素に設定を追記する。

`#{CATALINA_BASE}/conf/context.xml` (Tomcat のコンテキスト設定ファイル) に記載されている Context タグの子要素に、次の内容を追記してください。Context タグの子要素のほかタグとの記載順序は問いません。

```
<Loader loaderClass="com.cosminexus.appruntime.tomcat.tracer.WebappTraceClassLoader" />
```

### 5. Eclipse Jersey 以外の JAX-RS クライアントを使用する場合、ユーザプログラムを変更する。

Eclipse Jersey 以外の JAX-RS クライアントを使用する場合は、ユーザプログラム中の Client オブジェクトに JaxrsClientTraceFeature を登録する必要があります。登録しない場合、JAX-RS クライアントのトレースを取得できません。登録する例を次に示します。

```
import javax.ws.rs.client.Client;
import com.cosminexus.appruntime.common.tracer.JaxrsClientTraceFeature;

Client client = <Clientオブジェクトの取得>;
client.register(JaxrsClientTraceFeature.class); // JaxrsClientTraceFeatureの登録
```

なお、Eclipse Jersey 2.x 系を使用する場合は、ユーザプログラムの変更は不要です。この手順を実施する必要はありません。

## 6. 必要に応じて config.properties (本製品の設定ファイル) のプロパティを設定する。

config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。



## 9.3 トレース機能のアンセットアップ方法

---

トレース機能が不要になった場合、トレース機能のセットアップ時に追記および変更した設定を元に戻してください。

## 9.4 サーバの開始時および終了時のトレース

次の場合に、サーバに関するトレース情報が取得されます。

- Tomcat サーバコンポーネントの初期化処理が開始したとき
- Tomcat サーバコンポーネントの終了処理が完了したとき

サーバの開始時および終了時のトレースの一覧を次の表に示します。

表 9-1 サーバの開始時および終了時のトレースの一覧

イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
0xe200	サーバが起動するとき	INFO	0	-	-
0xe201	サーバが終了するとき	INFO	0	-	-

(凡例)

-: 出力なし

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

## 9.5 Web アプリケーションの開始時および終了時のトレース

次の場合に、Web アプリケーションに関するトレース情報が取得されます。

- Web アプリケーション (WAR 単位) の初期化処理が開始したとき
- Web アプリケーション (WAR 単位) の終了処理が完了したとき

Web アプリケーションの開始時および終了時のトレースの一覧を次の表に示します。

表 9-2 Web アプリケーションの開始時および終了時のトレースの一覧

イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
0xe210	Web アプリケーションが開始するとき	INFO	0	-	Web アプリケーションのコンテキストパス
0xe211	Web アプリケーションが停止するとき	INFO	0	-	Web アプリケーションのコンテキストパス

(凡例)

-: 出力なし

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

## 9.6 HTTP リクエストおよびサーブレット・フィルタのトレース

HTTP リクエストおよびサーブレット・フィルタのトレース情報が取得されます。HTTP リクエストおよびサーブレット・フィルタのトレースの一覧を次の表に示します。

表 9-3 HTTP リクエストおよびサーブレット・フィルタのトレースの一覧

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 1	0xe230	HTTP リクエストを受信した直後	FINE	0	HTTP メソッド	リクエスト URI <sup>※3</sup>
トレース 2	0xe231	HTTP レスポンスを送信する直前	FINE	<ul style="list-style-type: none"><li>正常リターンの場合：0</li><li>例外リターンの場合：1</li></ul>	HTTP メソッド	リクエスト URI <sup>※3</sup>
トレース 3	0xe240	最初のサーブレットまたはフィルタを呼び出す直前 <sup>※1</sup>	FINER	0	サーブレット・フィルタ名 <sup>※2</sup>	リクエスト URI <sup>※3</sup>
トレース 4	0xe241	最後のサーブレットの処理、または最初のフィルタの処理が完了した直後 <sup>※1</sup>	FINER	<ul style="list-style-type: none"><li>正常リターンの場合：0</li><li>例外リターンの場合：1</li></ul>	サーブレット・フィルタ名 <sup>※2</sup>	リクエスト URI <sup>※3</sup>

### 注※1

1 つのリクエストで複数のフィルタやサーブレットが呼び出される場合、最初に呼び出されるものだけがトレース取得の対象です。

### 注※2

Web アプリケーションデプロイ記述子に記載している場合は記載した名前が取得されます。記載していない場合は実装依存の名前が取得されます。

### 注※3

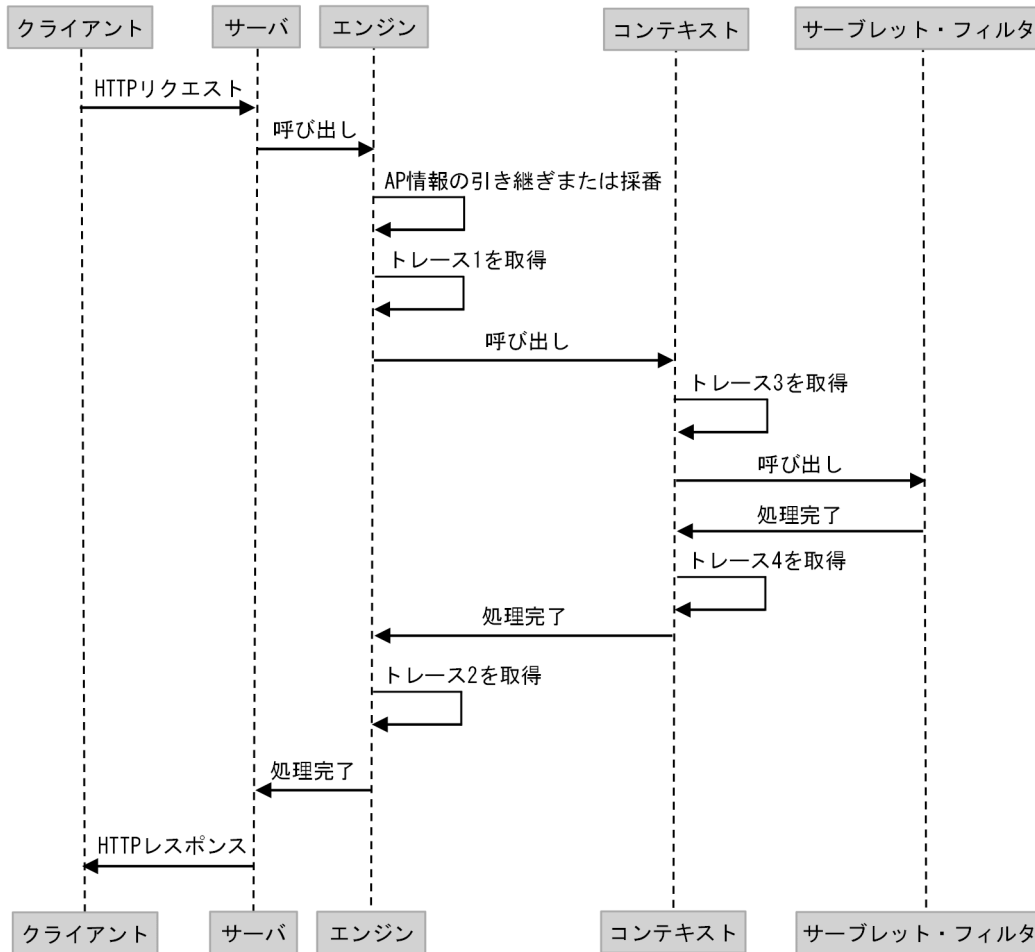
URI にパスワードなどの機密情報が含まれていないことを確認してください。

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

### トレース取得シーケンス

HTTP リクエストおよびサーブレット・フィルタのトレース取得シーケンスを次の図に示します。なお、図中の番号（トレース 1 など）は、「[表 9-3 HTTP リクエストおよびサーブレット・フィルタのトレースの一覧](#)」と対応しています。

図 9-2 HTTP リクエストおよびサーブレット・フィルタのトレース取得シーケンス (同期リクエスト)

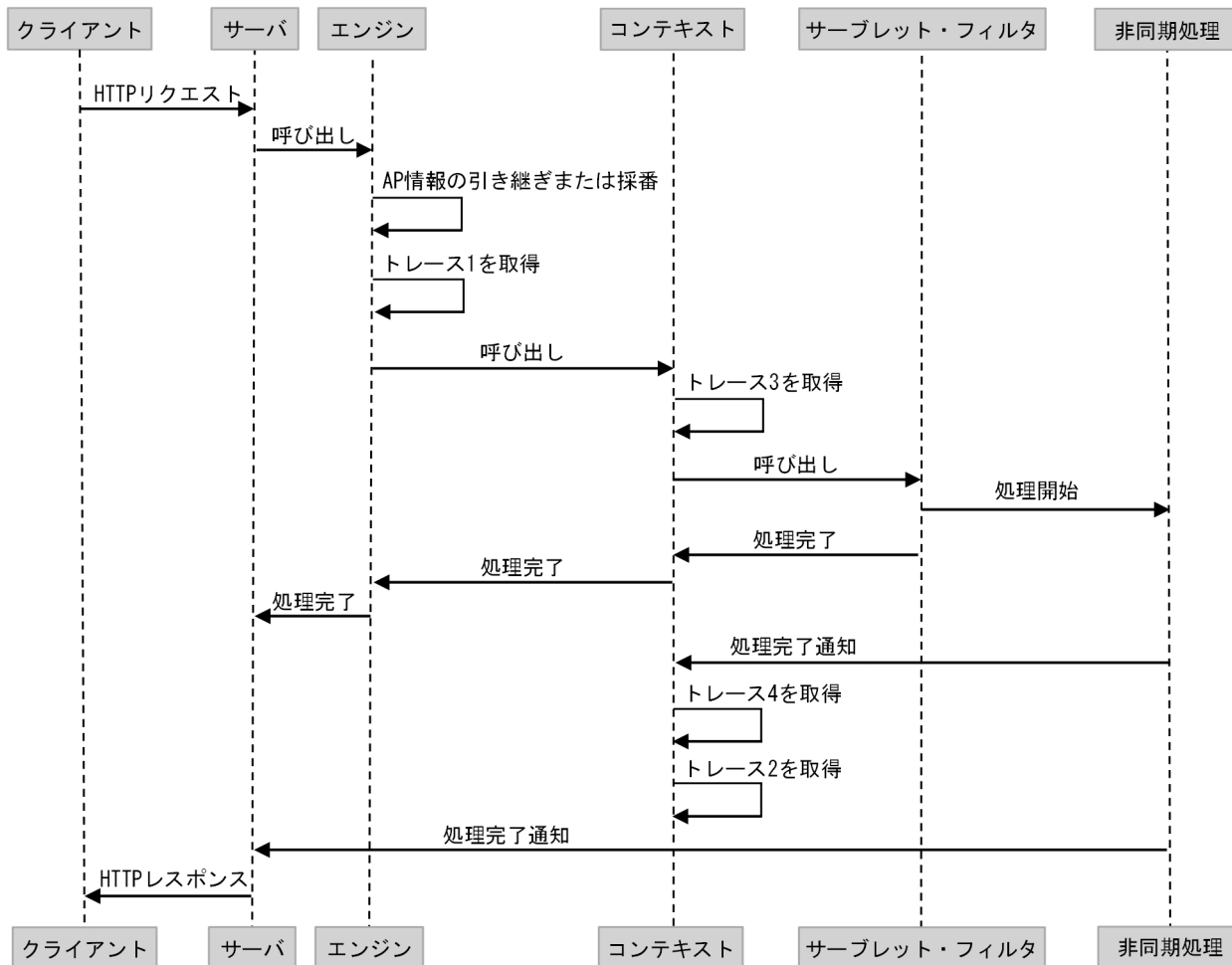


(凡例)

→ : 処理の流れ

AP情報 : アプリケーション情報

図 9-3 HTTP リクエストおよびサーブレット・フィルタのトレース取得シーケンス (非同期リクエスト)



(凡例)

→ : 処理の流れ

AP情報 : アプリケーション情報

## 9.7 HTTP セッションのトレース

---

HTTP セッションのトレース情報が取得されます。HTTP セッションのトレースの一覧を次の表に示します。

表 9-4 HTTP セッションのトレースの一覧

イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
0xe250	HTTP セッションが作成された直後	FINE	0	-	セッション ID
0xe251	HTTP セッションが無効にされる直前	FINE	0	-	セッション ID

(凡例)

-: 出力なし

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

## 9.8 JAX-RS クライアントのトレース

JAX-RS クライアントのトレース情報が取得されます。Eclipse Jersey 2.x 系を使用する場合は、自動的にトレース情報が取得されます。一方で、Eclipse Jersey 2.x 系以外の JAX-RS クライアントを使用する場合は、ユーザプログラムを変更する必要があります。ユーザプログラムの変更については、「[9.2 トレース機能のセットアップ方法](#)」を参照してください。

JAX-RS クライアントのトレースの一覧を次の表に示します。

表 9-5 JAX-RS クライアントのトレースの一覧

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 1	0xe260	JAX-RS クライアントで HTTP リクエストが発行される前	FINE	0	HTTP メソッド	エンドポイント URI <sup>※2</sup>
トレース 2	0xe261	JAX-RS クライアントで HTTP リクエストが発行され、HTTP レスポンスを受け取ったあと <sup>※1</sup>	FINE	0	HTTP メソッド	エンドポイント URI <sup>※2</sup>

注※1

HTTP レスポンスを受け取らなかった場合は、トレースを取得しません。

注※2

URI にパスワードなどの機密情報が含まれていないことを確認してください。

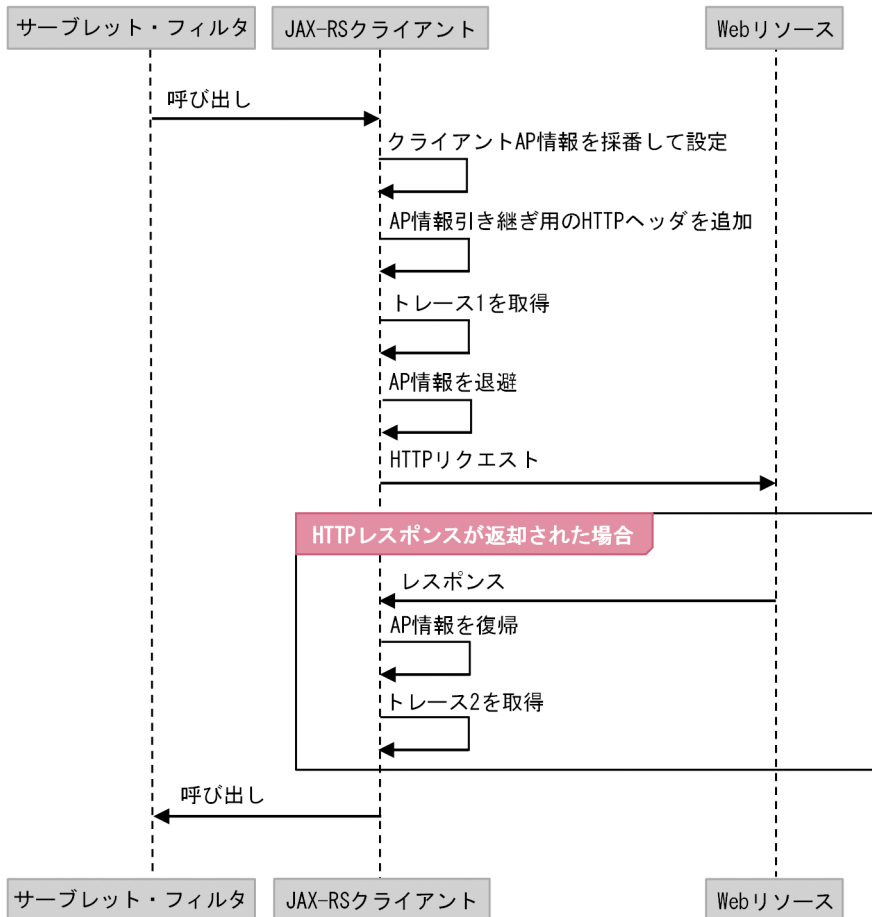
トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

### トレース取得シーケンス

JAX-RS クライアントのトレース取得シーケンスを次の図に示します。なお、図中の番号（トレース 1 など）は、「[表 9-5 JAX-RS クライアントのトレースの一覧](#)」と対応しています。



図 9-4 JAX-RS クライアントのトレース取得シーケンス



(凡例)

→ : 処理の流れ

AP情報 : アプリケーション情報

## 9.9 Tomcat JDBC Connection Pool のトレース

Tomcat JDBC Connection Pool に関しては、次のトレース情報が取得されます。

- javax.sql.DataSource インタフェース
- java.sql.Connection インタフェース
- java.sql.Statement インタフェース
- java.sql.PreparedStatement インタフェース
- java.sql.CallableStatement インタフェース

それぞれのトレースについて、説明します。

### 9.9.1 javax.sql.DataSource インタフェースのトレース

javax.sql.DataSource インタフェースのトレースの一覧を次の表に示します。

表 9-6 javax.sql.DataSource インタフェースのトレースの一覧

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 1	0xe270	javax.sql.DataSource .getConnection メソッドの呼び出しでコネクション取得に成功した直後	FINE	0	-	コネクション ID <sup>※</sup>

(凡例)

- : 出力なし

注※

物理コネクションオブジェクトのハッシュコードを 16 進数で表記した文字列を示します。

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

### 9.9.2 java.sql.Connection インタフェースのトレース

java.sql.Connection インタフェースのトレースの一覧を次の表に示します。

表 9-7 java.sql.Connection インタフェースのトレースの一覧

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 2	0xe271	java.sql.Connection.close メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	コネクション ID
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe272	java.sql.Connection.close メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe273	java.sql.Connection.commit メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	コネクション ID
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe274	java.sql.Connection.commit メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe275	java.sql.Connection.rollback メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	コネクション ID
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe276	java.sql.Connection.rollback メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe277	java.sql.Connection.createStatement メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	コネクション ID
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe278	java.sql.Connection.createStatement メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 2	0xe279	java.sql.Connection.prepareCall メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	コネクション ID
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe27a	java.sql.Connection.prepareCall メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe27b	java.sql.Connection.prepareStatement メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	コネクション ID
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe27c	java.sql.Connection.prepareStatement メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

### 9.9.3 java.sql.Statement インタフェースのトレース

java.sql.Statement インタフェースのトレースの一覧を次の表に示します。

表 9-8 java.sql.Statement インタフェースのトレースの一覧

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 2	0xe280	java.sql.Statement.execute メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe281	java.sql.Statement.execute メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 2	0xe282	java.sql.Statement.executeBatch メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe283	java.sql.Statement.executeBatch メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe284	java.sql.Statement.executeQuery メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe285	java.sql.Statement.executeQuery メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe286	java.sql.Statement.executeUpdate メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe287	java.sql.Statement.executeUpdate メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>

(凡例)

-: 出力なし

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

## 9.9.4 java.sql.PreparedStatement インタフェースのトレース

java.sql.PreparedStatement インタフェースのトレースの一覧を次の表に示します。

表 9-9 java.sql.PreparedStatement インタフェースのトレースの一覧

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 2	0xe290	java.sql.PreparedStatement.execute メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe291	java.sql.PreparedStatement.execute メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe292	java.sql.PreparedStatement.executeBatch メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe293	java.sql.PreparedStatement.executeBatch メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe294	java.sql.PreparedStatement.executeQuery メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe295	java.sql.PreparedStatement.executeQuery メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターン：空文字 ("")</li> <li>例外リターン：例外クラス名</li> </ul>
トレース 2	0xe296	java.sql.PreparedStatement.executeUpdate メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe297	java.sql.PreparedStatement.executeUpdate メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>

(凡例)

-：出力なし

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「13.3.1 Tomcat トレースログ」を参照してください。

## 9.9.5 java.sql.CallableStatement インタフェースのトレース

java.sql.CallableStatement インタフェースのトレースの一覧を次の表に示します。

表 9-10 java.sql.CallableStatement インタフェースのトレースの一覧

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
トレース 2	0xe2a0	java.sql.CallableStatement.execute メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe2a1	java.sql.CallableStatement.execute メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe2a2	java.sql.CallableStatement.executeBatch メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe2a3	java.sql.CallableStatement.executeBatch メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe2a4	java.sql.CallableStatement.executeQuery メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe2a5	java.sql.CallableStatement.executeQuery メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターンの場合：例外クラス名</li> </ul>
トレース 2	0xe2a6	java.sql.CallableStatement.executeUpdate メソッドを呼び出す前	FINE	0	メソッド引数のシグネチャ	-

シーケンス図との対応	イベント ID	トレース取得ポイント	取得レベル	リターンコード (Rc)	インタフェース名 (INT)	オペレーション情報 (OPR)
<ul style="list-style-type: none"> <li>正常リターンの場合：トレース 3</li> <li>例外リターンの場合：トレース 4</li> </ul>	0xe2a7	java.sql.CallableStatement.executeUpdate メソッドからリターンするとき	FINE	<ul style="list-style-type: none"> <li>正常リターンの場合：0</li> <li>例外リターンの場合：1</li> </ul>	メソッド引数のシグネチャ	<ul style="list-style-type: none"> <li>正常リターンの場合：空文字 ("")</li> <li>例外リターン：例外クラス名</li> </ul>

(凡例)

-：出力なし

トレース機能で取得された情報は、Tomcat トレースログに出力されます。詳細は、「[13.3.1 Tomcat トレースログ](#)」を参照してください。

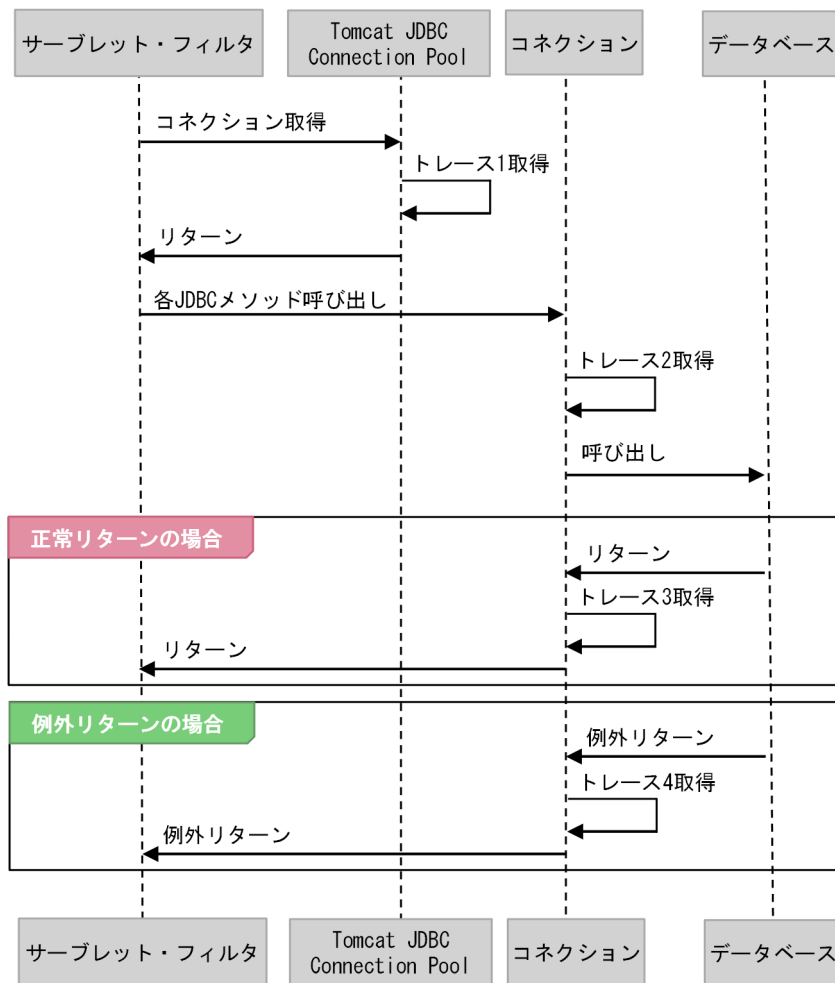
## 9.9.6 Tomcat JDBC Connection Pool のトレース取得シーケンス

Tomcat JDBC Connection Pool のトレース取得シーケンスを次の図に示します。なお、図中の番号（トレース 1 など）は、次の表と対応しています。

- 表 9-6 javax.sql.DataSource インタフェースのトレースの一覧
- 表 9-7 java.sql.Connection インタフェースのトレースの一覧
- 表 9-8 java.sql.Statement インタフェースのトレースの一覧
- 表 9-9 java.sql.PreparedStatement インタフェースのトレースの一覧
- 表 9-10 java.sql.CallableStatement インタフェースのトレースの一覧



図 9-5 Tomcat JDBC Connection Pool のトレース取得シーケンス



## 9.10 ユーザ作成スレッドおよび非同期処理 API 利用上の注意事項

---

「9.6 HTTP リクエストおよびサーブレット・フィルタのトレース」で示した、アプリケーション情報の引き継ぎまたは採番をしたスレッド外で、トレース情報を取得した場合の注意事項を示します。

- ユーザ作成スレッド上でトレースを取得する場合  
スレッド作成元にアプリケーション情報が設定されている場合、アプリケーション情報を引き継いで取得します。スレッド作成時点のアプリケーション情報を引き継ぐため、リクエスト処理が完了済みでも、スレッド作成時点のリクエストに対応するアプリケーション情報を取得します。
- `javax.servlet.AsyncContext` の `start` メソッドに指定している、`java.lang.Runnable` を実装したオブジェクトの `run` メソッド上でトレースを取得する場合  
アプリケーション情報を引き継ぎません。クライアントアプリケーション情報およびルートアプリケーション情報を情報なし (0.0.0.0, 0, 0x0000000000000000) として扱います。

なお、「HTTP リクエストおよびサーブレット・フィルタのトレース (0xe230~0xe241)」については、非同期リクエストモードに変更した (`javax.servlet.ServletRequest.startAsync` メソッドを呼び出した) 場合も正しいアプリケーション情報を取得します。

# 10

## 稼働監視機能

この章では、稼働監視機能の概要と、必要な設定について説明します。

## 10.1 稼働監視機能の概要

---

Tomcat 自身にリクエストを試行する Pull 型のヘルスチェックでは、Tomcat 自身がスローダウンやハングアップを起こしていると、障害発生の検知が遅れてしまうことがあります。本製品の稼働監視機能ではヘルスチェックを代行するほか、Push 型での障害通知ができます。この障害通知によって、エラーレートの低減や、素早い障害対策が図れるようになります。

稼働監視機能では、各監視項目が障害発生時などに発行するイベントを監視することで、次の問題を把握できます。

- 本製品に関する設定の誤り
- Tomcat サーバプロセスの起動失敗
- Tomcat サーバプロセスのハングアップ・スローダウン

また、イベント検知時に次の処理を実行させることもできます。

- スナップショットログ収集処理
- Tomcat サーバプロセス停止処理のトリガー
- ユーザコマンド実行（事前に定義されている場合）

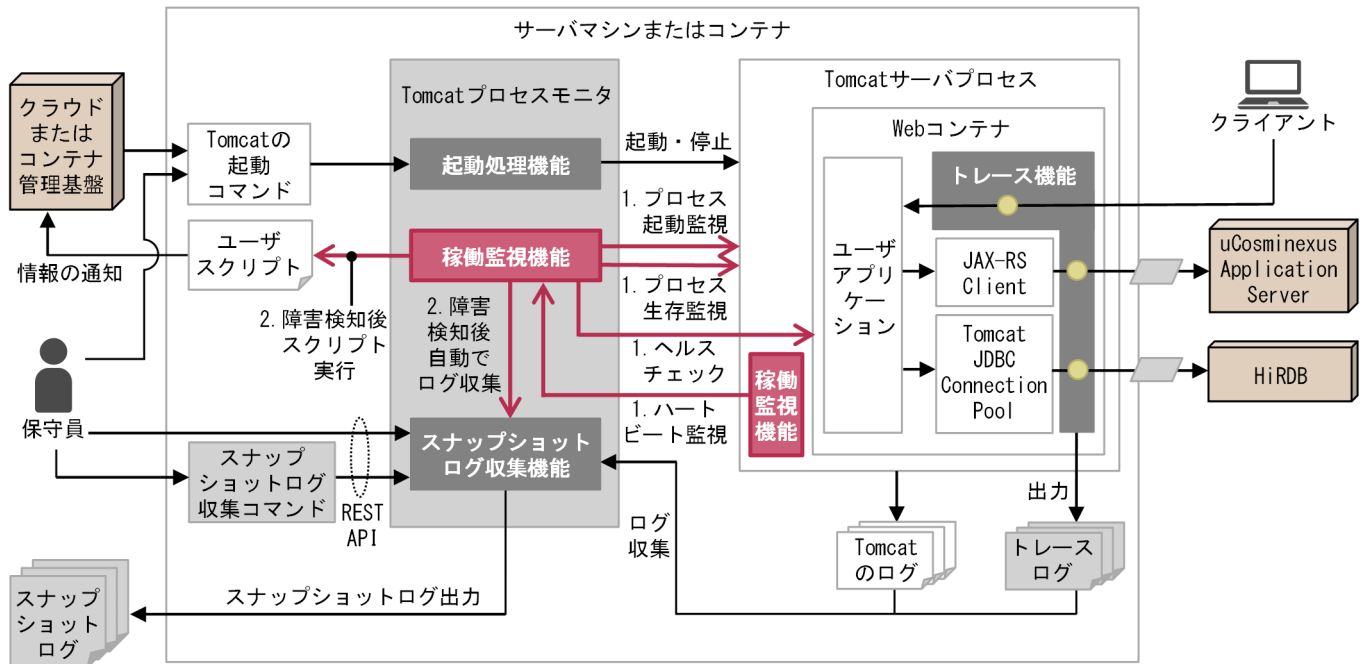
この節では、稼働監視機能の位置づけ、稼働監視機能の概要（監視項目、検知したイベントに対するアクション）、および稼働監視機能の検知内容の出力（イベントプロパティ）を説明します。

### 10.1.1 稼働監視機能の位置づけ

稼働監視機能は、Tomcat プロセスモニタが起動する稼働監視コンポーネントとして動作します。そして、Tomcat プロセスモニタが起動する Tomcat サーバプロセスを稼働監視します。

稼働監視機能の位置づけを次の図に示します。

図 10-1 稼働監視機能の位置づけ



(凡例)

- : 本製品の稼働監視機能
- : 本製品の機能 (稼働監視機能以外)
- : 本製品のコンポーネント、ログおよびコマンド
- : 本製品のトレース取得ポイント
- : Cosminexus Performance Tracer用アプリケーション情報
- : 稼働監視機能で実行する処理
- : そのほかの処理

図中の番号について次に説明します。

1. 稼働監視機能は、Tomcat サーバプロセスと、Tomcat プロセスモニタ上で起動する稼働監視コンポーネントとの間で通信を実行して、各種イベントを検知します。  
これによって、ヘルスチェックやハートビート監視などの機能を実現します。
2. 稼働監視機能では、イベントを検知した際に対応する動作（アクション）を定義できます。アクションとして、「クラウドまたはコンテナ管理基盤に情報を通知する処理」を実行するユーザコマンドを定義すれば、クラウドまたはコンテナ管理基盤に情報を通知できます。稼働監視機能で検知したイベントはログに出力されます。出力されたログは、スナップショットログとして収集されます。

## 10.1.2 稼働監視機能の概要

稼働監視機能が持つ監視項目の種類について説明します。また、監視によってイベントを検知した場合に実行させるアクションの種類についても説明します。

## (1) 稼働監視機能の監視項目

稼働監視機能の監視項目の種類を次に示します。設定方法などの詳細については、該当する項を参照してください。

監視項目	概要	設定方法の説明 箇所
プロセスの起動監視	Tomcat サーバプロセスの起動が成功したかどうかを監視します。 Tomcat サーバプロセス起動失敗時のスナップショットログ収集によって、失敗要因の解析・特定を容易にする目的があります。	10.3.1
ハートビート監視	Tomcat サーバプロセスのハングアップを監視します。 Tomcat サーバプロセス側からの定期的なハートビート送信によって、Tomcat サーバプロセスへの HTTP リクエストによるヘルスチェックよりも迅速に障害検知し、素早い障害対策を図る目的があります。	10.3.2
プロセス生存監視	Tomcat サーバプロセスの生存状況を監視します。 Tomcat サーバプロセスが異常終了した場合、スナップショットログが収集される前にユーザコマンドを実行することで、素早い障害対策を図る目的があります。	10.3.3
ヘルスチェック	Tomcat の HTTP リクエスト受付のヘルスチェックをします。 障害検知時のスナップショットログ収集によって、障害要因の解析・特定を容易にする目的があります。	10.3.4
リクエスト処理の停滞監視	Tomcat の HTTP リクエスト処理のスローダウン・ハングアップを監視します。 Tomcat サーバプロセス側からの定期的なリクエスト処理スレッド情報の送信によって、リクエスト処理のスローダウン・ハングアップを迅速に検知し、素早い障害対策を図る目的があります。	10.3.5

## (2) 稼働監視で検知したイベントに対するアクション

イベントを検知した場合の動作は、監視項目ごとの設定によって異なります。

稼働監視でイベントを検知すると、ログを出力し、定義されたアクションを実行します。定義できるアクションには次の種類があり、複数選択することもできます。

- スナップショットログ収集
- Tomcat サーバプロセス停止
- ユーザコマンドの実行

障害を検知した場合、あらかじめ指定したコマンドを実行させることで、情報通知ができます。ユーザコマンドの設定については、「[10.4 稼働監視機能の設定 \(ユーザコマンドの実行\)](#)」を参照してください。

ユーザコマンドを実行する場合は、ほかのアクションはユーザコマンドの実行終了後に実行されます。

### 10.1.3 稼働監視機能の検知内容の出力（イベントプロパティ）

稼働監視コンポーネントが各機能の監視時に検知した内容は、イベントプロパティとしてJSON形式で出力されます。

イベントプロパティは、メッセージログのKDLR20225-Iメッセージに出力されます。文字コードはUTF-8です。

「10.3 稼働監視機能の設定（監視項目）」で記述するイベントプロパティの、記述形式について次に説明します。

- この章ではイベントプロパティを複数行で記載しますが、本製品のログの出力として使う場合は、1行となります。
- <>で説明を付記している場合は、出力される文字列の一例を示しています。
- timestamp 属性については、出力される日付時刻のパターン文字列で表しています。
- succeeded 属性が false の例しかない項目の成功イベントについては、形式は同一で、succeeded 属性が true となります。

## 10.2 稼働監視機能の設定 (基本項目)

稼働監視機能は次の 2 つによって動作します。

- 本製品の稼働監視コンポーネント  
Tomcat プロセスモニタ機能を適用することで起動します。  
Tomcat サーバプロセスから稼働監視コンポーネントへの通信に関する設定は、Tomcat プロセスモニタで変更できます。
- 稼働監視用ライフサイクルリスナー  
Tomcat サーバプロセスに設定します。

この節では、これらの動作に関する設定について記載します。

### 10.2.1 Tomcat プロセスモニタ側の設定

稼働監視コンポーネントは、Tomcat プロセスモニタを起動することで開始します。

Tomcat プロセスモニタの適用と起動については、「[8.2 Tomcat プロセスモニタ機能の適用方法](#)」を参照してください。

なお、Tomcat サーバプロセスから稼働監視コンポーネントへの HTTP 通信に関して、次に示すタイムアウト時間を変更できます。変更は `config.properties` (本製品の設定ファイル) で実施します。

- 接続タイムアウト時間
- 読み込みタイムアウト時間

該当するプロパティと設定例を次に示します。この例ではデフォルト値を記載しています。

```
healthcheck.rest.connecttimeout=3000  
healthcheck.rest.readtimeout=10000
```

`config.properties` (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

これらのタイムアウトは、Tomcat プロセスモニタの HTTP 機能の受付ポートへの通信障害や、稼働監視コンポーネントでの処理遅延などによって、発生する場合があります。そのため、Tomcat プロセスモニタと Tomcat サーバプロセスの通信環境を考慮して、適切な時間を設定してください。デフォルト値は、一般的な環境を想定したタイムアウト時間が設定されています。

