HITACHI
Inspire the Next

Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator ユーザーズガイド

4010-1J-500-10

前書き

■ 対象製品

RT-1V63-5011 Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator 01-02 (適用 OS：Red Hat Universal Base Image 8(64-bit x86_64)、Red Hat Universal Base Image 8 Minimal(64-bit x86_64)、Almalinux 8 Minimal 8.8(64-bit x86_64))

■ 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

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 10 月 4010-1J-500-10

■ 著作権

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

変更内容

変更内容（4010-1J-500-10） Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator 01-02

追加・変更内容	変更個所
マイクロサービス化したアプリケーションから OpenTP1 のサービスを呼び出して使用するための TP1 連携機能をサポートしました。	1.1、1.2.4、1.3、1.3.3、1.4.1、1.4.2、1.5、2.2.1、2.2.7、2.2.13、2.6.2、2.6.3、2.6.6、3.1、3.2.3(2)、3.3、3.4.1、3.4.4、3.6.3、3.6.6、3.7、3.7.1、3.7.4、3.7.10、3.8.3(1)、3.8.3(3)、3.8.3(15)、3.8.3(20)、3.9.1(1)、3.9.1(6)、3.9.1(7)、3.9.1(8)、3.10.1(1)(c)、3.10.1(6)、3.10.1(7)、3.10.2、3.10.3、3.10.3(2)(c)、4.、5.1、5.3.8、6.2、6.4.1、6.4.2、6.4.6、8.1、8.2.1、8.2.3(1)、8.3.2、8.3.8(1)、8.5、8.6、9.1.1、10.1.1、10.3.4(1)(b)、10.4.1、10.5.3(3)、10.5.3(4)(b)、10.6.1、10.6.2(3)、10.6.3(1)、10.6.3(4)(b)、付録 A.1、付録 D
HMP-PCTO の前提プログラムおよび関連プログラムを追加・変更しました。	1.4.1、1.4.2、2.2.8(1)
SQL-Participant の接続先として、HiRDB および SQL-Server をサポートしました。	1.4.1、1.4.2、2.4.7(1)、3.9.1(3)、3.9.1(4)、8.1、8.2.4(2)、8.4、10.1.1、10.6.1、10.6.3(4)(b)
@PctoBranchRollbackFor アノテーションのロールバックしない場合の説明を変更しました。	2.2.8(3)、7.2.7
Kubernetes マニフェスト作成時の注意事項を追加しました。	2.6、2.6.4、2.6.5
次の Kubernetes マニフェストの記述例を変更しました。 • Orchestrator • Entity-Service（Participant コンテナを含まない Pod とする場合） • SQL-Participant • TCC-Participant • Mediator の ConsensusLog の永続ボリュームおよびストレージクラス	2.6.2、2.6.3、2.6.4、2.6.5、3.6.2
Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリを、NFS ストレージ上に作成できるようにしました。	3.5.2、3.6.2、3.8.1、6.4.5、8.3.1(1)
HMP-PCTO が提供する Helm チャートに、OSS パラメタを追加する手順の説明を追加しました。	3.7.1、3.7.2
Mediator の Helm チャートのパラメタに、次のパラメタを追加しました。 • resources	3.7.3、8.3.1(1)

追加・変更内容	変更個所
• ucars.enabled • tracing.queueSize • tracing.delayTime • tracing.batchSize	
Prometheus のバージョンアップに伴い、次のパラメタのデフォルト値を変更しました。 • prometheus.remoteWrite.queue.capacity • prometheus.remoteWrite.queue.maxShards • prometheus.remoteWrite.queue.maxSamplesPerSend	3.7.10、8.3.8(1)
バージョンアップ時の注意事項を追加しました。	3.8.3(17)(b)、3.8.3(17)(c)
チューニング対象のパラメタとメトリクス名の関連づけについての説明を追加しました。	3.10.1(3)(c)、3.10.1(4)(c)
トランザクション制御の中で使用するタイマの、次のパラメタの説明を変更しました。 • sql.transaction-termination.consensus.replay-interval-in-millis • tp1.transactionTermination.consensus.replayIntervalInMillis	3.10.2、8.2.4(2)、8.3.2(1)
次のコンテナイメージの前提プログラムを追加しました。 • SQL-Participant • Entity-Module/TCC-Participant	5.3.5、5.3.6
次の各コンテナの要件のコマンド要件を変更しました。 • Filebeat コンテナの要件 • Jaeger-query コンテナの要件 • Jaeger-collector コンテナの要件 • Prometheus コンテナの要件 • Metricbeat コンテナの要件	6.4.1、6.4.2、6.4.3、6.4.4、6.4.9、6.4.10、6.4.11、6.4.12、6.4.13
EADS サーバチャート用 values.yaml ファイルに次のパラメタを追加しました。 • eads.failureDetector.heartbeat.timeout • eads.failureDetector.connection.timeout • eads.failureDetector.read.timeout	8.3.3(1)(a)、8.3.3.(1)(b)
次のメッセージを変更しました。 KFSG60000-E (Y)M、KFSG60001-E (Y)M、KFSG60002-E (Y)M、KFSG60003-W (Y)M、KFSG60010-W (Y)M+J、KFSG60011-E (Y)M+J、KFSG60013-W (Y)M+J、KFSG60014-E (Y)M、KFSG60015-W (Y)M、KFSG60016-E (Y)M、KFSG60017-E (Y)M、KFSG60018-W (Y)M+J、KFSG60019-E (Y)M、KFSG60023-E (Y)M、KFSG60025-W (Y)M、KFSG60026-W (Y)M、KFSG60030-W (Y)M+J、KFSG61004-W (Y)M、KFSG61008-E (Y)M、KFSG61103-E (Y)M、KFSG62014-W (Y)M、KFSG62015-W (Y)M、KFSG64002-E (Y)M、KFSG64602-E (Y)M、KFSG74002-E (Y)M、KFSG74404-I (Y)M、KFSG74405-I (Y)M、	9.2.1、9.2.2、9.2.3

追加・変更内容	変更個所
KFSG74406-I (Y)M、KFSG74407-E (Y)M、KFSG74408-W (Y)M、KFSG80001-I (Y)M、KFSG80002-I (Y)M、KFSG80004-I (Y)M、KFSG82006-I (Y)M、KFSG82007-I (Y)M	
次のメッセージを追加しました。 KFSG60033-W (Y)M、KFSG63200-E (Y)M、KFSG63201-E (Y)M、KFSG63204-E (Y)M、KFSG63205-W (Y)M、KFSG63207-E (Y)M、KFSG63208-E (Y)M、KFSG63209-E (Y)M、KFSG63300-E (Y)C、KFSG63301-E (Y)C、KFSG63302-E (Y)C、KFSG63303-E (Y)C、KFSG63304-E (Y)C、KFSG63305-E (Y)C、KFSG63306-W (Y)C、KFSG63307-E (Y)C、KFSG63308-E (Y)C、KFSG63309-E (Y)C、KFSG63310-E (Y)C、KFSG63311-W (Y)C、KFSG63312-E (Y)C、KFSG83103-I (Y)C	9.2.1、9.2.3
次のメッセージを削除しました。 KFSG74208-I (Y)M	9.2.2
分散トレースで Process 情報を出力できるようにしました。	10.3.1、10.3.3(3)、10.3.4(1)(b)、10.5.2(3)(b)
分散トレースの取得に関するパラメタを追加しました。	10.3.4(4)
uCosminexus Application Runtime for Spring Boot のスナップショットログ収集機能での収集方法の説明を追加しました。	10.6.2(2)

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、マイクロサービス環境でシステム全体のトランザクションの一貫性を確保するために Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator（以降は HMP-PCTO）を適用する際、必要になる次の項目を説明したものです。

- HMP-PCTO を適用した Kubernetes アプリケーションの開発・デプロイ、および設定手順
- HMP-PCTO を適用した Kubernetes アプリケーションの開発・デプロイ、および設定時に必要なパラメタ
- Kubernetes クラスタ外の HMP-PCTO の運用方法
- Kubernetes クラスタ外の HMP-PCTO で設定が必要なパラメタ
- HMP-PCTO がサポートする API
- HMP-PCTO が出力するメッセージ
- HMP-PCTO の情報取得を含めたシステム運用

このマニュアルを利用することで、ユーザが HMP-PCTO を適用できるようになることを目的としています。

■ 対象読者

このマニュアルは次の方にお読みいただくことを前提に説明しています。

- HMP-PCTO を適用した Kubernetes アプリケーションを開発する方
- HMP-PCTO を適用したシステムを運用する方

また、次の知識をお持ちの方を前提としています。

- トランザクション制御に関する基本的な知識
- コンテナに関する知識
- Kubernetes に関する知識
- Java 言語のプログラミングの基本的な知識、および JDBC の基本的な知識

■ このマニュアルで使用する記号

このマニュアルで使用する記号を次のように定義しています。

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。

記号	意味
	(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。 (例) [A] 「何も指定しない」か「A 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。
<>	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。 (例 1) <プロパティ> 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。 (例 2) <ファイル名> ファイル名を指定します。

目次

前書き　2
変更内容　3
はじめに　6

第 1 編　HMP-PCTO の前提条件

1　HMP-PCTO の構成および前提条件　18
1.1　HMP-PCTO の Kubernetes アプリケーションの一覧と分類　19
1.2　構成要素　21
1.2.1　HMP-PCTO の Kubernetes クラスタの構成要素　21
1.2.2　SQL-Participant を使用する場合の構成要素（SQL-Participant 限定）　21
1.2.3　TCC-Participant を使用する場合の構成要素（TCC-Participant 限定）　22
1.2.4　TP1-Bridge を使用する場合の構成要素（通常版かつ TP1-Bridge 限定）　23
1.3　最小機器構成　24
1.3.1　SQL-Participant を使用する場合の最小機器構成（SQL-Participant 限定）　24
1.3.2　TCC-Participant を使用する場合の最小機器構成（TCC-Participant 限定）　28
1.3.3　TP1-Bridge を使用する場合の最小機器構成（通常版かつ TP1-Bridge 限定）　32
1.4　前提条件　35
1.4.1　前提 OS および前提プログラム　35
1.4.2　関連プログラム　37
1.4.3　前提とする Kubernetes クラスタ構成　38
1.4.4　前提とする共有ファイルシステム　39
1.5　HMP-PCTO のアプリケーション開発からシステム運用までの流れ　40

第 2 編　ユーザ責務の Kubernetes アプリケーション開発

2　ユーザ責務の Kubernetes アプリケーション開発　42
2.1　ユーザ責務の Kubernetes アプリケーションの開発の流れ　43
2.2　Java アプリケーションの開発　44
2.2.1　作業項目の一覧　44
2.2.2　事前準備　45
2.2.3　ライブラリのセットアップ（通常版限定）　45
2.2.4　コンポーネントスキャンの追加　45
2.2.5　DI に使用するアノテーションの注意事項　46
2.2.6　トランザクション境界の設定　46

2.2.7 トランザクションの伝搬 47
2.2.8 トランザクションブランチの属性の設定 50
2.2.9 先行プリペア機能の使用（SQL-Participant 限定） 51
2.2.10 JDBC Proxy Driver の使用（SQL-Participant 限定） 52
2.2.11 TccService インタフェースの実装クラスの作成（TCC-Participant 限定） 53
2.2.12 業務ロジックからの executeTry メソッドの呼び出し（TCC-Participant 限定） 59
2.2.13 TP1-Bridge の使用（通常版かつ TP1-Bridge 限定） 60
2.2.14 ロギングライブラリの設定ファイルのカスタマイズ 61
2.2.15 Quartz スケジューラ機能の利用 62
2.3 SQL-Participant のビルド（SQL-Participant 限定） 63
2.3.1 作業項目の一覧（SQL-Participant 限定） 63
2.3.2 事前準備（通常版かつ SQL-Participant 限定） 63
2.3.3 ライブラリのセットアップ（通常版かつ SQL-Participant 限定） 63
2.3.4 JAR ファイルのビルド（SQL-Participant 限定） 64
2.4 コンテナイメージ作成 65
2.4.1 作業項目の一覧 65
2.4.2 事前準備 65
2.4.3 uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成（通常版限定） 66
2.4.4 JDK コンテナベースイメージの作成 69
2.4.5 Orchestrator コンテナイメージの作成 72
2.4.6 Entity-Service コンテナイメージの作成 75
2.4.7 SQL-Participant コンテナイメージの作成（SQL-Participant 限定） 78
2.4.8 Entity-Module/TCC-Participant コンテナイメージの作成（TCC-Participant 限定） 83
2.5 共有ファイルシステムでの作業（通常版限定） 90
2.5.1 作業項目の一覧（通常版限定） 90
2.5.2 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定） 90
2.6 Kubernetes マニフェスト作成 91
2.6.1 作業項目の一覧 91
2.6.2 Orchestrator の Kubernetes マニフェストの作成 92
2.6.3 Entity-Service（Participant コンテナを含まない Pod とする場合）の Kubernetes マニフェストの作成 98
2.6.4 SQL-Participant の Kubernetes マニフェストの作成（SQL-Participant 限定） 104
2.6.5 TCC-Participant の Kubernetes マニフェストの作成（TCC-Participant 限定） 114
2.6.6 Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成 122
2.6.7 デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成 124
2.6.8 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定） 124

2.7 ユーザ責務の Kubernetes アプリケーションのデプロイとアンデプロイ 125
2.7.1 作業項目の一覧 125
2.7.2 Kubernetes コントロールプレーンでの作業 126

第 3 編 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイ

3 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイ 135
3.1 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイの流れ 136
3.2 Mediator のビルド（通常版限定） 137
3.2.1 作業項目の一覧（通常版限定） 137
3.2.2 事前準備（通常版限定） 137
3.2.3 ライブラリのセットアップ（通常版限定） 137
3.2.4 JAR ファイルのビルド（通常版限定） 138
3.3 TP1-Bridge のビルド（通常版かつ TP1-Bridge 限定） 140
3.3.1 作業項目の一覧（通常版かつ TP1-Bridge 限定） 140
3.3.2 事前準備（通常版かつ TP1-Bridge 限定） 140
3.3.3 ライブラリのセットアップ（通常版かつ TP1-Bridge 限定） 140
3.3.4 JAR ファイルのビルド（通常版かつ TP1-Bridge 限定） 141
3.4 コンテナイメージ作成（通常版限定） 142
3.4.1 作業項目の一覧（通常版限定） 142
3.4.2 事前準備（通常版限定） 142
3.4.3 Mediator コンテナイメージの作成（通常版限定） 143
3.4.4 TP1-Bridge コンテナイメージの作成（通常版かつ TP1-Bridge 限定） 146
3.4.5 Dependency-Checker コンテナイメージの作成（通常版限定） 150
3.4.6 EADS サーバ用コンテナイメージの作成（通常版限定） 151
3.4.7 EADS コマンド実行用コンテナイメージの作成（通常版限定） 153
3.5 共有ファイルシステムでの作業 156
3.5.1 作業項目の一覧 156
3.5.2 Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（NFS ストレージとする場合） 156
3.5.3 Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリの作成 157
3.5.4 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定） 157
3.6 Kubernetes マニフェスト作成 159
3.6.1 作業項目の一覧 159
3.6.2 Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成 160

3.6.3 Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成 167
3.6.4 Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成 171
3.6.5 EADS の永続ボリュームの Kubernetes マニフェストの作成（通常版限定） 174
3.6.6 デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成 180
3.6.7 コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成 186
3.6.8 メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成 187
3.6.9 メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストの作成 189
3.6.10 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定） 190
3.7 Helm チャートのカスタマイズ 194
3.7.1 作業項目の一覧 194
3.7.2 Helm チャートへのパラメタ追加 195
3.7.3 Mediator の Helm チャートのカスタマイズ 197
3.7.4 TP1-Bridge の Helm チャートのカスタマイズ（通常版かつ TP1-Bridge 限定） 200
3.7.5 EADS の Helm チャートのカスタマイズ（通常版限定） 204
3.7.6 Ext-Cons の Helm チャートのカスタマイズ（トライアル版限定） 204
3.7.7 Filebeat の Helm チャートのカスタマイズ 206
3.7.8 Jaeger-query の Helm チャートのカスタマイズ 207
3.7.9 Jaeger-collector の Helm チャートのカスタマイズ 209
3.7.10 Prometheus の Helm チャートのカスタマイズ 210
3.7.11 Metricbeat の Helm チャートのカスタマイズ 213
3.7.12 Elasticsearch の Helm チャートのカスタマイズ 214
3.7.13 Logstash の Helm チャートのカスタマイズ 219
3.8 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションの運用 222
3.8.1 作業項目の一覧 222
3.8.2 Kubernetes 各ワーカーノードでの作業 226
3.8.3 Kubernetes コントロールプレーンでの作業 229
3.9 ログの取得および監視の設定 264
3.9.1 ログおよびトラブルシュート情報の取得の設定 264
3.9.2 監視の設定 267
3.10 パラメタの設計とチューニング 271
3.10.1 スレッドプール 271
3.10.2 タイマ 311
3.10.3 G1GC 320
3.10.4 ディスク 322
3.10.5 システム運用に必要となる機能 322

第 4 編　Kubernetes クラスタ外の HMP-PCTO の運用

4　TP1-Converter の運用（通常版かつ TP1-Bridge 限定）　333
4.1　TP1-Converter の運用の流れ（通常版かつ TP1-Bridge 限定）　334
4.2　データ変換定義ファイルの作成（通常版かつ TP1-Bridge 限定）　335
4.2.1　データ変換で使用するファイル作成の流れ（通常版かつ TP1-Bridge 限定）　335
4.2.2　バイナリフォーマット定義ファイル（通常版かつ TP1-Bridge 限定）　336
4.2.3　XML スキーマ（通常版かつ TP1-Bridge 限定）　337
4.2.4　マッピング定義ファイル（通常版かつ TP1-Bridge 限定）　337
4.2.5　データ変換定義ファイル（通常版かつ TP1-Bridge 限定）　337
4.3　J2EE サーバのセットアップ（通常版かつ TP1-Bridge 限定）　338
4.4　TP1-Converter の資材の配置（通常版かつ TP1-Bridge 限定）　339
4.5　J2EE サーバの開始（通常版かつ TP1-Bridge 限定）　340
4.6　TP1-Converter のインポート（通常版かつ TP1-Bridge 限定）　341
4.7　TP1-Converter の開始（通常版かつ TP1-Bridge 限定）　342
4.8　データ変換定義の再作成（通常版かつ TP1-Bridge 限定）　343
4.9　キャッシュのリフレッシュ（通常版かつ TP1-Bridge 限定）　344
4.10　TP1-Converter の終了（通常版かつ TP1-Bridge 限定）　346
4.11　J2EE サーバの終了（通常版かつ TP1-Bridge 限定）　347
4.12　キャッシュの参照（通常版かつ TP1-Bridge 限定）　348
4.13　TP1-Converter の削除と J2EE サーバのアンセットアップ（通常版かつ TP1-Bridge 限定）　350
4.13.1　TP1-Converter の削除（通常版かつ TP1-Bridge 限定）　350
4.13.2　J2EE サーバのアンセットアップ（通常版かつ TP1-Bridge 限定）　350

第 5 編　リファレンス

5　コンテナイメージの作成要件　351
5.1　コンテナイメージの一覧　352
5.2　コンテナイメージの提供方式　355
5.3　各コンテナイメージの作成要件　357
5.3.1　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（通常版限定）　357
5.3.2　JDK コンテナベースイメージ　358
5.3.3　Orchestrator コンテナイメージ　359
5.3.4　Entity-Service コンテナイメージ　360
5.3.5　SQL-Participant コンテナイメージ　361
5.3.6　Entity-Module/TCC-Participant コンテナイメージ　363
5.3.7　Mediator コンテナイメージ　365
5.3.8　TP1-Bridge コンテナイメージ（通常版かつ TP1-Bridge 限定）　367
5.3.9　Dependency-Checker コンテナイメージ　368

5.3.10 Ext-Cons コンテナイメージ（トライアル版限定） 369
5.3.11 EADS サーバ用コンテナイメージ（通常版限定） 370
5.3.12 EADS コマンド実行用コンテナイメージ（通常版限定） 371

6 Kubernetes クラスタの作成要件 373
6.1 Kubernetes アプリケーションのデプロイ方法 374
6.2 Kubernetes クラスタ内外の通信暗号化方針 375
6.3 Kubernetes アプリケーションのタイムゾーン設定方針 376
6.4 各 Kubernetes アプリケーションの作成要件 377
6.4.1 Orchestrator の要件 377
6.4.2 Entity-Service（Participant コンテナを含まない Pod とする場合）の要件 381
6.4.3 SQL-Participant の要件 386
6.4.4 TCC-Participant の要件 393
6.4.5 Mediator の要件 398
6.4.6 TP1-Bridge の要件（通常版かつ TP1-Bridge 限定） 405
6.4.7 EADS の要件（通常版限定） 413
6.4.8 Ext-Cons の要件（トライアル版限定） 416
6.4.9 Filebeat の要件 419
6.4.10 Jaeger-query の要件 423
6.4.11 Jaeger-collector の要件 427
6.4.12 Prometheus の要件 431
6.4.13 Metricbeat の要件 436
6.4.14 Elasticsearch の要件 440
6.4.15 Logstash の要件 443
6.5 Kubernetes アプリケーションで使用するスクリプト 448
6.5.1 デプロイ依存関係チェックスクリプト（dependency_check.sh） 448
6.5.2 トランザクションヘルスチェックスクリプト（health_check.sh） 450
6.5.3 未決着トランザクション待機スクリプト（wait_transactions.sh） 451