#### ヒント

タイムアウト時間の設定に関する注意点を次に示します。



- タイムアウト時間を長く設定すると、Tomcat プロセスモニタ自体の障害の検知が遅れることがあります。
- タイムアウト時間を短く設定し過ぎると、Tomcat サーバプロセス側からの本来の障害通知の前に、Tomcat プロセスモニタ側の障害と誤検知することがあります。

## 10.2.2 Tomcat サーバプロセスへの稼働監視用ライフサイクルリスナーの設定

稼働監視機能では、稼働監視用にライフサイクルリスナーを提供します。

稼働監視用ライフサイクルリスナーは、次の個所に設定します。

項目	説明
設定対象のファイル	server.xml (Tomcat のサーバ設定ファイル) ※1
設定する要素	com.cosminexus.appruntime.tomcat.healthcheck.MonitoringListener※2

### 注※1

server.xml (Tomcat のサーバ設定ファイル) の記述方法については、Tomcat のドキュメントを参照してください。  
本製品を使用するため server.xml (Tomcat のサーバ設定ファイル) に追加が必要な要素については、「[12.5 server.xml \(Tomcat のサーバ設定ファイル\)](#)」を参照してください。

### 注※2

Server コンポーネントに設定します。設定個所は Listener 要素の className 属性です。

設定する要素について次に説明します。

### com.cosminexus.appruntime.tomcat.healthcheck.MonitoringListener

Tomcat サーバプロセス起動時に、稼働監視に必要な情報を稼働監視コンポーネントに送信します。また、停滞検出バルブで検出したリクエスト停滞情報を稼働監視コンポーネントに送信します。

この Listener は Server 要素にだけネストすることができます。この Listener 要素に className 属性以外の属性はありません。

リクエスト停滞情報および停滞検出バルブについては、「[10.3.5 リクエスト処理の停滞監視](#)」を参照してください。

設定例を次に示します。

```
<Server ...>
  ...
  <Listener className="com.cosminexus.appruntime.tomcat.healthcheck.MonitoringListener"
  ... />
  ...
</Server>
```

## 10.3 稼働監視機能の設定 (監視項目)

---

監視項目ごとに、検知できる内容と、設定できる内容を説明します。

### 10.3.1 プロセスの起動監視

プロセスの起動監視は、Tomcat サーバプロセスの起動が成功したかどうかを監視します。

Tomcat サーバプロセスの起動時に、次の理由で稼働監視が継続できないと判断した場合、Tomcat サーバプロセス起動時のエラーとして検知します。

- Tomcat サーバプロセス初期化完了通知待ちタイムアウトが発生した場合
- スナップショットで、Tomcat サーバプロセスが出力するログを収集する設定ができていない場合
- Tomcat サーバプロセス開始完了通知待ちタイムアウトが発生した場合

これらのエラー発生時の動作と、エラーを検知するための設定について次に説明します。

#### (1) Tomcat サーバプロセス初期化完了通知待ちタイムアウト

Tomcat サーバプロセス初期化完了通知待ちタイムアウトとは、Tomcat プロセスモニタ起動後、Tomcat サーバプロセス起動前の稼働監視コンポーネント開始時から、Tomcat サーバプロセス初期化完了通知を受信するまでにタイムアウトが発生することを指します。

この場合、ライフサイクルリスナーが正常に登録されていないため、Tomcat サーバプロセス側からの通知ができないと判断されます。この際、メッセージ KDLR20208-E を出力します。エラーを検知した場合は、「10.2.2 Tomcat サーバプロセスへの稼働監視用ライフサイクルリスナーの設定」の設定を見直してください。

#### (a) 障害検知時の動作

Tomcat サーバプロセス初期化完了通知待ちタイムアウトが発生した場合、障害検知後のアクションとして、次のように動作します。

1. スナップショットログを収集します。
2. Tomcat サーバプロセスが起動済みであれば Tomcat サーバプロセスを停止します。

これ以外のアクション (ユーザコマンドの指定を含む) を定義することはできません。

#### (b) 設定できる内容

Tomcat サーバプロセス初期化完了通知タイムアウト時間を変更することで、タイムアウト時間を設定できます。

config.properties (本製品の設定ファイル) に、稼働監視コンポーネントを開始してからのタイムアウト時間をミリ秒単位で指定します。0 を指定した場合はタイムアウトしません。

該当するプロパティと設定例を次に示します。この例ではデフォルト値を記載しています。

```
healthcheck.initdelay.timeout=60000
```

config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

## (2) Tomcat サーバプロセスが出力するログをスナップショットログで収集する設定の妥当性

Tomcat サーバプロセス起動前の稼働監視コンポーネント開始時に、本製品は common.base<sup>※</sup>と snapshot.include.paths<sup>※</sup>で指定されたディレクトリから更新チェック用ログファイルを探し、ファイルの状態を確認します。更新チェック用ログファイルの本製品のデフォルトのファイル名は catalina.\*.log です。ファイルにアクセスできない場合は、メッセージ KDLR20206-E を出力します。

注※

common.base、および snapshot.include.paths は、config.properties (本製品の設定ファイル) のプロパティです。common.base については「[12.2.4\(1\) 本製品全体に関するプロパティ](#)」を、snapshot.include.paths については「[12.2.4\(4\) スナップショットログ収集機能に関するプロパティ](#)」を参照してください。

Tomcat サーバプロセス初期化完了通知を受信した時に、更新チェック用ログファイルの内容に変更が発生していなかった場合、スナップショットで正常に Tomcat ログ (Tomcat サーバプロセスが出力するログ) を収集する設定ができていないと判断し、エラーが発生します。この際、メッセージ KDLR20207-E を出力します。エラーを検知した場合は、「[11. スナップショットログ収集機能](#)」に示す設定を見直してください。

### (a) 障害検知時の動作

スナップショットでログ収集が正しく設定されていない場合、障害検知後のアクションとして、次のように動作します。

1. スナップショットログを収集します。
2. Tomcat サーバプロセスが起動済みであれば Tomcat サーバプロセスを停止します。

これ以外のアクション (ユーザコマンドの指定を含む) を定義することはできません。

### (b) 設定できる内容

更新チェックに使用するログファイルを config.properties (本製品の設定ファイル) に glob 形式で設定することで、エラーを検知できます。

該当するプロパティの設定形式を次に示します。ログファイル名パターンのデフォルト値は「catalina.\*.log」です。

```
healthcheck.unchangedlogfile.logfilename.glob=<ログファイル名パターン>
```

config.properties（本製品の設定ファイル）については、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

### (3) Tomcat サーバプロセス開始完了通知待ちタイムアウト

Tomcat サーバプロセス開始完了通知待ちタイムアウトとは、Tomcat サーバプロセス初期化完了通知を受信してから、Tomcat サーバプロセス開始完了通知を受信するまでにタイムアウトが発生することを指します。この場合、Tomcat サーバプロセスの起動に失敗したと判断されます。この際、メッセージ KDLR20209-E を出力します。

Tomcat サーバプロセス開始完了通知待ちタイムアウトが発生すると、次に示す JSON 形式のイベントプロパティが出力されます。

```
{
  "type": "startdelay",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": false
}
```

#### (a) 障害検知時の動作

Tomcat サーバプロセス開始完了通知待ちタイムアウトが発生した場合、障害検知後のアクションとして、次のように動作します。

1. スナップショットログを収集します。
2. Tomcat サーバプロセス開始完了通知待ちをリトライします。
3. 最大リトライ回数分のタイムアウトが発生した場合は、スナップショットログを収集して Tomcat サーバプロセスを停止します。

デフォルトの設定では、1 分経過するたびにスナップショットログを収集し、5 分経過で Tomcat サーバプロセスを停止します。設定の変更方法については、「[\(b\) 設定できる内容](#)」を参照してください。

#### (b) 設定できる内容

Tomcat サーバプロセス開始完了通知待ちタイムアウトに関して設定できる内容を次に示します。

- Tomcat サーバプロセス開始完了通知待ちタイムアウト時間の変更  
Tomcat サーバプロセス開始完了通知待ちタイムアウト時間を変更することで、タイムアウトまでの時間を調整できます。

config.properties (本製品の設定ファイル) に、Tomcat サーバプロセス初期化完了通知を受信してからのタイムアウト時間をミリ秒単位で指定します。0 を指定した場合はタイムアウトしません。該当するプロパティと設定例を次に示します。この例ではデフォルト値を記載しています。

```
healthcheck.startdelay.timeout=60000
```

config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

- 障害検知時の動作の変更

障害検知時の動作のうち、次の項目を config.properties (本製品の設定ファイル) で変更できます。

- ユーザコマンドの実行  
ユーザコマンドを定義する場合は、別途「[10.4 稼働監視機能の設定 \(ユーザコマンドの実行\)](#)」の定義が必要です。
- 最大リトライ回数
- スナップショットログの収集有無
- Tomcat サーバプロセスの停止有無

該当するプロパティと設定例を次に示します。この例では、「[10.4 稼働監視機能の設定 \(ユーザコマンドの実行\)](#)」で設定する「ユーザコマンド定義の ID」として、「exec1」を設定しています。

```
healthcheck.startdelay.actions.failure.usercommand.idrefs.1=exec1
healthcheck.startdelay.actions.failure.retrymax=4
healthcheck.startdelay.actions.failure.snapshot=true
healthcheck.startdelay.actions.afterretry.terminate=true
```

config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

なお、healthcheck.startdelay.actions.afterretry.terminate を false に変更した場合は、最大リトライ回数分のタイムアウトが発生したあとでも、次の状態となります。

- Tomcat サーバプロセスが停止しない
- Tomcat プロセスモニタが動作中

そのため、「Tomcat サーバプロセス開始完了」の状態にならないことが Tomcat サーバプロセスが出力するログなどから分かっている場合には、明示的に Tomcat プロセスモニタを停止する必要があります。一方、最大リトライ回数分のタイムアウトが発生したあとに稼働監視コンポーネントが Tomcat サーバプロセス開始完了通知を受信できた場合には、通常どおり稼働監視を継続します。

## 10.3.2 ハートビート監視

稼働監視コンポーネントは、Tomcat サーバプロセスから定期的送信されるハートビートを監視し、ハートビート待ちタイムアウトが発生するかどうかによって、Tomcat サーバプロセスのハングアップを監視できます。

### (1) 障害発生 の判定基準

ハートビート待ちタイムアウトの計測を開始してから、Tomcat サーバプロセスからのハートビートを受信できなかった場合、障害発生と判定します。

詳細な状態は次のとおりです。

状態	説明
初期状態	Tomcat サーバプロセスからの開始完了通知受信時にハートビート待ちタイムアウトの計測を開始します。
正常状態	毎回のハートビート受信時にプロセスは正常に稼働していると判定し、ハートビート待ちタイムアウトをリセットして計測を開始します。プロセスが正常に稼働していると判定している間は、イベントは発行しません。
異常状態	ハートビート待ちタイムアウトが発生した場合、障害発生と判定し、メッセージ KDLR20223-W を出力して障害イベントを 1 回発行します。 その後、ハートビートを受信した場合は障害の回復と判定し、メッセージ KDLR20224-I を出力し回復イベントを 1 回発行して正常状態に戻ります。

障害イベント・回復イベントの発行時に出力されるイベントプロパティと、イベント検知後のアクションについては、「(2) 障害検知時の動作」を参照してください。

### (2) 障害検知時の動作

#### (a) 障害検知時に出力されるイベントプロパティ

障害イベント・回復イベントの発行時には、JSON 形式のイベントプロパティを出力します。障害イベントの例を次に示します。

```
{
  "type": "heartbeatdelay",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": false
}
```

#### (b) 障害検知後のアクション

障害イベントの通知を受けた場合、デフォルトでは次のアクションを実行します。

- スナップショットログを収集します。
- Tomcat サーバプロセスの停止指示は出さないで処理を続行します。

アクションの変更方法については「(3)(b) 障害イベント・回復イベントの通知後の動作」を参照してください。

### (3) 設定できる内容

ハートビート監視では、Tomcat サーバプロセスから定期的送信されるハートビートを監視します。ハートビートの送信は、「10.2.2 Tomcat サーバプロセスへの稼働監視用ライフサイクルリスナーの設定」で設定したライフサイクルリスナーによって実施されます。

ハートビート監視に関して設定できる内容を次に示します。

#### (a) ハートビートの監視設定

Tomcat プロセスモニタの config.properties (本製品の設定ファイル) で次の項目を設定できます。

- Tomcat プロセスモニタ側のハートビート監視の有効/無効
- Tomcat プロセスモニタ側のハートビート待ちタイムアウト時間
- Tomcat サーバプロセス側のハートビート送信間隔

該当するプロパティと設定例を次に示します。この例ではデフォルト値を記載しています。

```
healthcheck.heartbeatdelay.enabled=true
healthcheck.heartbeatdelay.timeout=60000
healthcheck.heartbeat.interval=10000
```

config.properties (本製品の設定ファイル) については、「12.2 config.properties (本製品の設定ファイル)」を参照してください。

#### (b) 障害イベント・回復イベントの通知後の動作

障害イベント・回復イベントの通知後の動作は、次の項目を組み合わせる config.properties (本製品の設定ファイル) で指定できます。

- ユーザコマンドの実行  
ユーザコマンドを定義する場合は、別途「10.4 稼働監視機能の設定 (ユーザコマンドの実行)」の定義が必要です。
- スナップショットログの収集有無
- Tomcat サーバプロセスの停止有無

回復イベントの通知時にアクションを実行する場合で、そのアクション実行時間が healthcheck.heartbeat.interval で指定したインターバルを超えると、次のハートビート送信が遅れることがあります。

該当するプロパティと設定例を次に示します。この例では、「10.4 稼働監視機能の設定 (ユーザコマンドの実行)」で設定する「ユーザコマンド定義の ID」として、「exec1」「exec2」を設定しています。

```
healthcheck.heartbeatdelay.actions.failure.usercommand.idrefs.1=exec1
healthcheck.heartbeatdelay.actions.failure.snapshot=true
healthcheck.heartbeatdelay.actions.failure.terminate=false
healthcheck.heartbeatdelay.actions.recovery.usercommand.idrefs.1=exec2
```

config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

### 10.3.3 プロセス生存監視

プロセス生存監視は、修正パッチの適用後に使用できる機能です。プロセス生存監視では、Tomcat サーバプロセスの生存状況を監視し、Tomcat サーバプロセスが異常終了した場合には、ユーザコマンドを実行します。

Tomcat サーバプロセスの生存状況が監視されている場合、次の順序で動作します。

1. Tomcat サーバプロセスが障害によって異常終了したと Tomcat プロセスモニタが判定する。
2. スナップショットログを収集する。
3. Tomcat プロセスモニタが終了する。

この場合、主にスナップショットログ収集に掛かる時間の分、Tomcat プロセスモニタの終了が遅れます。例えば、Tomcat プロセスモニタの終了後にロードバランサからのリクエストを閉塞する場合、スナップショットログ収集に掛かる時間の分、閉塞の開始が遅れます。

稼働監視機能のプロセス生存監視では、Tomcat サーバプロセスの異常を検知した場合、スナップショットログが収集される前にユーザコマンドを実行します。動作の順序を次に示します。

1. Tomcat サーバプロセスの生存状況を監視することで、Tomcat サーバプロセスの異常終了を検知する。
2. ユーザコマンドを実行する。
3. スナップショットログを収集する。
4. Tomcat プロセスモニタが終了する。

このように、プロセス生存監視によって、Tomcat プロセスモニタの終了を待つことなく、必要な処理を開始できます。

#### (1) 障害発生 の判定基準

Tomcat サーバプロセスの停止処理の実行時、次の場合に障害が発生したと判定します。

- 稼働監視機能を起因とする停止要求ではない場合、かつ、Tomcat サーバプロセスの終了ステータスが障害発生を示す値であった場合
- 稼働監視機能を起因とする停止要求ではない場合、かつ、Tomcat サーバプロセスがプロセス終了待ちタイムアウト時間内に終了しなかった場合



Tomcat サーバプロセスの終了ステータスの値、およびプロセス終了待ちタイムアウト時間は、config.properties（本製品の設定ファイル）で指定します。config.properties（本製品の設定ファイル）については、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

## (2) 障害検知時の動作

### (a) 障害検知時に出力されるイベントプロパティ

Tomcat サーバプロセスの異常終了を検知した時は、次の JSON 形式のイベントプロパティを出力します。

- イベントプロパティ（Tomcat サーバプロセスが終了した場合）

```
{
  "type": "shutdown",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": false,
  "exitstatus": 137<Tomcatサーバプロセスの終了ステータスの値>
}
```

- イベントプロパティ（Tomcat サーバプロセスがプロセス終了待ちタイムアウト時間内に終了しなかった場合）

```
{
  "type": "shutdown",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": false
}
```

### (b) 障害検知後のアクション

Tomcat サーバプロセスの障害発生を検知した場合、ユーザが定義したコマンドを実行します。コマンドを定義する方法については、「[10.4 稼働監視機能の設定（ユーザコマンドの実行）](#)」を参照してください。

## (3) 設定できる内容

config.properties（本製品の設定ファイル）で、次の項目を設定できます。

- Tomcat サーバプロセスの異常終了時に、ユーザコマンドを実行する終了ステータスの値
- Tomcat サーバプロセスの異常終了検知時に実行するコマンド

config.properties（本製品の設定ファイル）については、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

該当するプロパティと設定例を次に示します。この例では、「[10.4 稼働監視機能の設定（ユーザコマンドの実行）](#)」で設定する「ユーザコマンド定義の ID」を「exec1」としています。

```
healthcheck.shutdown.enabled=true
healthcheck.shutdown.actions.failure.condition=ERROR
healthcheck.shutdown.actions.failure.usercommand.idrefs.1=exec1
```

## 10.3.4 ヘルスチェック

Tomcat の HTTP リクエスト受付のヘルスチェックを実施できます。

稼働監視コンポーネントから OPTIONS メソッドによるヘルスチェック HTTP リクエストを送信することで、ヘルスチェックを実施します。

### (1) 障害発生 の判定基準

ヘルスチェック HTTP リクエストによる通信に失敗した場合、およびエラーステータスコードが返ってきた場合、障害発生と判定します。

詳細な状態は次のとおりです。

状態	説明
初期状態	Tomcat サーバプロセスからの開始完了通知受信時にヘルスチェック HTTP リクエストの送信を開始します。
正常状態	ヘルスチェックの結果、ステータスコードの値が 400 未満の HTTP レスポンスを受信した場合、ヘルスチェック成功と判定します。ヘルスチェック成功と判定している間は、イベントは発行しません。
異常状態	次の状態を障害発生と判定し、障害の種類に応じた障害イベントを 1 回発行します。 <ul style="list-style-type: none"><li>ヘルスチェック HTTP リクエストで接続失敗（接続タイムアウト発生を含む）やデータ送受信失敗（読み込みタイムアウト発生を含む）をした場合。この際、タイムアウトの場合はメッセージ KDLR20216-W を、タイムアウト以外の通信失敗の場合はメッセージ KDLR20214-W をそれぞれ出力します。</li><li>ステータスコードの値が 400 以上の HTTP レスポンスを受信した場合。この際、メッセージ KDLR20215-W を出力します。</li></ul> その後、ステータスコードの値が 400 未満の HTTP レスポンスを受信した場合は、障害の回復と判定し、メッセージ KDLR20218-I を出力し回復イベントを 1 回発行して正常状態に戻ります。

障害イベント・回復イベントの発行時に出力されるイベントプロパティと、イベント検知後のアクションについては、「(2) 障害検知時の動作」を参照してください。

### (2) 障害検知時の動作

#### (a) 障害検知時に出力されるイベントプロパティ

障害イベント・回復イベントの発行時には、JSON 形式のイベントプロパティを出力します。

ヘルスチェックの結果に応じて、出力するイベントプロパティは次のように異なります。

- イベントプロパティ（接続失敗またはデータ送受信失敗）

```
{
  "type": "httprequest",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": false,
  "port": 8080<リクエスト先のポート番号>,
  "virtualhost": "127.0.0.1",
```

```
"method": "GET",
"path": "*<Request-URIに指定するパス>",
"errorkind": "connectfailure<エラー種別:connectfailureまたはiofailure>",
"exceptionname": "xxx<発生した例外クラス名>",
"exceptionmsg": "yyy<発生した例外メッセージ>"
}
```

- イベントプロパティ (エラー応答)

```
{
  "type": "httprequest",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": false,
  "port": 8080<リクエスト先のポート番号>,
  "virtualhost": "127.0.0.1",
  "method": "GET",
  "path": "*<Request-URIに指定するパス>",
  "errorkind": "errorresponse",
  "statuscode": 500<ヘルスチェックレスポンスのステータスコード>
}
```

- 成功イベントプロパティ

```
{
  "type": "httprequest",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": true,
  "port": 8080<リクエスト先のポート番号>,
  "virtualhost": "127.0.0.1",
  "method": "GET",
  "path": "*<Request-URIに指定するパス>",
  "statuscode": 200<ヘルスチェックレスポンスのステータスコード>
}
```

## (b) 障害検知後のアクション

障害イベントの通知を受けた場合、デフォルトでは次のアクションを実行します。

- スナップショットログを収集します。
- Tomcat サーバプロセスの停止指示は出さないで処理を続行します。

アクションの変更方法については「(3)(b) [障害イベント・回復イベントの通知後の動作](#)」を参照してください。

## (3) 設定できる内容

ヘルスチェックに関して設定できる内容を次に示します。

### (a) ヘルスチェックの監視設定

Tomcat プロセスモニタの config.properties (本製品の設定ファイル) で、次に示す項目を設定できます。

- ヘルスチェックの有効/無効

- ヘルスチェックのインターバル
- ヘルスチェック HTTP リクエストの HTTP 接続タイムアウト時間
- ヘルスチェック HTTP リクエストの HTTP 読み込みタイムアウト時間

該当するプロパティと設定例を次に示します。この例ではデフォルト値を記載しています。

```
healthcheck.httprequest.enabled=true
healthcheck.httprequest.check.default.interval=30000
healthcheck.httprequest.check.default.connecttimeout=3000
healthcheck.httprequest.check.default.readtimeout=10000
```

config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

## (b) 障害イベント・回復イベントの通知後の動作

障害イベント・回復イベントの通知後の動作は、config.properties (本製品の設定ファイル) で変更できます。変更できる内容は次のとおりです。

- ユーザコマンドの実行  
ユーザコマンドを定義する場合は、別途「[10.4 稼働監視機能の設定 \(ユーザコマンドの実行\)](#)」の定義が必要です。
- スナップショットログの収集有無
- Tomcat サーバプロセスの停止有無

config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

なお、障害イベント・回復イベントの通知時にアクションを実行する場合は、そのアクション実行時間が経過したあとに healthcheck.httprequest.check.default.interval で指定したインターバルを空けて次のヘルスチェック HTTP リクエストを送信します。

該当するプロパティと設定例を次に示します。この例では、「[10.4 稼働監視機能の設定 \(ユーザコマンドの実行\)](#)」で設定する「ユーザコマンド定義の ID」として、「exec1」「exec2」を設定しています。

- 接続失敗の検知の場合

```
healthcheck.httprequest.actions.connectfailure.usercommand.idrefs.1=exec1
healthcheck.httprequest.actions.connectfailure.snapshot=true
healthcheck.httprequest.actions.connectfailure.terminate=false
```

- データ送受信失敗の検知の場合

```
healthcheck.httprequest.actions.iofailure.usercommand.idrefs.1=exec1
healthcheck.httprequest.actions.iofailure.snapshot=true
healthcheck.httprequest.actions.iofailure.terminate=false
```

- レスポンスステータスコードによるエラー検知の場合

```
healthcheck.httprequest.actions.errorresponse.usercommand.idrefs.1=exec1
healthcheck.httprequest.actions.errorresponse.snapshot=true
healthcheck.httprequest.actions.errorresponse.terminate=false
```

- 回復イベントに対するユーザコマンドを定義する場合

```
healthcheck.httprequest.actions.recovery.usercommand.idrefs.1=exec2
```

## 10.3.5 リクエスト処理の停滞監視

リクエスト処理の停滞監視は、Tomcat の HTTP リクエスト処理のスローダウン・ハングアップを監視します。

リクエスト処理の停滞を検知するには、リクエスト処理スレッドの停滞を検出する、次に示す Tomcat 標準のバルブ実装を定義します。このマニュアルでは停滞検出バルブと表記します。

org.apache.catalina.valves.StuckThreadDetectionValve

リクエスト処理の停滞を検知する方法について説明します。

### (1) 障害発生 の判定基準

停滞検出バルブからのリクエスト処理停滞通知を受信した場合、障害発生と判定します。

詳細な状態は次のとおりです。

状態	説明
初期状態	Tomcat サーバプロセスからの開始完了通知受信時にリクエスト停滞検知の監視を開始します。
正常状態	Tomcat サーバプロセスから停滞検出バルブの情報を受け取り、停滞しているスレッドや停滞検知バルブによってインタラプトされたスレッドがない場合は正常と判定します。正常と判定している間は、イベントは発行しません。
異常状態	Tomcat サーバプロセスから停滞検出バルブの情報を受け取り、停滞しているスレッドや停滞検知バルブによりインタラプトされたスレッドの情報がある場合、障害発生と判定し、メッセージ KDLR20220-W を出力して障害イベントを 1 回発行します。すでに障害が発生しているスレッドとは異なるスレッドの情報がある場合は、その度に新しい障害発生と判定し、メッセージ KDLR20220-W は出力しないで障害イベントを発行します。 その後、停滞検出バルブの情報から、停滞しているスレッドも停滞検知バルブによってインタラプトされたスレッドもなくなった場合は、障害の回復と判定し、メッセージ KDLR20221-I を出力し回復イベントを 1 回発行して正常状態に戻ります。

障害イベント・回復イベントの発行時に出力されるイベントプロパティと、イベント検知後のアクションについては、「(2) 障害検知時の動作」を参照してください。

## (2) 障害検知時の動作

### (a) 障害検知時に出力されるイベントプロパティ

障害イベント・回復イベントの発行時には、JSON形式のイベントプロパティを出力します。

リクエスト処理の結果に応じて、出力するイベントプロパティは次のように異なります。

- イベントプロパティ

```
{
  "type": "stuckthread",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": false,
  "objectName": "Catalina:context=/examples,host=localhost,name=StuckThreadDetectionValve,
type=valve",
  "count": 2<停滞しているスレッド数>,
  "ids": [1000,1001]<停滞しているスレッドID>,
  "interruptedcount": 0<前回報告時から新たにinterruptされたスレッド数>
}
```

- 成功イベントプロパティ

```
{
  "type": "stuckthread",
  "timestamp": "yyyy-MM-dd' T' HH:mm:ss.SSSXXX",
  "succeeded": true,
  "objectName": "Catalina:context=/examples,host=localhost,name=StuckThreadDetectionValve,
type=valve"
}
```

### (b) 障害検知後のアクション

稼働監視コンポーネントでリクエスト処理の停滞を検知した場合、スナップショットログを収集します。

アクションの変更方法については「(3)(c) 障害イベント・回復イベントの通知後の動作」を参照してください。

## (3) 設定できる内容

リクエスト処理の停滞監視に関して設定できる内容を次に示します。

### (a) Tomcat 標準の停滞検出バルブの定義

リクエスト処理の停滞監視をする場合、停滞検出バルブは次の個所に設定します。

項目	説明
設定対象のファイル	server.xml (Tomcat のサーバ設定ファイル) ※1 または context.xml (Tomcat のコンテキスト設定ファイル) ※1

項目	説明
設定する要素	org.apache.catalina.valves.StuckThreadDetectionValve*2

注※1

server.xml (Tomcat のサーバ設定ファイル) および context.xml (Tomcat のコンテキスト設定ファイル) の記述方法については、Tomcat のドキュメントを参照してください。

server.xml (Tomcat のサーバ設定ファイル) に本製品独自に追加が必要な要素については、「12.5 server.xml (Tomcat のサーバ設定ファイル)」を参照してください。

context.xml (Tomcat のコンテキスト設定ファイル) に本製品独自に追加が必要な要素については、「12.6 context.xml (Tomcat のコンテキスト設定ファイル)」を参照してください。

注※2

Engine, Host または Context コンポーネントに設定します。

設定する要素について次に説明します。

- org.apache.catalina.valves.StuckThreadDetectionValve

リクエスト処理の停滞を検知します。

停滞検出バルブを定義したコンポーネントのバックグラウンドスレッド実行 (Tomcat のデフォルト: 10 秒間隔) によって、リクエスト処理の停滞の通知が実施されます。

リクエスト処理停滞と見なす閾値 (単位: 秒) は Valve 要素に設定できます。Tomcat のデフォルトは 600 秒です。

停滞検出バルブの設定例を次に示します。

```
<Context>
  ...
  <Valve className="org.apache.catalina.valves.StuckThreadDetectionValve" .../>
  ...
</Context>
```

## (b) リクエスト処理停滞通知の監視設定

Tomcat プロセスモニタの config.properties (本製品の設定ファイル) で、停滞検出バルブからのリクエスト処理停滞通知を監視するかどうかを設定できます。

該当するプロパティと設定例を次に示します。デフォルトは無効 (false) です。

```
healthcheck.stuckthread.enabled=false
```

config.properties (本製品の設定ファイル) については、「12.2 config.properties (本製品の設定ファイル)」を参照してください。

## (c) 障害イベント・回復イベントの通知後の動作

障害イベント・回復イベントの通知後の動作は、次の項目を組み合わせる config.properties (本製品の設定ファイル) で指定できます。

- ユーザコマンドの実行

ユーザコマンドを定義する場合は、別途「[10.4 稼働監視機能の設定（ユーザコマンドの実行）](#)」の定義が必要です。

- スナップショットログの収集

- Tomcat サーバプロセスの停止

該当するプロパティと設定例を次に示します。この例では、「[10.4 稼働監視機能の設定（ユーザコマンドの実行）](#)」で設定する「ユーザコマンド定義の ID」として、「exec1」「exec2」を設定しています。

```
healthcheck.stuckthread.actions.failure.usercommand.idrefs.1=exec1
healthcheck.stuckthread.actions.failure.snapshot=true
healthcheck.stuckthread.actions.failure.terminate=false
healthcheck.stuckthread.actions.recovery.usercommand.idrefs.1=exec2
```

config.properties（本製品の設定ファイル）については、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。



## 10.4 稼働監視機能の設定（ユーザコマンドの実行）

稼働監視によって障害を検知した場合に、ユーザが定義したコマンドを実行させることができます。

ユーザコマンドは、稼働監視コンポーネントで管理するスレッドプールを使って、`java.lang.ProcessBuilder`によって実行します。クラウド・コンテナの管理インフラに情報を通知するコマンドを定義することで、情報通知ができます。

ユーザコマンドを定義するには、コマンドの内容とコマンドを実行するスレッドプールを `config.properties`（本製品の設定ファイル）に設定する必要があります。

設定できる項目は次のとおりです。

- スレッドプールのサイズ
- 実行するコマンド
- コマンドの引数
- ユーザコマンドの標準出力をリダイレクトするファイルパス
- ユーザコマンドの標準エラー出力をリダイレクトするファイルパス

該当するプロパティと設定例を次に示します。プロパティキーの「`healthcheck.usercommand.defs.`」の後ろに指定した<group-id>文字列がユーザコマンド定義の ID になります。この例では、「`exec1`」がユーザコマンド定義の ID です。

```
healthcheck.usercommand.threadpoolsize=10
healthcheck.usercommand.defs.exec1.command=senderrorCommand
healthcheck.usercommand.defs.exec1.args.1=param1
healthcheck.usercommand.defs.exec1.stdout.file.path=${common.base}/stdout.log
healthcheck.usercommand.defs.exec1.stderr.file.path=${common.base}/stderr.log
```

`config.properties`（本製品の設定ファイル）については、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

ユーザコマンド定義の ID は、各監視項目の障害検知時の動作のアクションとして複数指定できます。複数指定したユーザコマンドのうち、1つの実行に失敗した場合、そのユーザコマンドに失敗したことを示すメッセージ `KDLR20228-E` を出力して、次のコマンドの実行を継続します。スレッドプールはユーザコマンド全体で共有します。

# 11

## スナップショットログ収集機能

この章では、スナップショットログ収集機能の概要、収集方法、機密情報のマスキング、出力テスト、およびユースケース別の設定方法について説明します。

## 11.1 スナップショットログ収集機能の概要

---

スナップショットログ収集機能とは、障害を検知したときに、原因の解析に必要な保守情報（スナップショットログ）を一括で収集できる機能です。

障害が発生した場合は、原因の解析をサポートサービスに依頼するために、次の情報を収集する必要があります。

- ログの情報
- スレッドダンプの情報
- 実行環境の情報

スナップショットログ収集機能を利用すると、これらの情報を一括で自動収集できるため、迅速にサポートサービスへ問い合わせることができます。

また、この機能には、スナップショットログを自動収集する方法に加え、手動収集する方法も備わっています。手動収集では、次のどちらかを利用して任意のタイミングで情報を収集できます。

- スナップショットログ収集コマンド
- スナップショットログ収集 REST API

そのため、サポートサービスに依頼するときだけでなく、自身で障害の原因を解析するときにも、この機能を利用できます。

スナップショットログ収集機能には、このほかにも次の機能があります。

- 収集対象のカスタマイズ
- パスワードなどの機密情報のマスキング
- 障害発生時の、揮発性のある環境<sup>※1</sup>での情報の保持<sup>※2</sup>

注※1

コンテナ環境などを指します。

注※2

スナップショットログの出力先を永続化領域に指定する必要があります。

## 11.2 スナップショットログの出力

スナップショットログの出力先、および出力形式について説明します。

### 11.2.1 スナップショットログの出力先

スナップショットログの出力先は、次のとおりです。

```
<snapshot.log.filepathの指定値>_<yyyy-MM-dd_HH-mm-ss.SSS>_<n>.zip
```

#### 説明

- 「snapshot.log.filepath」は、config.properties（本製品の設定ファイル）のプロパティです。詳細は、「12.2.4(4) スナップショットログ収集機能に関するプロパティ」を参照してください。
- 「yyyy-MM-dd\_HH-mm-ss.SSS」は、時刻情報を示します。
- 「n」は、「何回目のスナップショットログ出力要求か」を表す通番<sup>※</sup>を示します。

#### 注※

Tomcat プロセスモニタを起動するたびにリセットされます。

スナップショットログの出力先を変更したい場合は、「11.9.1 スナップショットログの出力先を変更したい場合」を参照してください。

### 11.2.2 スナップショットログの出力形式

スナップショットログ収集機能では、収集された情報（スナップショットログ）が zip 形式で出力されます。スナップショットログの zip ファイル内のパスの規則を次の表に示します。スナップショットログの zip ファイルに含まれるファイルの詳細は、「11.3.1 ファイルによる情報の収集」および「11.3.2 コマンドの実行による情報の取得」を参照してください。

表 11-1 スナップショットログの zip ファイル内のパスの規則

形式	zip ファイル内のパス	備考
ファイル・ディレクトリ	絶対パスの先頭の「/」を除いた値（ルートディレクトリからの相対パス）	なし
アーカイブファイル内のファイル	アーカイブファイルの絶対パスの先頭の「/」を除いた値に、「_」およびアーカイブファイル内の絶対パスを加えた値	アプリケーションを war 形式でデプロイしている場合、デプロイメント・ディスクリプタファイルが該当します。

ファイル・ディレクトリの、zip ファイル内のパスの例を次に示します。

- 収集ファイルの絶対パス

```
/var/log/messages
```

- スナップショットログの zip ファイル内のパス

```
var/log/messages
```

アーカイブファイル内の、zip ファイル内のパスの例を次に示します。

- アーカイブファイルの絶対パス

```
/var/tomcat/webapps/sample.war
```

- アーカイブファイル内のファイルの絶対パス

```
/WEB-INF/web.xml
```

- スナップショットログの zip ファイル内のパス

```
var/tomcat/webapps/sample.war_/WEB-INF/web.xml
```