7 API 453
7.1 API の記述形式 454
7.2 トランザクション制御の API 仕様 456
7.2.1 クラス・インタフェースの一覧 456
7.2.2 アノテーションの一覧 456
7.2.3 例外の一覧 457
7.2.4 XidHeaders インタフェース（SQL-Participant 限定） 457
7.2.5 TxPropagationIscImperativePlugin クラス 458
7.2.6 TxPropagationRestTemplateInterceptor クラス 461
7.2.7 @PctoBranchRollbackFor アノテーション 463

7.2.8 @PctoTransactional アノテーション 464
7.2.9 RollbackTerminationException 例外 465
7.2.10 TerminationFailureException 例外 465
7.2.11 TransactionSystemException 例外 466
7.3 JDBC Proxy Driver の API 仕様（SQL-Participant 限定） 467
7.3.1 クラス・インタフェースの一覧（SQL-Participant 限定） 467
7.3.2 例外の一覧（SQL-Participant 限定） 468
7.3.3 PctoDataSource クラス（SQL-Participant 限定） 470
7.3.4 PctoConnection クラス（SQL-Participant 限定） 472
7.3.5 PctoStatement クラス（SQL-Participant 限定） 486
7.3.6 PctoPreparedStatement クラス（SQL-Participant 限定） 503
7.3.7 PctoCallableStatement クラス（SQL-Participant 限定） 530
7.3.8 PctoResultSet クラス（SQL-Participant 限定） 554
7.3.9 PctoDatabaseMetaData クラス（SQL-Participant 限定） 597
7.3.10 PctoResultSetMetadata クラス（SQL-Participant 限定） 599
7.3.11 PctoSQLXML クラス（SQL-Participant 限定） 603
7.3.12 PctoBlob クラス（SQL-Participant 限定） 607
7.3.13 PctoClob クラス（SQL-Participant 限定） 610
7.3.14 PctoInputStream クラス（SQL-Participant 限定） 612
7.3.15 PctoReader クラス（SQL-Participant 限定） 620
7.3.16 PctoArray クラス（SQL-Participant 限定） 629
7.3.17 PctoSqlFeatureNotSupportedException 例外（SQL-Participant 限定） 631
7.3.18 PctoSQLClientInfoException 例外（SQL-Participant 限定） 631
7.3.19 PctoCommunicationFailureException 例外（SQL-Participant 限定） 632
7.3.20 PctoInternalContradictionException 例外（SQL-Participant 限定） 632
7.3.21 PctoTransactionStartException 例外（SQL-Participant 限定） 633
7.3.22 PctoPrePreparedException 例外（SQL-Participant 限定） 633
7.3.23 PctoConnectionLackedException 例外（SQL-Participant 限定） 634
7.3.24 PctoIOException 例外（SQL-Participant 限定） 634
7.3.25 PctoIOCommunicationFailureException 例外（SQL-Participant 限定） 635
7.3.26 PctoIOInternalContradictionException 例外（SQL-Participant 限定） 635
7.3.27 PctoIOTransactionStartException 例外（SQL-Participant 限定） 636
7.3.28 SQLState・ErrorCode 一覧（SQL-Participant 限定） 636
7.4 TCC-Participant の API 仕様（TCC-Participant 限定） 638
7.4.1 クラス・インタフェースの一覧（TCC-Participant 限定） 638
7.4.2 アノテーションの一覧（TCC-Participant 限定） 638
7.4.3 例外の一覧（TCC-Participant 限定） 638
7.4.4 TccResponse<T extends java.io.Serializable>クラス（TCC-Participant 限定） 639
7.4.5 TccService<I, O extends java.io.Serializable>インタフェース（TCC-Participant 限定） 641

7.4.6 @Tcc アノテーション（TCC-Participant 限定） 644

8 パラメタ 646
8.1 HMP-PCTO の各コンポーネントおよび連携先製品に設定するパラメタ 647
8.2 ユーザのコンポーネントに設定するパラメタ 649
8.2.1 共通 649
8.2.2 Orchestrator 651
8.2.3 Entity-Service/Entity-Module 656
8.2.4 SQL-Participant 658
8.2.5 TCC-Participant 667
8.3 HMP-PCTO と RAS のコンポーネントに設定するパラメタ 673
8.3.1 Mediator 673
8.3.2 TP1-Bridge（通常版かつ TP1-Bridge 限定） 683
8.3.3 EADS（通常版限定） 698
8.3.4 Ext-Cons（トライアル版限定） 706
8.3.5 Filebeat 708
8.3.6 Jaeger-query 711
8.3.7 Jaeger-collector 715
8.3.8 Prometheus 718
8.3.9 Metricbeat 723
8.3.10 Elasticsearch 727
8.3.11 Logstash 731
8.4 各 RDBMS の使用に必要なパラメタ（SQL-Participant 限定） 736
8.4.1 PostgreSQL（SQL-Participant 限定） 736
8.4.2 SQL Server（SQL-Participant 限定） 736
8.4.3 HiRDB（SQL-Participant 限定） 737
8.5 TP1 連携時の連携先 OpenTP1 に設定が必要なパラメタ（通常版かつ TP1-Bridge 限定） 738
8.5.1 rap リスナーサービス定義（通常版かつ TP1-Bridge 限定） 738
8.5.2 システム共通定義（通常版かつ TP1-Bridge 限定） 738
8.5.3 トランザクションサービス定義（通常版かつ TP1-Bridge 限定） 739
8.5.4 HMP-PCTO のトランザクションに参加する SPP のユーザサービス定義（通常版かつ TP1-Bridge 限定） 739
8.6 uCosminexus Application Server に設定が必要な TP1-Converter のパラメタ（通常版かつ TP1-Bridge 限定） 740
8.6.1 usrconf.properties（通常版かつ TP1-Bridge 限定） 740
8.6.2 usrconf.cfg（通常版かつ TP1-Bridge 限定） 742

9 メッセージ 743
9.1 メッセージの概要 744
9.1.1 メッセージの記述形式 744

9.1.2 メッセージ系統 745
9.1.3 メッセージの説明 746
9.2 メッセージ一覧 749
9.2.1 KFSG60000～KFSG69999 749
9.2.2 KFSG70000～KFSG79999 809
9.2.3 KFSG80000～KFSG89999 817

第 6 編 システム運用に必要となる機能

10 システム運用に必要となる機能 828
10.1 概要 829
10.1.1 取得する情報および各情報の紐づけ方法 829
10.2 メッセージログ 842
10.2.1 概要 842
10.2.2 取得方針 843
10.3 分散トレース 844
10.3.1 概要 844
10.3.2 トレースの単位 845
10.3.3 Span へ付加する情報 846
10.3.4 取得方針 849
10.4 メトリクス 868
10.4.1 概要 868
10.4.2 パラメタ 872
10.5 データおよびファイル 873
10.5.1 メッセージログ 873
10.5.2 分散トレース 878
10.5.3 メトリクス 881
10.5.4 定義するデータ形式 886
10.6 HMP-PCTO の障害調査 889
10.6.1 取得する情報 889
10.6.2 情報の収集方法 891
10.6.3 障害原因の調査方法 893
10.7 システム運用に必要となる機能の障害対策 901
10.7.1 障害の原因および対策 901

付録 904
付録 A 提供ファイル 905
付録 A.1 提供ファイル一覧 905
付録 B HMP-PCTO のメトリクス取得項目 911
付録 C このマニュアルの参考情報 914

付録 C.1　関連ドキュメント　914
付録 C.2　マニュアルで使用している略語　916
付録 C.3　英略語　916
付録 C.4　KB（キロバイト）などの単位表記について　918
付録 D　HMP-PCTO で使用する用語　919

索引　923

1 HMP-PCTO の構成および前提条件

1.1 HMP-PCTO の Kubernetes アプリケーションの一覧と分類

HMP-PCTO は、ユーザアプリケーションに取り込んで使用する製品です。HMP-PCTO の Kubernetes アプリケーションには、次に示す分類があります。

- ユーザ責務の Kubernetes アプリケーション
  HMP-PCTO の業務ロジックを担う Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンの Kubernetes アプリケーション
  HMP-PCTO のトランザクション制御の機能を担う Kubernetes アプリケーション
- RAS の Kubernetes アプリケーション
  HMP-PCTO のシステム運用に必要な機能を担う Kubernetes アプリケーション

HMP-PCTO の Kubernetes アプリケーションの一覧と分類を次の表に示します。

表 1-1 HMP-PCTO の Kubernetes アプリケーションの一覧と分類

Kubernetes アプリケーション	Kubernetes アプリケーションの分類		
	ユーザ責務の Kubernetes アプリケーション	HMP-PCTO のコントロールプレーンの Kubernetes アプリケーション	RAS の Kubernetes アプリケーション
Orchestrator	○	−	−
Entity-Service（Participant コンテナを含まない Pod とする場合）※	○	−	−
SQL-Participant	○	−	−
TCC-Participant	○	−	−
Mediator	−	○	−
TP1-Bridge（TP1 連携機能を使用する場合）	−	○	−
EADS（通常版の場合）	−	○	−
Ext-Cons（トライアル版の場合）	−	○	−
Filebeat	−	−	○
Jaeger-query	−	−	○
Jaeger-collector	−	−	○
Prometheus	−	−	○
Metricbeat	−	−	○
Elasticsearch	−	−	○
Logstash	−	−	○

（凡例）

○：該当します。

－：該当しません。

注※

ここで対象としている Entity-Service は、サービス間を中継するサービスとしての Entity-Service を指します。詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のトランザクションのモデルに関する説明を参照してください。

1.2　構成要素

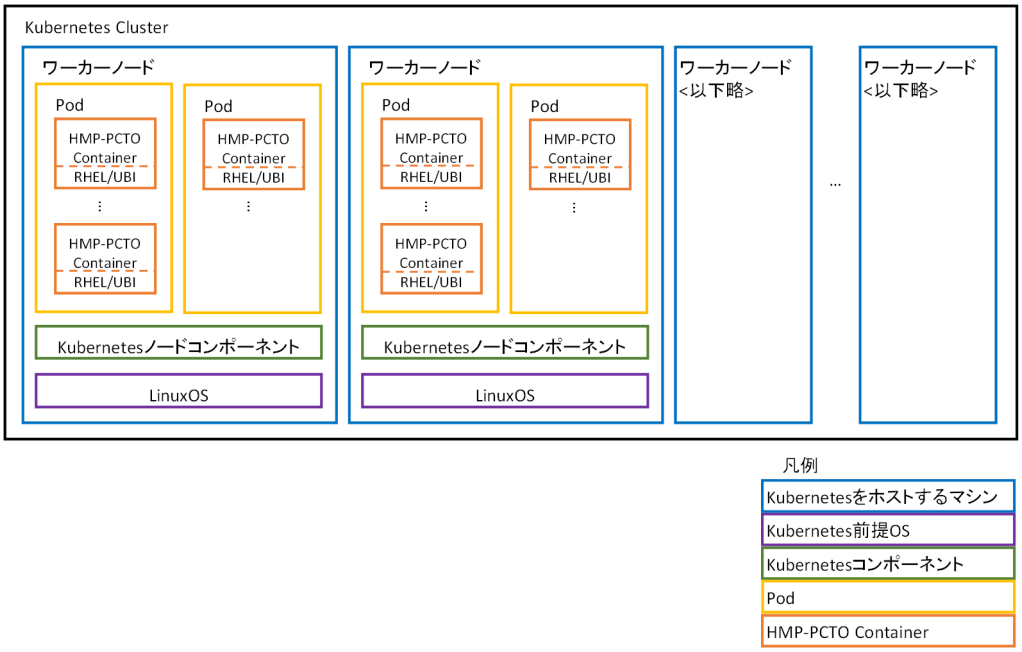
HMP-PCTO の Kubernetes クラスタの構成要素、SQL-Participant を使用する場合および TCC-Participant を使用する場合の構成要素（Pod および Container）について説明します。

1.2.1　HMP-PCTO の Kubernetes クラスタの構成要素

HMP-PCTO は 1 つの Kubernetes クラスタで構成します。

RHEL、Kubernetes、Java、前提 OS などの前提バージョンについては、「1.4　前提条件」を参照してください。

図 1-1　HMP-PCTO の Kubernetes クラスタの構成要素



注　トライアル版では、図中の RHEL/UBI の代わりに AlmaLinux を使用します。

Kubernetes マスターコンポーネントの配置については HMP-PCTO では規定しないため、Kubernetes マスターコンポーネントの記載は省略しています。

1.2.2　SQL-Participant を使用する場合の構成要素（SQL-Participant 限定）

SQL-Participant を使用する場合の構成要素（Pod および Container）を次の表に示します。

表 1-2　SQL-Participant を使用する場合の構成要素

Kubernetes cluster	Pod	Container	persistent volume
HMP-PCTO	Orchestrator	Orchestrator	−
	Participant	Entity-Service	−
		SQL-Participant	−
	Mediator	Mediator	○
	通常版の場合 　EADS トライアル版の場合 　Mock サーバ	通常版の場合 　EADS トライアル版の場合 　Mock サーバ	○
	Filebeat	Filebeat	−
	Jaeger-query	Jaeger-query	−
	Jaeger-collector	Jaeger-collector	−
	Prometheus	Prometheus	−
	Metricbeat	Metricbeat	−
	Elasticsearch	Elasticsearch	○
	Logstash	Logstash	−

（凡例）○：必要です　−：規定しません

1.2.3　TCC-Participant を使用する場合の構成要素（TCC-Participant 限定）

TCC-Participant を使用する場合の構成要素（Pod および Container）を次の表に示します。

表 1-3　TCC-Participant を使用する場合の構成要素

Kubernetes cluster	Pod	Container	persistent volume
HMP-PCTO	Orchestrator	Orchestrator	−
	Participant	Entity-Module/TCC-Participant	−
	Mediator	Mediator	○
	通常版の場合 　EADS トライアル版の場合 　Mock サーバ	通常版の場合 　EADS トライアル版の場合 　Mock サーバ	○

Kubernetes cluster	Pod	Container	persistent volume
	Filebeat	Filebeat	−
	Jaeger-query	Jaeger-query	−
	Jaeger-collector	Jaeger-collector	−
	Prometheus	Prometheus	−
	Metricbeat	Metricbeat	−
	Elasticsearch	Elasticsearch	○
	Logstash	Logstash	−

（凡例）○：必要です　−：規定しません

1.2.4　TP1-Bridge を使用する場合の構成要素（通常版かつ TP1-Bridge 限定）

TP1-Bridge を使用する場合の構成要素（Pod および Container）を次の表に示します。

表 1–4　TP1-Bridge を使用する場合の構成要素

Kubernetes cluster	Pod	Container	persistent volume
HMP-PCTO	Orchestrator	Orchestrator	−
	TP1-Bridge	TP1-Bridge	−
	Mediator	Mediator	○
	EADS	EADS	○
	Filebeat	Filebeat	−
	Jaeger-query	Jaeger-query	−
	Jaeger-collector	Jaeger-collector	−
	Prometheus	Prometheus	−
	Metricbeat	Metricbeat	−
	Elasticsearch	Elasticsearch	○
	Logstash	Logstash	−

（凡例）○：必要です　−：規定しません

1.3　最小機器構成

SQL-Participant、TCC-Participant、および TP1-Bridge を使用する場合の最小機器構成について説明します。

機器構成には、HMP-PCTO に組み込む Kubernetes アプリケーションの責務境界についても示しています。HMP-PCTO の Kubernetes アプリケーションの分類については、「1.1　HMP-PCTO の Kubernetes アプリケーションの一覧と分類」を参照してください。

1.3.1　SQL-Participant を使用する場合の最小機器構成（SQL-Participant 限定）

SQL-Participant を使用する場合の最小機器構成について説明します。

各 Mediator の Pod は可用性を確保するため、別々の Kubernetes ワーカーノードに配置し、物理的にも別のマシンとする必要があります。

Mediator の最小個数は分散合意方式によって異なります。

DB サーバは、HMP-PCTO と同一の Kubernetes クラスタ内に配置するか、または Kubernetes クラスタ外に配置するかは任意です（HMP-PCTO では規定しません）。

各 Kubernetes アプリケーションをデプロイする Namespace の名称および数は任意とします。

(1)　通常版の場合

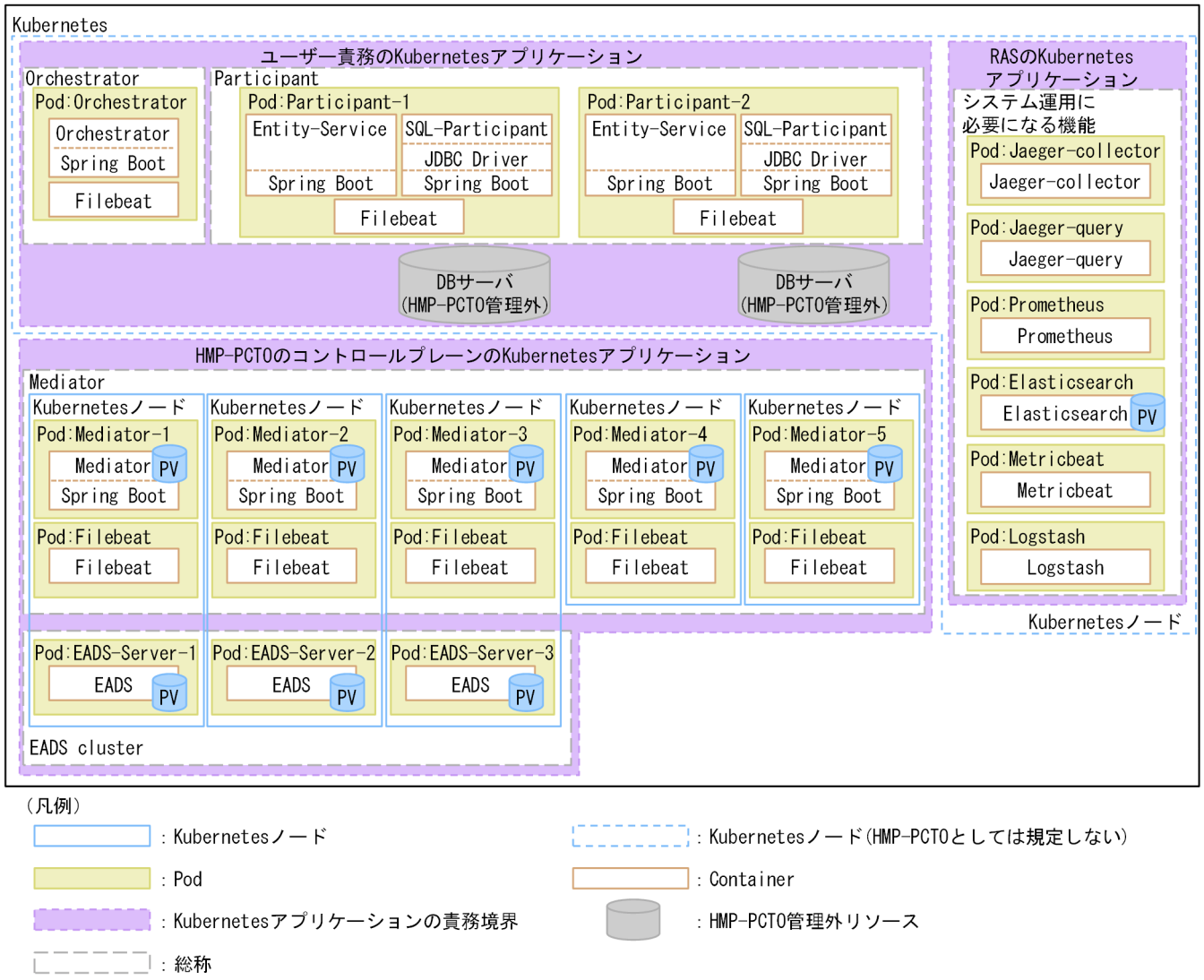
補助合意用の EADS クラスタ内の EADS サーバの Pod は可用性を確保するため、別々の Kubernetes ワーカーノードに配置し、物理的にも別のマシンとする必要があります。

Mediator の Pod と EADS サーバの Pod は、同一のワーカーノードにも配置できます。

なお、各 Mediator の Pod と、補助合意用の EADS クラスタ内の Pod（EADS サーバの Pod を除く）は、任意に Kubernetes ワーカーノードに配置できます（HMP-PCTO では規定しません）。

- ThreeQuarters 合意方式での最小機器構成
  SQL-Participant を使用する場合の、ThreeQuarters 合意方式での最小機器構成を次の図に示します。