## 11.3 スナップショットログの収集対象

スナップショットログ収集機能で収集される情報を次の表に示します。表中の情報を収集できる場合、それらがスナップショットログに格納された状態で出力されます。

表 11-2 スナップショットログ収集機能の収集対象情報と収集方法

カテゴリ	収集対象情報	収集方法※
ホストマシン	<ul style="list-style-type: none"> <li>OS バージョン情報</li> <li>インストール PP 情報</li> </ul>	コマンド
	定義情報	ファイル
	環境変数	コマンド
	ホストマシンのリソース使用状況	
	ログ (syslog)	ファイル
JavaVM	システムプロパティ	コマンド
	ログ	ファイル
	コアダンプ	
	スレッドダンプ	コマンド
Tomcat	バージョン情報	コマンド
	定義情報	ファイル
	標準出力・標準エラー出力	
	ログ <ul style="list-style-type: none"> <li>Tomcat サーバプロセスが出力するログファイル</li> <li>独自ログファイル</li> <li>JavaVM ログ (日立 JavaVM を使用している場合だけ)</li> </ul>	
	トレース機能のログ	
Web アプリケーション	定義情報	ファイル
	ログ	
Tomcat プロセスモニタ	バージョン情報	ファイル
	定義情報	
	ログ <ul style="list-style-type: none"> <li>インストーラログ</li> <li>メッセージログ</li> <li>保守ログ</li> </ul>	

カテゴリ	収集対象情報	収集方法※
	<ul style="list-style-type: none"> <li>JavaVM ログ (日立JavaVM を使用している場合だけ)</li> </ul>	

注※

収集方法が「コマンド」になっている収集対象情報は、本製品が OS のコマンドを実行することで取得されます。

収集方法が「ファイル」の場合の詳細は、「11.3.1 ファイルによる情報の収集」を参照してください。収集方法が「コマンド」の場合の詳細は、「11.3.2 コマンドの実行による情報の取得」を参照してください。

## 11.3.1 ファイルによる情報の収集

「表 11-2 スナップショットログ収集機能の収集対象情報と収集方法」で収集方法が「ファイル」になっている収集対象情報は、ファイルに含まれた状態で収集されます。ここでは、スナップショットログ収集機能で収集されるファイルについて説明します。

収集対象を変更したい場合は、「11.8.1 収集対象を変更したい場合」を参照してください。

### ❗ 重要

収集対象外のファイルは収集されません。また、収集対象および収集対象外の両方に当てはまるファイルは収集されません。例えば、`snapshot.include.paths.<n>*`と `snapshot.exclude.globs.<n>*`にそれぞれ同じファイルを指定した場合は、そのファイルは収集されません。

注※

`config.properties` (本製品の設定ファイル) のプロパティです。詳細は、「12.2.4(4) スナップショットログ収集機能に関するプロパティ」を参照してください。

## (1) 収集対象のパス

スナップショットログ収集機能の収集対象のパスを次の表に示します。glob 形式で表記しています。

表 11-3 スナップショットログ収集機能の収集対象のパス

収集対象のパス (glob 形式)	パスの説明
<code>/etc/hosts</code>	ホスト定義ファイルのパス
<code>/var/log/messages*</code>	syslog のパス
<code>/var/log/syslog*</code>	syslog のパス
<code>\${CATALINA_BASE}/bin/setenv.sh</code>	Tomcat の環境設定スクリプトのパス
<code>\${CATALINA_HOME}/bin/setenv.sh</code>	Tomcat の環境設定スクリプトのパス

収集対象のパス (glob 形式)	パスの説明
<code>\${CATALINA_BASE}/conf</code>	Tomcat の設定ディレクトリのパス
<code>\${CATALINA_HOME}/conf</code>	Tomcat の設定ディレクトリのパス
<common.base <sup>*1</sup> の指定値>	ログ出力先ディレクトリのパス
<monitor.log.maintenance.filepath <sup>*1</sup> の指定値>*	Tomcat プロセスモニタの保守ログのパス
<monitor.log.message.filepath <sup>*1</sup> の指定値>*	Tomcat プロセスモニタのメッセージログのパス
<snapshot.include.paths.<n> <sup>*1</sup> の指定値>	追加で指定した収集対象のファイルのパス <sup>*2</sup>
<tracer.log.filepath <sup>*1</sup> の指定値>*	トレース機能のログのパス

(凡例)

\* : ワイルドカードを示します。

注※1

config.properties (本製品の設定ファイル) のプロパティです。詳細は、「[12.2.4 指定可能なプロパティ](#)」の各プロパティの説明を参照してください。

注※2

複数のパスを収集対象に指定できます。収集対象のファイルを追加したい場合は、「[11.8.1\(2\) スナップショットログの収集対象を追加する](#)」を参照してください。

なお、これらのパスがディレクトリを指す場合、サブディレクトリの情報も収集されます。収集対象のパスにファイルが存在しない場合は、情報が収集されません。

## (2) その他の収集対象

「[表 11-3 スナップショットログ収集機能の収集対象のパス](#)」の情報に加え、次の表に示す情報も収集されます。

表 11-4 スナップショットログ収集機能のその他の収集対象

収集対象	説明
config.properties (本製品の設定ファイル)	なし
本製品のバージョン情報ファイル	<ul style="list-style-type: none"> <li>• /etc/.hitachi/pplistd/pplistd</li> <li>• /etc/.hitachi/ppinfo</li> <li>• &lt;本製品のインストールディレクトリ&gt;/internal/version.dat</li> </ul>
本製品のインストールログ	<本製品のインストールディレクトリ>/install.log
各アプリケーションのデプロイメント・ディスクリプタファイル	デプロイされたアプリケーション内の次のファイルが収集されます。 <ul style="list-style-type: none"> <li>• /META-INF/context.xml</li> <li>• /WEB-INF/web.xml</li> <li>• /WEB-INF/tomcat-web.xml</li> </ul>
コマンド実行結果格納ディレクトリ	スナップショットログ収集中にコマンドが実行され、結果がファイルに出力されます。コマンド実行結果格納ディレクトリには、これらのファイルが格納されます。



収集対象	説明
	コマンド実行結果格納ディレクトリの詳細は、「11.3.2 コマンドの実行による情報の取得」を参照してください。

### (3) 独自の収集対象パス (日立 JavaVM を使用する場合)

日立 JavaVM を利用している場合、「(1) 収集対象のパス」および「(2) その他の収集対象」で示した情報に加え、スレッドダンプファイルも収集されます。詳細を次の表に示します。

表 11-5 スナップショットログ収集機能で日立 JavaVM を利用する場合だけ収集される情報

収集対象	説明
スレッドダンプファイル	Tomcat サーバプロセスが起動してから出力されたスレッドダンプファイルがすべて収集されます。次のファイルが収集対象です。 <ul style="list-style-type: none"> <li>スナップショットログ収集時に要求したスレッドダンプファイル</li> <li>スナップショットログ収集要求以前に出力されたスレッドダンプファイル</li> </ul>

なお、スレッドダンプファイルの出力ファイル名および出力内容は、マニュアル「uCosminexus Application Runtime - Cosminexus Developer's Kit for Java 機能解説・リファレンス」を参照してください。

### (4) 収集対象外ファイル

スナップショットログ収集機能で収集対象外のファイルパスを次の表に示します。パスは glob 形式です。

表 11-6 収集対象外のパス

収集対象外のパス (glob 形式)	説明
<code>\${CATALINA_BASE}/conf/tomcat-users.xml</code>	Tomcat Manager のログインユーザ定義ファイルのパス
<code>\${CATALINA_HOME}/conf/tomcat-users.xml</code>	Tomcat Manager のログインユーザ定義ファイルのパス
<code>&lt;snapshot.log.filepath<sup>※1</sup>の指定値&gt;*.zip</code>	過去に出力されたスナップショットファイルのパス
<code>&lt;snapshot.exclude.globs.&lt;n&gt;<sup>※1</sup>の指定値&gt;</code>	収集対象外に指定したファイルのパス <sup>※2</sup>

注※1

config.properties (本製品の設定ファイル) のプロパティです。詳細は、「12.2.4(4) スナップショットログ収集機能に関するプロパティ」を参照してください。

注※2

複数のパスを収集対象外に指定できます。収集対象外のパスを指定したい場合は、「11.8.1(1) 特定のファイルおよび特定のディレクトリを収集対象外にする」を参照してください。

## 11.3.2 コマンドの実行による情報の取得

「表 11-2 スナップショットログ収集機能の収集対象情報と収集方法」で収集方法が「コマンド」になっている収集対象情報は、本製品が OS のコマンドを実行することで取得されます。ここでは、OS のコマンド実行による情報取得の仕組みと、取得される情報について説明します。

次のファイルに記載されない、障害の解析に必要な情報は、本製品が OS のコマンドを実行することで取得されます。

- 定義ファイル
- ログファイル

コマンドを実行した場合の標準出力および標準エラー出力の内容は、コマンド実行結果格納ディレクトリ以下に出力され、スナップショットログに収集されます。コマンド実行結果格納ディレクトリは、次のパスに作成されます。

```
<Tomcat プロセスモニタの一時領域>/snapshot_<yyyy-MM-dd_HH-mm-ss.SSS>_<n>
```

<Tomcat プロセスモニタの一時領域>については、「8.8 Tomcat プロセスモニタの一時領域」を参照してください。パスの<yyyy-MM-dd\_HH-mm-ss.SSS>の部分は、時刻情報を示します。<n>の部分は、Tomcat プロセスモニタを起動してから何番目の取得要求かを表す通番<sup>\*</sup>を示します。

注※

一番目の取得要求を 1 とします。

コマンド実行結果格納ディレクトリは、スナップショットログの取得要求時に作成され、スナップショットログのファイルの出力後に削除されます。出力処理中にエラーが発生した場合もコマンド実行結果格納ディレクトリは削除されます。

次に、本製品が OS のコマンドを実行することで取得される情報について説明します。

### (1) Tomcat 稼働中情報の取得

スナップショットログの取得時に、Tomcat サーバプロセスが稼働している場合だけ Tomcat 稼働中情報が取得されます。Tomcat 稼働中情報とは、次に示す「モニタリング情報」と「スレッドダンプ情報」を指します。Tomcat 稼働中情報は、コマンド実行結果格納ディレクトリ以下に出力されます。

#### (a) モニタリング情報

モニタリング情報の取得について説明します。

実行環境のメモリおよび CPU の使用状況を測定し、マシンリソースの使用情報が取得されます。測定には少なくとも 5 秒は掛かるため、次の場合だけ測定します。

表 11-7 マシンリソースの使用情報の出力トリガーと取得条件

出力トリガー	取得条件
Tomcat サーバプロセスの停止要求時	snapshot.onshutdownrequest.watchcommand.enabled <sup>*</sup> の指定値が true のとき
稼働監視の異常検知時	snapshot.onhealthcheck.watchcommand.enabled <sup>*</sup> の指定値が true のとき
スナップショットログ収集コマンド	「16.2 スナップショットログ収集コマンド」で指定した値
スナップショットログ収集 REST API	「17.2 スナップショットログ収集 REST API」で指定した値

注※

config.properties (本製品の設定ファイル) のプロパティです。詳細は、「12.2.4(4) スナップショットログ収集機能に関するプロパティ」を参照してください。

出力トリガーについては、「11.5.1(2) Tomcat 稼働中情報の取得」を参照してください。

モニタリング情報の取得時に実行されるコマンド、出力先、およびその説明を次の表に示します。なお、各コマンドは並列で実行されます。

表 11-8 モニタリング情報の取得時のコマンド、出力先、およびその説明

コマンド	出力先	説明
vmstat 1 5	vmstat.txt	CPU、メモリ、およびディスク I/O などのマシンリソースの使用状況をモニタリングして取得します。
iostat 1 5	iostat.txt	I/O デバイスの使用状況を取得します。
top -b -n 5	top.txt	プロセスごとの CPU の使用状況をモニタリングして取得します。
sar -A 1 5	sar.txt	各種マシンリソースの使用状況をモニタリングして取得します。 <sup>*</sup>

注※

Tomcat プロセスモニタの実行ユーザが root 権限を持っている場合に、結果が格納されます。

## (b) スレッドダンプ情報

Tomcat サーバプロセスのスレッドダンプ情報が取得されます。スレッドダンプ情報は Tomcat サーバプロセスに負荷をかけるため、稼働監視で障害を検知した場合だけ取得されます。ただし、スレッドダンプ情報の取得回数、および取得間隔を設定することで、そのほかのタイミングでも取得できます。

スレッドダンプ情報の出力トリガー、取得回数、および取得間隔を次の表に示します。

表 11-9 スレッドダンプ情報の出力トリガー、取得回数、および取得間隔

出力トリガー	取得回数	取得間隔
稼働監視の異常検知時	<ul style="list-style-type: none"> <li>デフォルトの場合： 3 回</li> <li>変更した場合：</li> </ul>	<ul style="list-style-type: none"> <li>デフォルトの場合： 1000 ミリ秒</li> <li>変更した場合：</li> </ul>

出力トリガー	取得回数	取得間隔
	<snapshot.onhealthcheck.threaddumpnum*の指定値>回	<snapshot.default.threaddumpinterval*の指定値>ミリ秒
Tomcat サーバプロセスの停止要求時	<ul style="list-style-type: none"> <li>デフォルトの場合：取得されない</li> <li>設定した場合：&lt;snapshot.onshutdownrequest.threaddumpnum*の指定値&gt;回</li> </ul>	<ul style="list-style-type: none"> <li>デフォルトの場合：取得されない</li> <li>変更した場合：&lt;snapshot.default.threaddumpinterval*の指定値&gt;ミリ秒</li> </ul>
スナップショットログ収集コマンド	「16.2 スナップショットログ収集コマンド」で指定した値	<ul style="list-style-type: none"> <li>デフォルトの場合：1000 ミリ秒</li> <li>変更した場合：&lt;snapshot.default.threaddumpinterval*の指定値&gt;ミリ秒</li> </ul>
スナップショットログ収集 REST API	「17.2 スナップショットログ収集 REST API」で指定した値	

注※

config.properties（本製品の設定ファイル）のプロパティです。詳細は、「12.2.4(4) スナップショットログ収集機能に関するプロパティ」を参照してください。

表の内容について説明します。

- 稼働監視の障害検知時のスレッドダンプの取得  
デフォルトでは、1000 ミリ秒間隔で3回取得されます。ただし、取得回数および取得間隔は変更できます。取得回数および取得間隔を変更する方法については、「11.9.4 稼働監視で障害を検知したときの Tomcat 稼働中情報取得時の条件をカスタマイズしたい場合」を参照してください。
- Tomcat サーバプロセスの停止要求時のスレッドダンプの取得  
デフォルトでは取得されません。設定すれば取得できます。設定方法については、「11.9.3 Tomcat サーバプロセスの停止要求時、停止する前に Tomcat 稼働中情報を取得したい場合」を、出力トリガーについては、「11.5.1(2) Tomcat 稼働中情報の取得」を参照してください。

出力トリガーの詳細については、「11.5.1(2) Tomcat 稼働中情報の取得」を参照してください。

### (c) スレッドダンプ情報（日立 JavaVM を使用する場合）

日立 JavaVM を利用する場合、kill -3 コマンドの実行によって Tomcat サーバプロセスのスレッドダンプ情報が取得されます。

スレッドダンプ情報は、次のとおりに出力されます。

```
<スレッドダンプの出力先>/<スレッドダンプのファイル名>
```

#### スレッドダンプの出力先

スレッドダンプの出力先は、次のどちらかです。

- 環境変数 JAVACOREDİR の値
- common.java.hitachi.javacoredir\*の指定値

注※ config.properties（本製品の設定ファイル）のプロパティです。詳細は、「12.2.4(1) 本製品全体に関するプロパティ」を参照してください。

スレッドダンプの出力先を変更したい場合は、「11.8.5 スレッドダンプの出力先を変更したい場合（日立 JavaVM 使用時）」を参照してください。

## スレッドダンプのファイル名

「表 11-10 kill -3 <pid> コマンド実行時の出力先とその説明（日立 JavaVM を使用する場合）」を参照してください。

取得されたスレッドダンプファイルは、スナップショットログの出力処理が完了したあとに削除されます。スレッドダンプのファイル名、およびスレッドダンプに出力される内容については、マニュアル「uCosminexus Application Runtime - Cosminexus Developer's Kit for Java 機能解説・リファレンス」を参照してください。

次の表に、kill -3 <pid> コマンド実行時の出力先およびその説明を示します。

表 11-10 kill -3 <pid> コマンド実行時の出力先とその説明（日立 JavaVM を使用する場合）

コマンド	出力先	説明
kill -3 <pid>	sigquit_<n>.txt	sigquit_<n>.txt には、シグナルを送信した結果を出力します。

(凡例)

- <n>：ある出力要求での何回目の取得かを示します。
- <pid>：Tomcat サーバプロセスの PID を示します。

## (d) スレッドダンプ情報（他社製 JavaVM を使用する場合）

他社製 JavaVM を使用する場合で、jcmd コマンドを利用できるときは、jcmd コマンドの実行によって Tomcat サーバプロセスのスレッドダンプ情報が取得されます。スレッドダンプに出力される内容は、使用している JavaVM のマニュアルを参照してください。

次の表に、jcmd コマンド実行時の出力先およびその説明を示します。

表 11-11 jcmd コマンド実行時の出力先とその説明（他社製 JavaVM を使用する場合）

コマンド	出力先	説明
jcmd <pid> Thread.print	jcmd_thread_<n>.txt	スレッドダンプ情報を出力します。
jcmd <pid> GC.heap_info	jcmd_heapinfo_<n>.txt	ヒープ情報を出力します。

(凡例)

- <n>：ある出力要求での何回目の取得かを示します。
- <pid>：Tomcat サーバプロセスの PID を示します。

次のどちらかの場合は、kill -3 <pid> コマンドの実行によってスレッドダンプが取得されます。

- jcmd コマンドが存在しない場合
- jcmd コマンドが失敗した場合

kill -3 <pid> コマンド実行時の出力先およびその説明を次の表に示します。

表 11-12 kill -3 <pid> コマンド実行時の出力先およびその説明（他社製 JavaVM を使用する  
場合）

コマンド	出力先	説明
kill -3 <pid>	sigquit_<n>.txt	<ul style="list-style-type: none"><li>Tomcat サーバプロセスの標準出力にもスレッドダンプを出力します。</li><li>sigquit_&lt;n&gt;.txt には、シグナルを送信した結果を出力します。</li></ul>

(凡例)

<n> : ある出力要求での何回目の取得かを示します。

<pid> : Tomcat サーバプロセスの PID を示します。

## (2) 環境情報の取得

スナップショットログの取得時に、Tomcat サーバプロセスが稼働しているかどうかに関係なく、必ず環境情報が取得されます。環境情報は、コマンド実行結果格納ディレクトリ以下に出力されます。

### (a) ホストマシン情報

ホストマシン情報およびネットワーク使用状況が取得されます。これらの情報の取得時に実行されるコマンド、出力先、およびその説明を次の表に示します。なお、各コマンドは並列で実行されます。

表 11-13 ホストマシン情報の取得時のコマンド、出力先、およびその説明

コマンド	出力先	説明
df	df.txt	ディスクの使用状況を取得します。
ps -eflm	ps.txt	プロセスの状況を取得します。
netstat -s	netstat_s.txt	ネットワーク統計情報を取得します。
netstat -an	netstat_an.txt	ネットワークの使用状況を取得します。
sysctl -a	sysctl.txt	サービスの状況を取得します。
rpm -qa	rpm_qa.txt	ホストマシンにインストールされている PP 情報を取得します。
rpm -qai	rpm_qai.txt	ホストマシンにインストールされている PP 情報の詳細を取得します。
dpkg -l	dpkg.txt	ホストマシンにインストールされている PP 情報を取得します。
uname -a	uname_a.txt	OS のバージョン情報を取得します。
env	env.txt	環境変数を取得します。
set	set.txt	シェル変数およびシェル関数を取得します。

コマンド	出力先	説明
ipcs	ipcs.txt	プロセス間通信機能の状況を取得します。
ipcs -t	ipcs_t.txt	プロセス間通信機能の制御時刻を取得します。
ipcs -p	ipcs_p.txt	プロセス間通信機能のプロセス情報を取得します。
ipcs -c	ipcs_c.txt	プロセス間通信機能のユーザ情報を取得します。
ipcs -u	ipcs_u.txt	プロセス間通信機能のサマリを取得します。
ipcs -l	ipcs_l.txt	プロセス間通信機能の制限値情報を取得します。
\${CATALINA_HOME}/bin/ version.sh	tomcat_version.txt	Tomcat のバージョン情報を取得します。

## (b) Java の実行環境情報

Java の実行環境情報について説明します。

Java の実行環境情報を取得するために、システムプロパティの値が取得されます。ここで取得される対象は、Tomcat プロセスモニタのシステムプロパティです。

すべてのシステムプロパティのキーおよび値が「<key>=<value>」形式で、system\_properties.txt に出力されます。

### ❗ 重要

Tomcat サーバプロセスのシステムプロパティは出力されません。

## 11.4 機密情報のマスキング

定義ファイル中に機密情報を平文で記述している場合、スナップショットログに機密情報が含まれないようにマスキングする必要があります。

### 11.4.1 定義情報のマスキング

接続先情報や認証情報などの機密情報が、スナップショットログに平文で格納されないように、指定したルールと合致する文字列をマスキングできます。ルールは、正規表現で指定します。正規表現に合致した文字列の最初のグループが「\*\*\*\*\*」に置換されます。

#### デフォルトのマスキングルール

次のルールに合致する情報は、スナップショットログに含まれないように必ずマスキングされます。

```
password="(.+?)"
```

このルールは、ユーザがマスキングルールを追加するかどうかに関係なく、必ず適用されます。

#### 追加のマスキングルール

デフォルトのマスキングルールに合致しない情報を追加でマスキングしたい場合は、マスキングルールを追加してください。マスキングルールの追加方法については、「[11.8.2 機密情報のマスキングルールを追加したい場合](#)」を参照してください。

#### マスキングルールの適用対象

マスキングルールが適用される対象は、次のとおりです。

表 11-14 マスキングルールの適用対象

適用対象	説明
<code>\${CATALINA_BASE}/conf</code> 以下のすべてのファイル	Tomcat の設定ディレクトリを指します。
<code>\${CATALINA_HOME}/conf</code> 以下のすべてのファイル	
デプロイされたアプリケーション内の次のファイル <ul style="list-style-type: none"><li><code>/META-INF/context.xml</code></li><li><code>/WEB-INF/web.xml</code></li><li><code>/WEB-INF/tomcat-web.xml</code></li></ul>	各アプリケーションのデプロイメント・ディスクリプタファイルを指します。
<code>\${CATALINA_BASE}/bin/setenv.sh</code>	Tomcat の環境設定スクリプトを指します。
<code>\${CATALINA_HOME}/bin/setenv.sh</code>	
env コマンドの実行結果	env コマンドを実行して取得する環境変数を指します。env コマンドの詳細は、「 <a href="#">11.3.2(2)(a) ホストマシン情報</a> 」を参照してください。
set コマンドの実行結果	set コマンドを実行して取得するシェル変数およびシェル関数を指します。



適用対象	説明
	set コマンドの詳細は、「11.3.2(2)(a) ホストマシン情報」を参照してください。
システムプロパティの一覧	システムプロパティについては、「11.3.2(2)(b) Javaの実行環境情報」を参照してください。

### ❗ 重要

ログファイルにはマスキングルールが適用されません。

## 11.4.2 config.properties (本製品の設定ファイル) のマスキング

config.properties (本製品の設定ファイル) の次のプロパティキーの指定値は、「11.4.1 定義情報のマスキング」で示したマスキングルールに関係なく、マスキングされます。

- snapshot.maskingrule.regexes.<n>

例えば、config.properties (本製品の設定ファイル) を「config.properties (本製品の設定ファイル) の指定例」のように作成した場合、「スナップショットログの出力例」のように出力されます。

config.properties (本製品の設定ファイル) の指定例

```
snapshot.log.filepath=${common.base}/snapshot
snapshot.maskingrule.regexes.1=secretToken="(.)+?"
snapshot.maskingrule.regexes.2=^DB_PASSWORD=(.)+$
snapshot.default.threaddumpinterval=1000
```

スナップショットログの出力例

```
snapshot.log.filepath=${common.base}/snapshot
snapshot.maskingrule.regexes.1=*****
snapshot.maskingrule.regexes.2=*****
snapshot.default.threaddumpinterval=1000
```

上記のように、太字のプロパティキーの指定値が「\*\*\*\*\*」に置換され、マスキングされます。

## 11.5 スナップショットログの自動収集

スナップショットログの自動収集について説明します。

### 11.5.1 障害時の保守情報の収集

Tomcat プロセスモニタが Tomcat サーバプロセスの障害を検知すると、保守情報（スナップショットログ）が次のパスに自動で出力されます。

```
<snapshot.log.filepath※の指定値>_<yyyy-MM-dd_HH-mm-ss.SSS>_<n>.zip
```

#### 説明

- 「snapshot.log.filepath」は、config.properties（本製品の設定ファイル）のプロパティです。詳細は、「[12.2.4\(4\) スナップショットログ収集機能に関するプロパティ](#)」を参照してください。
- このパスの<yyyy-MM-dd\_HH-mm-ss.SSS>\_<n>の値を「スナップショットログ ID」と呼びます。
- 「yyyy-MM-dd\_HH-mm-ss.SSS」は、時刻情報を示します。
- 「n」は、「Tomcat プロセスモニタを起動してから何番目の収集要求か」を表す通番※を示します。  
注※ 一番目の収集要求を 1 とします。
- スナップショット収集中のログは、Tomcat プロセスモニタのメッセージログに出力されます。

スナップショットログのログファイルの活用方法を次に示します。

- サポートサービスに障害の原因解析を依頼する場合  
zip ファイルをサポートサービスに送付してください。
- 自身で障害の原因を解析する場合  
zip ファイルを展開して調査してください。スナップショットログに含まれる情報については、「[11.3 スナップショットログの収集対象](#)」を参照してください。

#### (1) スナップショットログの自動収集の条件

スナップショットログが出力される条件は、「[8.5 自動でスナップショットログが収集されるタイミング](#)」を参照してください。

#### (2) Tomcat 稼働中情報の取得

本製品が障害を検知したときに、Tomcat サーバプロセスが稼働中であれば Tomcat 稼働中情報が取得されます。ただし、config.properties（本製品の設定ファイル）のプロパティで設定すれば、Tomcat プロセスモニタが Tomcat サーバプロセスの停止要求を検知したとき、停止する前に Tomcat 稼働中情報を取

得できます。設定方法は、「11.9.3 Tomcat サーバプロセスの停止要求時、停止する前に Tomcat 稼働中情報を取得したい場合」を参照してください。

また、稼働監視で障害を検知したときの Tomcat 稼働中情報の取得時の条件をカスタマイズしたい場合は、「11.9.4 稼働監視で障害を検知したときの Tomcat 稼働中情報取得時の条件をカスタマイズしたい場合」を参照してください。

Tomcat 稼働中情報の出力トリガーと取得可否を次の表に示します。

表 11-15 Tomcat 稼働中情報の出力トリガーと取得可否

Tomcat 稼働中情報の出力トリガー	Tomcat 稼働中情報の取得可否	備考
Tomcat サーバプロセスの停止要求を受けた	○	Tomcat プロセスモニタが停止シグナルを受ける終了方式の場合に取得されます。 Tomcat サーバプロセスの終了方式によっては、Tomcat 稼働中情報を取得できない場合があります。例えば、Tomcat サーバプロセスがシャットダウンポートによって終了するときは、Tomcat 稼働中情報を取得できません。
Tomcat サーバプロセスが終了した	×	なし
稼働監視が異常を検知した	○	なし

(凡例)

- ：Tomcat 稼働中情報が取得されます。
- ×：Tomcat 稼働中情報が取得されません。

## 11.6 スナップショットログの手動収集

スナップショットログを手動で収集する方法を説明します。

### ❗ 重要

スナップショットログを手動で収集する前に、「[11.6.3 スナップショットログの多重実行に関する注意事項](#)」を必ず確認してください。

### 11.6.1 Tomcat プロセスモニタが稼働しているときの収集方法

Tomcat プロセスモニタが稼働中の場合、次の方法でスナップショットログを収集できます。

- スナップショットログ収集コマンドを実行して収集する
- HTTP リクエストを送信して収集する

それぞれの収集方法について説明します。

#### (1) スナップショットログ収集コマンドを実行して収集する

本製品のコマンドを実行することで、スナップショットログを収集できます。実行するコマンドとスナップショットログの出力先は次のとおりです。

実行するコマンド

```
<本製品のインストールディレクトリ>/bin/collect-snapshot.sh
```

スナップショットログの出力先

```
<カレントディレクトリ>/snapshot.zip
```

スナップショットログ収集コマンドのオプションは、「[16.2 スナップショットログ収集コマンド](#)」を参照してください。また、スナップショットログのファイルに含まれる情報については、「[11.3 スナップショットログの収集対象](#)」を参照してください。

収集中のログは、Tomcat プロセスモニタのメッセージログに出力されます。メッセージログの詳細は、「[13.2.1 メッセージログ](#)」を参照してください。

#### (2) HTTP リクエストを送信して収集する

次の URL に HTTP リクエストを送信すると、スナップショットログが収集され、レスポンスとして出力されます。

送信するリクエストの内容を次の表に示します。

表 11-16 リクエストの内容

項目	内容
Method	GET
URL	http://<IP アドレス>:<ポート番号>/api/v1/snapshot

<IP アドレス>には Tomcat プロセスモニタを起動させているマシンの IP アドレスを指定してください。<ポート番号>には、Tomcat プロセスモニタの HTTP 機能の受付ポート番号 (monitor.rest.port の値の指定値) と同じ値を指定してください。monitor.rest.port については、「[12.2.4\(2\) Tomcat プロセスモニタに関するプロパティ](#)」を参照してください。

出力されるレスポンスの内容を次の表に示します。

表 11-17 レスポンスの内容

項目	内容
Status	200
Response Header : Content-Type	application/zip
Response Body	<スナップショットログデータ>

HTTP リクエストで指定できるオプションについては、「[17.2 スナップショットログ収集 REST API](#)」を参照してください。

また、スナップショットログのファイルに含まれる情報については、「[11.3 スナップショットログの収集対象](#)」を参照してください。

## 11.6.2 Tomcat プロセスモニタが稼働していないときの取得方法

Tomcat プロセスモニタが稼働していない環境で、スナップショットログと同等の内容の保守情報を取得する場合は、次の (1) から (3) までの手順を実行してください。

### (1) コマンドの実行と結果の保存

1. 次のコマンドを実行して、コマンド実行結果格納ディレクトリを作成する。

```
mkdir -p <Tomcatプロセスモニタの一時領域>/snapshot_$(date +%Y-%m-%d_%H-%M-%S)_manual
```

Tomcat プロセスモニタの一時領域については、「[8.8 Tomcat プロセスモニタの一時領域](#)」を参照してください。

2. (Tomcat サーバプロセス稼働時) 実行するコマンドおよびその出力先を確認する。

Tomcat サーバプロセスが稼働しているときは、次の表に記載されているコマンドを実行します。コマンドおよびその出力先を確認してください。

## 日立 JavaVM を使用している場合

「表 11-8 モニタリング情報の取得時のコマンド、出力先、およびその説明」

「表 11-10 kill -3 <pid> コマンド実行時の出力先とその説明（日立 JavaVM を使用する場合）」

「表 11-13 ホストマシン情報の取得時のコマンド、出力先、およびその説明」

## 他社製 JavaVM を使用している場合

「表 11-8 モニタリング情報の取得時のコマンド、出力先、およびその説明」

「表 11-11 jcmd コマンド実行時の出力先とその説明（他社製 JavaVM を使用する場合）」

「表 11-13 ホストマシン情報の取得時のコマンド、出力先、およびその説明」

### 3. (Tomcat サーバプロセス非稼働時) 実行するコマンドおよびその出力先を確認する。

Tomcat サーバプロセスが稼働していないときは、次の表に記載されているコマンドを実行します。コマンドおよびその出力先を確認してください。

「表 11-13 ホストマシン情報の取得時のコマンド、出力先、およびその説明」

### 4. 手順 2 または 3 で確認したコマンドを次の形式で実行する。出力ファイル名は、手順 2 または 3 で確認した出力先と一致させてください。

```
<実行するコマンド> 1><コマンド実行結果格納ディレクトリ>/<出力ファイル名> 2>&1
```

### 5. 標準出力および標準エラー出力の内容をリダイレクトして、ファイルに保存する。

### 6. 次のコマンドを実行し、Tomcat の起動に使用している Java 環境のバージョンを取得する。

```
<Tomcatの起動に使用しているjavaコマンド> -version 1><コマンド実行結果格納ディレクトリ>/java_version.txt 2>&1
```

## (2) ファイルの取得

### 1. コピーするファイルを次の表で確認する。

- 「表 11-3 スナップショットログ収集機能の収集対象のパス」
- 「表 11-4 スナップショットログ収集機能のその他の収集対象」
- 「表 11-5 スナップショットログ収集機能で日立 JavaVM を利用する場合だけ収集される情報」

### 2. 次のコマンドを実行して、手順 1 で確認したファイルをカレントディレクトリにコピーする。そのとき、コピー元のパスが分かるようにしてください。

```
cp -r --parents <ディレクトリまたはファイルパス> .
```

### 3. アプリケーションのデプロイメント・ディスクリプタが war ファイル内だけに格納されている場合、その war ファイルだけを解凍し、コピーします。

コピー先は、次のとおりです。

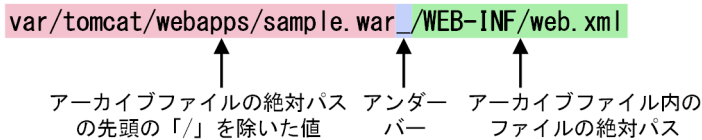
```
<アーカイブファイルの絶対パスの先頭の「/」を除いた値>_/<アーカイブファイル内のファイルの絶対パス>
```

war ファイルを解凍し、コピーする例を次に示します。

### 図 11-1 war ファイルを解凍・コピーする例

例) /var/tomcat/webapps/sample.war を解凍し、コピーする場合

- ① /var/tomcat/webapps/sample.war (アーカイブファイルの絶対パス) を解凍する。  
→ /WEB-INF/web.xml (アーカイブファイル内のファイルの絶対パス) が得られる。
- ② 下記にコピーする。



4. 「表 11-6 収集対象外のパス」に記載されているファイルを削除する。

5. 次に記載されているファイル中の機密情報をマスキングする。

- 「表 11-14 マスキングルールの適用対象」
- 「11.4.2 config.properties (本製品の設定ファイル) のマスキング」

## (3) zip ファイルの作成

次のコマンドを実行して、取得したファイルをアーカイブしてください。

```
zip -r snapshot_$(date +%Y-%m-%d_%H-%M-%S)_manual.zip .
```

## 11.6.3 スナップショットログの多重実行に関する注意事項

本製品では、Tomcat サーバプロセスのパフォーマンスの劣化を防止するため、スナップショットログ収集中は、新規のスナップショットログ収集要求を受け付けません。

ただし、障害検知後に時間的猶予がない場合もあるため、障害検知時には既存の処理と並行して収集します。障害検知後に時間的猶予がない場合の例として、オーケストレーションツール管理下のコンテナ環境などで、コンテナが強制削除されるなどが挙げられます。

スナップショットログ多重実行時の既存の処理と新規の処理の動作を次の表に示します。

表 11-18 スナップショットログ多重実行時の既存の処理と新規の処理の動作

既存の処理	新規の処理	多重実行時の既存の処理の動作	多重実行時の新規の処理の動作
<ul style="list-style-type: none"> <li>• スナップショットログ収集コマンド</li> <li>• スナップショットログ収集 REST API</li> </ul>	スナップショットログ収集コマンド	実行中の処理を継続する。	収集処理を開始しない。
	スナップショットログ収集 REST API		

既存の処理	新規の処理	多重実行時の 既存の処理の動作	多重実行時の 新規の処理の動作
<ul style="list-style-type: none"> <li>障害検知</li> </ul>	障害検知		収集処理を開始する。