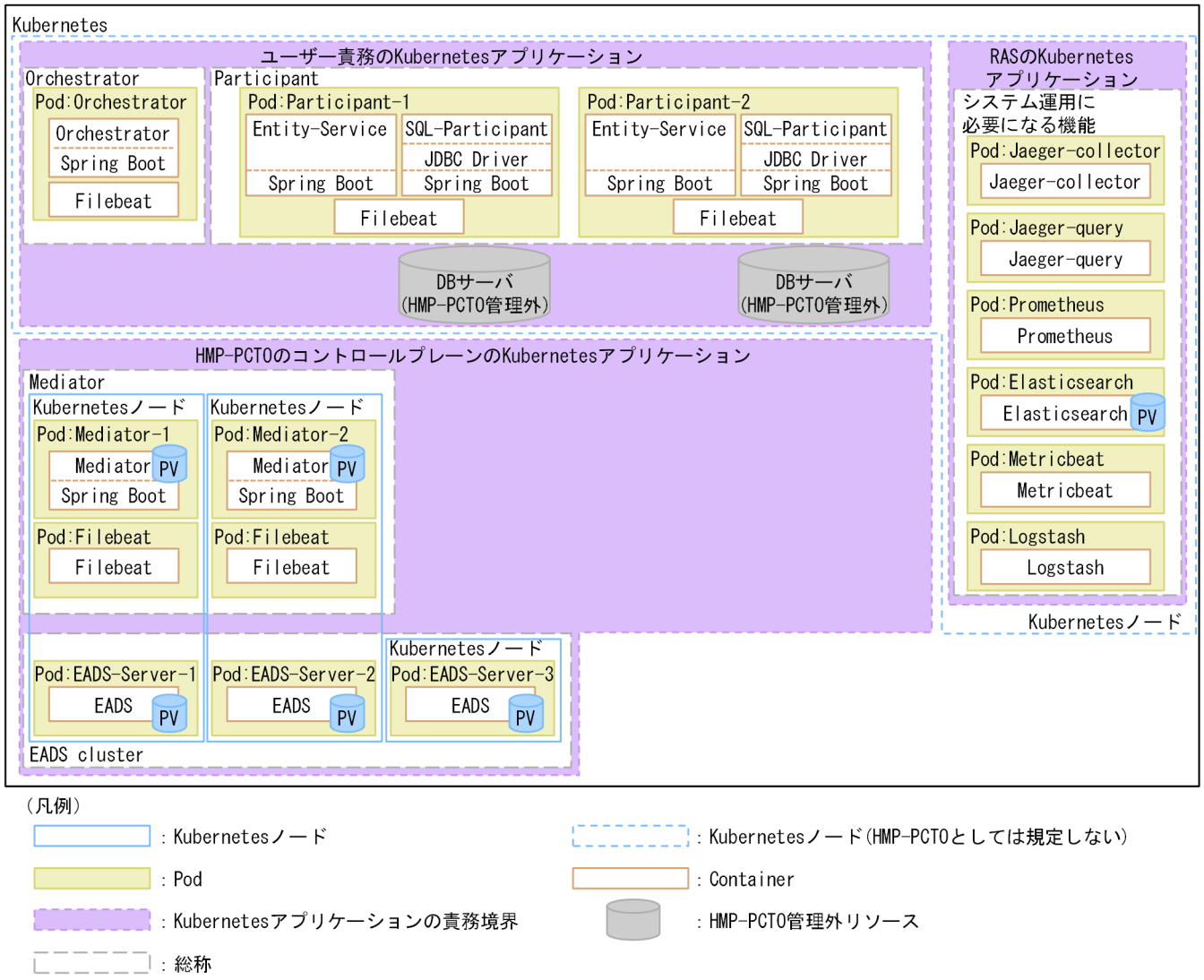
図 1-2　SQL-Participant を使用する場合の最小機器構成（ThreeQuarters 合意方式（通常版））



- PrivilegedValue 合意方式での最小機器構成

  SQL-Participant を使用する場合の、PrivilegedValue 合意方式での最小機器構成を次の図に示します。

図 1-3　SQL-Participant を使用する場合の最小機器構成（PrivilegedValue 合意方式（通常版））



(2)　トライアル版の場合

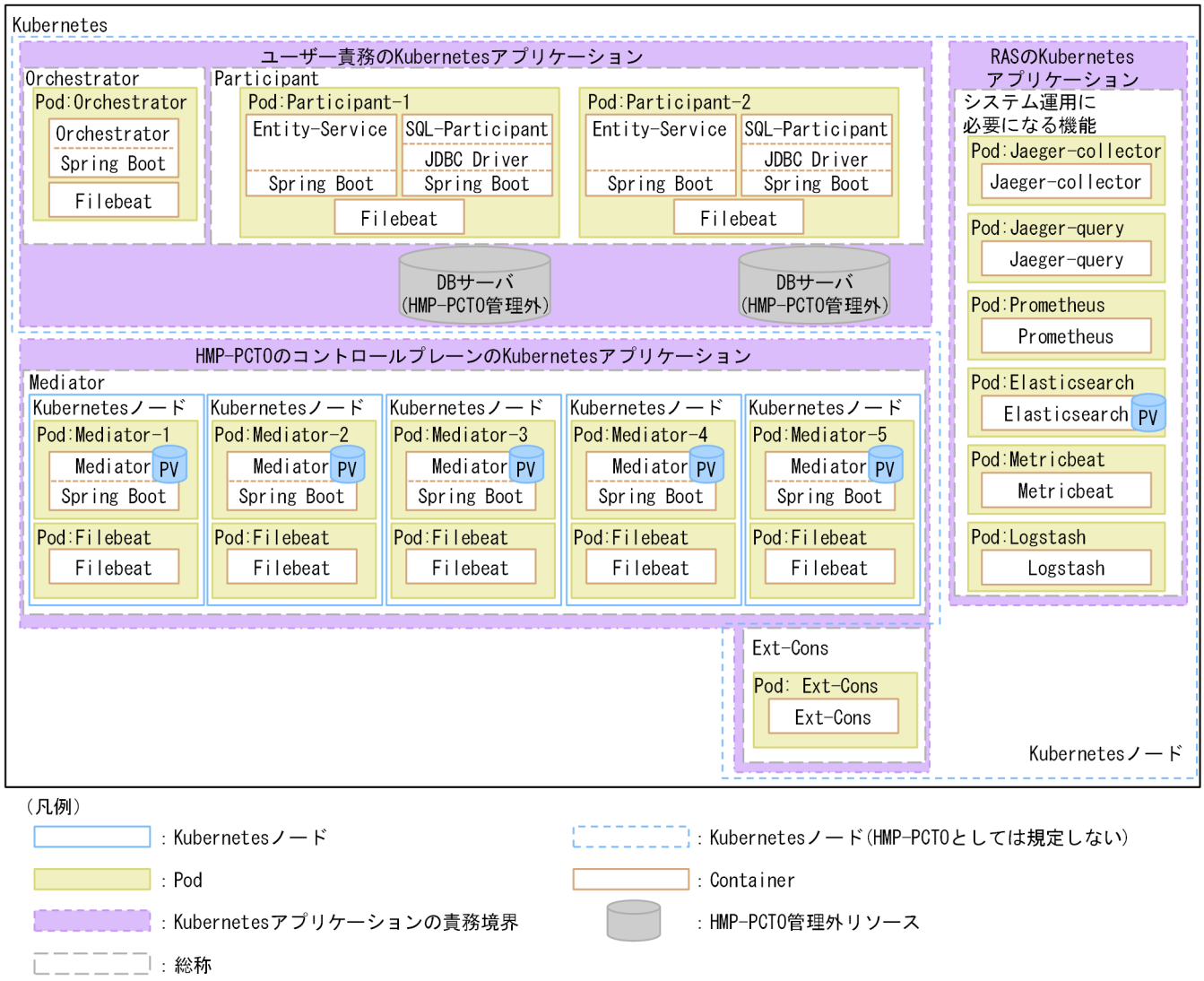
トライアル版では、EADS の代わりに Mock サーバを使用します。Mock サーバの場合、コンテナは 1 つで可用性を考慮しないため、別ノードとする必要はありません。

HMP-PCTO では Mediator を別ノードに配置することを推奨していますが、同一ノードにも配置できます。

- ThreeQuarters 合意方式での最小機器構成

  SQL-Participant を使用する場合の、ThreeQuarters 合意方式での最小機器構成を次の図に示します。

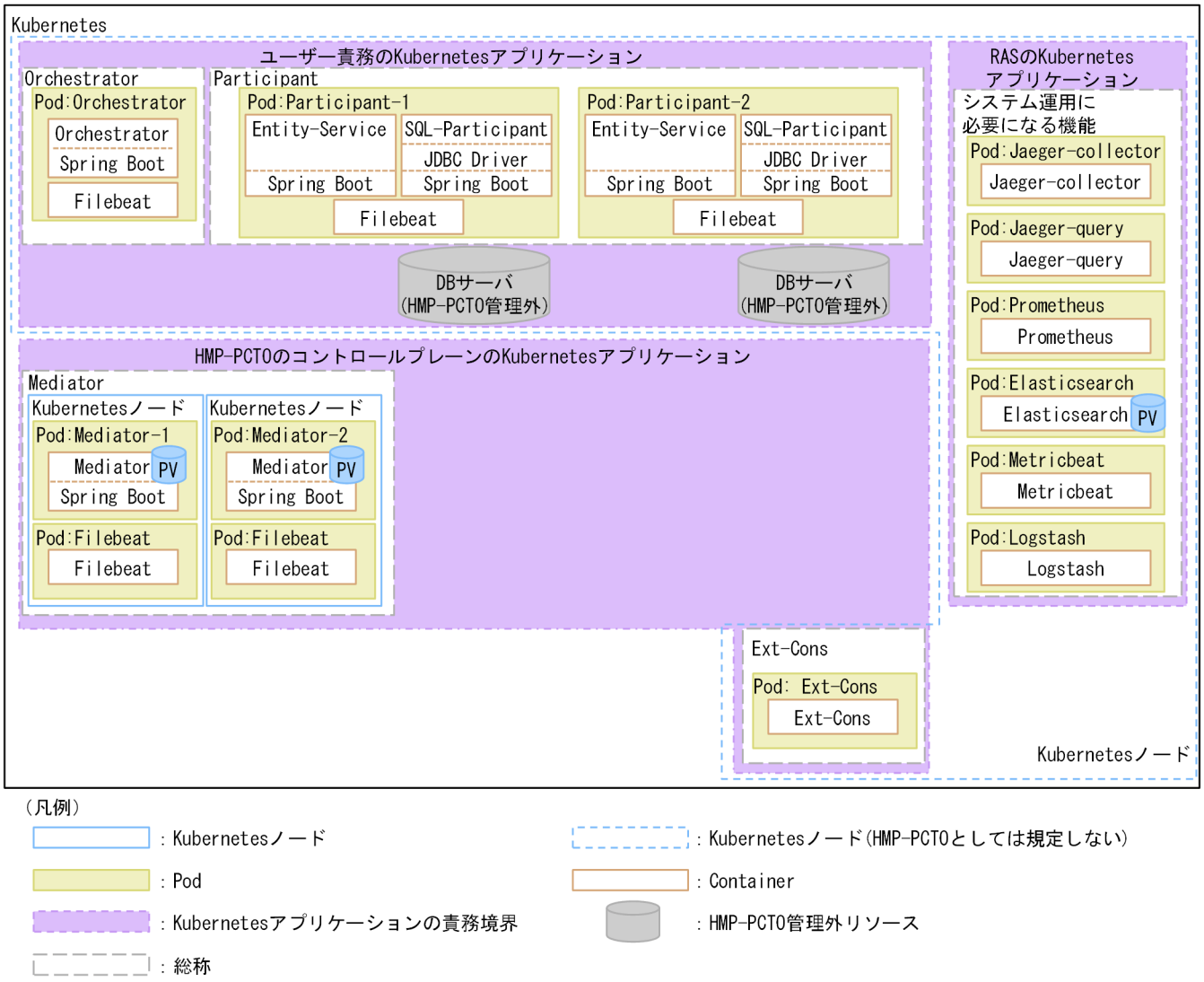
図 1-4　SQL-Participant を使用する場合の最小機器構成（ThreeQuarters 合意方式（トライアル版））



- PrivilegedValue 合意方式での最小機器構成

  SQL-Participant を使用する場合の、PrivilegedValue 合意方式での最小機器構成を次の図に示します。

図 1-5　SQL-Participant を使用する場合の最小機器構成（PrivilegedValue 合意方式（トライアル版））



1.3.2　TCC-Participant を使用する場合の最小機器構成（TCC-Participant 限定）

TCC-Participant を使用する場合の最小機器構成について説明します。

各 Mediator の Pod は可用性を確保するため、別々の Kubernetes ワーカーノードに配置し、物理的にも別のマシンとする必要があります。Mediator の最小個数は分散合意方式によって異なります。

各 Kubernetes アプリケーションをデプロイする Namespace の名称および数は任意とします。

(1) 通常版の場合

補助合意用の EADS クラスタ内の EADS サーバの Pod は可用性を確保するため、別々の Kubernetes ワーカーノードに配置し、物理的にも別のマシンとする必要があります。

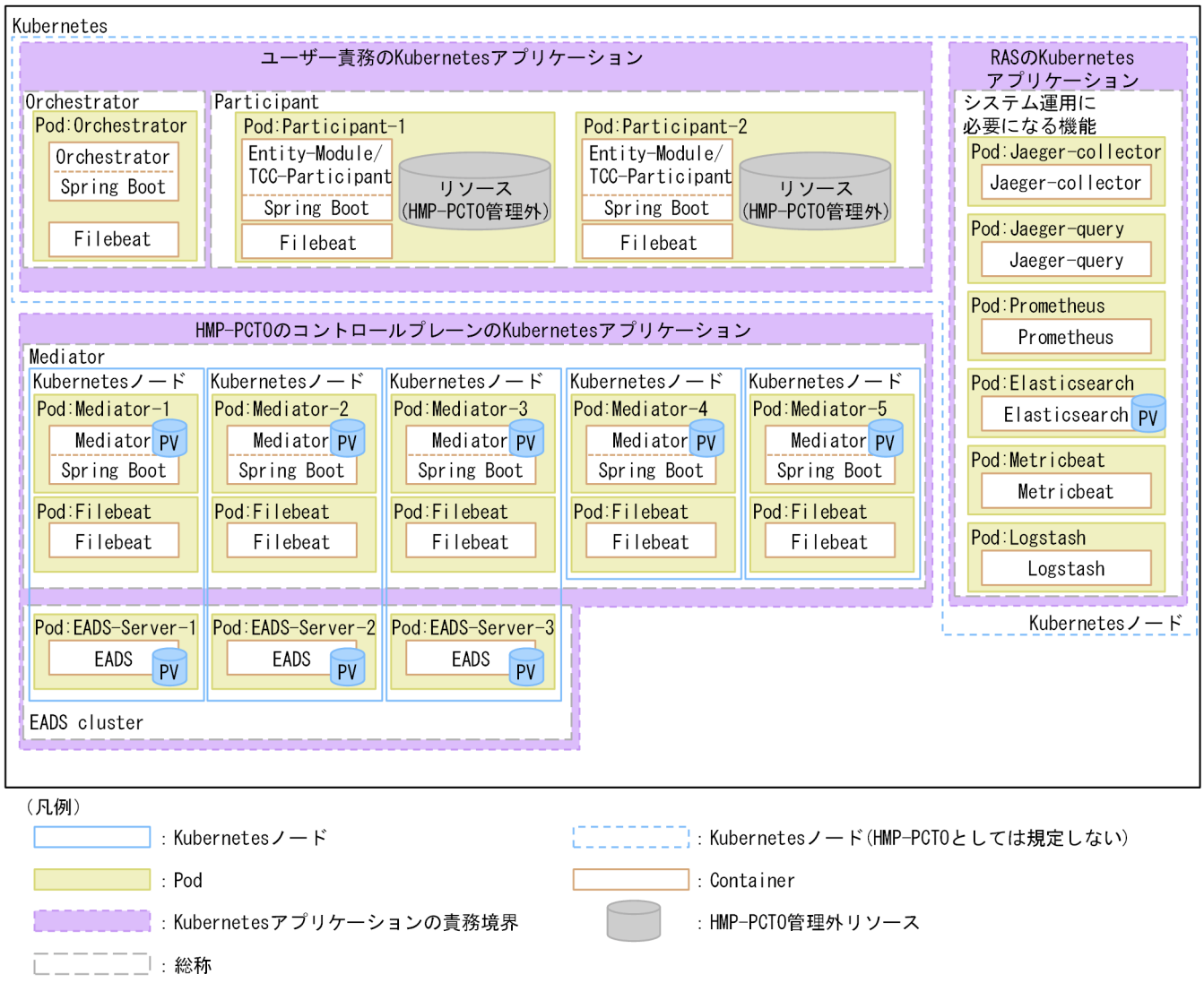
Mediator の Pod と EADS サーバの Pod は、同一のワーカーノードにも配置できます。

なお、各 Mediator の Pod と、補助合意用の EADS クラスタ内の Pod（EADS サーバの Pod を除く）は、任意に Kubernetes ワーカーノードに配置できます（HMP-PCTO では規定しません）。

- ThreeQuarters 合意方式での最小機器構成

  TCC-Participant を使用する場合の、ThreeQuarters 合意方式での最小機器構成を次の図に示します。

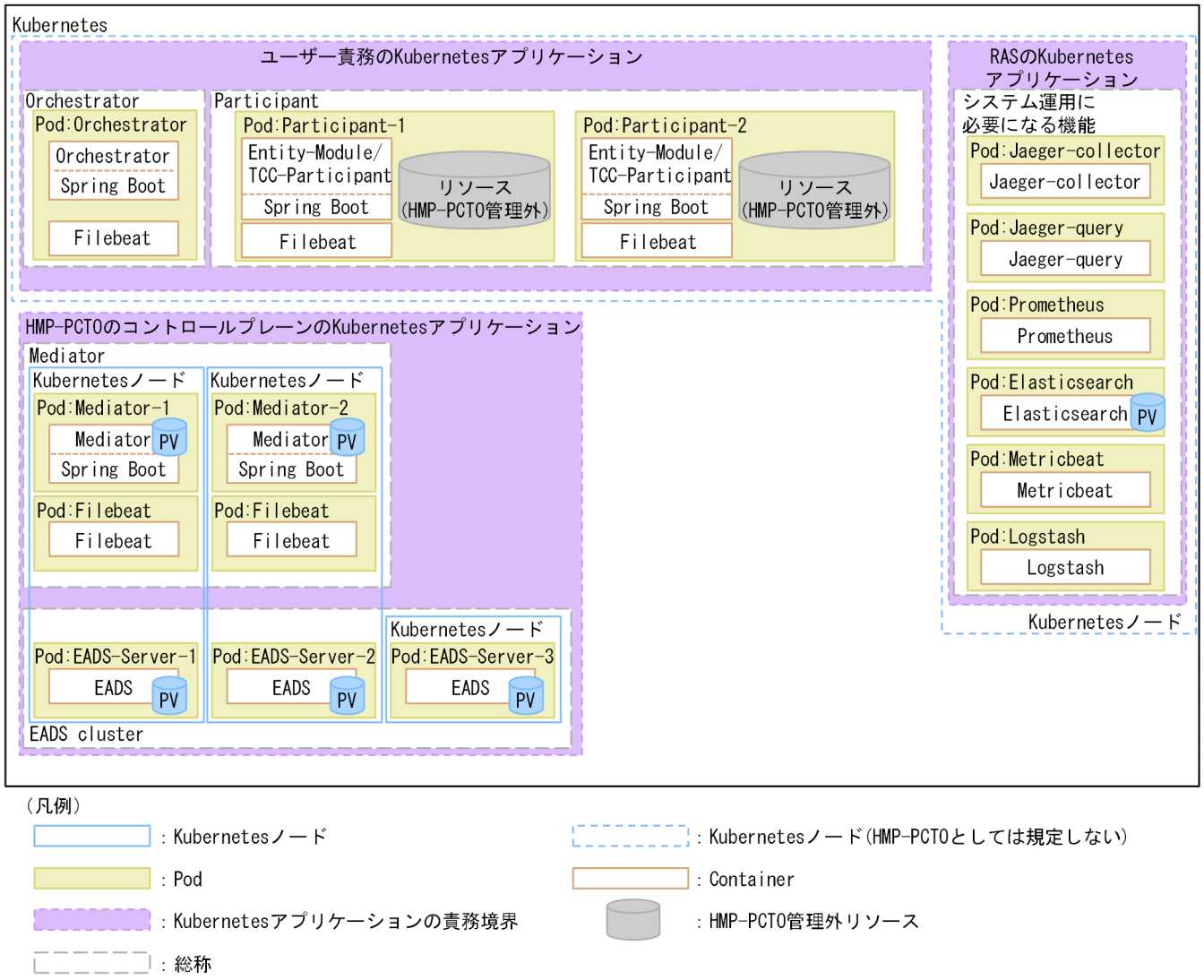
  図 1-6 TCC-Participant を使用する場合の最小機器構成（ThreeQuarters 合意方式（通常版））