## 11.6.4 手動収集のユースケース

スナップショットログの手動収集のユースケースについて説明します。

### (1) Tomcat サーバプロセスの通常稼働時に保守情報を収集する場合

Tomcat サーバプロセスが通常どおり稼働しているときも、手動で保守情報（スナップショットログ）を収集できます。これによって、Tomcat サーバプロセスが正常に稼働しているかどうかを確認できます。

Tomcat サーバプロセス稼働時の保守情報を収集したいときは、次の方法で収集できます。

- スナップショットログ収集コマンドを実行して収集する  
収集方法は、「[11.6.1\(1\) スナップショットログ収集コマンドを実行して収集する](#)」を参照してください。
- HTTP リクエストを送信して収集する  
収集方法は、「[11.6.1\(2\) HTTP リクエストを送信して収集する](#)」を参照してください。

次に、収集時の前提条件とセキュリティ対策について説明します。

#### 収集時の前提条件

Tomcat プロセスモニタが起動している必要があります。

#### 収集時のセキュリティ対策

デフォルトでは、セキュリティ対策の観点からローカルホスト上のスナップショットログだけを収集できます。リモートマシンから収集するためには、`monitor.rest.bindaddress` に適した値を設定してください。`monitor.rest.bindaddress` については、「[12.2.4\(2\) Tomcat プロセスモニタに関するプロパティ](#)」を参照してください。

あわせてファイアウォールなどを使用し、次の対策を実施してください。

- アクセス元を制限する
- 通信経路の安全性を確保する



## 11.7 スナップショットログの出力テスト

---

本番稼働する前にスナップショットログの出力テストを実施してください。目的は、意図したファイルが収集、マスキングされているかどうかを確認することです。手順は次のとおりです。

### 1. スナップショットログを手動で収集する。

スナップショットログを手動で収集する方法については、「[11.6 スナップショットログの手動収集](#)」を参照してください。

### 2. 出力結果を次の観点で確認する。

- 必要なファイルが収集対象に含まれているか
- 除外したいファイルが収集されていないか
- 正しくマスキングされているか

## 11.8 ユースケース別の設定（自動収集・手動収集共通）

自動収集および手動収集に共通する、ユースケース別の設定方法について説明します。これらは、`config.properties`（本製品の設定ファイル）に設定します。`config.properties`（本製品の設定ファイル）については、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

### 11.8.1 収集対象を変更したい場合

#### (1) 特定のファイルおよび特定のディレクトリを収集対象外にする

特定のファイルおよび特定のディレクトリをスナップショットログの収集対象から除外する場合、`config.properties`（本製品の設定ファイル）の次のプロパティに除外するファイルパターン（glob形式で指定したファイル名）を指定してください。<n>には、自然数（1以上の整数）を指定します。

```
snapshot.exclude.globs.<n>=<除外するファイルパターン>
```

なお、プロパティを複数定義することで、複数の除外ファイルパターンを定義できます。

#### (2) スナップショットログの収集対象を追加する

特定のファイルおよび特定のディレクトリをスナップショットログの収集対象に追加する場合、`config.properties`（本製品の設定ファイル）の次のプロパティに追加するファイルのパスを指定してください。<n>には、自然数（1以上の整数）を指定します。

```
snapshot.include.paths.<n>=<追加収集するファイルパス>
```

なお、プロパティを複数定義することで、複数の収集対象パスを定義できます。

### 11.8.2 機密情報のマスキングルールを追加したい場合

機密情報のマスキングルールを追加したい場合、`config.properties`（本製品の設定ファイル）の次のプロパティを指定してください。

```
snapshot.maskingrule.regexes.<n>=<正規表現文字列>
```

指定時は、正規表現を用いてください。正規表現に合致した文字列の最初のグループが「\*\*\*\*\*」に置換されます。また、<n>には、自然数（1以上の整数）を指定します。

設定後、システムの稼働前にスナップショットログ収集コマンドを実行し、正しくマスキングされていることを確認してください。スナップショットログ収集コマンドの詳細は、「[16.2 スナップショットログ収集コマンド](#)」を参照してください。

### 11.8.3 Context 要素の altDDName を設定している場合

context.xml (Tomcat のコンテキスト設定ファイル) などの Context 要素の altDDName を指定している場合、スナップショットログにアプリケーションのデプロイメント・ディスクリプタ情報を含める必要があります。そのため、config.properties (本製品の設定ファイル) の次のプロパティを追加してください。<n>には、自然数 (1 以上の整数) を指定します。

```
snapshot.include.paths.<n>=<altDDNameのパス>
```

### 11.8.4 monitor.tomcat.change.work.directory.enabled を false に変更している場合

障害の原因の解析に必要なコアダンプや JavaVM のエラーレポートなどの一部の保守情報は、カレントディレクトリに出力されます。monitor.tomcat.change.work.directory.enabled を false に設定する場合、スナップショットログにこれらのログが出力されるように設定する必要があります。そのため、config.properties (本製品の設定ファイル) の次のプロパティを追加してください。<n>には、自然数 (1 以上の整数) を指定します。

```
snapshot.include.paths.<n>=<追加収集パス>
```

設定例は、次のとおりです。

```
snapshot.include.paths.1=/tmp/core  
snapshot.include.paths.2=/tmp/hs_err.log
```

### 11.8.5 スレッドダンプの出力先を変更したい場合 (日立 JavaVM 使用時)

日立 JavaVM を利用する場合、次のどちらかの方法でスレッドダンプの出力先を変更できます。

- 環境変数 JAVACOREDİR に出力先のディレクトリを設定する
- config.properties (本製品の設定ファイル) の common.java.hitachi.javacoredir に出力先のディレクトリを設定する

環境変数 JAVACOREDİR が定義されている場合、common.java.hitachi.javacoredir の設定値は使用されません。また、上記のどちらも定義されていない場合は、common.java.hitachi.javacoredir のデフォルト値が出力先のディレクトリになります。

common.java.hitachi.javacoredir は、config.properties (本製品の設定ファイル) プロパティです。詳細は、「12.2.4(1) 本製品全体に関するプロパティ」を参照してください。

## 11.8.6 ログ出力先を<common.base の指定値>以外に設定している場合

`#{CATALINA_BASE}/conf/logging.properties` を変更するなどして<common.base の指定値>以外にログを出力する場合、障害の原因の解析に必要なログを収集できるように、`config.properties`（本製品の設定ファイル）の次のプロパティを追加してください。<n>には、自然数（1以上の整数）を指定します。

```
snapshot.include.paths.<n>=<追加収集パス>
```

`common.base`、および `snapshot.include.paths.<n>` は、`config.properties`（本製品の設定ファイル）のプロパティです。`common.base` については「[12.2.4\(1\) 本製品全体に関するプロパティ](#)」を、`snapshot.include.paths.<n>` については「[12.2.4\(4\) スナップショットログ収集機能に関するプロパティ](#)」を参照してください。

### メモ

`#{CATALINA_BASE}`の値は、Tomcat プロセスモニタの起動時に決定されます。

## 11.8.7 同一環境で複数の Tomcat プロセスモニタを動作させる場合

`config.properties`（本製品の設定ファイル）の `snapshot.log.filepath` を設定している場合、スナップショットログの出力先が競合しないようにする必要があります。`snapshot.log.filepath` には、Tomcat プロセスモニタごとに一意な値を設定してください。

`snapshot.log.filepath` は、`config.properties`（本製品の設定ファイル）のプロパティです。詳細は、「[12.2.4\(4\) スナップショットログ収集機能に関するプロパティ](#)」を参照してください。

## 11.9 ユースケース別の設定（自動収集）

自動収集に関する、ユースケース別の設定方法について説明します。目的に応じて、`config.properties`（本製品の設定ファイル）のプロパティの値を変更してください。`config.properties`（本製品の設定ファイル）については、「[12.2 config.properties（本製品の設定ファイル）](#)」を参照してください。

### 11.9.1 スナップショットログの出力先を変更したい場合

スナップショットログをログディレクトリと異なるディレクトリに出力する場合、`config.properties`（本製品の設定ファイル）の次のプロパティを変更してください。

```
snapshot.log.filepath=<変更先ファイルプリフィックス>
```

### 11.9.2 Tomcat サーバプロセスの正常終了時にもスナップショットログを出力したい場合

通常は、次のどちらでもない場合にスナップショットログが出力されます。

- Tomcat サーバプロセスの終了ステータスが 0 の場合
- Tomcat サーバプロセスの終了ステータスが 143 の場合（SIGTERM による終了）

終了ステータスによって、スナップショットログの出力条件を変更できます。`config.properties`（本製品の設定ファイル）の次のプロパティ値を「ALWAYS」に指定することで、Tomcat サーバプロセスが正常終了したとき\*も、スナップショットログが出力されます。

```
snapshot.onshutdownrequest.collect.condition=[ALWAYS|ERROR|NONZERO]
```

注※

Tomcat サーバプロセスの終了ステータスが 0 の場合を指します。

### 11.9.3 Tomcat サーバプロセスの停止要求時、停止する前に Tomcat 稼働中情報を取得したい場合

Tomcat プロセスモニタが Tomcat サーバプロセスの停止要求を検知し、Tomcat サーバプロセスが停止する前に Tomcat 稼働中情報を取得できます。`config.properties`（本製品の設定ファイル）のプロパティで設定できます。デフォルトでは、取得されません。

Tomcat サーバプロセスの停止要求時に、Tomcat 稼働中情報を取得するための設定方法を次に示します。

## モニタリング情報の取得を有効にする設定

Tomcat サーバプロセスの停止要求時にモニタリング情報を取得したい場合は、config.properties（本製品の設定ファイル）の次のプロパティに true を設定してください。

```
snapshot.onshutdownrequest.watchcommand.enabled=[true|false]
```

## スレッドダンプ情報の取得回数の設定

Tomcat サーバプロセスの停止要求時にスレッドダンプ情報を取得する回数は、次のプロパティに設定してください。

```
snapshot.onshutdownrequest.threaddumpnum=<取得回数>
```

### ❗ 重要

これらのプロパティを設定すると、Tomcat サーバプロセスが停止するまでに時間が掛かります。

また、設定しても Tomcat サーバプロセスの終了方式によっては、Tomcat 稼働中情報を取得できない場合があります。Tomcat 稼働中情報の詳細については、「[11.5.1\(2\) Tomcat 稼働中情報の取得](#)」を参照してください。

## 11.9.4 稼働監視で障害を検知したときの Tomcat 稼働中情報取得時の条件をカスタマイズしたい場合

稼働監視で障害を検知したときに、Tomcat 稼働中情報が取得されます。この場合の取得時の条件をカスタマイズできます。

通常、本製品では次のとおりに Tomcat 稼働中情報が取得されます。

- モニタリング情報が取得される。  
モニタリング情報の詳細は、「[11.3.2\(1\)\(a\) モニタリング情報](#)」を参照してください。
- スレッドダンプ情報が 1000 ミリ秒間隔で 3 回取得される。  
スレッドダンプ情報の詳細は、「[11.3.2\(1\)\(b\) スレッドダンプ情報](#)」を参照してください。

これらの取得時の条件をカスタマイズする方法を次に示します。

### モニタリング情報の取得有無の変更

モニタリング情報を取得するかどうかを config.properties（本製品の設定ファイル）の次のプロパティで変更できます。

```
snapshot.onhealthcheck.watchcommand.enabled=[true|false]
```

### スレッドダンプ情報の取得回数の変更

スレッドダンプ情報の取得回数を変更したい場合は、config.properties（本製品の設定ファイル）の次のプロパティで設定を変更できます。

```
snapshot.onhealthcheck.threaddumpnum=<取得回数>
```

### スレッドダンプ情報の取得間隔の変更

スレッドダンプ情報の取得間隔を変更したい場合は、`config.properties`（本製品の設定ファイル）次のプロパティで設定を変更できます。

```
snapshot.default.threaddumpinterval=<取得間隔ミリ秒>
```

#### **!** 重要

スレッドダンプの取得間隔を変更すると、Tomcat サーバプロセスの終了要求時の、スナップショットログ収集 REST API 呼び出し時の取得間隔にもその変更が反映されます。

## 11.10 ユースケース別の設定 (手動収集)

手動収集に関する、ユースケース別の設定方法について説明します。目的に応じて、`config.properties` (本製品の設定ファイル) のプロパティの値を変更してください。`config.properties` (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

### 11.10.1 オプションの指定を省略したときのデフォルト値を変更したい場合

スナップショットログ収集コマンド、またはスナップショットログ収集 REST API を使ってスナップショットログを収集する場合、次の情報を収集できます。

- モニタリング情報  
モニタリング情報の詳細は、「[11.3.2\(1\)\(a\) モニタリング情報](#)」を参照してください。
- スレッドダンプ情報  
スレッドダンプ情報の詳細は、「[11.3.2\(1\)\(b\) スレッドダンプ情報](#)」を参照してください。

また、オプションの指定を省略したときの値 (デフォルト値) を変更できます。通常、オプションの指定を省略したときはモニタリング情報は取得されますが、スレッドダンプ情報は取得されません。ただし、次の設定でオプションを指定しなかった場合の動作を変更できます。

#### デフォルトの取得回数の変更

スナップショットログ収集 REST API の呼び出し時に、スレッドダンプの取得回数のオプションを指定しなかった場合の取得回数を次のプロパティで指定します。

```
snapshot.rest.default.threaddumpnum=<取得回数>
```

#### モニタリング情報をデフォルトで取得するかどうかの変更

スナップショットログ収集 REST API の呼び出し時に、モニタリング情報の取得オプションを指定しなかった場合の、マシンリソースの使用情報の取得有無を指定します。取得しない場合は、「`false`」を指定してください。

```
snapshot.rest.default.watchcommand.enabled=[true|false]
```

スナップショットログ収集コマンドのオプションの指定については、「[16.2 スナップショットログ収集コマンド](#)」を参照してください。また、スナップショットログ収集 REST API のオプションの指定については、「[17.2 スナップショットログ収集 REST API](#)」を参照してください。



# 12

## 定義ファイル

この章では、本製品で使用する定義ファイルについて説明します。

## 12.1 定義ファイルの種類

本製品で使用する定義ファイルの一覧を次の表に示します。Tomcat の設定ファイルは、「3.3 Tomcat に組み込む」の操作手順で使用する順番で説明します。

表 12-1 本製品で使用する定義ファイルの一覧

ファイル名	分類	概要	参照先
config.properties	本製品の設定ファイル	本製品の各種機能の設定値を指定します。ファイルの内容を変更しない場合は、デフォルト値が適用されます。	12.2
setenv.sh	Tomcat 起動時の環境変数定義ファイル	Tomcat 起動時の環境変数を指定します。このファイルは必ず作成してください。	12.3
catalina.properties	Tomcat のプロパティ定義ファイル	本製品から提供されるライブラリ格納先を common.loader プロパティに追加します。このファイルは、環境に応じて必ず設定を変更してください。	12.4
server.xml	Tomcat のサーバ設定ファイル	本製品から提供される Listener や Valveなどを追加します。このファイルは、環境に応じて必ず設定を変更してください。	12.5
context.xml	Tomcat のコンテキスト設定ファイル	本製品から提供される Valve や jdbcInterceptorなどを追加します。このファイルは、環境に応じて必ず設定を変更してください。	12.6

各ファイルは、システムを起動する前に作成しておく必要があります。起動後に内容を変更した場合は、システムを停止したあとで、再起動してください。

## 12.2 config.properties (本製品の設定ファイル)

config.properties (本製品の設定ファイル) では、本製品の各種機能に対する設定をカスタマイズできます。

### 12.2.1 形式

Java SE のプロパティファイル形式です。

次のようにキーを指定します。

```
<キー名称>=<値>
```

プロパティファイル内で非 ASCII 文字を記述する場合は、UTF-8 (BOM なし) でエンコードする必要があります。

キー名称には、可変値が含まれる場合があります。また、値には、変数を含めることができます。プロパティキーに含まれる可変値およびプロパティ値の変数展開について次に示します。

#### (1) プロパティキーに含まれる可変値

各プロパティキーには、次の可変値が含まれる場合があります。可変値について次に示します。

<n>

複数の値を設定できるプロパティキーの末尾にある可変値です。プロパティを設定する場合は、<n>の部分を一意な自然数 (1 以上の整数) に置き換えてください。

<group-id>

複数のプロパティをグループとして定義 (グループ化) するための値で、プロパティキー中にある可変値です。プロパティを設定する場合は、<group-id>の部分を任意の文字列 (1 文字以上) に置き換えてください。

次の点を満たすプロパティが 1 つのグループとして扱われます。

- 先頭から<group-id>までの文字列が同じプロパティ
- 置き換えた<group-id>が一致している、複数のプロパティ

なお、<group-id>を置き換える文字列には、次の文字を使用できません。

- . (0x2e)
- : (0x3a)
- = (0x3d)

## (2) プロパティ値の変数展開について

プロパティの値に\${XXX}形式を含めた場合は、変数として値が置換されます。XXXの部分に指定できる文字列は次のどれかです。

- catalina.base
- catalina.home
- <任意の環境変数名>
- <プロパティキー名>

次に、変数を指定した場合の詳細を説明します。

- 指定した環境変数が存在しない場合、または、指定したプロパティが定義されていない場合はエラーとなります。
- 指定した環境変数と同名のプロパティが存在した場合はエラーとなります。
- \${XXX}形式のネストは置換できません。「\${」から、最初に出現した「}」までの文字列を、XXXの部分として扱います。
- プロパティキーへの指定は無効です。指定した場合、変数展開しないで文字列として扱います。
- 展開先の文字列に\${XXX}の形式を含めることはできません。指定した場合、そのままの文字列となります。
- OSの仕様に関係なく、大文字・小文字を区別します。
- 変数「\${catalina.base}」はTomcatのベースディレクトリとして展開できます。  
\${catalina.base}の値は、Tomcatによって次のように決定されます。
  1. 環境変数 CATALINA\_BASE が設定済みの場合：環境変数 CATALINA\_BASE の値
  2. 1.以外で、環境変数 CATALINA\_HOME が設定済みの場合：環境変数 CATALINA\_HOME の値
  3. 1.および 2.以外の場合：Tomcat が提供する catalina.sh のディレクトリの親ディレクトリパス
- 変数「\${catalina.home}」はTomcatのホームディレクトリとして展開できます。  
\${catalina.home}の値は、Tomcatによって次のように決定されます。
  1. 環境変数 CATALINA\_HOME が設定済みの場合：環境変数 CATALINA\_HOME の値
  2. 1.以外の場合：Tomcat が提供する catalina.sh のディレクトリの親ディレクトリパス

### ポイント

環境変数に変数展開できる機能を利用することで、config.properties（本製品の設定ファイル）を直接編集しなくても起動時に設定値を決定できます。また、同一マシン・同一ホスト上に複数のTomcatインスタンスがある場合に、config.properties（本製品の設定ファイル）を1つに集約できます。例えば、config.properties（本製品の設定ファイル）に次を定義した場合、monitor.rest.portはTomcat起動時の環境変数MY\_MONITOR\_PORTの値で動作します。

```
monitor.rest.port=${MY_MONITOR_PORT}
```

変数は、環境変数である必要があります。bash の場合、次に示すとおり、先頭に export を付与してください。

```
export MY_MONITOR_PORT=28888
```

## 12.2.2 ファイルの格納先, および格納先の変更方法

デフォルトの設定の場合の格納先, および格納先の変更方法を説明します。

### (1) ファイルの格納先 (デフォルトの設定)

デフォルトの設定の場合は、次の config.properties (本製品の設定ファイル) が適用されます。

```
<本製品のインストールディレクトリ>/conf/config.properties
```

### (2) ファイルの格納先の変更方法

config.properties (本製品の設定ファイル) の格納先は、環境変数 PROCESS\_MONITOR\_CONFIG\_PATH で変更できます。次に、変更する手順を説明します。

1. config.properties (本製品の設定ファイル) のテンプレートをコピーして、任意のディレクトリに格納する。

テンプレートは次のファイルを使用してください。

```
<本製品のインストールディレクトリ>/template/config.properties
```

2. 手順 1 でコピーしたファイルの絶対パスを、環境変数 PROCESS\_MONITOR\_CONFIG\_PATH に指定する。

#### ヒント

同一マシン, または, 同一ホスト上に複数の Tomcat インスタンスを作成する場合, インスタンス間で一意とする必要があるプロパティが存在するため, 次のどちらかを実施する必要があります。

- config.properties (本製品の設定ファイル) を Tomcat インスタンスごとに作成する
- config.properties (本製品の設定ファイル) を環境変数で動的に展開できるように編集し, Tomcat サーバプロセス起動時に環境変数を設定する

## 12.2.3 機能

本製品の各種機能に対する設定をカスタマイズできます。

Tomcat プロセスモニタおよび Tomcat サーバプロセスの稼働中にこのファイルの内容を変更した場合、変更した内容は次に Tomcat プロセスモニタを起動したときに反映されます。

また、次の場合は、バリデーションエラーになり、Tomcat プロセスモニタの起動に失敗します。

- 指定可能値以外の値を指定した場合
- プロパティ値の変数展開でエラーとなった場合

## 12.2.4 指定可能なプロパティ

各機能で指定可能なプロパティについて説明します。

### (1) 本製品全体に関するプロパティ

本製品全体に関するプロパティを次に示します。

表 12-2 本製品全体に関するプロパティ

キー名称	内容	指定可能値	省略可否	デフォルト
common.base	ログ出力先のデフォルトのルートディレクトリ、および Tomcat プロセスモニタや Tomcat サーバプロセスの作業ディレクトリとして利用します。 複数の Tomcat プロセスモニタのプロセスを同一環境で動かす場合は、プロセスごとに一意となる値を指定してください。	絶対パス	可	\$ {catalina.base} /logs
common.java.hitachi.javacoredir	日立 JavaVM を使用する場合、スレッドダンプの出力先ディレクトリを指定します。環境変数 JAVACOREDIR が指定されていた場合、このプロパティの値は使用されません。	絶対パス	可	\$ {common.base} /threaddump

### (2) Tomcat プロセスモニタに関するプロパティ

Tomcat プロセスモニタに関するプロパティを次に示します。

表 12-3 Tomcat プロセスモニタに関するプロパティ

キー名称	内容	指定可能値	省略可否	デフォルト
monitor.log.message.filepath	メッセージログファイルの出力先ディレクトリとファイルプリフィックスを指定します。出力先のファイルパスは <monitor.log.message.filepath の指定	絶対パス	可	\$ {common.base} /ucarmessage

キー名称	内容	指定可能値	省略可否	デフォルト
	値<ローテーション番号>.log になります。 複数の Tomcat プロセスモニタのプロセスを同一環境で動かす場合は、プロセスごとに一意となる値を指定してください。			
monitor.log.message.filenum	メッセージログファイルのローテーション数を指定します。	1~16の整数で指定します。	可	4
monitor.log.message.filesize	メッセージログの1ファイル当たり書き込むおおよその最大量(単位:バイト)を示す数値を指定します。「おおよそ」とは、行の途中でファイルをローテーションさせないため、超過することがあることを意味します。	1~2147483647の整数で指定します。	可	33554432
monitor.log.message.level	メッセージログのログ取得レベル(SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALLのどれか)を示す文字列を指定します。各レベルの重要度はjava.util.logging.Levelに定義されているとおりです。詳細は、「 <a href="#">12.2.5 ログ取得レベル</a> 」を参照してください。	次のどれかを指定します。 <ul style="list-style-type: none"><li>• SEVERE</li><li>• WARNING</li><li>• INFO</li><li>• CONFIG</li><li>• FINE</li><li>• FINER</li><li>• FINEST</li><li>• ALL</li></ul>	可	INFO
monitor.log.maintenance.filepath	保守ログファイルの出力先ディレクトリとファイルプリフィクスを指定します。出力先のファイルパスは<monitor.log.maintenance.filepathの指定値><ローテーション番号>.log になります。 複数の Tomcat プロセスモニタのプロセスを同一環境で動かす場合は、プロセスごとに一意となる値を指定してください。	絶対パス	可	\${common.base}/ucarmaintenance
monitor.log.maintenance.filenum	保守ログファイルのローテーション数を指定します。	1~16の整数で指定します。	可	4
monitor.log.maintenance.filesize	保守ログの1ファイル当たり書き込むおおよその最大量(単位:バイト)を示す数値を指定します。「おおよそ」とは、行の途中でファイルをローテーションさせないため、超過することがあることを意味します。	1~2147483647の整数で指定します。	可	33554432

キー名称	内容	指定可能値	省略可否	デフォルト
monitor.log.maintenance.level	<p>保守ログのログ取得レベル (SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL のどれか) を示す文字列を指定します。各レベルの重要度は java.util.logging.Level に定義されているとおりです。</p> <p>詳細は、「<a href="#">12.2.5 ログ取得レベル</a>」を参照してください。</p>	<p>次のどれかを指定します。</p> <ul style="list-style-type: none"> <li>• SEVERE</li> <li>• WARNING</li> <li>• INFO</li> <li>• CONFIG</li> <li>• FINE</li> <li>• FINER</li> <li>• FINEST</li> <li>• ALL</li> </ul>	可	INFO
monitor.jvm.options	<p>Tomcat プロセスモニタの Java プロセスに指定する JavaVM オプションを指定します。</p> <p>複数のオプションを指定する場合、空白区切りで指定してください。</p> <p>自動的に設定される JavaVM オプション (「<a href="#">13.2.3 JavaVM ログ</a>」に記載があるもの) を指定した場合は、デフォルト値ではなく、このプロパティでの指定値が適用されます。</p>	任意文字列	可	-
monitor.tomcat.change.work.directory.enabled	<p>Tomcat の作業ディレクトリの変更可否を指定します。</p> <p>true を指定した場合： common.base の指定値を作業ディレクトリとします。</p> <p>false を指定した場合： Tomcat プロセスモニタ起動時のカレントディレクトリを作業ディレクトリとします。</p> <p>注 false に変更する場合、障害時に JavaVM が出力するログファイルが収集できなくなるおそれがあります。false に変更する場合は、次のプロパティにカレントディレクトリを追加してください。</p> <ul style="list-style-type: none"> <li>• snapshot.include.paths.&lt;n&gt;</li> </ul>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	true
monitor.tomcat.forcestop.timeout	<p>異常検知時に Tomcat サーバプロセスを終了する場合の、プロセス終了待ちタイムアウト時間 (単位：ミリ秒) を指定します。</p>	1~3600000の整数で指定します。	可	10000



キー名称	内容	指定可能値	省略可否	デフォルト
monitor.rest.port	Tomcat プロセスモニタの HTTP 機能の受付ポート番号を指定します。	1~65535 の整数で指定します。	可	28081
monitor.rest.bindaddress	Tomcat プロセスモニタの HTTP 機能に割り当てる IP アドレスを指定します。 リモートマシンから接続する場合に指定してください。その場合、ファイアウォールなどでアクセス元を制限したり、通信経路の安全性を確保したりするように注意してください。 localhost にループバックアドレス以外の IP アドレスを割り当てている場合、ループバックアドレスの値を設定してください。	任意文字列	可	localhost

(凡例)

- : なし

### (3) 稼働監視機能に関するプロパティ

稼働監視機能に関するプロパティを次に示します。

表 12-4 稼働監視機能に関するプロパティ

キー名称	内容	指定可能値	省略可否	デフォルト
healthcheck.rest.connecttimeout	Tomcat サーバプロセスから稼働監視コンポーネントへの HTTP 通信の接続タイムアウト時間 (単位: ミリ秒) を指定します。0 を指定した場合はタイムアウトしません。	0~ 2147483647 の整数で指定します。	可	3000
healthcheck.rest.readtimeout	Tomcat サーバプロセスから稼働監視コンポーネントへの HTTP 通信の読み込みタイムアウト時間 (単位: ミリ秒) を指定します。0 を指定した場合はタイムアウトしません。	0~ 2147483647 の整数で指定します。	可	10000
healthcheck.heartbeat.interval	Tomcat サーバプロセスから稼働監視コンポーネントへ送信するハートビートの送信間隔 (単位: ミリ秒) を指定します。	1~ 2147483647 の整数で指定します。	可	10000
healthcheck.heartbeatdelay.enabled	稼働監視コンポーネントが Tomcat サーバプロセスからのハートビートを監視するかどうかを指定します。	次のどちらかを指定します。 • true • false	可	true

キー名称	内容	指定可能値	省略可否	デフォルト
	<p>true を指定した場合：</p> <p>稼働監視コンポーネントで受信する Tomcat サーバプロセスからのハートビートを監視します。</p> <p>false を指定した場合：</p> <p>稼働監視コンポーネントで受信する Tomcat サーバプロセスからのハートビートを監視しません。</p>			
healthcheck.heartbeatdelay.timeout	<p>ハートビート待ちタイムアウトを指定します。</p> <p>稼働監視コンポーネントが Tomcat サーバプロセスからのハートビートを受信してから次のハートビートを受信するまで待つ時間（単位：ミリ秒）を指定します。</p>	0～2147483647の整数で指定します。	可	60000
healthcheck.heartbeatdelay.actions.failure.usercommand.idrefs.<n>	<p>稼働監視コンポーネントでハートビート待ちタイムアウトが発生した場合に実行するユーザコマンド定義の ID (&lt;group-id&gt;) を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.&lt;group-id&gt;.command キーの欄を参照してください。</p>	healthcheck.usercommand.defs.<group-id>	可	-
healthcheck.heartbeatdelay.actions.failure.snapshot	<p>稼働監視コンポーネントでハートビート待ちタイムアウトが発生した場合に、スナップショットログを収集するかどうかを指定します。</p> <p>true を指定した場合：</p> <p>スナップショットログを収集します。</p> <p>false を指定した場合：</p> <p>スナップショットログを収集しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	true
healthcheck.heartbeatdelay.actions.failure.terminate	<p>稼働監視コンポーネントでハートビート待ちタイムアウトが発生した場合に、Tomcat サーバプロセスを停止するかどうかを指定します。</p> <p>true を指定した場合：</p> <p>Tomcat サーバプロセスを停止します。</p> <p>false を指定した場合：</p> <p>Tomcat サーバプロセスを停止しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	false
healthcheck.heartbeatdelay.actions.recovery.usercommand.idrefs.<n>	<p>稼働監視コンポーネントでハートビート待ちタイムアウトが発生したあとに、ハートビートを受信した初回に実行するユーザコマンド定義の ID (&lt;group-id&gt;) を指定</p>	healthcheck.usercommand.defs.<group-id>	可	-

キー名称	内容	指定可能値	省略可否	デフォルト
	します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command キーの欄を参照してください。	定義の ID (<group-id>)		
healthcheck.unchangedlogfile.logfilename.glob	Tomcat サーバプロセス起動監視でログファイル更新チェックに使用するログファイル名を glob 形式で指定します。	glob 形式で指定したファイル名	可	catalina.*.log
healthcheck.initdelay.timeout	Tomcat サーバプロセス初期化完了通知待ちタイムアウトを指定します。 稼働監視コンポーネントを開始してから、Tomcat サーバプロセス初期化完了通知を受信するまでの時間（単位：ミリ秒）を指定します。0 を指定した場合はタイムアウトしません。	0～ 2147483647 の整数で指定します。	可	60000
healthcheck.startdelay.timeout	Tomcat サーバプロセス開始完了通知待ちタイムアウトを指定します。 Tomcat サーバプロセス初期化完了通知を受信してから、Tomcat サーバプロセス開始完了通知を受信するまでの時間（単位：ミリ秒）を指定します。0 を指定した場合はタイムアウトしません。	0～ 2147483647 の整数で指定します。	可	60000
healthcheck.startdelay.actions.failure.usercommand.idrefs.<n>	Tomcat サーバプロセス開始完了通知待ちタイムアウトが発生した場合に実行するユーザコマンド定義の ID (<group-id>) を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command キーの欄を参照してください。	healthcheck.usercommand.defs で始まるユーザコマンド定義の ID (<group-id>)	可	-
healthcheck.startdelay.actions.failure.retrymax	Tomcat サーバプロセス開始完了通知待ちタイムアウトが発生した場合に、Tomcat サーバプロセス開始完了待ちをリトライする最大回数を指定します。	0～ 2147483647 の整数で指定します。	可	4
healthcheck.startdelay.actions.failure.snapshot	Tomcat サーバプロセス開始完了待ちタイムアウトが発生した場合に、スナップショットログを収集するかどうかを指定します。  true を指定した場合： スナップショットログを収集します。  false を指定した場合： スナップショットログを収集しません。	次のどちらかを指定します。 <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	true

キー名称	内容	指定可能値	省略可否	デフォルト
healthcheck.startdelay.actions.afterretry.terminate	Tomcat サーバプロセス開始完了待ちタイムアウトが発生した場合に、タイムアウトの回数が healthcheck.startdelay.actions.failure.retrymax で指定した最大リトライ回数を超えていたとき、Tomcat サーバプロセスを停止するかどうかを指定します。  true を指定した場合： Tomcat サーバプロセスを停止します。  false を指定した場合： Tomcat サーバプロセスを停止しません。	次のどちらかを指定します。 <ul style="list-style-type: none"><li>• true</li><li>• false</li></ul>	可	true
healthcheck.httprequest.enabled	稼働監視コンポーネントからヘルスチェックリクエストを送信するかどうかを指定します。  true を指定した場合： 稼働監視コンポーネントからヘルスチェックリクエストを送信します。  false を指定した場合： 稼働監視コンポーネントからヘルスチェックリクエストを送信しません。	次のどちらかを指定します。 <ul style="list-style-type: none"><li>• true</li><li>• false</li></ul>	可	true
healthcheck.httprequest.check.default.interval	稼働監視コンポーネントから送信するヘルスチェックリクエストの送信間隔（単位：ミリ秒）を指定します。	1～2147483647の整数で指定します。	可	30000
healthcheck.httprequest.check.default.connecttimeout	稼働監視コンポーネントから送信するヘルスチェックリクエストのHTTP 接続タイムアウト時間（単位：ミリ秒）を指定します。	0～2147483647の整数で指定します。	可	3000
healthcheck.httprequest.check.default.readtimeout	稼働監視コンポーネントから送信するヘルスチェックリクエストのHTTP 読み込みタイムアウト時間（単位：ミリ秒）を指定します。	0～2147483647の整数で指定します。	可	10000
healthcheck.httprequest.actions.connectfailure.usercommand.idrefs.<n>	稼働監視コンポーネントから送信するヘルスチェックリクエストのHTTP 接続タイムアウトが発生した場合に実行するユーザコマンド定義の ID (<group-id>) を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command キーの欄を参照してください。	healthcheck.usercommand.defs で始まるユーザコマンド定義の ID (<group-id>)	可	-

キー名称	内容	指定可能値	省略可否	デフォルト
healthcheck.httprequest.actions.connectfailure.snapshot	稼働監視コンポーネントから送信するヘルスチェックリクエストの HTTP 接続タイムアウトが発生した場合に、スナップショットログを収集するかどうかを指定します。  true を指定した場合： スナップショットログを収集します。  false を指定した場合： スナップショットログを収集しません。	次のどちらかを指定します。 <ul style="list-style-type: none"><li>• true</li><li>• false</li></ul>	可	true
healthcheck.httprequest.actions.connectfailure.terminate	稼働監視コンポーネントから送信するヘルスチェックリクエストの HTTP 接続タイムアウトが発生した場合に、Tomcat サーバプロセスを停止するかどうかを指定します。  true を指定した場合： Tomcat サーバプロセスを停止します。  false を指定した場合： Tomcat サーバプロセスを停止しません。	次のどちらかを指定します。 <ul style="list-style-type: none"><li>• true</li><li>• false</li></ul>	可	false
healthcheck.httprequest.actions.iofailure.usercommand.idrefs.<n>	稼働監視コンポーネントから送信するヘルスチェックリクエストの HTTP 読み込みタイムアウトが発生した場合に実行するユーザコマンド定義の ID (<group-id>) を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command キーの欄を参照してください。	healthcheck.usercommand.defs で始まるユーザコマンド定義の ID (<group-id>)	可	-
healthcheck.httprequest.actions.iofailure.snapshot	稼働監視コンポーネントから送信するヘルスチェックリクエストの HTTP 読み込みタイムアウトが発生した場合に、スナップショットログを収集するかどうかを指定します。  true を指定した場合： スナップショットログを収集します。  false を指定した場合： スナップショットログを収集しません。	次のどちらかを指定します。 <ul style="list-style-type: none"><li>• true</li><li>• false</li></ul>	可	true
healthcheck.httprequest.actions.iofailure.terminate	稼働監視コンポーネントから送信するヘルスチェックリクエストの HTTP 読み込みタイムアウトが発生した場合に、Tomcat サーバプロセスを停止するかどうかを指定します。	次のどちらかを指定します。 <ul style="list-style-type: none"><li>• true</li><li>• false</li></ul>	可	false

キー名称	内容	指定可能値	省略可否	デフォルト
	<p>true を指定した場合： Tomcat サーバプロセスを停止します。</p> <p>false を指定した場合： Tomcat サーバプロセスを停止しません。</p>			
healthcheck.httprequest.actions.errorresponse.usercommand.idrefs.<n>	稼働監視コンポーネントから送信するヘルスチェックリクエストの結果、エラーレスポンスを受信した場合に実行するユーザコマンド定義の ID (<group-id>) を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command キーの欄を参照してください。	healthcheck.usercommand.defs で始まるユーザコマンド定義の ID (<group-id>)	可	-
healthcheck.httprequest.actions.errorresponse.snapshot	<p>稼働監視コンポーネントから送信するヘルスチェックリクエストの結果、エラーレスポンスを受信した場合に、スナップショットログを収集するかどうかを指定します。</p> <p>true を指定した場合： スナップショットログを収集します。</p> <p>false を指定した場合： スナップショットログを収集しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	true
healthcheck.httprequest.actions.errorresponse.terminate	<p>稼働監視コンポーネントから送信するヘルスチェックリクエストの結果、エラーレスポンスを受信した場合に、Tomcat サーバプロセスを停止するかどうかを指定します。</p> <p>true を指定した場合： Tomcat サーバプロセスを停止します。</p> <p>false を指定した場合： Tomcat サーバプロセスを停止しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	false
healthcheck.httprequest.actions.recovery.usercommand.idrefs.<n>	稼働監視コンポーネントから送信するヘルスチェックリクエストで異常を検知したあとに、ヘルスチェックリクエストが成功した初回に実行する、ユーザコマンド定義の ID (<group-id>) を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command キーの欄を参照してください。	healthcheck.usercommand.defs で始まるユーザコマンド定義の ID (<group-id>)	可	-
healthcheck.stuckthread.enabled	稼働監視コンポーネントが Tomcat サーバプロセスからのリクエスト処理停滞通知を監視するかどうかを指定します。	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> </ul>	可	false

キー名称	内容	指定可能値	省略可否	デフォルト
	<p>true を指定した場合：</p> <p>稼働監視コンポーネントで受信する Tomcat からのリクエスト処理停滞通知を監視します。</p> <p>false を指定した場合：</p> <p>稼働監視コンポーネントで受信する Tomcat からのリクエスト処理停滞通知を監視しません。</p>	<ul style="list-style-type: none"> <li>• false</li> </ul>		
healthcheck.stuckthread.actions.failure.usercommand.idrefs.<n>	稼働監視コンポーネントが Tomcat サーバプロセスからのリクエスト処理停滞通知を検知した場合に実行する、ユーザコマンド定義の ID (<group-id>) を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command キーの欄を参照してください。	healthcheck.usercommand.defs.<group-id>	可	-
healthcheck.stuckthread.actions.failure.snapshot	<p>稼働監視コンポーネントが Tomcat サーバプロセスからのリクエスト処理停滞通知を検知した場合に、スナップショットログを収集するかどうかを指定します。</p> <p>true を指定した場合：</p> <p>スナップショットログを収集します。</p> <p>false を指定した場合：</p> <p>スナップショットログを収集しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	true
healthcheck.stuckthread.actions.failure.terminate	<p>稼働監視コンポーネントが Tomcat サーバプロセスからのリクエスト処理停滞通知を検知した場合に、Tomcat サーバプロセスを停止するかどうかを指定します。</p> <p>true を指定した場合：</p> <p>Tomcat サーバプロセスを停止します。</p> <p>false を指定した場合：</p> <p>Tomcat サーバプロセスを停止しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	false
healthcheck.stuckthread.actions.recovery.usercommand.idrefs.<n>	稼働監視コンポーネントが Tomcat サーバプロセスからのリクエスト処理停滞通知を検知したあとに、停滞解消通知を受信した際に実行するユーザコマンド定義の ID (<group-id>) を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command キーの欄を参照してください。	healthcheck.usercommand.defs.<group-id>	可	-

キー名称	内容	指定可能値	省略可否	デフォルト
healthcheck.shutdown.enabled	稼働監視コンポーネントが Tomcat サーバプロセスの生存状態を監視するかどうかを指定します。	次のどちらかを指定します。 <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	false
healthcheck.shutdown.actions.failure.condition	Tomcat サーバプロセスが終了したとき、自動的にユーザコマンドを実行する Tomcat サーバプロセスの終了ステータスの条件を選択します。ただし、稼働監視機能を起因とした停止要求の場合、この設定は無視されます。  ALWAYS を指定した場合： 終了ステータスによらず、常にユーザコマンドを実行します。  ERROR を指定した場合： 終了ステータスが 0 でも 143 (SIGTERM による終了) でもないときに、ユーザコマンドを実行します。  NONZERO を指定した場合： 終了ステータスが 0 以外のときに、ユーザコマンドを実行します。	次のどれかを指定します。 <ul style="list-style-type: none"> <li>• ALWAYS</li> <li>• ERROR</li> <li>• NONZERO</li> </ul>	可	ERROR
healthcheck.shutdown.actions.failure.usercommand.idrefs.<n>	稼働監視コンポーネントが Tomcat サーバプロセスの異常終了を検知したあとに、実行するユーザコマンド定義の<group-id>を指定します。ユーザコマンド定義の指定方法は healthcheck.usercommand.defs.<group-id>.command を参照してください。	healthcheck.usercommand.defs.<group-id>	可	-
healthcheck.usercommand.defs.<group-id>.command	ユーザコマンド定義の実行コマンドを指定します。指定したコマンドは healthcheck.usercommand.defs.<group-id>.args.<n>で指定したコマンド引数と共に、java.lang.ProcessBuilder で実行されます。コマンド実行時の作業ディレクトリは common.base で指定したディレクトリです。  healthcheck.usercommand.defs.<group-id>から始まるプロパティを、可変値<group-id>の値でグループ化します。このプロパティは、<group-id>の値ごとに必須です。また、このプロパティキーに含まれる<group-id>の値は「ユーザコマンド定義の ID」になります。	任意文字列	可	-
healthcheck.usercommand.defs.<group-id>.args.<n>	ユーザコマンド定義のコマンド引数を指定します。	任意文字列	可	-



キー名称	内容	指定可能値	省略可否	デフォルト
	healthcheck.usercommand.defs.<group-id>.command で指定した実行コマンドに渡す、コマンド引数を指定します。<n>が連番ではない場合も間を空けず昇順でコマンド引数とします。 同じ<group-id>を持つ healthcheck.usercommand.defs.<group-id>.command も指定する必要があります。			
healthcheck.usercommand.defs.<group-id>.stdout.file.path	ユーザコマンドの標準出力をリダイレクトするファイルパスを絶対パスで指定します。指定がない場合は、Tomcat プロセスモニタの標準出力にリダイレクトします。 同じ<group-id>を持つ healthcheck.usercommand.defs.<group-id>.command も指定する必要があります。	絶対パス	可	-
healthcheck.usercommand.defs.<group-id>.stderr.file.path	ユーザコマンドの標準エラー出力をリダイレクトする絶対パスを指定します。指定がない場合は、Tomcat プロセスモニタの標準エラー出力にリダイレクトします。 同じ<group-id>を持つ healthcheck.usercommand.defs.<group-id>.command も指定する必要があります。	絶対パス	可	-
healthcheck.usercommand.defs.<group-id>.timeout	ユーザコマンドの終了待ちのタイムアウト時間（単位：ミリ秒）を指定します。 同じ<group-id>を持つ healthcheck.usercommand.defs.<group-id>.command も指定する必要があります。	1～3600000の整数で指定します。	可	30000
healthcheck.usercommand.threadpoolsize	ユーザコマンドの実行に使用する、稼働監視コンポーネントが管理するスレッドプールのサイズを指定します。	1～1024の整数で指定します。	可	10

(凡例)

- : なし

## (4) スナップショットログ収集機能に関するプロパティ

スナップショットログ収集機能に関するプロパティを次に示します。

表 12-5 スナップショットログ収集機能に関するプロパティ

キー名称	内容	指定可能値	省略可否	デフォルト
snapshot.log.filepath	<p>スナップショットログの出力先ディレクトリとファイルプリフィックスを指定します。スナップショットログの出力パスは「&lt;snapshot.log.filepath の指定値&gt;_&lt;yyyy-MM-dd_HH-mm-ss.SSS&gt;_&lt;n&gt;.zip」です。</p> <p>注 1 yyyy-MM-dd_HH-mm-ss.SSS は出力要求時刻です。</p> <p>注 2 n は Tomcat プロセスモニタを起動してから何回目の収集要求か示す通番です。</p> <p>注 3 複数の Tomcat プロセスモニタのプロセスを同一環境で動かす場合は、プロセスごとに一意となる値を指定してください。</p>	絶対パス	可	\$ {common.base}/snapshot
snapshot.exclude.globs.<n>	<p>スナップショットログの収集対象外とするファイル名を glob 形式で指定します。このプロパティに該当するファイルは snapshot.include.paths.&lt;n&gt;に含まれていても収集されません。\$ {CATALINA_BASE}/conf/tomcat-users.xml は収集対象外のため、このプロパティによる設定は不要です。このプロパティ設定変更時はスナップショットログを手動で収集し、スナップショットログに意図したファイルが含まれるかどうかを確認してください。</p> <ul style="list-style-type: none"> <li>定義例 snapshot.exclude.globs.1=\$ {catalina.base}/conf/ my_private.key snapshot.exclude.globs.2=\$ {catalina.base}/logs/my_secret.log*</li> </ul>	glob 形式で指定したファイル名	可	-
snapshot.include.paths.<n>	<p>スナップショットログに追加で収集するディレクトリまたはファイルパスを指定します。snapshot.exclude.globs.&lt;n&gt;に含まれるファイルは収集されません。このプロパティ設定変更時はスナップショットログを手動で収集し、スナップショットログに意図したファイルが含まれるかどうかを確認してください。</p>	絶対パス	可	-

キー名称	内容	指定可能値	省略可否	デフォルト
	<ul style="list-style-type: none"> <li>定義例  snapshot.include.paths.1=/etc/  myapp/common-web.xml  snapshot.include.paths.2=/var/log  /myextralogs</li> </ul>			
snapshot.maskingrule.regexes.<n>	<p>スナップショットログに収集する際、マスキングするルールを正規表現で指定します。マッチした最初のグループを「*****」に置き換えます。この設定は次に示す収集対象に適用されます。</p> <ul style="list-style-type: none"> <li>• \${CATALINA_BASE}/bin/setenv.sh</li> <li>• \${CATALINA_BASE}/conf 以下のファイル</li> <li>• アプリケーションに含まれるデプロイメント・ディスクリプタ</li> <li>• env, set コマンド実行結果の環境変数一覧が出力されたファイル</li> <li>• システムプロパティの値一覧が出力されたファイル</li> </ul> <p>このプロパティの定義有無に関係なく password="(.+?)" が適用されます。このプロパティ設定変更時はスナップショットログを手動で収集し、スナップショットログ内のファイルが意図したマスキング結果になることを確認してください。</p> <ul style="list-style-type: none"> <li>• xml ファイル中の特定要素をマスキングする場合の例  &lt;Resource secret="mysecret" /&gt; を  &lt;Resource secret="*****" /&gt; と置き換える場合  snapshot.maskingrule.regexes.1=secret="(.+?)"</li> <li>• properties ファイル中の特定プロパティをマスキングする場合の例  com.example.secret=mysecret を  com.example.secret=***** と置き換える場合  snapshot.maskingrule.regexes.1=^com%.example%.secret=(.+)\$</li> <li>• 特定の環境変数をマスキングする場合の例  MY_PASSWORD=mysecret を  MYPASSWORD=***** と置き換える場合</li> </ul>	正規表現	可	-

キー名称	内容	指定可能値	省略可否	デフォルト
	snapshot.maskingrule.regexes.1= MYPASSWORD=(.+)\$			
snapshot.onshutdownrequest.threaddumpnum	<p>シャットダウン要求時、Tomcat が稼働している場合にスレッドダンプを取得する回数を指定します。取得したスレッドダンプはシャットダウンの結果が snapshot.onshutdownrequest.collect.condition に該当する場合、スナップショットログとして出力されます。</p> <p>Tomcat プロセスモニタが停止シグナルを受ける終了方式の場合、スレッドダンプが取得されます。それ以外の終了方式の場合、Tomcat プロセスモニタが取得を開始する前に Tomcat が終了するため、スレッドダンプは取得されません。例えばシャットダウンポートによる終了が該当します。</p> <p>0 を指定した場合、取得しません。複数回を指定した場合、snapshot.default.threaddumpinterval の間隔で取得します。</p>	0~60 の整数で指定します。	可	0
snapshot.onshutdownrequest.watchcommand.enabled	<p>シャットダウン要求時、Tomcat が稼働している場合にマシンリソースの使用状況を取得するかどうかを指定します。</p> <p>true を指定した場合：</p> <p>マシンリソースの使用状況を取得します。</p> <p>取得した情報はシャットダウンの結果が snapshot.onshutdownrequest.collect.condition に該当する場合、スナップショットログとして出力されます。</p> <p>Tomcat プロセスモニタが停止シグナルを受ける終了方式の場合、マシンリソース情報が取得されます。それ以外の終了方式の場合、Tomcat プロセスモニタが取得を開始する前に Tomcat が終了するため、マシンリソース情報は取得されません。例えばシャットダウンポートによる終了が該当します。</p> <p>false を指定した場合：</p> <p>マシンリソースの使用状況を取得しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	false

キー名称	内容	指定可能値	省略可否	デフォルト
snapshot.onshutdownrequest.collect.condition	<p>Tomcat サーバプロセスの終了を検知したとき、自動的にスナップショットログを収集する Tomcat サーバプロセス終了ステータスの条件を選択します。</p> <p>ALWAYS を指定した場合： 終了ステータスによらず、常にスナップショットログを収集します。</p> <p>ERROR を指定した場合： 終了ステータスが 0 でも 143 (SIGTERM による終了) でもないときに、スナップショットログを収集します。</p> <p>NONZERO を指定した場合： 終了ステータスが 0 以外のときに、スナップショットログを収集します。</p>	<p>次のどれかを指定します。</p> <ul style="list-style-type: none"> <li>• ALWAYS</li> <li>• ERROR</li> <li>• NONZERO</li> </ul>	可	ERROR
snapshot.onhealthcheck.threaddumpnum	<p>稼働監視による障害検知時、Tomcat が稼働している場合にスレッドダンプを取得する回数を指定します。</p> <p>0 を指定した場合、取得しません。複数回を指定した場合、snapshot.default.threaddumpinterval の間隔で取得します。</p>	0~60 の整数で指定します。	可	3
snapshot.onhealthcheck.watchcommand.enabled	<p>稼働監視による障害検知時、Tomcat が稼働している場合にマシンリソースの使用状況を取得するかどうかを指定します。</p> <p>true を指定した場合： マシンリソースの使用状況を取得します。</p> <p>false を指定した場合： マシンリソースの使用状況を取得しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	true
snapshot.rest.default.threaddumpnum	<p>スナップショットログ収集 REST API 呼び出し時、スレッドダンプ取得回数オプションを省略した場合の取得回数を指定します。</p> <p>0 を指定した場合、取得しません。複数回を指定した場合、snapshot.default.threaddumpinterval の間隔で取得します。</p>	0~60 の整数で指定します。	可	0
snapshot.rest.default.watchcommand.enabled	<p>スナップショットログ収集 REST API 呼び出し時、モニタリング情報取得オプションを省略した場合、マシンリソースの使用状況を取得するかどうかを指定します。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	true

キー名称	内容	指定可能値	省略可否	デフォルト
	<p>true を指定した場合： マシンリソースの使用状況を取得します。</p> <p>false を指定した場合： マシンリソースの使用状況を取得しません。</p>			
snapshot.default.threaddumpinterval	複数回スレッドダンプを取得する際、取得を完了してから次の取得を開始するまでの待機時間（単位：ミリ秒）を指定します。0を指定した場合、待機時間なしで連続してスレッドダンプを取得します。	0～60000の整数で指定します。	可	1000

(凡例)

-: なし

## (5) トレース機能に関するプロパティ

トレース機能に関するプロパティを次に示します。

表 12-6 トレース機能に関するプロパティ

キー名称	内容	指定可能値	省略可否	デフォルト
tracer.enabled	<p>トレースを取得するかどうかを示す真偽値を指定します。</p> <p>true を指定した場合： トレースを取得します。</p> <p>false を指定した場合： トレースを取得しません。</p>	<p>次のどちらかを指定します。</p> <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	可	true
tracer.queuesize	トレースの非同期出力で使う待ちキューのサイズ（行単位）を示す数値を指定します。実行待ちキューのサイズに比例して、Java ヒープが消費されます。	1～2147483647の整数で指定します。	可	16384
tracer.log.filepath	出力先ディレクトリとログファイルプリフィックスを指定します。出力先のファイルパスは<tracer.log.filepathの指定値><ローテーション番号>.log になります。	絶対パス	可	\$ {common.base} }/ucartrace
tracer.log.filenum	ログファイルのローテーション数を示す数値を示します。	1～16の整数で指定します。	可	4
tracer.log.filesize	1つのログファイルに書き込むおおよその最大量（単位：バイト）を示す数値を指定します。「おおよそ」とは、行の途中でファ	1～2147483647の整数で指定します。	可	33554432

キー名称	内容	指定可能値	省略可否	デフォルト
	イルをローテーションさせないため、超過することがあることを意味します。			
tracer.log.level	<p>ログ取得レベル (SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL のどれか) を示す文字列を指定します。各レベルの重要度は java.util.logging.Level に定義されているとおりです。</p> <p>詳細は、「<a href="#">12.2.5 ログ取得レベル</a>」を参照してください。</p>	<p>次のどれかを指定します。</p> <ul style="list-style-type: none"> <li>• SEVERE</li> <li>• WARNING</li> <li>• INFO</li> <li>• CONFIG</li> <li>• FINE</li> <li>• FINER</li> <li>• FINEST</li> <li>• ALL</li> </ul>	可	FINE

(凡例)

-: なし

## 12.2.5 ログ取得レベル

指定可能なプロパティには、ログ取得レベルを指定できるプロパティがあります。指定可能なプロパティについては、「[12.2.4 指定可能なプロパティ](#)」を参照してください。

指定できるログ取得レベルについて次に示します。

表 12-7 ログ取得レベルの一覧

ログ取得レベル	意味
ALL	すべてのメッセージのログを取ることを示すログ取得レベルです。
SEVERE	<p>重大な障害を意味するログ取得レベルです。</p> <p>メッセージ・レベル*が「SEVERE」に指定されているメッセージをログファイルに出力します。</p>
WARNING	<p>潜在的な問題を意味するログ取得レベルです。</p> <p>次のメッセージ・レベル*が指定されているメッセージをログファイルに出力します。</p> <ul style="list-style-type: none"> <li>• SEVERE</li> <li>• WARNING</li> </ul>
INFO	<p>メッセージを情報として提供するログ取得レベルです。</p> <p>次のメッセージ・レベル*が指定されているメッセージをログファイルに出力します。</p>

ログ取得レベル	意味
	<ul style="list-style-type: none"> <li>• SEVERE</li> <li>• WARNING</li> <li>• INFO</li> </ul>
CONFIG	<p>静的な構成メッセージのログ取得レベルです。</p> <p>次のメッセージ・レベル※が指定されているメッセージをログファイルに出力します。</p> <ul style="list-style-type: none"> <li>• SEVERE</li> <li>• WARNING</li> <li>• INFO</li> <li>• CONFIG</li> </ul>
FINE	<p>トレース情報を提供するログ取得レベルです。</p> <p>次のメッセージ・レベル※が指定されているメッセージをログファイルに出力します。</p> <ul style="list-style-type: none"> <li>• SEVERE</li> <li>• WARNING</li> <li>• INFO</li> <li>• CONFIG</li> <li>• FINE</li> </ul>
FINER	<p>より詳細なトレース情報を提供するログ取得レベルです。</p> <p>次のメッセージ・レベル※が指定されているメッセージをログファイルに出力します。</p> <ul style="list-style-type: none"> <li>• SEVERE</li> <li>• WARNING</li> <li>• INFO</li> <li>• CONFIG</li> <li>• FINE</li> <li>• FINER</li> </ul>
FINEST	<p>非常に詳細なトレース情報を提供するログ取得レベルです。</p> <p>次のメッセージ・レベル※が指定されているメッセージをログファイルに出力します。</p> <ul style="list-style-type: none"> <li>• SEVERE</li> <li>• WARNING</li> <li>• INFO</li> <li>• CONFIG</li> <li>• FINE</li> <li>• FINER</li> <li>• FINEST</li> </ul>

注※

メッセージ・レベルは、「[14.2 メッセージの詳細](#)」に示す `java.util.logging` のレベルのことを指します。



## 12.3 setenv.sh (Tomcat 起動時の環境変数定義ファイル)

setenv.sh (Tomcat 起動時の環境変数定義ファイル) では, Tomcat 起動時の環境変数を定義できます。

### 12.3.1 形式

シェルスクリプト形式のファイルです。setenv.sh (Tomcat 起動時の環境変数定義ファイル) 自体の使い方や仕様については, 使用するバージョンの Tomcat のドキュメントを参照してください。

### 12.3.2 ファイルの格納先

```
${CATALINA_BASE}/bin/setenv.sh
```

または

```
${CATALINA_HOME}/bin/setenv.sh
```

setenv.sh (Tomcat 起動時の環境変数定義ファイル) がない場合は新規に作成してください。なお, このファイルのテンプレートが次の場所に格納されているため, こちらのテンプレートをコピーして使うこともできます。

```
<本製品のインストールディレクトリ>/template/tomcat/setenv.sh
```

このテンプレートを使用した場合, /opt/Cosminexus/jdk ディレクトリ※に日立 JavaVM がインストールされていると, 環境変数の定義に関係なく /opt/Cosminexus/jdk ディレクトリ※にある日立 JavaVM が優先して使用されます。

注※

JDK17 ベースの日立 JavaVM がインストールされている場合は, /opt/Cosminexus/jdk17 ディレクトリになります。

#### 重要

- 環境変数 CATALINA\_BASE を設定し, かつ \${CATALINA\_BASE}/bin に setenv.sh (Tomcat 起動時の環境変数定義ファイル) が存在する場合は, \${CATALINA\_BASE}/bin の setenv.sh (Tomcat 起動時の環境変数定義ファイル) が修正対象となります。\${CATALINA\_BASE}/bin に setenv.sh (Tomcat 起動時の環境変数定義ファイル) を作成した場合, \${CATALINA\_HOME}/bin の setenv.sh (Tomcat 起動時の環境変数定義ファイル) は読み込まれなくなります。

- 環境変数 CATALINA\_BASE に環境変数 CATALINA\_HOME (Tomcat のインストールディレクトリ) とは別のパスを設定していて、\${CATALINA\_BASE}/bin/setenv.sh (Tomcat 起動時の環境変数定義ファイル) が存在する場合は、\${CATALINA\_HOME}/bin/setenv.sh (Tomcat 起動時の環境変数定義ファイル) は読み込まれません。

### 12.3.3 機能

Tomcat 起動時の環境変数を定義できます。

Tomcat プロセスモニタおよび Tomcat サーバプロセスの稼働中にこのファイルの内容を変更した場合、変更した内容は次に Tomcat プロセスモニタを起動したときに反映されます。

### 12.3.4 指定可能な環境変数

本製品で指定可能な環境変数を次に示します。

表 12-8 setenv.sh (Tomcat 起動時の環境変数定義ファイル) に記述する変数定義

setenv.sh に記述する変数名	説明	省略可否
JRE_HOME	Java のインストールディレクトリを指定します。 export を先頭に付与して指定してください。	必ず指定してください。 ただし、次のどちらかの条件を満たす場合は、「export JRE_HOME」と指定できます。 <ul style="list-style-type: none"> <li>• 環境変数 JAVA_HOME にご利用の JDK のインストールディレクトリが指定されている</li> <li>• 環境変数 PATH にご利用の java コマンドが格納されたディレクトリが指定されている</li> </ul>
_RUNJAVA	本製品が提供するラッパースクリプトを起動します。 <本製品のインストールディレクトリ>/script/wrapper.sh を必ず指定してください。	必ず指定してください。
PROCESS_MONITOR_CONFIG_PATH	config.properties (本製品の設定ファイル) のパスを絶対パスで指定します。 export を先頭に付与して指定してください。	指定を省略できます。 省略した場合、次のパスのファイルを指定したものと見なします。 <本製品のインストールディレクトリ>/conf/config.properties

## ❗ 重要

環境変数 CATALINA\_BASE を設定し、かつ、`${CATALINA_BASE}/bin` に `setenv.sh` (Tomcat 起動時の環境変数定義ファイル) が存在する場合は、`${CATALINA_BASE}/bin` にある `setenv.sh` (Tomcat 起動時の環境変数定義ファイル) が修正対象となります。その場合、`${CATALINA_HOME}/bin` の `setenv.sh` (Tomcat 起動時の環境変数定義ファイル) は読み込まれなくなります。

## 12.4 catalina.properties (Tomcat のプロパティ定義ファイル)

---

本製品から提供されるライブラリを Tomcat サーバプロセスの Common クラスローダでロードさせるために、catalina.properties (Tomcat のプロパティ定義ファイル) 内のプロパティ「common.loader」に対して、次に示す下線部分の追記が必要です。

```
common.loader=<元の値>,"${com.cosminexus.appruntime.lib.path}/tomcat/common/*.jar"
```

設定例：

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/*.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${com.cosminexus.appruntime.lib.path}/tomcat/common/*.jar"
```

catalina.properties (Tomcat のプロパティ定義ファイル) の仕様や、Tomcat が定義しているプロパティの仕様については、使用するバージョンの Tomcat のドキュメントを参照してください。

## 12.5 server.xml (Tomcat のサーバ設定ファイル)

本製品の機能を Tomcat サーバプロセス上で動作させるために、本製品から提供される実装クラスを server.xml (Tomcat のサーバ設定ファイル) に設定する必要があります。server.xml (Tomcat のサーバ設定ファイル) に追加する必要がある要素、およびその親要素を次の表に示します。

表 12-9 server.xml (Tomcat のサーバ設定ファイル) に追加が必要な要素

要素, 属性 (太字の部分が追加が必要な要素)				参照先
<Server>				-
ト	<Listener className="com.cosminexus.appruntime.tomcat.tracer.ServerComponentHandler" />			9.2 トレース機能の セットアップ方法
ト	<Listener className="com.cosminexus.appruntime.tomcat.healthcheck.MonitoringListener" />			10.2.2 Tomcat サー バプロセスへの稼働監 視用ライフサイクルリ スナーの設定
	:			-
ト	<Server>			-
		:		-
	ト	<Engine>		-
		ト	<Valve className="com.cosminexus.appruntime.tomcat.tracer .RequestTraceValve" />	9.2 トレース機能の セットアップ方法
			:	-
:	:	:	:	-

(凡例)

-: 該当なし

なお、このファイルのテンプレートが次の場所に格納されています。こちらのテンプレートをコピーして使うこともできます。

Tomcat 9.x の場合：

<本製品のインストールディレクトリ>/template/tomcat9/server.xml

Tomcat 10.1.x の場合：

<本製品のインストールディレクトリ>/template/tomcat10.1/server.xml

## ! 重要

テンプレートファイル内のコメントに、テンプレートファイルの元となった Tomcat のバージョンが記載されています。コメントに記載されているバージョンが、実際に使用する Tomcat のバージョンよりも古い場合、Tomcat の仕様が変更されているおそれがあります。

バージョンが書かれているコメントの例は次のとおりです。

```
<!-- original Tomcat version: 10.1.15 -->
```

また、次に示す要素は、必要に応じて追加してください。

- Tomcat JDBC Connection Pool 機能を使用する場合  
server.xml (Tomcat のサーバ設定ファイル) に<Resource>要素を追加します。該当する<Resource>要素の jdbcInterceptors 属性に "com.cosminexus.appruntime.tomcat.tracer.JdbcTraceHandler" を追記してください。詳細は、「[9.2 トレース機能のセットアップ方法](#)」を参照してください。
- 稼働監視機能でリクエスト処理の停滞を検知する場合  
server.xml (Tomcat のサーバ設定ファイル) または context.xml (Tomcat のコンテキスト設定ファイル) で Tomcat の Stuck Thread Detection Valve (className 属性が "org.apache.catalina.valves.StuckThreadDetectionValve" の<Valve>要素) を追加してください。詳細は、「[10.3.5\(3\) 設定できる内容](#)」および Tomcat のドキュメントを参照してください。

## 12.6 context.xml (Tomcat のコンテキスト設定ファイル)

本製品の機能を Tomcat サーバプロセス上で動作させるために、本製品から提供される実装クラスを context.xml (Tomcat のコンテキスト設定ファイル) に設定する必要があります。context.xml (Tomcat のコンテキスト設定ファイル) に追加する必要がある要素、およびその親要素を次の表に示します。

表 12-10 context.xml (Tomcat のコンテキスト設定ファイル) に追加が必要な要素

要素, 属性 (太字の部分追加が必要な要素)	参照先
<Context>	-
:	-
└ <Loader loaderClass="com.cosminexus.appruntime.tomcat.tracer.Webapp TraceClassLoader" />	9.2 <a href="#">トレース機能のセットアップ方法</a>
:	-

(凡例)

-: 該当なし

なお、このファイルのテンプレートが次の場所に格納されています。こちらのテンプレートをコピーして使うこともできます。

Tomcat 9.x の場合 :

```
<本製品のインストールディレクトリ>/template/tomcat9/context.xml
```

Tomcat 10.1.x の場合 :

```
<本製品のインストールディレクトリ>/template/tomcat10.1/context.xml
```

### ❗ 重要

テンプレートファイル内のコメントに、テンプレートファイルの元となった Tomcat のバージョンが記載されています。コメントに記載されているバージョンが、実際に使用する Tomcat のバージョンよりも古い場合、Tomcat の仕様が変更されているおそれがあります。

バージョンが書かれているコメントの例は次のとおりです。

```
<!-- original Tomcat version: 10.1.15 -->
```

また、次に示す要素は、必要に応じて追加してください。

- Tomcat JDBC Connection Pool 機能を使用する場合  
context.xml (Tomcat のコンテキスト設定ファイル) に<Resource>要素を追加します。該当する<Resource>要素の jdbcInterceptors 属性

に"com.cosminexus.appruntime.tomcat.tracer.JdbcTraceHandler"を追記してください。詳細は、[「9.2 トレース機能のセットアップ方法」](#)を参照してください。

- 稼働監視機能でリクエスト処理の停滞を検知する場合

server.xml (Tomcat のサーバ設定ファイル) または context.xml (Tomcat のコンテキスト設定ファイル) で Tomcat の Stuck Thread Detection Valve (className 属性が"org.apache.catalina.valves.StuckThreadDetectionValve"の<Valve>要素) を追加してください。詳細は、[「10.3.5\(3\) 設定できる内容」](#) および Tomcat のドキュメントを参照してください。



# 13

## ログファイル

この章では、本製品が出力するログファイルについて説明します。

## 13.1 ログファイルの種類

Tomcat プロセスモニタ，および本製品を適用した Tomcat サーバプロセスから出力されるログファイルの一覧を次の表に示します。環境変数 CATALINA\_BASE を定義していない場合は，\${CATALINA\_BASE}を\${CATALINA\_HOME}に読み替えてください。

表 13-1 Tomcat プロセスモニタから出力されるログファイル

分類	デフォルトのログ出力先およびログファイル名	デフォルトのサイズ×面数
メッセージログ	\${CATALINA_BASE}/logs/ucarmessage*.log	32MB×4 面
保守ログ	\${CATALINA_BASE}/logs/ucarmaintenance*.log	32MB×4 面
JavaVM ログ*	\${CATALINA_BASE}/logs/pmjavavm*.log	256KB×4 面

注※

日立 JavaVM を使用している場合だけ出力されます。

表 13-2 本製品を適用した Tomcat サーバプロセスから出力される独自ログファイル

分類	デフォルトのログ出力先およびログファイル名	デフォルトのサイズ×面数
トレースログ	\${CATALINA_BASE}/logs/ucartrace*.log	32MB×4 面
JavaVM ログ*	\${CATALINA_BASE}/logs/javavm*.log	256KB×4 面

注※

日立 JavaVM を使用している場合だけ出力されます。

Tomcat サーバプロセスからは，「表 13-2 本製品を適用した Tomcat サーバプロセスから出力される独自ログファイル」に示したもの以外に，Tomcat 自身が出力しているログファイルも出力されます。Tomcat のログファイルの仕様については，使用するバージョンの Tomcat のドキュメントを参照してください。

## 13.2 Tomcat プロセスモニタのログ

Tomcat プロセスモニタが提供するログファイルについて説明します。

### 13.2.1 メッセージログ

メッセージログについて説明します。

#### ファイルの文字コード

ファイルの文字コードは UTF-8 です。

#### フォーマット

メッセージログの各行フォーマットを次に示します。

```
<日付>△<時刻>△<プロセスID>△<スレッドID>△<メッセージID>△<メッセージ本文>
```

注 △：半角空白

各項目の詳細を次の表に示します。

表 13-3 メッセージログの各項目の詳細

項目	詳細
日付	日付を「yyyy-MM-dd」形式で出力します。
時刻	時刻を「HH:mm:ss.SSS」形式で出力します。
プロセス ID	Tomcat プロセスモニタのプロセス ID を 8 桁の 10 進数形式で出力します。
スレッド ID	ログ出力元のスレッド ID を 8 桁の 10 進数形式で出力します。100000000 以上の場合は下 8 桁だけ出力します。
メッセージ ID	メッセージ ID を「XXXXnnnnn-Y」形式で出力します。
メッセージ本文	メッセージ本文を出力します。文字数制限はありません。

#### プロパティ

メッセージログの設定に関しては、「monitor.log.message」で始まるプロパティで変更できます。詳細は、「[12.2.4\(2\) Tomcat プロセスモニタに関するプロパティ](#)」を参照してください。

### 13.2.2 保守ログ

保守ログについて説明します。保守ログは、サポートサービス内で参照することを目的としています。そのため、通常は参照する必要はありません。

## ファイルの文字コード

ファイルの文字コードは UTF-8 です。

## フォーマット

保守ログの各行のフォーマットは規定されません。そのため、ログの内容を独自に解析したときの動作は保証されません。

## プロパティ

保守ログの設定に関しては、「[monitor.log.maintenance](#)」で始まるプロパティで変更できます。詳細は、「[12.2.4\(2\) Tomcat プロセスモニタに関するプロパティ](#)」を参照してください。

## 13.2.3 JavaVM ログ

JavaVM ログの設定方法について説明します。

### 日立 JavaVM の場合

日立 JavaVM を利用する場合、デフォルトとして次に示す日立 JavaVM オプションでログ出力定義が設定されます。

- -XX:+HitachiJavaClassLibTrace
- -XX:+HitachiLocalsInStackTrace
- -XX:+HitachiLocalsSimpleFormat
- -XX:+HitachiOutOfMemoryAbort
- -XX:+HitachiOutOfMemoryStackTrace
- -XX:+HitachiOutputMilliTime
- -XX:-HitachiThreadDumpToStdout
- -XX:+HitachiVerboseGC
- -XX:+StandardLogToHitachiJavaLog
- -XX:+HitachiFullCore
- -XX:HitachiJavaLog:\${common.base}/pmjavalog

### 他社製 JavaVM の場合

次の設定ファイルのプロパティで設定できます。複数の JavaVM オプションを指定する場合は、空白で区切って指定してください。設定値の詳細は、利用する JavaVM のマニュアルを参照してください。

- config.properties (本製品の設定ファイル) のプロパティ `monitor.jvm.options`

config.properties (本製品の設定ファイル) については、「[12.2 config.properties \(本製品の設定ファイル\)](#)」を参照してください。

## 13.3 Tomcat サーバプロセスのログ

Tomcat サーバプロセスから出力される独自ログファイルについて説明します。

### 13.3.1 Tomcat トレースログ

本製品のトレース機能は、Tomcat の動作状況を示すトレース情報をログファイルに出力します。ログファイルのヘッダ、およびログファイルに出力されるトレース情報の詳細について説明します。なお、ログファイルに書き込む文字コードは、プラットフォームのデフォルトエンコーディングを使用します。

#### (1) ヘッダの詳細

##### ヘッダを出力するタイミング

ヘッダを出力するタイミングは、次のとおりです。

- トレース機能を初期化したとき
- ローテーションで次のファイルに書き込むとき

##### ヘッダの出力内容

ヘッダの出力内容は、次のとおりです。