- PrivilegedValue 合意方式での最小機器構成

  TCC-Participant を使用する場合の、PrivilegedValue 合意方式での最小機器構成を次の図に示します。

図 1-7　TCC-Participant を使用する場合の最小機器構成（PrivilegedValue 合意方式（通常版））



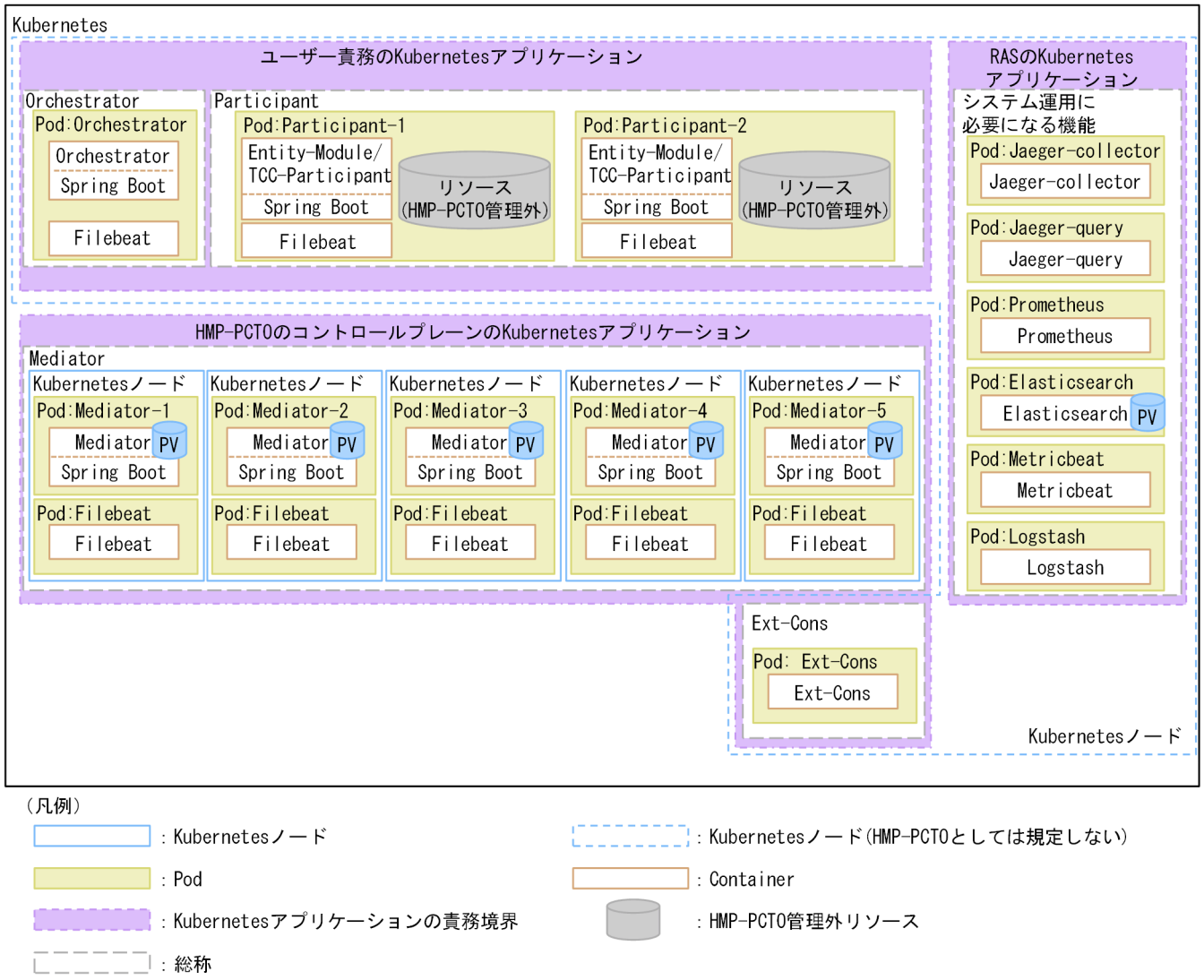
(2)　トライアル版の場合

トライアル版では、EADS の代わりに Mock サーバを使用します。Mock サーバの場合、コンテナは 1 つで可用性を考慮しないため、別ノードとする必要はありません。

HMP-PCTO では Mediator を別ノードに配置することを推奨していますが、同一ノードにも配置できます。

- ThreeQuarters 合意方式での最小機器構成
  TCC-Participant を使用する場合の、ThreeQuarters 合意方式での最小機器構成を次の図に示します。

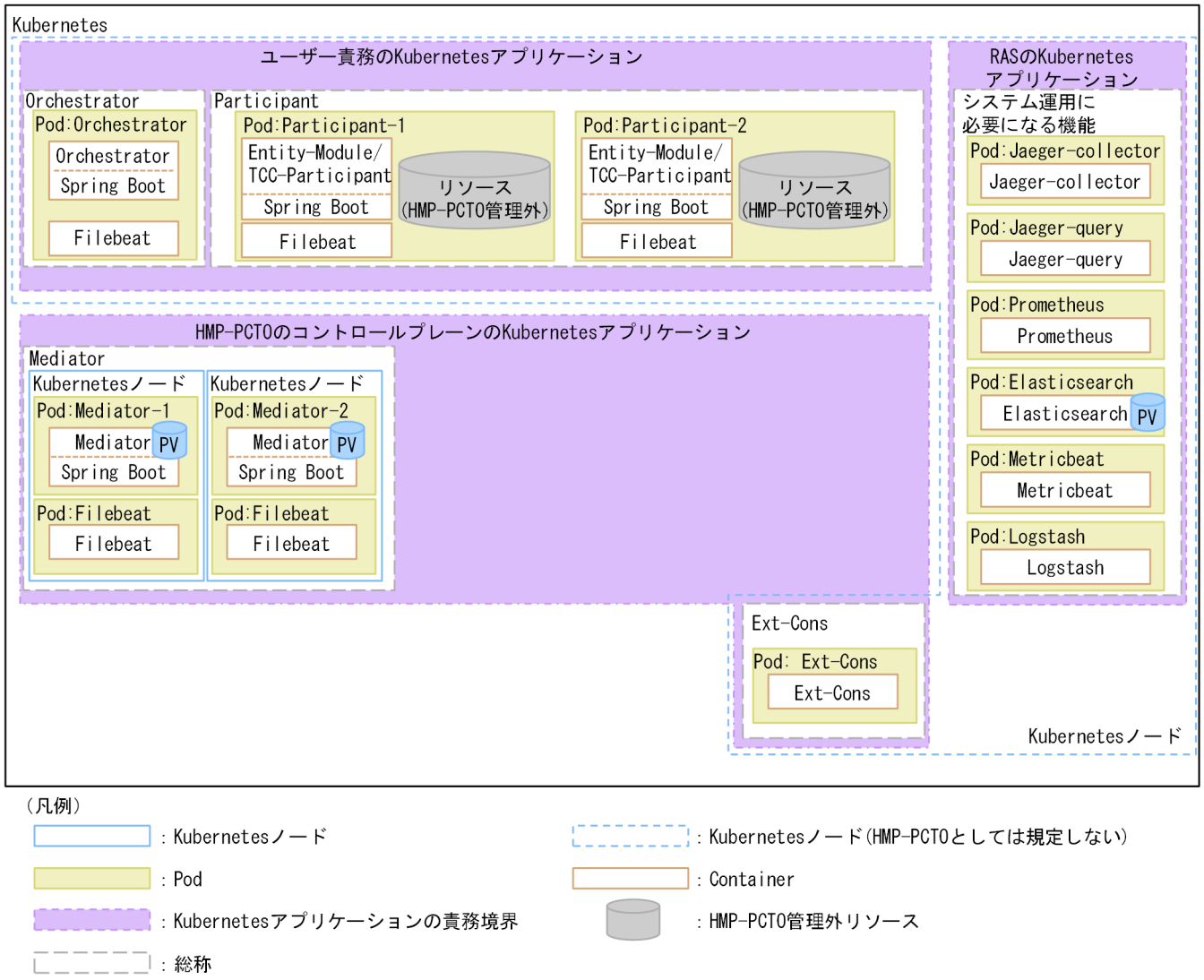
図 1-8　TCC-Participant を使用する場合の最小機器構成（ThreeQuarters 合意方式（トライアル版））



- PrivilegedValue 合意方式での最小機器構成

  TCC-Participant を使用する場合の、PrivilegedValue 合意方式での最小機器構成を次の図に示します。

図 1-9　TCC-Participant を使用する場合の最小機器構成（PrivilegedValue 合意方式（トライアル版））



1.3.3　TP1-Bridge を使用する場合の最小機器構成（通常版かつ TP1-Bridge 限定）

TP1-Bridge を使用する場合の最小機器構成について説明します。

各 Mediator の Pod は可用性を確保するため、別々の Kubernetes ワーカーノードに配置し、物理的にも別のマシンとする必要があります。

補助合意用の EADS クラスタ内の EADS サーバの Pod は可用性を確保するため、別々の Kubernetes ワーカーノードに配置し、物理的にも別のマシンとする必要があります。

Mediator の Pod と EADS サーバの Pod は、同一のワーカーノードにも配置できます。

Mediator の最小個数は分散合意方式によって異なります。

なお、各 Mediator の Pod と、補助合意用の EADS クラスタ内の Pod（EADS サーバの Pod を除く）は、任意に Kubernetes ワーカーノードに配置できます（HMP-PCTO では規定しません）。

OpenTP1 は、Kubernetes クラスタ外に配置してください。

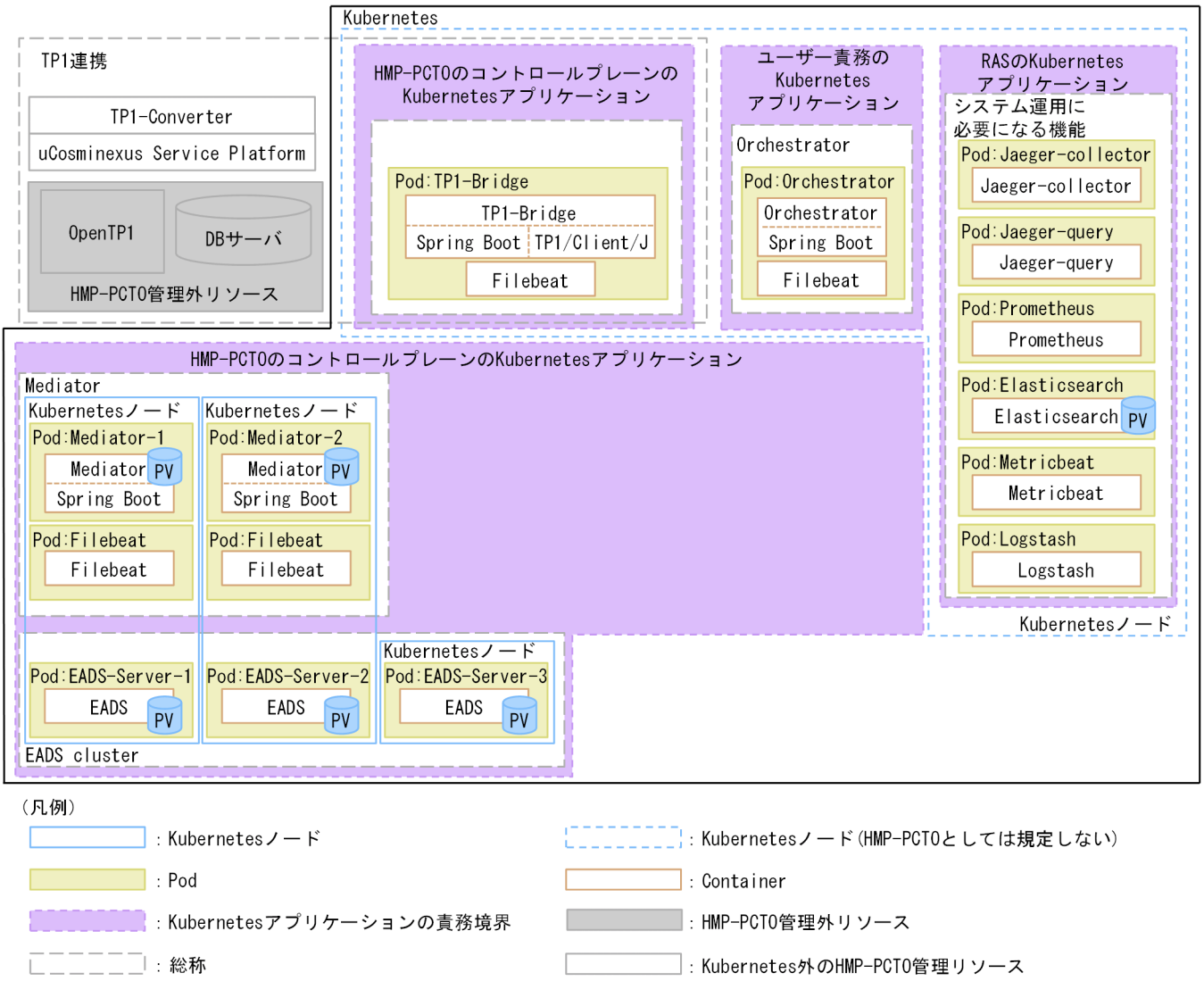
各 Kubernetes アプリケーションをデプロイする Namespace の名称および数は任意とします。