```
PRF, Process, Thread(hashcode), Trace, ProcessName, Event, Date, Time, Time(msec/usec/nsec), Rc, ClientAP IP, ClientAP PID, ClientAP CommNo., RootAP IP, RootAP PID, RootAP CommNo., SendSCD IP, SendSCD PID, ReceiveSCD IP, ReceiveSCD PID, INT, OPR, LookupName, OPT, ASCII
```

#### (2) トレース情報の詳細

ログファイルのヘッダを出力したあと、トレース情報を出力します。

##### トレース情報の形式

トレースの形式は、次のとおりです。

- 1 トレース 1 行で出力
- 日時は、プラットフォームのデフォルトタイムゾーンを使用

##### トレース情報の出力内容

トレース情報の出力内容と値の範囲を次の表に示します。

表 13-4 トレース情報の出力内容と値の範囲

トレース情報のヘッダ	出力内容	値の範囲
PRF	取得したトレース情報の状態。	Rec だけが出力されます。Rec は「正常」を意味します。
Process	トレース情報を取得したプロセスのプロセス ID。	10 進数形式で出力されます。

トレース情報のヘッダ	出力内容	値の範囲
Thread (hashcode)	<ul style="list-style-type: none"> <li>• トレース情報を取得したスレッドの情報。</li> <li>• Java スレッド ID</li> <li>• トレース情報を取得したスレッドのスレッドオブジェクトのハッシュコード</li> </ul>	<p>次の形式で出力されます。</p> <p>&lt;Java スレッド ID&gt; (&lt;スレッドオブジェクトのハッシュコード※1&gt;)</p> <p>なお、Java スレッド ID は 10 進数、スレッドオブジェクトのハッシュコードは 0x で始まる 16 進数で表記されます。</p>
Trace	トレース情報を取得したプロセス内スレッドでのトレース通番。	10 進数形式で出力されます。
ProcessName	トレース情報を取得したプロセスの名前。	java だけが出力されます。
Event	処理方式の各トレースで記載したイベント ID。	0x で始まる 16 進数形式で出力されます。
Date	トレース情報を取得した日付。	<西暦>/<月>/<日>形式で出力されます。
Time	トレース情報を取得した時刻 (時:分:秒)。	<時>:<分>:<秒>形式で出力されます。
Time (msec/usec/nsec)	トレース情報を取得した時刻 (ミリ秒/マイクロ秒/ナノ秒)。	<ミリ秒>/<マイクロ秒>/<ナノ秒>形式で出力されます。
Rc	各トレース取得ポイントに記載した Rc 項目。	16 進数形式で出力されます。ただし、0 は 0 で出力されます。
ClientAP IP	トレース情報を取得したクライアントアプリケーションの IP アドレス※2。	IPv4 の IP アドレス形式(a.b.c.d)で出力されます。
ClientAP PID	トレース情報を取得したクライアントアプリケーションのプロセス ID※3。	10 進数形式で出力されます。
ClientAP CommNo.	トレース情報を取得したクライアントアプリケーションの通信番号※4。	0x で始まる 16 進数形式で出力されます。
RootAP IP	トレース情報を取得したルートアプリケーションの IP アドレス※2。	IPv4 の IP アドレス形式(a.b.c.d)で出力されます。
RootAP PID	トレース情報を取得したルートアプリケーションのプロセス ID※3。	10 進数形式で出力されます。
RootAP CommNo.	トレース情報を取得したルートアプリケーションの通信番号※4。	0x で始まる 16 進数形式で出力されます。
SendSCD IP	リクエスト要求元 CTM の IP アドレス。	****だけが出力されます。
SendSCD PID	リクエスト要求元 CTM のプロセス ID。	****だけが出力されます。
ReceiveSCD IP	リクエスト要求先 CTM の IP アドレス。	****だけが出力されます。
ReceiveSCD PID	リクエスト要求先 CTM のプロセス ID。	****だけが出力されます。
INT	各トレース取得ポイントに記載した INT 項目を""で囲んだ※5 もの。	出力文字数に制限はありません。

トレース情報のヘッダ	出力内容	値の範囲
OPR	各トレース取得ポイントに記載した OPR 項目を""で囲んだ※5 もの。	出力文字数に制限はありません。
LookupName	ルックアップ名。	****だけが出力されます。
OPT	取得ポイントごとの付加情報。	空文字 ("" ) だけが出力されます。
ASCII	各トレース取得ポイントに記載した ASCII 項目を""で囲んだ※5 もの。	記載がない場合、空文字 ("" ) が出力されます。

注※1

スレッドダンプの jid 項目と一致します。この項目を利用してスレッドダンプと突き合わせを実施してください。

注※2

情報がない場合 (IPv4 アドレスが取得できなかった場合も含む), 0.0.0.0 を出力します。

注※3

情報がない場合は, 0 を出力します。

注※4

情報がない場合は, 0x0000000000000000 を出力します。

注※5

各トレース取得ポイントの項目に"- "が記載されている場合, ""は出力されず空文字 ("" ) が出力されます。項目に含まれるダブルクォーテーション ( " ) はエスケープした文字 ( "" ) に置き換えて出力されます。

## 13.3.2 JavaVM ログ

JavaVM ログの設定方法について説明します。

### 日立 JavaVM の場合

日立 JavaVM を利用する場合, デフォルトとして次に示す日立 JavaVM オプションでログ出力定義が設定されます。

- -XX:+HitachiJavaClassLibTrace
- -XX:+HitachiLocalsInStackTrace
- -XX:+HitachiLocalsSimpleFormat
- -XX:+HitachiOutOfMemoryAbort
- -XX:+HitachiOutOfMemoryStackTrace
- -XX:+HitachiOutputMilliTime
- -XX:-HitachiThreadDumpToStdout
- -XX:+HitachiVerboseGC
- -XX:+StandardLogToHitachiJavaLog
- -XX:+HitachiFullCore

- `-XX:HitachiJavaLog:${common.base}/javalog`

デフォルトの設定値を変更する場合、または設定を追加する場合は、次の環境変数で設定を上書きできます。複数の日立 JavaVM オプションを指定する場合は、空白で区切って指定してください。

- 環境変数 `CATALINA_OPTS`

#### 他社製 JavaVM の場合

次の環境変数で設定できます。複数の JavaVM オプションを指定する場合は、空白で区切って指定してください。設定値の詳細は、利用する JavaVM のマニュアルを参照してください。

- 環境変数 `CATALINA_OPTS`



# 14

## メッセージ

本製品で出力されるメッセージについて説明します。

## 14.1 メッセージの形式

---

メッセージの記述形式、および `java.util.logging` のレベルを説明します。

### 14.1.1 メッセージの記述形式

以降の「[14.2 メッセージの詳細](#)」では、メッセージを次の形式で記述します。

KDLRnnnnn-Y

メッセージテキスト

可変値に関する説明

説明

メッセージテキストに対する補足説明

対処

ユーザが実施する対処

`java.util.logging` のレベル

`java.util.logging` のレベル

なお、「可変値に関する説明」、「説明」、「対処」および「`java.util.logging` のレベル」はメッセージによって記述しないものもあります。

次に、各項目について説明します。

KDLRnnnnn

メッセージ ID を表します。

メッセージ ID を構成する要素について、次に説明します。

KDLR

本製品で出力されるメッセージのプリフィックスを示します。

nnnnn

本製品で管理するメッセージ番号を表します。それぞれのメッセージには、5桁の固有の番号が付いています。

Y

メッセージのレベルを表します。メッセージのレベルは英字 1 文字で示します。

メッセージのレベルを示す文字とその意味を次に示します。

E (Error)

エラーレベルのトラブルが発生したことを通知するメッセージです。

このメッセージが出力されたときは、処理を中断します。

## W (Warning)

警告レベルのトラブルが発生したことを通知するメッセージです。

メッセージが出力されたあとも処理を続行します。

## I (Information)

システムの動作を通知するメッセージです。

メッセージが出力されたあとも処理を続行します。

## メッセージテキスト

本製品で出力されるメッセージテキストを示します。

なお、メッセージテキスト中の可変値（メッセージが出力される状況によって変わる値）は、「xx...xx」（xx は英小文字）の形式で示します。

## 可変値に関する説明

メッセージテキスト中の可変値に表示される情報を「xx...xx：表示される情報」（xx は英小文字）の形式で示します。可変値に関する説明の記述例を次に示します。

（例）

aa...aa：ファイル名

bb...bb：アプリケーション名

## 説明

メッセージが通知された要因やメッセージを出力した構成ソフトウェアの動作など、メッセージに対する補足説明を示します。

## 対処

ユーザが実施する対処を表します。なお、対処方法の「保守員に連絡してください」とは、購入時の契約に基づいて、システム管理者が弊社問い合わせ窓口へ連絡することを示します。

## java.util.logging のレベル

java.util.logging のレベルを表します。詳細については「[14.1.2 java.util.logging のレベル](#)」を参照してください。

## 14.1.2 java.util.logging のレベル

各メッセージのロギング・レベルを示します。次の表に java.util.logging のレベルを高いほうから順に示します。レベルが高いほど、重大な問題であることを意味します。

表 14-1 java.util.logging のレベルとその意味

レベルの高低	java.util.logging のレベル	意味
高	SEVERE	重大な障害を意味するメッセージです。

レベルの 高低	java.util.logging のレベル	意味
	WARNING	潜在的な問題を意味するメッセージです。
	INFO	情報レベルのメッセージです。
	CONFIG	静的な構成メッセージです。
	FINE	トレース情報を提供するメッセージです。
	FINER	より詳細なトレース情報を提供するメッセージです。
低	FINEST	非常に詳細なトレース情報を提供するメッセージです。

## 14.2 メッセージの詳細

### KDLR00000-E

The JRE\_HOME environment variable is not defined, or it's value is incorrect. If the JAVA\_HOME environment variable is defined, then write simply "export JRE\_HOME" to setenv.sh.

#### 説明

JRE\_HOME が定義されていません。または、正しい値ではありません。

#### 対処

- 環境変数 JAVA\_HOME が定義されている場合：  
setenv.sh (Tomcat 起動時の環境変数定義ファイル) に export JRE\_HOME を追加してください。
- 環境変数 PATH で Java へのパスが追加されている場合：  
setenv.sh (Tomcat 起動時の環境変数定義ファイル) に export JRE\_HOME を追加してください。
- JAVA\_HOME が示す Java や、環境変数 PATH で指定する Java 以外の Java を利用する場合：  
setenv.sh (Tomcat 起動時の環境変数定義ファイル) に export JRE\_HOME=<使用する Java のディレクトリ>を追加してください。指定するディレクトリは、bin ディレクトリの親ディレクトリです。

### KDLR00001-E

The required command cannot be found. (command = aa....aa)

aa....aa : 必要なコマンド名

#### 説明

必要なコマンドが見つかりません。

#### 対処

- コマンドがインストールされているかどうかを確認してください。
- コマンドへのパスが PATH 環境変数に指定されているかどうかを確認してください。

### KDLR00100-I

[Default] aa....aa = bb....bb

aa....aa : config.properties (本製品の設定ファイル) のプロパティキー

bb....bb : デフォルト値

## 説明

config.properties (本製品の設定ファイル) によって値が設定されていないプロパティとデフォルト値です。

## 対処

なし

## java.util.logging のレベル

CONFIG

## KDLR00101-I

```
[Updated] aa....aa = bb....bb
```

aa....aa : config.properties (本製品の設定ファイル) のプロパティキー

bb....bb : 設定値

## 説明

config.properties (本製品の設定ファイル) によって値が設定されているプロパティと設定値です。

## 対処

なし

## java.util.logging のレベル

CONFIG

## KDLR00102-I

```
[Undefined] aa....aa
```

aa....aa : config.properties (本製品の設定ファイル) のプロパティキー

## 説明

config.properties (本製品の設定ファイル) によって値が設定されていないプロパティです。

## 対処

なし

## java.util.logging のレベル

CONFIG

## KDLR00103-E

```
An invalid value was found during validation of the property value. (key = aa....aa, value = bb....bb)
```

aa....aa : config.properties (本製品の設定ファイル) のプロパティキー

bb....bb：設定された値

#### 説明

プロパティ値の検証で不正な値が見つかりました。

#### 対処

ファイルの設定内容を確認して、プロパティキーに応じた値の範囲で設定値を記載してください。\${XXX}形式を使用している場合は、XXX が示す値も確認してください。

#### java.util.logging のレベル

SEVERE

### KDLR00104-E

```
Failed to resolve the embedded variable. (key = aa....aa, value = bb....bb)
```

aa....aa：config.properties（本製品の設定ファイル）のプロパティキー

bb....bb：設定値

#### 説明

変数の解決に失敗しました。

#### 対処

原因として、\${XXX}形式で指定した XXX が存在しないことが考えられます。XXX が次のどれかに定義されている必要があります。

- catalina.home
- catalina.base
- <環境変数名>
- config.properties（本製品の設定ファイル）のキー

<環境変数名> を指定した場合は、環境変数が定義されている必要があります。

#### java.util.logging のレベル

SEVERE

### KDLR00105-E

```
The required property is not defined. (key = aa....aa)
```

aa....aa：config.properties（本製品の設定ファイル）のプロパティキー

#### 説明

必須プロパティが未設定です。

## 対処

必須プロパティを config.properties (本製品の設定ファイル) で定義してください。

## java.util.logging のレベル

SEVERE

## KDLR00106-E

An error occurred during validation of the configuration file. (path = aa....aa)

aa....aa : ファイルパス

## 説明

config.properties (本製品の設定ファイル) の検証でエラーが発生しました。

## 対処

このメッセージの前に出力されているエラーメッセージを確認してください。

## java.util.logging のレベル

SEVERE

## KDLR00107-I

The common.java.hitachi.javacoredir property in the configuration file will be ignored because the JAVACOREDİR environment variable is already defined. (JAVACOREDİR = aa....aa)

aa....aa : JAVACOREDİR 値

## 説明

環境変数 JAVACOREDİR が設定済みのため、config.properties (本製品の設定ファイル) のプロパティ common.java.hitachi.javacoredir は無視されます。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR00108-E

The value of the JAVACOREDİR environment variable is not an absolute path. (JAVACOREDİR = aa....aa)

aa....aa : JAVACOREDİR の値



## 説明

環境変数 JAVACOREDIR の値が絶対パスではありません。

## 対処

環境変数 JAVACOREDIR を絶対パスで指定してください。

## java.util.logging のレベル

SEVERE

## KDLR00109-E

The variable value <n> in the property key is not a natural number. (key = aa....aa, specification-key = bb....bb)

aa....aa : config.properties (本製品の設定ファイル) のプロパティキー

bb....bb : プロパティキーの仕様上表記

## 説明

プロパティキーに含まれる可変値<n>が自然数 (1 以上の整数) ではありません。

## 対処

プロパティキーの可変値<n>には、自然数 (1 以上の整数) を指定してください。先頭に 0 が付く値は指定できません。

## java.util.logging のレベル

SEVERE

## KDLR00110-E

The variable value <group-id> in the required property key is invalid. (key = aa....aa, required-key = bb....bb)

aa....aa : config.properties (本製品の設定ファイル) のプロパティキー

bb....bb : <group-id>を定義するための必須プロパティキーの仕様上表記

## 説明

必須プロパティキーに含まれる可変値<group-id>が不正です。

## 対処

プロパティキーの可変値<group-id>は、次の文字が含まれない 1 文字以上の文字列にしてください。

- . (0x2e)
- : (0x3a)
- = (0x3d)

## java.util.logging のレベル

SEVERE

### KDLR00111-E

The variable value <group-id> in the property key is not defined by another required property. (key = aa....aa, specification-key = bb....bb, required-key = cc....cc)

aa....aa : config.properties (本製品の設定ファイル) のプロパティキー

bb....bb : プロパティキーの仕様上表記

cc....cc : <group-id>を定義するための必須プロパティキーの仕様上表記

#### 説明

プロパティキーに含まれる可変値<group-id>がほかの必須プロパティによって定義されていません。

#### 対処

プロパティキーの可変値<group-id>は、定義するための必須プロパティキーと同一の値を指定してください。対応する必須プロパティについては、エラーとなったプロパティの説明を「[12.2 config.properties \(本製品の設定ファイル\)](#)」で確認してください。

## java.util.logging のレベル

SEVERE

### KDLR00112-E

There is an environment variable with the same name as the property key specified as the variable name. (name = aa....aa)

aa....aa : 環境変数名

#### 説明

変数名として指定されたプロパティキーと同名の環境変数が存在します。

#### 対処

config.properties (本製品の設定ファイル) のプロパティキー名と同一の環境変数名は、\${XXX}形式で指定できません。別の環境変数名で\${XXX}形式を指定してください。

## java.util.logging のレベル

SEVERE

### KDLR00113-E

Two related properties have value which cannot be specified together. (key1 = aa....aa, value1 = bb....bb, key2 = cc....cc, value2 = dd....dd)

aa....aa : 1 つ目の config.properties (本製品の設定ファイル) のプロパティキー

bb....bb : 1 つ目の設定値

cc....cc : 2 つ目の config.properties (本製品の設定ファイル) のプロパティキー

dd....dd : 2 つ目の設定値

#### 説明

2 つの関連するプロパティの値に、同時に設定できない値が設定されています。

#### 対処

ファイルの設定内容を確認して、正しい値の組み合わせで設定値を記載してください。

#### java.util.logging のレベル

SEVERE

### KDLR00200-E

```
Failed to read the file, or it does not exist. (path = aa....aa, cause = bb....bb)
```

aa....aa : ファイルパス

bb....bb : 原因例外メッセージ

#### 説明

ファイルの読み込みに失敗しました。または、ファイルが存在しません。

#### 対処

次の内容を確認してください。

- メッセージ内のファイルパスにファイルが存在するか確認してください。
- アクセス権が正しく設定されているか確認してください。
- ファイルの種類によっては、絶対パスで指定する必要があります。設定値を確認してください。

#### java.util.logging のレベル

SEVERE

### KDLR00201-E

```
Failed to create a directory. (path = aa....aa, cause = bb....bb)
```

aa....aa : ディレクトリパス

bb....bb : 原因例外メッセージ

#### 説明

ディレクトリの作成に失敗しました。

## 対処

メッセージ内のディレクトリパスに、ファイルが存在していないことを確認してください。また、アクセス権が正しく設定されているか確認してください。

### java.util.logging のレベル

SEVERE

## KDLR00202-E

```
Failed to initialize a log file. (logging-type = aa....aa, property-value = bb...bb, cause = cc....cc)
```

aa....aa : ログの種類 (MESSAGE, MAINTENANCE)

bb....bb : config.properties (本製品の設定ファイル) のパス指定値

cc....cc : 原因例外メッセージ

## 説明

ログファイルの初期化に失敗しました。

## 対処

次の内容を確認してください。

- ログ出力先ディレクトリが存在するか確認してください。また、出力先のアクセス権が正しく設定されているか確認してください。
- プリフィックスにファイル名として不正な文字が含まれていないか確認してください。

### java.util.logging のレベル

SEVERE

## KDLR00203-E

```
Failed to delete the file or directory. (path = aa....aa, cause = bb...bb)
```

aa....aa : ファイルパス, またはディレクトリパス

bb....bb : 原因例外メッセージ

## 説明

ファイルまたはディレクトリの削除に失敗しました。

## 対処

メッセージ内のパスが使用中か確認してください。また、アクセス権が正しく設定されているか確認してください。

メッセージ内のパスがディレクトリの場合は、ディレクトリ内のすべてのファイル、およびディレクトリについても確認してください。

## java.util.logging のレベル

SEVERE

### KDLR00204-E

The file path contains an invalid character. (path = aa....aa, cause = bb....bb)

aa....aa：ファイルパス，またはディレクトリパス

bb....bb：原因例外メッセージ

#### 説明

ファイルパスに不正な文字が含まれています。

#### 対処

ファイルパスとして利用できない文字が含まれていないか確認してください。マルチバイト文字を利用する場合は，OS に設定された文字コードで利用可能であることを確認してください。

## java.util.logging のレベル

SEVERE

### KDLR10000-I

Exporting the snapshot log has finished successfully. (id = aa....aa, path = bb....bb)

aa....aa：スナップショットログ ID

bb....bb：出力先ファイルパス

#### 説明

スナップショットログ出力が完了しました。

#### 対処

なし

## java.util.logging のレベル

INFO

### KDLR10001-E

An error occurred during output of the snapshot log. (id = aa....aa, path = bb....bb, cause = cc....cc)

aa....aa：スナップショットログ ID

bb....bb：出力先ファイルパス

cc....cc：原因例外メッセージ

## 説明

スナップショットログ出力中にエラーが発生しました。

## 対処

次の内容を確認し、スナップショットログを手動で再収集してください。

- 出力先ディレクトリパスが存在するか確認してください。
- 出力先のアクセス権が正しく設定されているか確認してください。
- 原因例外メッセージを参照し、原因を取り除いてください。

## java.util.logging のレベル

SEVERE

## KDLR10002-I

```
Output of the snapshot log will now start. (id = aa....aa, trigger = bb....bb)
```

aa....aa：スナップショットログ ID

bb....bb：出力契機

## 説明

スナップショットログ出力を開始します。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR10003-W

```
Failed to create a temporary directory for storing the results of command execution for the snapshot log. (id = aa....aa, path = bb....bb, cause = cc....cc)
```

aa....aa：スナップショットログ ID

bb....bb：コマンド実行結果格納ディレクトリパス

cc....cc：原因例外メッセージ

## 説明

スナップショットログのためのコマンド実行結果格納ディレクトリの作成に失敗しました。

## 対処

次の内容を確認し、スナップショットログを手動で再収集してください。

- コマンド実行結果格納ディレクトリを格納するディレクトリパスが存在するか確認してください。

- 出力先のアクセス権が正しく設定されているか確認してください。
- 原因例外メッセージを参照し、原因を取り除いてください。

## java.util.logging のレベル

WARNING

### KDLR10004-W

```
Failed to delete the intermediate file created during output of the snapshot log. (id = aa....aa, path = bb....bb, cause = cc....cc)
```

aa....aa : スナップショットログ ID

bb....bb : 中間ファイルのパス

cc....cc : 原因例外メッセージ

#### 説明

スナップショットログ出力中に作成した中間ファイルの削除に失敗しました。

#### 対処

中間ファイルのパスに残存するファイルを削除してください。

## java.util.logging のレベル

WARNING

### KDLR10005-W

```
An error occurred during execution of a command for the snapshot log. (id = aa....aa, command = bb....bb, path = cc....cc, cause = dd....dd)
```

aa....aa : スナップショットログ ID

bb....bb : 実行コマンド

cc....cc : 出力先ファイルパス

dd....dd : 原因例外メッセージ

#### 説明

スナップショットログのためのコマンド実行時にエラーが発生しました。

#### 対処

次の内容を確認し、スナップショットログを手動で再収集してください。

- 出力先ディレクトリパスが存在するか確認してください。
- 出力先のアクセス権が正しく設定されているか確認してください。

- 原因例外メッセージを参照し、原因を取り除いてください。

## java.util.logging のレベル

WARNING

### KDLR10006-W

```
The executed command for the snapshot log exited with error status. (id = aa....aa, command = bb....bb, exit-status = cc....cc)
```

aa....aa : スナップショットログ ID

bb....bb : 実行コマンド

cc....cc : 終了ステータス

#### 説明

スナップショットログのために実行したコマンドがエラーステータスで終了しました。

#### 対処

次の内容を確認し、必要があればスナップショットログを手動で再収集してください。

- 実行ユーザにコマンドの実行権限が正しく設定されているか確認してください。
- 別プロセスがコマンド実行中でないか確認してください（同時実行できないコマンドの場合）。

環境によっては、常に終了ステータスが異常コードとなるコマンドがあります。その場合、この警告メッセージは無視してください。

## java.util.logging のレベル

WARNING

### KDLR10007-W

```
Failed to output a temporary file for the snapshot log. (id = aa....aa, path = bb....bb, cause = cc....cc)
```

aa....aa : スナップショットログ ID

bb....bb : 出力先ファイルパス

cc....cc : 原因例外メッセージ

#### 説明

スナップショットログのための一時ファイルの出力に失敗しました。

#### 対処

次の内容を確認し、スナップショットログを手動で再収集してください。

- 出力先ディレクトリパスが存在するか確認してください。



- 出力先のアクセス権が正しく設定されているか確認してください。
- 原因例外メッセージを参照し、原因を取り除いてください。

## java.util.logging のレベル

WARNING

### KDLR10008-W

An error occurred during the storing of an entry in the snapshot log. (id = aa....aa, path = bb....bb, entry = cc....cc, cause = dd....dd)

aa....aa : スナップショットログ ID

bb....bb : 出力先ファイルパス

cc....cc : エントリパス

dd....dd : 原因例外メッセージ

#### 説明

スナップショットログファイルにエントリを格納する際、エラーが発生しました。

#### 対処

次の内容を確認し、スナップショットログを手動で再収集してください。

- 出力先ファイルパスが存在するか確認してください。
- 出力先のアクセス権が正しく設定されているか確認してください。
- エントリパスのアクセス権が正しく設定されているか確認してください。
- 原因例外メッセージを参照し、原因を取り除いてください。

## java.util.logging のレベル

WARNING

### KDLR10009-I

Collection of the snapshot log will now start.

#### 説明

スナップショットログ収集を開始します。

#### 対処

なし

### KDLR10010-I

Collection of the snapshot log has finished successfully.

## 説明

スナップショットログ収集が完了しました。

## 対処

なし

## KDLR10011-E

An invalid argument has been specified. (argument = aa....aa)

aa....aa : 引数

## 説明

不正な引数が指定されました。

## 対処

引数を修正してください。

## KDLR10012-E

An invalid option value has been specified. (option = aa....aa, value = bb....bb)

aa....aa : オプション名

bb....bb : オプション値

## 説明

不正なオプション値が指定されました。

## 対処

オプション値を修正してください。

## KDLR10013-E

An error occurred during communication with the endpoint of the snapshot log collection REST API. (endpoint = aa....aa)

aa....aa : エンドポイント

## 説明

スナップショットログ収集 REST API のエンドポイントとの通信中にエラーが発生しました。

## 対処

次の内容を確認してください。

- Tomcat プロセスモニタが起動しているか確認してください。
- --port オプション指定値が正しいか確認してください。

- --endpoint オプション指定値が正しいか確認してください。

## KDLR10014-E

An error was returned from the snapshot log collection REST API.

### 説明

スナップショットログ収集 REST API がエラーを返しました。

### 対処

このメッセージに続いて表示されるエラーメッセージを確認してください。

## KDLR10015-E

The collected snapshot log file is broken.

### 説明

収集したスナップショットログファイルが壊れています。

### 対処

再度スナップショットログ収集コマンドを実行してください。

## KDLR10016-E

An invalid value has been specified for the request parameter of the snapshot log collection REST API. (name = aa....aa, value = bb....bb)

aa....aa : パラメタ名

bb....bb : パラメタ値

### 説明

スナップショットログ収集 REST API のリクエストパラメタに不正な値が指定されました。

### 対処

パラメタ値を修正してください。

java.util.logging のレベル

SEVERE

## KDLR10017-E

Failed to output the snapshot log with the snapshot log collection REST API. (cause = aa....aa)

aa....aa : 原因例外メッセージ, または原因情報

## 説明

スナップショットログ収集 REST API からのスナップショットログ出力に失敗しました。

## 対処

原因例外メッセージ, または原因情報を参照し, 原因を取り除いてください。

## java.util.logging のレベル

SEVERE

## KDLR10018-I

```
Sending the snapshot log has finished successfully. (id = aa....aa)
```

aa....aa : スナップショットログ ID

## 説明

スナップショットログの送信が完了しました。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR10019-I

```
The snapshot log file has been deleted. (id = aa....aa, path = bb....bb)
```

aa....aa : スナップショットログ ID

bb....bb : 出力先ファイルパス

## 説明

スナップショットログファイルが削除されました。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR10020-E

```
Failed to send a snapshot log. (id = aa....aa, path = bb....bb, cause = cc....cc)
```

aa....aa : スナップショットログ ID

bb....bb : 出力先ファイルパス

cc....cc：原因例外メッセージ

#### 説明

スナップショットログの送信が失敗しました。

#### 対処

原因例外メッセージを参照し、原因を取り除いてください。

java.util.logging のレベル

SEVERE

### KDLR10021-W

```
Failed to delete the snapshot log file. (id = aa....aa, path = bb....bb, cause = cc....cc)
```

aa....aa：スナップショットログ ID

bb....bb：出力先ファイルパス

cc....cc：原因例外メッセージ

#### 説明

スナップショットログファイルの削除が失敗しました。

#### 対処

必要に応じて、手動でファイルを削除してください。

java.util.logging のレベル

WARNING

### KDLR10022-E

```
An error occurred during execution of the command. (command = aa....aa, exit-status = bb....bb, message = cc....cc)
```

aa....aa：コマンド

bb....bb：終了ステータス

cc....cc：エラーメッセージ

#### 説明

コマンドの実行中にエラーが発生しました。

#### 対処

次に出力されるメッセージの対処を参照してください。

## KDLR10023-I

Deployed application was detected. (context = aa....aa)

aa....aa : アプリケーションのコンテキストパス

### 説明

アプリケーションのデプロイを検知しました。

### 対処

なし

java.util.logging のレベル

INFO

## KDLR10024-I

Application's deployment descriptor was detected. (context = aa....aa, path = bb....bb)

aa....aa : アプリケーションのコンテキストパス

bb....bb : デプロイメント・ディスクリプタのパス

### 説明

アプリケーション中に含まれるデプロイメント・ディスクリプタを検知しました。

### 対処

なし

java.util.logging のレベル

INFO

## KDLR10025-W

Failed to collect the application's deployment descriptor. (context = aa....aa, path = bb....bb, cause = cc....cc)

aa....aa : アプリケーションのコンテキストパス

bb....bb : デプロイメント・ディスクリプタのパス

cc....cc : 原因例外メッセージ

### 説明

アプリケーション中に含まれるデプロイメント・ディスクリプタの収集に失敗しました。

## 対処

スナップショットログを保守員に送付する際、収集に失敗したデプロイメント・ディスクリプタファイルも添付してください。

## java.util.logging のレベル

WARNING

## KDLR10026-I

```
Sending of the path of the application's deployment descriptor to the Tomcat process monitor will now start. (context = aa....aa)
```

aa....aa : アプリケーションのコンテキストパス

## 説明

Tomcat プロセスモニタへアプリケーションのデプロイメント・ディスクリプタパスの送信を開始します。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR10027-I

```
Sending of the path of the application's deployment descriptor to the Tomcat process monitor has been finished successfully. (context = aa....aa)
```

aa....aa : アプリケーションのコンテキストパス

## 説明

Tomcat プロセスモニタへのアプリケーションのデプロイメント・ディスクリプタパスの送信が完了しました。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR10028-W

```
An error occurred during the sending of the path of the application's deployment descriptor to the Tomcat process monitor. (context = aa....aa, cause = bb....bb)
```

aa....aa : アプリケーションのコンテキストパス

bb....bb：原因例外メッセージ

#### 説明

Tomcat プロセスモニタへアプリケーションのデプロイメント・ディスクリプタパスを送信する際、エラーが発生しました。

#### 対処

スナップショットログを保守員に送付する際、送信に失敗したアプリケーションのデプロイメント・ディスクリプタファイルも送付してください。

#### java.util.logging のレベル

WARNING

#### KDLR10029-W

```
Failed to delete the output file. (path = aa....aa)
```

aa....aa：ファイルパス

#### 説明

出力ファイルの削除に失敗しました。

#### 対処

必要に応じて、手でファイルを削除してください。

#### KDLR10030-E

```
Failed to send an error message. (cause = aa....aa)
```

aa....aa：原因例外メッセージ

#### 説明

エラーメッセージの送信が失敗しました。

#### 対処

原因例外メッセージを参照し、原因を取り除いてください。

#### java.util.logging のレベル

SEVERE

#### KDLR10031-I

```
Output of the snapshot log was canceled. (id = aa....aa, reason = bb....bb)
```

aa....aa：スナップショットログ ID

bb....bb：キャンセルした理由



## 説明

スナップショットログの出力はキャンセルされました。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR20100-I

```
The health check of the Tomcat process will now start.
```

## 説明

Tomcat サーバプロセス上の稼働監視が開始します。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR20101-E

```
An error occurred during communication with the health check component. (URL = aa....aa, HTTP-status-code = bb....bb)
```

aa....aa : URL

bb....bb : HTTP ステータスコード

## 説明

稼働監視コンポーネントへの通信中にエラーが発生しました。

## 対処

Tomcat プロセスモニタが正常に稼働していることを確認してください。

## java.util.logging のレベル

SEVERE

## KDLR20102-E

```
A timeout occurred during communication with the health check component. (URL = aa....aa, cause = bb....bb)
```

aa....aa : URL

bb....bb : 原因例外メッセージ

## 説明

稼働監視コンポーネントへの通信中にタイムアウトが発生しました。

## 対処

タイムアウトに関連するパラメタ値を見直してください。KDLR20100-I の直後に出力されている場合は、Tomcat プロセスモニタのポートのパラメタ値と、指定したポートのファイアウォールの設定を見直してください。

## java.util.logging のレベル

SEVERE

## KDLR20103-E

```
An error occurred during communication with the health check component. (port = aa....aa, request-uri = bb....bb, cause = cc....cc)
```

aa....aa : ポート番号

bb....bb : リクエスト URI

cc....cc : 原因例外メッセージ

## 説明

稼働監視コンポーネントへの通信中にエラーが発生しました。

## 対処

Tomcat プロセスモニタが正常に稼働していることを確認してください。解決しない場合は、保守員に連絡してください。

## java.util.logging のレベル

SEVERE

## KDLR20105-E

```
An error occurred while processing lifecycle event. (type = aa....aa, data = bb....bb, src = cc....cc, cause = dd....dd)
```

aa....aa : イベント種別

bb....bb : イベントデータ

cc....cc : イベント発行元

dd....dd : 原因例外メッセージ

## 説明

ライフサイクルイベント処理中にエラーが発生しました。

## 対処

保守員に連絡してください。

java.util.logging のレベル

SEVERE

## KDLR20106-E

```
An error occurred while processing container event. (type = aa....aa, data = bb....bb, src = cc....cc, cause = dd....dd)
```

aa....aa : イベント種別

bb....bb : イベントデータ

cc....cc : イベント発行元

dd....dd : 原因例外メッセージ

## 説明

コンテナイベント処理中にエラーが発生しました。

## 対処

保守員に連絡してください。

java.util.logging のレベル

SEVERE

## KDLR20200-I

```
The health check component will now start.
```

## 説明

稼働監視コンポーネントが開始します。

## 対処

なし

java.util.logging のレベル

INFO

## KDLR20201-E

```
An error occurred while starting health check component. (cause = aa....aa)
```

aa....aa : 原因例外メッセージ

## 説明

稼働監視コンポーネントの開始中にエラーが発生しました。

## 対処

KDLR20205-E または KDLR20206-E が出力されている場合は、各メッセージに従って対処してください。それ以外の場合は、保守員に連絡してください。

## java.util.logging のレベル

SEVERE

## KDLR20202-E

An error occurred while receiving data on the servlet. (request-uri = aa....aa, cause = bb....bb)

aa....aa：リクエスト URI

bb....bb：原因例外メッセージ

## 説明

サーブレットでのデータ受信中に例外が発生しました。

## 対処

不正なアクセスが起きていないか確認してください。

## java.util.logging のレベル

SEVERE

## KDLR20203-E

An error occurred while processing data on the servlet. (request-uri = aa....aa, cause = bb....bb)

aa....aa：リクエスト URI

bb....bb：原因例外メッセージ

## 説明

サーブレットでのデータ処理中に例外が発生しました。

## 対処

保守員に連絡してください。

## java.util.logging のレベル

SEVERE

## KDLR20204-E

An error occurred while sending data on the servlet. (request-uri = aa....aa, cause = bb....bb)

aa....aa：リクエスト URI

bb....bb：原因例外メッセージ

#### 説明

サーブレットでのデータ送信中に例外が発生しました。

#### 対処

保守員に連絡してください。

java.util.logging のレベル

SEVERE

### KDLR20205-E

```
The Tomcat log directory cannot be found. (log-directory = aa....aa)
```

aa....aa：ログディレクトリ

#### 説明

Tomcat ログディレクトリが見つかりません。

#### 対処

Tomcat ログ (Tomcat サーバプロセスが出力するログ) に関連するパラメタ値を見直してください。

java.util.logging のレベル

SEVERE

### KDLR20206-E

```
An error occurred while trying to access Tomcat log files. (log-directory = aa....aa, log-file-name = bb....bb, cause = cc....cc)
```

aa....aa：ログディレクトリ

bb....bb：ログファイル名 (glob 形式)

cc....cc：原因例外メッセージ

#### 説明

Tomcat ログファイルへのアクセス中にエラーが発生しました。

#### 対処

common.base と snapshot.include.paths で指定されたディレクトリや更新チェック用ログファイルのアクセス権と、Tomcat ログに関連するパラメタ値を見直してください。

common.base, および snapshot.include.paths は、config.properties (本製品の設定ファイル) のプロパティです。common.base については「[12.2.4\(1\) 本製品全体に関するプロパティ](#)」を、

snapshot.include.paths については「12.2.4(4) スナップショットログ収集機能に関するプロパティ」を参照してください。

#### java.util.logging のレベル

SEVERE

#### KDLR20207-E

```
Any updated Tomcat log files cannot be found since startup. (log-directory = aa....aa, log-file-name = bb....bb)
```

aa....aa : ログディレクトリ

bb....bb : ログファイル名 (glob 形式)

#### 説明

起動してから更新された Tomcat ログファイルが見つかりません。

#### 対処

スナップショットログを収集する設定と、Tomcat ログに関連するパラメタ値を見直してください。

#### java.util.logging のレベル

SEVERE

#### KDLR20208-E

```
A timeout occurred during the wait for server initialization to finish. (init-delay = aa....aa)
```

aa....aa : サーバ初期化完了待ちタイムアウト時間

#### 説明

サーバ初期化完了待ちがタイムアウトしました。

#### 対処

Tomcat サーバプロセスへの稼働監視用ライフサイクルリスナーの設定を見直してください。ライフサイクルリスナーの設定が正常な場合は、タイムアウトに関連するパラメタ値を見直してください。

#### java.util.logging のレベル

SEVERE

#### KDLR20209-E

```
A timeout occurred during the wait for server startup to finish. (start-delay = aa....aa)
```

aa....aa : サーバ開始完了待ちタイムアウト時間

#### 説明

サーバ開始完了待ちがタイムアウトしました。

## 対処

Tomcat サーバプロセス開始が完了しない原因が Tomcat サーバプロセスが出力するログなどから分かる場合には、その原因を取り除いてください。Tomcat サーバプロセス開始に時間が掛かっているだけの場合は、タイムアウトに関連するパラメタ値を見直してください。

### java.util.logging のレベル

SEVERE

## KDLR20210-I

Monitoring the HTTP connector by health check request will now start.

## 説明

ヘルスチェックリクエストによる HTTP コネクタの監視を開始します。

## 対処

なし

### java.util.logging のレベル

INFO

## KDLR20211-W

Available HTTP connectors cannot be found on the Tomcat process.

## 説明

Tomcat サーバプロセスに HTTP プロトコルが使えるコネクタが存在しません。

## 対処

HTTP プロトコルが使えるコネクタを追加してください。Connector コンポーネントの SSLEnabled 属性が true の場合は HTTPS 用のコネクタと判断します。また、ポートにバインドできていないコネクタは、有効な HTTP コネクタではないと判断しますので、Connector コンポーネントの port 属性や Server コンポーネントの portOffset 属性を見直してください。

### java.util.logging のレベル

WARNING

## KDLR20214-W

An error occurred during communication with the HTTP connector. (URL = aa....aa, cause = bb....bb)

aa....aa : URL

bb....bb : 原因例外メッセージ

## 説明

HTTP コネクタへの通信中にエラーが発生しました。

## 対処

Tomcat サーバプロセスが正常に稼働していることを確認してください。ヘルスチェック HTTP リクエストが失敗する原因がログなどから分かる場合は、その原因を取り除いてください。

## java.util.logging のレベル

WARNING

## KDLR20215-W

```
An error occurred during communication with the HTTP connector. (URL = aa....aa, HTTP-status-code = bb....bb)
```

aa....aa : URL

bb....bb : HTTP ステータスコード

## 説明

HTTP コネクタへの通信中にエラーが発生しました。

## 対処

Tomcat サーバプロセスが正常に稼働していることを確認してください。ヘルスチェック HTTP リクエストが失敗する原因がログなどから分かる場合は、その原因を取り除いてください。

## java.util.logging のレベル

WARNING

## KDLR20216-W

```
A timeout occurred during communication with the HTTP connector. (URL = aa....aa, cause = bb....bb)
```

aa....aa : URL

bb....bb : 原因例外メッセージ

## 説明

HTTP コネクタへの通信中にタイムアウトが発生しました。

## 対処

Tomcat サーバプロセスが正常に稼働していることを確認してください。正常に稼働している場合は、タイムアウトに関連するパラメタ値を見直してください。

Tomcat サーバプロセス開始直後に出力されている場合は、Tomcat サーバプロセスのポートのパラメタ値と、指定したポートのファイアウォールの設定を見直してください。



## java.util.logging のレベル

WARNING

### KDLR20217-E

An error occurred during communication with the HTTP connector. (hostname = aa....aa, port = bb....bb, request-uri = cc....cc, cause = dd....dd)

aa....aa : ホスト名

bb....bb : ポート番号

cc....cc : リクエスト URI

dd....dd : 原因例外メッセージ

#### 説明

HTTP コネクタへの通信中にエラーが発生しました。

#### 対処

Tomcat サーバプロセスが正常に稼働していることを確認してください。解決しない場合は、保守員に連絡してください。

## java.util.logging のレベル

SEVERE

### KDLR20218-I

Communication with the HTTP connector was successfully established for the first time since an error or timeout occurred. (URL = aa....aa)

aa....aa : URL

#### 説明

エラーまたはタイムアウトが発生してから初めて HTTP コネクタへの通信が成功しました。

#### 対処

なし

## java.util.logging のレベル

INFO

### KDLR20219-I

The StuckThreadDetectionValve monitoring of the health check component will now start.

## 説明

稼働監視コンポーネントの停滞検知バルブ監視を開始します。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR20220-W

```
A stuck thread was detected by the StuckThreadDetectionValve. (valve-object-name = aa....aa)
```

aa....aa : バルブの ObjectName

## 説明

停滞したスレッドを検知しました。

## 対処

Tomcat サーバプロセスが正常に稼働しているか確認してください。停滞したスレッドやインタラプトされたスレッドの原因がログなどから分かる場合は、その原因を取り除いてください。

## java.util.logging のレベル

WARNING

## KDLR20221-I

```
Stuck thread was no longer detected by the StuckThreadDetectionValve for the first time since the stuck thread detection. (valve-object-name = aa....aa)
```

aa....aa : バルブの ObjectName

## 説明

停滞スレッド検知以降で初めて、停滞したスレッドが検知されませんでした。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR20222-I

```
The heartbeat monitoring of the health check component will now start.
```

## 説明

稼働監視コンポーネントのハートビート監視を開始します。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR20223-W

A timeout occurred during the wait for the heartbeat from the Tomcat process. (heartbeat-delay = aa....aa, last-heartbeat-received-time = bb....bb)

aa....aa : ハートビートタイムアウト時間

bb....bb : ハートビート最終受信時刻 (初回の受信時は"-")

## 説明

ハートビートがタイムアウトしました。

## 対処

Tomcat サーバプロセスが正常に稼働しているか確認してください。正常に稼働している場合は、ハートビートに関連するパラメタ値を見直してください。

## java.util.logging のレベル

WARNING

## KDLR20224-I

The heartbeat has been received for the first time since a timeout occurred.

## 説明

タイムアウト発生以降で初めて、ハートビートを受信しました。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR20225-I

The aa....aa event occurred. The specified action will be executed. (action = [bb....bb], event-property = cc....cc)

aa....aa : イベントタイプ (次のうち 1 つ)

- heartbeatdelay : ハートビート監視
- httprequest : ヘルスチェック
- stuckthread : リクエスト停滞監視
- startdelay : プロセス起動監視
- shutdown : プロセス生存監視

bb....bb : 実行されるアクション (次のうち 1 つまたは複数でコンマ区切りで表示)

- snapshot : スナップショットログ収集
- terminate : Tomcat サーバプロセス停止
- ユーザコマンド定義の ID : ユーザコマンドの実行

cc....cc : JSON 形式のイベントプロパティ

#### 説明

aa....aa イベントが発生しました。指定されたアクションが実行されます。

#### 対処

なし

#### java.util.logging のレベル

INFO

#### KDLR20226-I

Execution of the user command for the aa....aa event will now start. (id = bb....bb, reason = cc....cc)

aa....aa : イベントタイプ (次のうち 1 つ)

- heartbeatdelay : ハートビート監視
- httprequest : ヘルスチェック
- stuckthread : リクエスト停滞監視
- startdelay : プロセス起動監視
- shutdown : プロセス生存監視

bb....bb : ユーザコマンド定義の ID

- config.properties (本製品の設定ファイル) 内のプロパティ healthcheck.usercommand.defs.<group-id>.command の<group-id>部分  
このプロパティについては、「[12.2.4\(3\) 稼働監視機能に関するプロパティ](#)」を参照してください。

cc....cc : イベント発行要因

- failure : 障害イベント
- recovery : 回復イベント

#### 説明

aa....aa イベントに対するユーザコマンドの実行を開始します。

#### 対処

なし

#### java.util.logging のレベル

INFO

### KDLR20227-I

```
Execution of the user command for the aa....aa event has finished. (id = bb....bb, exit-status = cc....cc)
```

aa....aa : イベントタイプ (次のうち 1 つ)

- heartbeatdelay : ハートビート監視
- httprequest : ヘルスチェック
- stuckthread : リクエスト停滞監視
- startdelay : プロセス起動監視
- shutdown : プロセス生存監視

bb....bb : ユーザコマンド定義の ID

- config.properties (本製品の設定ファイル) 内のプロパティ healthcheck.usercommand.defs.<group-id>.command の<group-id>部分  
このプロパティについては、「[12.2.4\(3\) 稼働監視機能に関するプロパティ](#)」を参照してください。

cc....cc : コマンドの終了ステータス

#### 説明

aa....aa イベントに対するユーザコマンドの実行を終了しました。

#### 対処

なし

#### java.util.logging のレベル

INFO

### KDLR20228-E

```
Failed to execute a user command for the aa....aa event. (id = bb....bb, cause = cc....cc)
```

aa....aa : イベントタイプ (次のうち 1 つ)

- heartbeatdelay : ハートビート監視
- httprequest : ヘルスチェック
- stuckthread : リクエスト停滞監視
- startdelay : プロセス起動監視
- shutdown : プロセス生存監視

bb....bb : ユーザコマンド定義の ID

- config.properties (本製品の設定ファイル) 内のプロパティ healthcheck.usercommand.defs.<group-id>.command の<group-id>部分  
このプロパティについては、「[12.2.4\(3\) 稼働監視機能に関するプロパティ](#)」を参照してください。

cc....cc : 原因例外メッセージ

## 説明

aa....aa イベントに対するユーザコマンドの実行に失敗しました。

## 対処

ユーザコマンド定義の各プロパティを次の観点で見直し、失敗の原因を取り除いてください。

- プロパティで指定した実行コマンドが配置されているかどうかを確認してください。
- プロパティで指定した実行コマンドに適切な実行権限が与えられているかどうかを確認してください。
- ユーザコマンドの実行中に Tomcat プロセスモニタが終了した場合には、原因例外 `InterruptedException` を伴って出力されることがあります。その場合は、ユーザコマンド定義の ID を基に、ユーザコマンドが正常に完了したかどうかを確認してください。確認の方法は、各コマンドによって異なります。各コマンドに必要な回復手順がある場合は、その回復手順を実施してください。

問題が解決しない場合は、保守員に連絡してください。

## java.util.logging のレベル

SEVERE

## KDLR20229-E

The group ID specified for the property is not defined. (property = aa....aa, group-id = bb....bb)

aa....aa : グループ ID を参照先として指定するプロパティキー

bb....bb : 指定されたグループ ID の値

## 説明

プロパティで指定されたグループ ID が定義されていません。

## 対処

config.properties（本製品の設定ファイル）を確認し、指定されたグループ ID（<group-id>）を持つ次のユーザコマンド定義が存在するか確認してください。

```
healthcheck.usercommand.defs.<group-id>.command
```

存在しない場合は、プロパティの説明に従ってユーザコマンドを定義し、適切な<group-id>を指定してください。

healthcheck.usercommand.defs.<group-id>.command については、「[12.2.4\(3\) 稼働監視機能に関するプロパティ](#)」を参照してください。

## java.util.logging のレベル

SEVERE

## KDLR20230-W

Some user commands might be running while the health check component is stopped.

## 説明

稼働監視コンポーネントの停止中に、幾つかのユーザコマンドが実行中の可能性があります。

## 対処

KDLR20228-E や KDLR20226-I メッセージに出力されるユーザコマンド定義の ID を基に、ユーザコマンドが正常に完了したかどうかを確認してください。確認の方法は、各コマンドによって異なります。各コマンドに必要な回復手順がある場合は、その回復手順を実施してください。

## java.util.logging のレベル

WARNING

## KDLR20231-W

Execution of the user commands waiting execution has canceled. (id = aa....aa)

aa....aa：ユーザコマンド定義の ID

- config.properties（本製品の設定ファイル）内のプロパティ healthcheck.usercommand.defs.<group-id>.command の<group-id>部分（カンマ区切り表示）このプロパティについては、「[12.2.4\(3\) 稼働監視機能に関するプロパティ](#)」を参照してください。

## 説明

実行待ちのユーザコマンドをキャンセルしました。

## 対処

ユーザコマンド定義の ID を基に、キャンセルされたユーザコマンドを確認し、各コマンドに手動実行などの必要な手順がある場合は、実施してください。

キャンセルの原因がユーザコマンドの実行に時間が掛かっているだけの場合は、ユーザコマンドのタイムアウトに関連するパラメタ値やユーザコマンドの実行で使用するスレッドプールのサイズを見直してください。

## java.util.logging のレベル

WARNING

### KDLR20232-W

```
A timeout occurred during execution of the user command for the aa....aa event. (id = bb....bb, timeout = cc....cc)
```

aa....aa : イベントタイプ (次のうち 1 つ)

- heartbeatdelay : ハートビート監視
- httprequest : ヘルスチェック
- stuckthread : リクエスト停滞監視
- startdelay : プロセス起動監視
- shutdown : プロセス生存監視

bb....bb : ユーザコマンド定義の ID

- config.properties (本製品の設定ファイル) 内のプロパティ healthcheck.usercommand.defs.<group-id>.command の<group-id>部分  
このプロパティについては、「[12.2.4\(3\) 稼働監視機能に関するプロパティ](#)」を参照してください。

cc....cc : タイムアウト時間

#### 説明

aa....aa イベントに対するユーザコマンドの実行がタイムアウトしました。

#### 対処

ユーザコマンド定義の ID を基に、ユーザコマンドが正常に完了したかどうかを確認してください。確認の方法は、コマンドによって異なります。各コマンドに必要な回復手順がある場合は、その回復手順を実施してください。

ユーザコマンドの実行に時間が掛かっているだけの場合は、ユーザコマンド定義の ID を基に、ユーザコマンドのタイムアウトに関連するパラメタ値を見直してください。

## java.util.logging のレベル

WARNING

### KDLR30000-I

```
The tracer has started.
```



#### 説明

トレーサを開始しました。

#### 対処

なし

java.util.logging のレベル

INFO

### KDLR30001-E

An error occurred during the processing to start the tracer.

#### 説明

トレーサの開始処理でエラーが発生しました。

#### 対処

保守員に連絡してください。

java.util.logging のレベル

SEVERE

### KDLR30002-I

The tracer has stopped.

#### 説明

トレーサを停止しました。

#### 対処

なし

java.util.logging のレベル

INFO

### KDLR30003-E

An error occurred during the processing to stop the tracer.

#### 説明

トレーサの停止処理でエラーが発生しました。

#### 対処

保守員に連絡してください。

java.util.logging のレベル

SEVERE

## KDLR30004-W

The local address cannot be obtained. The local address will be logged as "0.0.0.0".

### 説明

ローカルアドレスを取得できません。ローカルアドレスを 0.0.0.0 に設定します。

### 対処

java.net.InetAddress.getLocalHost メソッドが java.net.UnknownHostException をスローする問題を解決してください。

### java.util.logging のレベル

WARNING

## KDLR30005-W

The IPv4 local address cannot be obtained. The local address will be logged as "0.0.0.0".

### 説明

IPv4 のローカルアドレスを取得できません。ローカルアドレスを 0.0.0.0 に設定します。

### 対処

java.net.InetAddress.getLocalHost メソッドが java.net.Inet4Address オブジェクトを返却しない問題を解決してください。

### java.util.logging のレベル

WARNING

## KDLR40000-E

Failed to start the Tomcat process monitor.

### 説明

Tomcat プロセスモニタの起動に失敗しました。

### 対処

このメッセージの前に出力されているエラーメッセージを確認してください。

### java.util.logging のレベル

SEVERE

## KDLR40001-W

A signal from the OS has been received.

### 説明

OS からのシグナルを受信しました。

## 対処

なし

## java.util.logging のレベル

WARNING

## KDLR40002-W

A request to stop the Tomcat process will be sent. (process-id = aa....aa)

aa....aa : Tomcat サーバプロセスのプロセス ID

## 説明

Tomcat サーバプロセスの停止を試みます。

## 対処

なし

## java.util.logging のレベル

WARNING

## KDLR40003-I

The Tomcat process will now start.

## 説明

Tomcat サーバプロセスを開始します。

## 対処

なし

## java.util.logging のレベル

INFO

## KDLR40004-I

The end of the Tomcat process was detected. (process-id = aa....aa, exit-status = bb....bb)

aa....aa : Tomcat サーバプロセスのプロセス ID

bb....bb : 終了ステータス

## 説明

Tomcat サーバプロセスの終了を検知しました。

## 対処

なし

## java.util.logging のレベル

INFO

### KDLR40005-I

The monitor will request collection of the snapshot log. (process-id = aa....aa)

aa....aa : Tomcat サーバプロセスのプロセス ID

#### 説明

スナップショットログ収集要求を発行します。

#### 対処

なし

## java.util.logging のレベル

INFO

### KDLR40006-I

A request to stop the Tomcat process has been received from the health check module.  
(process-id = aa....aa)

aa....aa : Tomcat サーバプロセスのプロセス ID

#### 説明

稼働監視モジュールから Tomcat サーバプロセスの終了要求を受信しました。

#### 対処

なし

## java.util.logging のレベル

INFO

### KDLR40007-E

Failed to start the Tomcat process. (cause = aa....aa)

aa....aa : 原因例外メッセージ

#### 説明

Tomcat サーバプロセスの起動に失敗しました。

#### 対処

Tomcat のスクリプトまたは本製品が提供するスクリプトに、適切な実行権限が設定されているか確認してください。

## java.util.logging のレベル

SEVERE

### KDLR40008-I

```
The Tomcat process has started. (process-id = aa....aa)
```

aa....aa : Tomcat サーバプロセスのプロセス ID

#### 説明

Tomcat サーバプロセスを起動しました。

#### 対処

なし

## java.util.logging のレベル

INFO

### KDLR40009-E

```
Failed to initialize the HTTP function of the Tomcat process monitor. (cause = aa....aa)
```

aa....aa : 原因例外メッセージ, または原因情報

#### 説明

Tomcat プロセスモニタの HTTP 機能の初期化に失敗しました。

#### 対処

Tomcat プロセスモニタの HTTP 機能に関する設定値を確認してください。

## java.util.logging のレベル

SEVERE

### KDLR40010-W

```
The Tomcat process monitor will stop soon. However, the Tomcat process might continue to run. (process-id = aa....aa)
```

aa....aa : Tomcat サーバプロセスのプロセス ID

#### 説明

Tomcat プロセスモニタはまもなく停止します。ただし, Tomcat サーバプロセスは動き続けている可能性があります。

#### 対処

必要に応じて, OS のコマンドを利用して Tomcat サーバプロセスを停止してください。

## java.util.logging のレベル

WARNING

### KDLR40012-E

SecurityManager is not allowed on the Tomcat process monitor.

#### 説明

Tomcat プロセスモニタでセキュリティマネージャは許可されていません。

#### 対処

Tomcat プロセスモニタの config.properties (本製品の設定ファイル) 内のプロパティ monitor.jvm.options から、-Djava.security.manager を削除してください。

## java.util.logging のレベル

SEVERE

### KDLR40013-W

Failed to finalize the HTTP function of the Tomcat process monitor. (cause = aa....aa)

aa....aa : 原因例外メッセージ

#### 説明

Tomcat プロセスモニタの HTTP 機能の終了処理に失敗しました。

#### 対処

このメッセージ以前にエラーメッセージが出力されて、Tomcat プロセスモニタの起動に失敗している場合は、前のエラーメッセージの対処を実施してください。

このメッセージ以前にエラーメッセージが出力されていない場合は、この警告メッセージを無視してください。

## java.util.logging のレベル

WARNING

### KDLR99998-E

An unexpected exception occurred. (cause = aa....aa)

aa....aa : 原因例外メッセージ

#### 説明

予期しない例外が発生しました。

#### 対処

保守員に連絡してください。

java.util.logging のレベル

SEVERE

# 15

## JavaVM 起動オプション

JavaVM 起動オプションについて説明します。日立 JavaVM を使用する場合と、他社製 JavaVM を使用する場合とで、指定できる JavaVM 起動オプションが異なります。



## 15.1 日立 JavaVM を使用する場合

日立 JavaVM 独自起動オプションのデフォルト値と変更可否について説明します。

表 15-1 日立 JavaVM 独自起動オプションのデフォルト値と変更可否 (Tomcat プロセスモニタ)

オプション名称	日立 JavaVM のデフォルト値	Tomcat プロセスモニタ	
		デフォルト値	変更可否
-XX:[+ -]HitachiThreadDump	-XX:+HitachiThreadDump	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiThreadDumpToStdout	-XX:+HitachiThreadDumpToStdout	-XX:-HitachiThreadDumpToStdout	不可
-XX:[+ -]HitachiThreadDumpWithHashCode	-XX:+HitachiThreadDumpWithHashCode	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiThreadDumpWithCpuTime	-XX:+HitachiThreadDumpWithCpuTime	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiThreadDumpWithBlockCount	-XX:+HitachiThreadDumpWithBlockCount	JavaVM デフォルト値と同じ	不可
-XX:HitachiJavaLog	-XX:HitachiJavaLog:javalog	-XX:HitachiJavaLog:\$ {common.base}/pmjavalog	可※
-XX:HitachiJavaLogFileSize	-XX:HitachiJavaLogFileSize=256k	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiJavaLogNoMoreOutput	-XX:+HitachiJavaLogNoMoreOutput	JavaVM デフォルト	不可

オプション名称	日立 JavaVM のデフォルト値	Tomcat プロセスモニタ	
		デフォルト値	変更可否
		ト値と同じ	
-XX:HitachiJavaLogNumberOfFile	-XX:HitachiJavaLogNumberOfFile=4	JavaVM デフォルト値と同じ	不可
-XX:[+ -]JavaLogAsynchronous	-XX:-JavaLogAsynchronous	JavaVM デフォルト値と同じ	不可
-XX:[+ -]StandardLogToHitachiJavaLog	-XX:-StandardLogToHitachiJavaLog	-XX:+StandardLogToHitachiJavaLog	不可
-XX:[+ -]HitachiOutputMilliTime	-XX:-HitachiOutputMilliTime	-XX:+HitachiOutputMilliTime	不可
-XX:[+ -]HitachiVerboseGC	-XX:-HitachiVerboseGC	-XX:+HitachiVerboseGC	不可
-XX:[+ -]HitachiCommaVerboseGC	-XX:-HitachiCommaVerboseGC	JavaVM デフォルト値と同じ	不可
-XX:HitachiVerboseGCIntervalTime	-XX:HitachiVerboseGCIntervalTime=0	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiVerboseGCPrintCause	-XX:+HitachiVerboseGCPrintCause	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiVerboseGCPrintDate	-XX:+HitachiVerboseGCPrintDate	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiVerboseGCCpuTime	-XX:+HitachiVerboseGCCpuTime	JavaVM デフォルト	不可

オプション名称	日立 JavaVM のデフォルト値	Tomcat プロセスモニタ	
		デフォルト値	変更可否
		ト値と同じ	
-XX:[+ -]HitachiVerboseGCPrintTenuringDistribution	-XX:-HitachiVerboseGCPrintTenuringDistribution	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiVerboseGCPrintJVMInternalMemory	-XX:+HitachiVerboseGCPrintJVMInternalMemory	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiVerboseGCPrintThreadCount	-XX:+HitachiVerboseGCPrintThreadCount	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiVerboseGCPrintDeleteOnExit	-XX:+HitachiVerboseGCPrintDeleteOnExit	JavaVM デフォルト値と同じ	不可
-XX:[+ -]PrintCodeCacheInfo	-XX:+PrintCodeCacheInfo	JavaVM デフォルト値と同じ	不可
-XX:CodeCacheInfoPrintRatio	-XX:CodeCacheInfoPrintRatio=80	JavaVM デフォルト値と同じ	不可
-XX:[+ -]PrintCodeCacheFullMessage	-XX:+PrintCodeCacheFullMessage	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiOutOfMemoryCause	-XX:-HitachiOutOfMemoryCause	-XX:+HitachiOutOfMemoryCause	不可
-XX:[+ -]HitachiOutOfMemoryStackTrace	-XX:-HitachiOutOfMemoryStackTrace	-XX:+HitachiOutOfMemoryStackTrace	不可

オプション名称	日立 JavaVM のデフォルト値	Tomcat プロセスモニタ	
		デフォルト値	変更可否
-XX:HitachiOutOfMemoryStackTraceLineSize	-XX:HitachiOutOfMemoryStackTraceLineSize=1024	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiOutOfMemorySize	-XX:-HitachiOutOfMemorySize	-XX:+HitachiOutOfMemorySize	不可
-XX:[+ -]HitachiOutOfMemoryAbort	-XX:-HitachiOutOfMemoryAbort	-XX:+HitachiOutOfMemoryAbort	不可
-XX:[+ -]HitachiOutOfMemoryAbortThreadDump	-XX:+HitachiOutOfMemoryAbortThreadDump	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiOutOfMemoryAbortThreadDumpWithJHeapProf	-XX:-HitachiOutOfMemoryAbortThreadDumpWithJHeapProf	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiOutOfMemoryHandling	-XX:-HitachiOutOfMemoryHandling	JavaVM デフォルト値と同じ	不可
-XX:HitachiOutOfMemoryHandlingMaxThrowCount	-XX:HitachiOutOfMemoryHandlingMaxThrowCount=60	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiJavaClassLibTrace	-XX:-HitachiJavaClassLibTrace	-XX:+HitachiJavaClassLibTrace	不可
-XX:HitachiJavaClassLibTraceLineSize	-XX:HitachiJavaClassLibTraceLineSize=1024	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiLocalsInThrowable	-XX:-HitachiLocalsInThrowable	JavaVM デフォルト値と同じ	不可

オプション名称	日立 JavaVM のデフォルト値	Tomcat プロセスモニタ	
		デフォルト値	変更可否
-XX:[+ -]HitachiLocalsInStackTrace	-XX:-HitachiLocalsInStackTrace	-XX:+HitachiLocalsInStackTrace	不可
-XX:[+ -]HitachiLocalsSimpleFormat	-XX:-HitachiLocalsSimpleFormat	-XX:+HitachiLocalsSimpleFormat	不可
-XX:[+ -]HitachiTrueTypeInLocals	-XX:-HitachiTrueTypeInLocals	JavaVM デフォルト値と同じ	不可
-XX:HitachiCallToString	-XX:HitachiCallToString=minimal	JavaVM デフォルト値と同じ	不可
-XX:[+ -]HitachiFullCore	-XX:-HitachiFullCore	-XX:+HitachiFullCore	不可
-XX:HitachiJITCompileMaxMemorySize	-XX:HitachiJITCompileMaxMemorySize=0	JavaVM デフォルト値と同じ	不可
-XX:HitachiThreadLimit	-XX:HitachiThreadLimit=0	JavaVM デフォルト値と同じ	不可
-XX:[+ -]JITCompilerContinuation	-XX:+JITCompilerContinuation	JavaVM デフォルト値と同じ	不可
Java 17 以降 -XX:[+ -]HitachiVerboseGCPrintDirectBuffer	-XX:+HitachiVerboseGCPrintDirectBuffer	JavaVM デフォルト値と同じ	不可

注

変更可否が「不可」となっているオプションは、本製品の安定稼働やサポートサービスへの保守情報の提供のために必要です。保守員からの指示がない場合は、値を変更しないでください。

注※

ログ出力先を common.base で指定したディレクトリ直下以外の場所に変更した場合は、スナップショットログの取得対象に含まれるように、設定を必ず追加してください。詳細は、「11.8.6 ログ出力先を<common.base の指定値>以外に設定している場合」を参照してください。

common.base は、config.properties (本製品の設定ファイル) のプロパティです。詳細は、「12.2.4(1) 本製品全体に関するプロパティ」を参照してください。

表 15-2 日立 JavaVM 独自起動オプションのデフォルト値と変更可否 (Tomcat サーバプロセス)

オプション名称	日立 JavaVM のデフォルト値	Tomcat サーバプロセス	
		デフォルト値	変更可否
-XX:[+ -]HitachiThreadDump	-XX:+HitachiThreadDump	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiThreadDumpToStdout	-XX:+HitachiThreadDumpToStdout	-XX:-HitachiThreadDumpToStdout	不可
-XX:[+ -]HitachiThreadDumpWithHashCode	-XX:+HitachiThreadDumpWithHashCode	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiThreadDumpWithCpuTime	-XX:+HitachiThreadDumpWithCpuTime	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiThreadDumpWithBlockCount	-XX:+HitachiThreadDumpWithBlockCount	JavaVM デフォルト値と同じ	可
-XX:HitachiJavaLog	-XX:HitachiJavaLog:javalog	-XX:HitachiJavaLog:\${common.base}/javalog	可※1
-XX:HitachiJavaLogFileSize	-XX:HitachiJavaLogFileSize=256k	JavaVM デフォルト値と同じ	可

オプション名称	日立 JavaVM のデフォルト値	Tomcat サーバプロセス	
		デフォルト値	変更可否
-XX:[+ -]HitachiJavaLogNoMoreOutput	-XX:+HitachiJavaLogNoMoreOutput	JavaVM デフォルト値と同じ	可
-XX:HitachiJavaLogNumberOfFile	-XX:HitachiJavaLogNumberOfFile=4	JavaVM デフォルト値と同じ	可
-XX:[+ -]JavaLogAsynchronous	-XX:-JavaLogAsynchronous	JavaVM デフォルト値と同じ	可
-XX:[+ -]StandardLogToHitachiJavaLog	-XX:-StandardLogToHitachiJavaLog	-XX:+StandardLogToHitachiJavaLog	不可
-XX:[+ -]HitachiOutputMilliTime	-XX:-HitachiOutputMilliTime	-XX:+HitachiOutputMilliTime	不可
-XX:[+ -]HitachiVerboseGC	-XX:-HitachiVerboseGC	-XX:+HitachiVerboseGC	不可
-XX:[+ -]HitachiCommaVerboseGC	-XX:-HitachiCommaVerboseGC	JavaVM デフォルト値と同じ	可
-XX:HitachiVerboseGCIntervalTime	-XX:HitachiVerboseGCIntervalTime=0	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiVerboseGCPrintCause	-XX:+HitachiVerboseGCPrintCause	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiVerboseGCPrintDate	-XX:+HitachiVerboseGCPrintDate	JavaVM デフォルト値と同じ	可

オプション名称	日立 JavaVM のデフォルト値	Tomcat サーバプロセス	
		デフォルト値	変更可否
-XX:[+ -]HitachiVerboseGCCpuTime	-XX:+HitachiVerboseGCCpuTime	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiVerboseGCPrintTenuringDistribution	-XX:-HitachiVerboseGCPrintTenuringDistribution	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiVerboseGCPrintJVMinernalMemory	-XX:+HitachiVerboseGCPrintJVMinernalMemory	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiVerboseGCPrintThreadCount	-XX:+HitachiVerboseGCPrintThreadCount	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiVerboseGCPrintDeleteOnExit	-XX:+HitachiVerboseGCPrintDeleteOnExit	JavaVM デフォルト値と同じ	可
-XX:[+ -]PrintCodeCacheInfo	-XX:+PrintCodeCacheInfo	JavaVM デフォルト値と同じ	可
-XX:CodeCacheInfoPrintRatio	-XX:CodeCacheInfoPrintRatio=80	JavaVM デフォルト値と同じ	可
-XX:[+ -]PrintCodeCacheFullMessage	-XX:+PrintCodeCacheFullMessage	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiOutOfMemoryCause	-XX:-HitachiOutOfMemoryCause	-XX:+HitachiOutOfMemoryCause	不可
-XX:[+ -]HitachiOutOfMemoryStackTrace	-XX:-HitachiOutOfMemoryStackTrace	-XX:+HitachiOutOfMemoryStackTrace	不可



オプション名称	日立 JavaVM のデフォルト値	Tomcat サーバプロセス	
		デフォルト値	変更可否
		ryStackTrace	
-XX:HitachiOutOfMemoryStackTraceLineSize	-XX:HitachiOutOfMemoryStackTraceLineSize=1024	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiOutOfMemorySize	-XX:-HitachiOutOfMemorySize	-XX:+HitachiOutOfMemorySize	不可
-XX:[+ -]HitachiOutOfMemoryAbort	-XX:-HitachiOutOfMemoryAbort	-XX:+HitachiOutOfMemoryAbort	可※2
-XX:[+ -]HitachiOutOfMemoryAbortThreadDump	-XX:+HitachiOutOfMemoryAbortThreadDump	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiOutOfMemoryAbortThreadDumpWithJHeapProf	-XX:-HitachiOutOfMemoryAbortThreadDumpWithJHeapProf	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiOutOfMemoryHandling	-XX:-HitachiOutOfMemoryHandling	JavaVM デフォルト値と同じ	可
-XX:HitachiOutOfMemoryHandlingMaxThrowCount	-XX:HitachiOutOfMemoryHandlingMaxThrowCount=60	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiJavaClassLibTrace	-XX:-HitachiJavaClassLibTrace	-XX:+HitachiJavaClassLibTrace	不可
-XX:HitachiJavaClassLibTraceLineSize	-XX:HitachiJavaClassLibTraceLineSize=1024	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiLocalsInThrowable	-XX:-HitachiLocalsInThrowable	JavaVM デフォルト	可

オプション名称	日立 JavaVM のデフォルト値	Tomcat サーバプロセス	
		デフォルト値	変更可否
		ト値と同じ	
-XX:[+ -]HitachiLocalsInStackTrace	-XX:-HitachiLocalsInStackTrace	-XX:+HitachiLocalsInStackTrace	不可
-XX:[+ -]HitachiLocalsSimpleFormat	-XX:-HitachiLocalsSimpleFormat	-XX:+HitachiLocalsSimpleFormat	不可
-XX:[+ -]HitachiTrueTypeInLocals	-XX:-HitachiTrueTypeInLocals	JavaVM デフォルト値と同じ	可
-XX:HitachiCallToString	-XX:HitachiCallToString=minimal	JavaVM デフォルト値と同じ	可
-XX:[+ -]HitachiFullCore	-XX:-HitachiFullCore	-XX:+HitachiFullCore	不可
-XX:HitachiJITCompileMaxMemorySize	-XX:HitachiJITCompileMaxMemorySize=0	JavaVM デフォルト値と同じ	可
-XX:HitachiThreadLimit	-XX:HitachiThreadLimit=0	JavaVM デフォルト値と同じ	可
-XX:[+ -]JITCompilerContinuation	-XX:+JITCompilerContinuation	JavaVM デフォルト値と同じ	可
Java 17 以降 -XX:[+ -]HitachiVerboseGCPrintDirectBuffer	-XX:+HitachiVerboseGCPrintDirectBuffer	JavaVM デフォルト値と同じ	可

#### 注

変更可否が「不可」となっているオプションは、本製品の安定稼働やサポートサービスへの保守情報の提供のために必要です。保守員からの指示がない場合は、値を変更しないでください。

注※1

ログ出力先を common.base で指定したディレクトリ直下以外の場所に変更した場合は、スナップショットログの取得対象に含まれるように、設定を必ず追加してください。詳細は、「11.8.6 ログ出力先を<common.base の指定値>以外に設定している場合」を参照してください。

common.base は、config.properties (本製品の設定ファイル) のプロパティです。詳細は、「12.2.4(1) 本製品全体に関するプロパティ」を参照してください。

注※2

OutOfMemory エラーが発生すると、JavaVM は、空領域が枯渇している中で GC を繰り返すものの空領域が確保できなくなり、Java アプリケーションの身動きが取れなくなった結果、プロセスがハングアップに陥ることがあります。このオプションを有効にした場合、OutOfMemory エラーが発生すると、無条件に Java プロセスを強制停止します。アプリケーションサーバの自動再起動や待機系システムへの自動切り替えなどの仕組みを前提として、アプリケーションサーバプロセスの生存を監視している場合には、このオプションを有効にすることで、通常の状態への回復を促す効果があります。

Java HotSpot VM 共通起動オプションのデフォルト値と変更可否について説明します。

本製品または日立 JavaVM が独自にデフォルト値を変えているオプションだけを次の表に示します。そのほかの Java Hot Spot VM 共通の起動オプションについては、Oracle Java SE 11 のドキュメントを参照してください。

表 15-3 Java HotSpot VM 共通起動オプションのデフォルト値と変更可否 (Tomcat プロセスモニタ)

オプション名称	日立 JavaVM のデフォルト値	Tomcat プロセスモニタ	
		デフォルト値	変更可否
-Xmx<size>	83M	64M	不可
-Xms<size>	7.8M	64M	不可
-XX:MaxMetaspaceSize=<size>	2 <sup>64</sup> -1	2 <sup>64</sup> -1	不可
-XX:MetaspaceSize=<size>	<ul style="list-style-type: none"> <li>Java 11 以前 20.8M</li> <li>Java 17 以降 21M</li> </ul>	48M	不可
-XX:CompressedClassSpaceSize=<size>	1G	1G	不可
-Xss<size>	1M	1M	不可
-XX:NewRatio=<value>	2	2	不可
-XX:SurvivorRatio=<value>	32	32	不可
-XX:TargetSurvivorRatio=<value>	50	50	不可
-XX:MaxTenuringThreshold=<value>	14	14	不可
-XX:ReservedCodeCacheSize=<size>	48M	48M	不可

オプション名称	日立 JavaVM のデフォルト値	Tomcat プロセスモニタ	
		デフォルト値	変更可否
-XX:[+ -]UseSerialGC	<ul style="list-style-type: none"> <li>Java 8 -XX:-UseSerialGC</li> <li>Java 11 以降 -XX:+UseSerialGC</li> </ul>	-XX:+UseSerialGC	不可
-XX:[+ -]UseG1GC	-XX:-UseG1GC	-XX:-UseG1GC	不可
-XX:[+ -]UseCompressedOops	-XX:-UseCompressedOops	-XX:+UseCompressedOops	不可

注 1

<size>は、自然数（1 以上の整数）の値を次に示す単位を使って指定してください。

- ・キロ [k]
- ・メガ [m]
- ・ギガ [g]
- ・テラ [t]

なお、大文字・小文字は区別されません。

注 2

変更可否が「不可」となっているオプションは、本製品の安定稼働やサポートサービスへの保守情報の提供のために必要です。保守員からの指示がない場合は、値を変更しないでください。

表 15-4 Java HotSpot VM 共通起動オプションのデフォルト値と変更可否（Tomcat サーバプロセス）

オプション名称	日立 JavaVM のデフォルト値	Tomcat サーバプロセス	
		デフォルト値	変更可否
-Xmx<size>	83M	JavaVM デフォルト値と同じ	可
-Xms<size>	7.8M	JavaVM デフォルト値と同じ	可
-XX:MaxMetaspaceSize=<size>	2 <sup>64</sup> -1	JavaVM デフォルト	可

オプション名称	日立 JavaVM のデフォルト値	Tomcat サーバプロセス	
		デフォルト値	変更可否
		ト値と同じ	
-XX:MetaspaceSize=<size>	<ul style="list-style-type: none"> <li>• Java 11 以前 20.8M</li> <li>• Java 17 以降 21M</li> </ul>	JavaVM デフォルト値と同じ	可
-XX:CompressedClassSpaceSize=<size>	1G	JavaVM デフォルト値と同じ	可
-Xss<size>	1M	JavaVM デフォルト値と同じ	可
-XX:NewRatio=<value>	2	JavaVM デフォルト値と同じ	可
-XX:SurvivorRatio=<value>	32	JavaVM デフォルト値と同じ	可
-XX:TargetSurvivorRatio=<value>	50	JavaVM デフォルト値と同じ	可
-XX:MaxTenuringThreshold=<value>	14	JavaVM デフォルト値と同じ	可
-XX:ReservedCodeCacheSize=<size>	48M	JavaVM デフォルト値と同じ	可
-XX:[+ -]UseSerialGC	<ul style="list-style-type: none"> <li>• Java 8 -XX:-UseSerialGC</li> <li>• Java 11 以降 -XX:+UseSerialGC</li> </ul>	JavaVM デフォルト値と同じ	可

オプション名称	日立 JavaVM のデフォルト値	Tomcat サーバプロセス	
		デフォルト値	変更可否
-XX:[+ -]UseG1GC	-XX:-UseG1GC	JavaVM デフォルト値と同じ	可
-XX:[+ -]UseCompressedOops	-XX:-UseCompressedOops	-XX:+UseCompressedOops	可

注

<size>は、自然数（1 以上の整数）の値を次に示す単位を使って指定してください。

- ・キロ [k]
- ・メガ [m]
- ・ギガ [g]
- ・テラ [t]

なお、大文字・小文字は区別されません。

## 15.2 他社製 JavaVM を使用する場合

Java HotSpot VM 共通起動オプションのデフォルト値と変更可否について説明します。

本製品を使用することで、Java Hot Spot VM 共通起動オプションのデフォルト値とは異なる値が設定されるオプションがあります。そのオプションを次に示します。その他の Java Hot Spot VM 共通の起動オプションについては、使用している他社製 JavaVM のドキュメントを参照してください。

表 15-5 Java HotSpot VM 共通起動オプションのデフォルト値と変更可否 (Tomcat プロセスモニタ)

オプション名称	Tomcat プロセスモニタ	
	デフォルト値	変更可否
-Xmx<size>	64M	不可
-Xms<size>	64M	不可
-XX:MaxMetaspaceSize=<size>	2 <sup>64</sup> -1	不可
-XX:MetaspaceSize=<size>	48M	不可
-XX:CompressedClassSpaceSize=<size>	1G	不可
-Xss<size>	1M	不可
-XX:NewRatio=<value>	2	不可
-XX:SurvivorRatio=<value>	32	不可
-XX:TargetSurvivorRatio=<value>	50	不可
-XX:MaxTenuringThreshold=<value>	14	不可
-XX:ReservedCodeCacheSize=<size>	48M	不可
-XX:[+ -]UseSerialGC	-XX:+UseSerialGC	不可
-XX:[+ -]UseG1GC	-XX:-UseG1GC	不可
-XX:[+ -]UseCompressedOops	-XX:+UseCompressedOops	不可
-XX:[+ -]TieredCompilation	-XX:-TieredCompilation	不可

注 1

<size>は、自然数 (1 以上の整数) の値を次に示す単位を使って指定してください。

- ・キロ [k]
- ・メガ [m]
- ・ギガ [g]
- ・テラ [t]

なお、大文字・小文字は区別されません。

注 2

変更可否が「不可」となっているオプションは、本製品の安定稼働やサポートサービスへの保守情報の提供のために必要です。保守員からの指示がない場合は、値を変更しないでください。

表 15-6 Java HotSpot VM 共通起動オプションのデフォルト値と変更可否 (Tomcat サーバプロセス)

オプション名称	Tomcat サーバプロセス	
	デフォルト値	変更可否
-Xmx<size>	JavaVM デフォルト値と同じ	可
-Xms<size>	JavaVM デフォルト値と同じ	可
-XX:MaxMetaspaceSize=<size>	JavaVM デフォルト値と同じ	可
-XX:MetaspaceSize=<size>	JavaVM デフォルト値と同じ	可
-XX:CompressedClassSpaceSize=<size>	JavaVM デフォルト値と同じ	可
-Xss<size>	JavaVM デフォルト値と同じ	可
-XX:NewRatio=<value>	JavaVM デフォルト値と同じ	可
-XX:SurvivorRatio=<value>	JavaVM デフォルト値と同じ	可
-XX:TargetSurvivorRatio=<value>	JavaVM デフォルト値と同じ	可
-XX:MaxTenuringThreshold=<value>	JavaVM デフォルト値と同じ	可
-XX:ReservedCodeCacheSize=<size>	JavaVM デフォルト値と同じ	可
-XX:[+ -]UseSerialGC	JavaVM デフォルト値と同じ	可
-XX:[+ -]UseG1GC	JavaVM デフォルト値と同じ	可
-XX:[+ -]UseCompressedOops	JavaVM デフォルト値と同じ	可
-XX:[+ -]TieredCompilation	JavaVM デフォルト値と同じ	可

注

<size>は、自然数 (1 以上の整数) の値を次に示す単位を使って指定してください。

- ・キロ [k]
- ・メガ [m]
- ・ギガ [g]



・テラ [t]

なお、大文字・小文字は区別されません。

# 16

## 運用管理用コマンド

この章では、本製品の運用管理用コマンドについて説明します。

## 16.1 運用管理用コマンドの概要

ここでは、次の内容について説明します。

- 運用管理用コマンド文法の記述形式
- 運用管理用コマンドの入力形式

### 16.1.1 運用管理用コマンド文法の記述形式

運用管理用コマンドの文法の記述形式と使用する記号について説明します。

#### (1) 記述形式

コマンドの文法について次の形式で説明します。なお、各コマンドは、アルファベットの順に説明します。

##### 形式

コマンドの入力形式を示します。

##### 機能

コマンドの機能について説明します。

##### 引数

コマンドの引数およびオプションについて説明します。

##### 出力形式

コマンドの出力形式を示します。

##### 入力例・出力例

コマンドの入力例および出力例を示します。

##### 戻り値

コマンドの戻り値について説明します。

#### (2) 使用する記号

コマンドの文法は次の表に示す記号および構文要素を使用して記述します。

表 16-1 文法で使用している構文要素

構文要素	定義
英字	A~Z a~z
英小文字	a~z
英大文字	A~Z

構文要素	定義
数字	0~9
英数字	A~Z a~z 0~9
記号	! " # \$ % & ' ( ) + , _ . / : ; < = > @ [ ] ^ - { } <code>—</code> タブ 空白

注

すべて半角文字を使用してください。

## 16.1.2 運用管理用コマンドの入力形式

運用管理用コマンドの入力形式を次に示します。

コマンド名称 [引数…]

各項目について説明します。なお、コマンドプロンプトを「\$」、コマンド名称を「cmd」と表記します。

### (1) コマンド名称

実行するコマンドのファイル名を指定します。

### (2) 引数

引数には、オプションも含まれます。オプションの入力形式および指定規則を次に示します。

#### (a) オプションの入力形式

オプションは、「--」（ハイフン 2 個）で始まる文字列です。オプションには、1 個のオプション値を指定してください。

```
$ cmd --オプション名=オプション値
```

(凡例)

- オプション名  
半角英字の文字列です。大文字と小文字が区別されます。
- オプション値  
オプション名に対する引数です。

#### (b) オプションの指定規則

- オプション名の前には 2 つの「-」（ハイフン）が必要です。  
誤った指定例：\$ cmd -key=value

正しい指定例：`$ cmd --key=value`

- オプション名を指定した場合、オプション値は省略できません。

誤った指定例：`$ cmd --key`

正しい指定例：`$ cmd --key=value`

- オプション名とオプション値の間には「=」（イコール）が必要です。

誤った指定例：`$ cmd --key value`

正しい指定例：`$ cmd --key=value`

- 同じオプション名は、複数指定できません。

誤った指定例：`$ cmd --key=value1 --key=value2`

- オプション値に空白を含む場合、オプション値全体を"で囲む必要があります。

誤った指定例：`$ cmd --key=file 1`

正しい指定例：`$ cmd --key="file 1"`

## 16.2 スナップショットログ収集コマンド

スナップショットログ収集コマンドについて説明します。

### collect-snapshot.sh (スナップショットログ収集)

#### 形式

```
collect-snapshot.sh [--help] [--port=<port>|--endpoint=<endpoint>] [--threaddumpnum=<number>] [--watchcommand=<boolean>] [--timeout-sec=<timeout>] [--file=<file>]
```

#### 機能

スナップショットログを収集し、指定したパスに出力します。また、標準出力または標準エラー出力にメッセージを出力します。

#### 引数

collect-snapshot.sh コマンドのオプションを次の表に示します。

表 16-2 collect-snapshot.sh コマンドのオプション

オプション	説明	省略の可否	省略時の動作
--help	使用方法 (Usage) を出力します。	省略可	使用方法 (Usage) を出力しません。
--port=<port>	スナップショットログ収集 REST API のポート番号を指定します。1~65535 の範囲で指定します。 <ul style="list-style-type: none"><li>• monitor.rest.port<sup>*</sup>に値を設定している場合、その値を指定します。</li><li>• --endpoint と同時に指定した場合、このオプションは無視されます。</li></ul>	省略可	「28081」を使用します。
--endpoint=<endpoint>	スナップショットログ収集 REST API のエンドポイントを指定します。 <ul style="list-style-type: none"><li>• monitor.rest.bindaddress<sup>*</sup>に値を設定している場合は、ホスト名部分に、Tomcat プロセスモニタを起動させているマシンの IP アドレスを指定します。</li></ul>	省略可	「http://localhost:28081/api/v1/snapshot」を使用します。

オプション	説明	省略の可否	省略時の動作
	<ul style="list-style-type: none"> <li>• --port と同時に指定した場合、このオプションが使用されます。</li> </ul>		
--threaddumpnum=<number>	スレッドダンプの取得回数を指定します。0~60 の範囲で指定します。	省略可	収集対象の Tomcat プロセスモニタに設定されている <snapshot.rest.default.threaddumpnum <sup>*</sup> の指定値>の値を使用します。
--watchcommand=<boolean>	マシンリソースの使用状況を取得するかどうかを、true または false で指定します。	省略可	収集対象の Tomcat プロセスモニタに設定されている <snapshot.rest.default.watchcommand.enabled <sup>*</sup> の指定値>の値を使用します。
--timeout-sec=<timeout>	コマンドを実行後、指定した時間を経過しても収集処理が終わらない場合のタイムアウト時間を指定します。単位は秒です。1~2147483647 の範囲で指定します。	省略可	タイムアウトしません。
--file=<file>	出力先パスを指定します。	省略可	「./snapshot.zip」を使用します。

#### 注※

config.properties (本製品の設定ファイル) のプロパティです。

monitor.rest.port, および monitor.rest.bindaddress については、「12.2.4(2) Tomcat プロセスモニタに関するプロパティ」を参照してください。

snapshot.rest.default.threaddumpnum および snapshot.rest.default.watchcommand.enabled については、「12.2.4(4) スナップショットログ収集機能に関するプロパティ」を参照してください。

## 出力形式

- 標準出力に処理中のメッセージを出力します。
- 標準エラー出力にエラーメッセージを出力します。
- --file オプションのパス (省略時はカレントディレクトリの snapshot.zip) にスナップショットログを出力します。

## 入力例

- カレントディレクトリの snapshot.zip にスナップショットログを出力します。

```
$ collect-snapshot.sh
```

- オプションを指定して snapshot1.zip にスナップショットログを出力します。

```
$ collect-snapshot.sh --port=28082 --threaddumpnum=3 --watchcommand=true --timeout-sec=60 --file=snapshot1.zip
```

## 戻り値

- 0 :  
正常終了しました。
- 1 :  
引数が不正です。
- 2 :  
異常終了しました。



# 17

## 運用管理用 REST API

この章では、本製品の運用管理用 REST API について説明します。

## 17.1 運用管理用 REST API の概要

---

運用管理用 REST API の概要について説明します。

### 17.1.1 運用管理用 REST API の記述形式

運用管理用 REST API について次の形式で説明します。

#### 説明

API の機能について説明します。

#### 形式

API の記述形式を示します。

#### パラメタ

API のパラメタについて説明します。

#### 実行例

API の実行例について説明します。

#### 戻り値

API の戻り値について説明しています。

## 17.2 スナップショットログ収集 REST API

Tomcat プロセスモニタは、スナップショットログ収集 REST API を提供します。

### GET メソッド

#### 説明

次の表のリクエストを受け付けるとスナップショットログを収集し、レスポンスとして出力します。

表 17-1 スナップショットログ収集 REST API が収集処理を実行するリクエスト

項目	値
Method	GET
URL	/api/v1/snapshot
パラメタ	「表 17-2 スナップショットログ収集 REST API のパラメタ」に示すパラメタと値

#### 形式

```
http://<IPアドレス>:<ポート番号>/api/v1/snapshot
```

<IP アドレス>には Tomcat プロセスモニタを起動させているマシンの IP アドレスを指定してください。  
<ポート番号>には、Tomcat プロセスモニタの HTTP 機能の受付ポート番号 (monitor.rest.port の値の指定値) と同じ値を指定してください。

monitor.rest.port については、「12.2.4(2) Tomcat プロセスモニタに関するプロパティ」を参照してください。

#### パラメタ

スナップショットログ収集 REST API のパラメタを次の表に示します。

表 17-2 スナップショットログ収集 REST API のパラメタ

パラメタ	指定できる値	省略の可否	デフォルト値	説明
threaddumpnum	0~60	省略可	<snapshot.rest.default.threaddumpnum*の指定値>	スレッドダンプ情報を取得する回数を指定します。取得すると Tomcat サーバプロセスに負荷が掛かります。 スレッドダンプ情報については、「11.3.2(1)(b) スレッドダンプ情報」を参照してください。

パラメタ	指定できる値	省略の可否	デフォルト値	説明
watchcommand	次のどちらかを指定します。 <ul style="list-style-type: none"> <li>• true</li> <li>• false</li> </ul>	省略可	<snapshot.rest.default.watchcommand.enabled*の指定値>	モニタリング情報を取得するかどうかを指定します。取得すると所要時間が長くなります。 モニタリング情報については、「11.3.2(1)(a) モニタリング情報」を参照してください。

注※  
 config.properties（本製品の設定ファイル）のプロパティです。詳細は、「12.2.4(4) スナップショットログ収集機能に関するプロパティ」を参照してください。

## 実行例

リクエストの例を次に示します。

- デフォルト値で収集

```
/api/v1/snapshot
```

- オプションを指定して収集

```
/api/v1/snapshot?threaddumpnum=3&watchcommand=true
```

## 戻り値

### 正常時

スナップショットログの収集に成功した場合、レスポンスボディに出力されるスナップショットログ（zip形式）を次の表に示します。

表 17-3 収集に成功した場合に出力されるスナップショットログ

項目	値
Status	200
Content-Type	application/zip
Body	<スナップショットログ（zip形式）>

### 異常時

スナップショットログの収集に失敗した場合に出力されるエラーメッセージ（JSON形式）を次の表に示します。

表 17-4 収集に失敗した場合に出力されるエラーメッセージ（リクエストのパラメタが指定できない値だった場合）

項目	値
Status	400
Content-Type	application/json
Body	{ "message": "<メッセージ KDLR10016-E>" }

表 17-5 収集に失敗した場合に出力されるエラーメッセージ（スナップショットログ出力に失敗した場合）

項目	値
Status	500
Content-Type	application/json
Body	{ "message": "<メッセージ KDLR10017-E>" }

# 18

## ユーザアプリケーションで使用できる API

この章では、ユーザアプリケーションで使用できる API について説明します。

## 18.1 ユーザアプリケーションで使用できる API の概要

---

ユーザアプリケーションで使用できる API の概要について説明します。

### 18.1.1 ユーザアプリケーションで使用できる API の記述形式

ユーザアプリケーションで使用できる API について次の形式で説明します。

#### 説明

API の機能について説明します。

#### 形式

API の記述形式を示します。

#### パラメタ

API のパラメタについて説明します。

#### 例外

API を利用する際に発生する例外について説明します。

#### 戻り値

API の戻り値について説明しています。

## 18.2 性能解析トレースで使用する API

性能解析トレースのアプリケーション情報取得機能で使用する API について説明します。主に、HiRDB の JDBC ドライバから使用されることを想定しています。この機能では、uCosminexus Application Server V11-00 以降で使用できる `com.hitachi.software.javaee.util.prf.PrfTrace` クラスだけをサポートします。

この API を含む Java のソースコードをコンパイルする場合、<本製品のインストールディレクトリ>/lib/tomcat/system/ucar-javaprpf-api.jar をクラスパスに設定してください。本製品の実行時に、Tomcat プロセスモニタがこの API の jar をデフォルトでクラスパスに設定します。そのため、クラスパスを設定する必要はありません。

性能解析トレースで使用する API について説明します。メソッドの記載順は、アルファベット順です。

### getClientApInfo メソッド

#### 説明

getPrfTrace メソッド実行時に取得したクライアントアプリケーション情報を文字列表現で返します。

#### 形式

```
public final String getClientApInfo()
```

#### パラメタ

なし

#### 例外

なし

#### 戻り値

クライアントアプリケーション情報の文字列表現。

次の場合には、null を返します。

- クライアントアプリケーション情報が設定されていない場合※
- トレース機能が動作していない場合

#### 注※

クライアントアプリケーション情報を引き継ぐシーケンスの範囲外の場合を指します。



## getPrfTrace メソッド

### 説明

現在のスレッドが保持しているルートアプリケーション情報とクライアントアプリケーション情報を保持する PrfTrace のインスタンスを返します。

### 形式

```
public static PrfTrace getPrfTrace()
```

### パラメタ

なし

### 例外

なし

### 戻り値

PrfTrace のインスタンス。

## getRootApInfo メソッド

### 説明

getPrfTrace メソッド実行時に取得したルートアプリケーション情報を文字列表現で返します。

### 形式

```
public final String getRootApInfo()
```

### パラメタ

なし

### 例外

なし

### 戻り値

ルートアプリケーション情報の文字列表現。

次の場合には、null を返します。

- ルートアプリケーション情報が設定されていない場合※
- トレース機能が動作していない場合

注※

ルートアプリケーション情報を引き継ぐシーケンスの範囲外の場合を指します。

# 付録

## 付録 A GUI を使用したインストール

ここでは、「3.2 インストールする」にあるコマンドでのインストールではなく、GUI を用いた本製品のインストール手順を説明します。

### 操作手順

1. インストール CD-ROM のデータを、本製品をインストールするマシンにコピーする。

本製品のインストール CD-ROM をマウントし、CD-ROM に格納されている X64LIN ディレクトリを、インストール先のマシン上の任意のディレクトリに X64LIN ディレクトリごとコピーしてください。これ以降、X64LIN ディレクトリのコピー先を<インストーラのパス>と表記します。

2. setup コマンドに実行権限を付ける。

「<インストーラのパス>/X64LIN/setup」という実行ファイルに対して、管理者権限で実行権限を付与します。

実行例：

```
$ sudo chmod +x <インストーラのパス>/X64LIN/setup
```

3. PP インストーラの setup コマンドを実行する。

次に示す引数を指定して setup コマンドを実行します。

```
$ sudo <インストーラのパス>/X64LIN/setup <インストーラのパス>
```

### 注

「3.2 インストールする」のコマンドでのインストール手順とはコマンドの引数が異なります。

4. PP インストーラのメインメニューで、[I] キーを押す。

PP インストール画面が表示されます。

5. インストールする製品を選択して [スペース] キーを押す。

「uCAR for Apache Tomcat」または「uCAR with Java for Apache Tomcat」にカーソルを移動させて選択してください。選択した製品の左側には「<@>」が表示されます。

6. インストールする製品の左側に「[@]」が表示されていることを確認して [I] キーを押す。

画面の最下行に、「Install PP? (y: install, n: cancel)==>」が表示されます。

7. [Y] キーを押す。

インストールが開始されます。

8. インストール終了を示すメッセージが表示されたら [Q] キーを押す。

PP インストーラのメインメニューに戻ります。

9. PP インストーラのメインメニューで [L] キーを押す。

PP 一覧表示画面にインストール済みの製品一覧が表示されます。

10. 本製品がインストールされていることを確認して [Q] キーを押す。

PP インストーラのメインメニューが表示されます。

11. PP インストーラのメインメニューで [Q] キーを押す。

PP インストーラが終了し、本製品のインストールが完了します。

12. インストールが正常に完了しているかどうかを確認する。

インストールが正常に完了しているかどうかは、install.log で確認できます。

実行例：

```
$ sudo cat /opt/hitachi/ucart/install.log
```

install.log に「rc=0 msg=I:Installation completed.」と出力されていれば、インストールが正常に完了しています。

出力例：

```
2022/06/30 12:34:56 rc=0 msg=I:Installation completed.
```

13. 最新の修正パッチを適用する。

ソフトウェアサポートサービスの Web サイトから修正パッチが提供されている場合は、最新の修正パッチを入手して適用してください。修正パッチの適用方法については、「[4.7 修正パッチを適用する](#)」を参照してください。ソフトウェアサポートサービスの Web サイトにアクセスできない場合は、本製品に同梱されている修正パッチ CD を利用してください。

## 付録 B GUI を使用したアンセットアップ

ここでは、「4.8 アンセットアップする」にあるコマンドでのアンセットアップではなく、GUI を用いた本製品のアンセットアップ手順を説明します。

ここで説明するアンセットアップ手順は、次に示す条件を満たしていることを前提としています。

- Tomcat および Tomcat プロセスモニタを停止している
- 「3.3 Tomcat に組み込む」のセットアップ手順に従って、次の3つの定義ファイルについて、編集前の状態でバックアップを作成している
  - `${CATALINA_BASE}/conf/catalina.properties` (Tomcat のプロパティ定義ファイル)
  - `${CATALINA_BASE}/conf/server.xml` (Tomcat のサーバ設定ファイル)
  - `${CATALINA_BASE}/conf/context.xml` (Tomcat のコンテキスト設定ファイル)
- 次のファイルがセットアップ前にすでに存在していた場合は、バックアップを作成している
  - `${CATALINA_HOME}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)
  - `${CATALINA_BASE}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)

### 操作手順

#### 1. Tomcat の定義ファイルを編集前の状態に戻す。

次の3つの定義ファイルについて、セットアップ時に取得したバックアップを使用して、編集前の状態に戻します。

- `${CATALINA_BASE}/conf/catalina.properties` (Tomcat のプロパティ定義ファイル)
- `${CATALINA_BASE}/conf/server.xml` (Tomcat のサーバ設定ファイル)
- `${CATALINA_BASE}/conf/context.xml` (Tomcat のコンテキスト設定ファイル)

#### 2. setenv.sh (Tomcat 起動時の環境変数定義ファイル) を編集前の状態に戻す。

次のファイルがセットアップ前にすでに存在していた場合は、バックアップを使用して、編集前の状態に戻します。

- `${CATALINA_HOME}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)
- `${CATALINA_BASE}/bin/setenv.sh` (Tomcat 起動時の環境変数定義ファイル)

セットアップ前には存在しなかった場合は、setenv.sh (Tomcat 起動時の環境変数定義ファイル) 自体を削除してください。

#### 3. PP インストーラを使用して本製品をアンインストールする。

次のコマンドを管理者権限で実行し、画面の指示に従って「uCAR for Apache Tomcat」または「uCAR with Java for Apache Tomcat」を削除してください。

```
$ sudo /etc/hitachi_x64setup
```

これでアンセットアップは完了です。

## 付録 C 本製品のアーカイブファイルを用いたインストールおよびアンセットアップ

---

PP インストーラを利用できない環境に本製品をインストールするため、アーカイブファイルを提供します。本製品のアーカイブファイルを用いることにより、次のメリットがあります。

- 一部の前提ライブラリのインストールが不要となる。
- インストール、上書きインストールおよびアンセットアップ時に管理者権限が不要となる。
- 本製品のインストールディレクトリのパスを変更できる。

アーカイブファイルは、本製品に添付されています。具体的な使用方法については、添付品に同梱されているドキュメントを参照してください。

マニュアルに記載されている次のパスは、アーカイブファイルを展開したディレクトリのパスに読み替えてください。

```
/opt/hitachi/ucart
```



## 付録 D このマニュアルの参考情報

このマニュアルを読むに当たっての参考情報を示します。

### 付録 D.1 関連マニュアル

関連マニュアルを次に示します。必要に応じてお読みください。

- uCosminexus Application Runtime - Cosminexus Developer's Kit for Java 機能解説・リファレンス (3021-3-K02)  
本製品とあわせて日立 JavaVM を使用する場合に、お読みください。
- Cosminexus V11 アプリケーションサーバ 機能解説 保守／移行編 (3021-3-J11)  
本製品のトレース機能で uCosminexus Application Server と連携する際に、お読みください。

### 付録 D.2 このマニュアルでの表記

サービス名、製品名、機能名などの名称を、次のように表記しています。

表記	サービス名、製品名、機能名など	
AWS	Amazon Web Services	
Azure	Microsoft Azure	
HiRDB	HiRDB/Single Server Version 10	
Linux	Red Hat Enterprise Linux 7 (AMD/Intel 64)	Red Hat Enterprise Linux 7.1 (AMD/Intel 64) 以降
	Red Hat Enterprise Linux 8 (AMD/Intel 64)	Red Hat Enterprise Linux 8.1 (AMD/Intel 64) 以降
	Red Hat Enterprise Linux 9 (AMD/Intel 64)	Red Hat Enterprise Linux 9.1 (AMD/Intel 64) 以降
Tomcat	Apache Tomcat	
Ubuntu 20.04	Ubuntu 20.04 以降	
Ubuntu 22.04	Ubuntu 22.04 以降	
サポートサービス	<ul style="list-style-type: none"><li>• OSS Support Service with uCosminexus Application Runtime for Apache Tomcat</li><li>• OSS Support Service with uCosminexus Application Runtime with Java for Apache Tomcat</li></ul>	

表記	サービス名, 製品名, 機能名など
日立 JavaVM	Cosminexus Developer's Kit for Java
保守員	uCosminexus Application Runtime サポートサービスの契約に基づくお問い合わせ窓口のこと。

## 付録 D.3 英略語

このマニュアルで使用する英略語を次に示します。

英略語	英字での表記
API	Application Programming Interface
BOM	Byte Order Mark
CPU	Central Processing Unit
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
Java EE, Java Platform, または Enterprise Edition	Java Platform, Enterprise Edition
Java SE	Java Platform, Standard Edition
JAX-RS	Java API for RESTful Web Services
JDBC	Java Database Connectivity
JRE	Java Runtime Environment
NFS	Network File System
OS	Operating System
OSS	Open Source Software
REST	Representational State Transfer
WAR	Web Application Archive

## 付録 D.4 KB (キロバイト) などの単位表記について

1KB (キロバイト), 1MB (メガバイト), 1GB (ギガバイト), 1TB (テラバイト) はそれぞれ  $1,024$  バイト,  $1,024^2$  バイト,  $1,024^3$  バイト,  $1,024^4$  バイトです。

# 索引

## C

catalina.properties 162, 188  
com.cosminexus.appruntime.tomcat.healthcheck.MonitoringListener 113  
config.properties 162, 163  
context.xml 162, 191

## H

HTTP セッションのトレース 95  
HTTP リクエストおよびサーブレット・フィルタのトレース 92

## J

java.util.logging のレベル 203  
JavaVM ログ (Tomcat サーバプロセス) 199  
JavaVM ログ (Tomcat プロセスモニタ) 196  
Java の実行環境情報 143  
JAX-RS クライアントのトレース 96

## O

org.apache.catalina.valves.StuckThreadDetectionValve 127

## S

server.xml 162, 189  
setenv.sh 162, 185

## T

Tomcat JDBC Connection Pool のトレース 98  
Tomcat 稼働中情報 138  
Tomcat 起動時の環境変数定義ファイル 185  
Tomcat サーバプロセス開始完了通知待ちタイムアウト 116  
Tomcat サーバプロセス初期化完了通知待ちタイムアウト 114  
Tomcat サーバプロセス内のフィルタ 16  
Tomcat サーバプロセスのログ 197  
Tomcat トレースログ 197

Tomcat のコンテキスト設定ファイル 191  
Tomcat のサーバ設定ファイル 189  
Tomcat のプロパティ定義ファイル 188  
Tomcat プロセスモニタ 16, 76  
Tomcat プロセスモニタ機能 75  
Tomcat プロセスモニタに関するプロパティ 166  
Tomcat プロセスモニタのログ 195

## U

uCosminexus Application Server との連携 86

## W

Web アプリケーションの開始時および終了時のトレース 91

## あ

アクション 109

## い

イベントプロパティ 111

## か

稼働監視機能 107  
稼働監視機能に関するプロパティ 169  
稼働監視用ライフサイクルリスナー 113  
監視項目 114

## き

機密情報のマスキング 144

## こ

コマンドの実行による情報の取得 138

## さ

サーバの開始時および終了時のトレース 90

## し

自動収集 131, 154, 157

手動収集 131, 154, 160

## す

スナップショットログ 131  
スナップショットログ収集機能 131  
スナップショットログ収集機能に関するプロパティ 177  
スナップショットログ収集コマンド 148  
スレッドダンプ情報 139

## て

定義ファイルの種類 162  
停滞検出バルブ 125

## と

独自ログファイル 197  
トレース機能 85  
トレース機能に関するプロパティ 182

## は

ハートビート監視 118

## ふ

ファイルによる情報の収集 135  
プロセス生存監視 120  
プロセスの起動監視 114  
プロパティキーに含まれる可変値 163  
プロパティ値の変数展開について 164

## へ

ヘルスチェック 122

## ほ

保守ログ 195  
ホストマシン情報 142  
本製品全体に関するプロパティ 166  
本製品の設定ファイル 163

## め

メッセージ 201

メッセージログ 195

## も

モニタリング情報 138

## ゆ

ユーザコマンドの実行 129  
ユーザスレッドおよび非同期処理 API 利用上の注意事項 106

## り

リクエスト処理の停滞監視 125

## ろ

ログ取得レベル 183  
ログファイルの種類 194

---

 株式会社 日立製作所

〒100-8280 東京都千代田区丸の内一丁目6番6号

---