- ThreeQuarters 合意方式での最小機器構成
  TP1-Bridge を使用する場合の、ThreeQuarters 合意方式での最小機器構成を次の図に示します。

  図 1-10　TP1-Bridge を使用する場合の最小機器構成（ThreeQuarters 合意方式）

- PrivilegedValue 合意方式での最小機器構成
  TP1-Bridge を使用する場合の、PrivilegedValue 合意方式での最小機器構成を次の図に示します。

図 1-11　TP1-Bridge を使用する場合の最小機器構成（PrivilegedValue 合意方式）



1.4　前提条件

OS、プログラム、Kubernetes クラスタ構成、共有ファイルシステムなどの前提条件について説明します。

1.4.1　前提 OS および前提プログラム

HMP-PCTO の前提 OS および前提プログラムを次の表に示します。

表 1-5　HMP-PCTO の前提 OS および前提プログラム

項番	種別	前提 OS および前提プログラム
1	Kubernetes コンテナプラットフォーム	• Red Hat OpenShift Container Platform（オプション）
2	Kubernetes ディストリビューション	• Kubernetes 1.24.0 以降
3	Kubernetes パッケージマネージャ	• Helm
4	前提 OS （コンテナベースイメージ）	通常版の場合 • ユーザ責務のコンテナイメージのとき Red Hat Universal Base Image 8 8.1 以降 (64-bit x86_64)※1 • HMP-PCTO のコントロールプレーンのコンテナイメージのとき Red Hat Universal Base Image 8 Minimal 8.8-860 (64-bit x86_64) トライアル版の場合 Almalinux 8 Minimal 8.8
5	Java フレームワーク （AP サーバを含む）	• Spring Boot 2.7.18
6	Java ランタイム	通常版の場合 • uCosminexus Application Runtime with Java for Spring Boot 導入ありのとき※2 Java SE 8 の uCosminexus Application Runtime with Java for Spring Boot 01-00 以降 • uCosminexus Application Runtime with Java for Spring Boot 導入なしのとき ・ユーザ責務のコンテナイメージのとき：Java SE 8 準拠の JDK ・HMP-PCTO のコントロールプレーンのコンテナイメージのとき：java-1.8.0-openjdk-headless-1.8.0.372.b07 トライアル版の場合 • ユーザ責務のコンテナイメージのとき Java SE 8 準拠の JDK • HMP-PCTO のコントロールプレーンのコンテナイメージのとき java-1.8.0-openjdk-headless-1.8.0.392.b08
7	Java ライブラリ※3	同梱の表記がないものはこの製品に同梱しない。

項番	種別	前提 OS および前提プログラム
		• Hitachi Elastic Application Data Store Client for Java 06-20 以降（通常版の場合だけ同梱） • com.google.protobuf:protobuf-java-util 3.24.0 • io.grpc:grpc-api 1.59.1 • io.grpc:grpc-netty 1.59.1 • io.grpc:grpc-netty-shaded 1.59.1 • io.grpc:grpc-protobuf 1.59.1 • io.grpc:grpc-services 1.59.1 • io.grpc:grpc-stub 1.59.1 • io.kubernetes:client-java 14.0.0 • io.micrometer:micrometer-core 1.9.17 • io.micrometer:micrometer-registry-prometheus 1.9.17 • io.opentelemetry.instrumentation:opentelemetry-grpc-1.6 2.4.0-alpha • io.opentelemetry:opentelemetry-api 1.38.0 • io.opentelemetry:opentelemetry-sdk 1.38.0 • io.opentelemetry:opentelemetry-exporter-jaeger-thrift:1.34.1 • javax.annotation:javax.annotation-api 1.3.2 • javax.json.bind:javax.json.bind-api 1.0 • javax.ws.rs:javax.ws.rs-api 2.1.1 • net.devh:grpc-client-spring-boot-autoconfigure 2.13.1.RELEASE • net.devh:grpc-server-spring-boot-autoconfigure 2.13.1.RELEASE • org.eclipse.microprofile.config:microprofile-config-api 1.4 • org.eclipse:yasson 1.0.11 • org.globallogic.microservice:msa-imperative-modules-isc-rest 1.6.5 • org.osgi:org.osgi.annotation.bundle 2.0.0 • org.projectlombok:lombok 1.18.30 • org.quartz-scheduler:quartz 2.3.2 • org.springframework.boot:spring-boot-starter-actuator 2.7.18 • org.springframework.boot:spring-boot-starter-web 2.7.18 • org.springframework:spring-aspects 5.3.31 • org.springframework:spring-web 5.3.31
8	RPM パッケージ	• curl 7.61.1 • glibc 2.28 • jq 1.6 • ncompress • ncurses • tar • cpio • gzip

項番	種別	前提 OS および前提プログラム
9	JDBC Driver	• PostgreSQL 13 に接続する場合 PostgreSQL JDBC Driver 42.x.x（x.x は 7.1 以降） • SQL Server 2022 に接続する場合 Microsoft JDBC Driver 12.x for SQL Server（x は 6 以降） • HiRDB 10 に接続する場合 HiRDB type4 JDBC Driver（10-xx）（xx は 07 以降）
10	TP1/Client/J	• TP1 連携機能を使用する場合 P-2464-73B4 uCosminexus TP1/Client/J 07-53 以降
11	uCosminexus Service Platform	• TP1 連携機能を使用する場合 P-9W43-7SY1 uCosminexu Service Platform11-30 以降 （TP1-Converter 実行環境に必要です）
12	uCosminexus Service Architect	• TP1 連携機能を使用する場合 P-2943-7TY4 uCosminexus Service Architect 11-40 以降 （データ変換定義ファイルを作成するために開発環境に必要です）

注　前提 OS および前提プログラムにバージョンの記載がない場合、バージョンに制限はありません。

注※1

UBI 8 の任意のベースイメージ（UBI 8 Minimal など）

注※2

Mediator では uCosminexus Application Runtime with Java for Spring Boot の HTTP 機能は使用できません。

注※3

ユーザアプリケーションを開発する場合は，記載されているバージョンを使用してください。記載されていないバージョンを使用したときは、動作保証の対象外となります。

1.4.2　関連プログラム

HMP-PCTO の関連プログラムを次の表に示します。

表 1-6　HMP-PCTO の関連プログラム

項番	種別	関連プログラム
1	関連プログラム （同一 Kubernetes クラスタ内に起動）	• Hitachi Elastic Application Data Store 06-20（通常版の場合だけ） • Hitachi Microservices Platform-Application Distributed Integration Framework 02-04（オプション） • Jaeger-collector 1.56.0 以降※1 • Jaeger-query 1.56.0 以降※1

項番	種別	関連プログラム
		• Prometheus 2.53.1 以降※1※2 • Elasticsearch 8.13.0 以降※1 • Filebeat 8.13.0 以降※1 • Logstash 8.13.0 以降※1 • Metricbeat 8.13.0 以降※1
		• PostgreSQL に接続する場合 PostgreSQL 13 • SQL Server に接続する場合 SQL Server 2022
2	関連プログラム (Kubernetes クラスタ外に起動)	• P-8164-2111 uCosminexus TP1/Server Base 07-56 以降または P-8264-2111 uCosminexus TP1/Server Base(64) 07-56 以降 • HiRDB に接続する場合 HiRDB 10

注※1

公式ページからリリースノートを参照し、後方互換のあるバージョンを使用してください。また、メジャーアップデートバージョンが異なるバージョンは使用しないでください。

注※2

Prometheus は、LTS 版の 2.53 系列を使用してください。2.54 などの異なるマイナーバージョンは使用しないでください。

重要

OSS バージョンアップ時の注意事項

OSS のバージョンアップをする際の手順については、各 OSS の公式ドキュメントを参照してください。

1.4.3 前提とする Kubernetes クラスタ構成

HMP-PCTO が前提とする Kubernetes クラスタ構成を次の表に示します。

表 1-7 HMP-PCTO が前提とする Kubernetes クラスタ構成

項番	項目	前提条件
1	Kubernetes クラスタ数	1 クラスタだけ
2	リージョン数 (パブリッククラウドを使用する場合)	1 リージョンだけ
3	ゾーン数 (パブリッククラウドを使用する場合)	1 ゾーンだけ
4	Kubernetes ワーカーノード数	分散合意方式によって、次のとおり異なります。

項番	項目	前提条件
		Kubernetes ワーカーノードはそれぞれ物理的に別マシンである必要があります。 • ThreeQuarters 合意方式の場合 Kubernetes ワーカーノード数：5 以上 • PrivilegedValue 合意方式の場合 Kubernetes ワーカーノード数：3 以上

1.4.4　前提とする共有ファイルシステム

HMP-PCTO が前提とする共有ファイルシステムを次の表に示します。

表 1-8　HMP-PCTO が前提とする共有ファイルシステム

項番	項目	前提条件
1	NFS	Kubernetes クラスタからアクセス可能

1.5 HMP-PCTO のアプリケーション開発からシステム運用までの流れ

HMP-PCTO の Kubernetes アプリケーションの開発を含む運用は次の流れで実施します。

1. Kubernetes アプリケーションをデプロイする
   - ユーザ責務の Kubernetes アプリケーションを開発する場合（2. ユーザ責務の Kubernetes アプリケーション開発）

     HMP-PCTO の業務ロジックを担う Kubernetes アプリケーションを開発し、コンテナイメージを作成して任意の方法でデプロイします。
   - HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションをデプロイする場合（3. HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイ）

     HMP-PCTO のトランザクション制御の機能を担うコントロールプレーンの Kubernetes アプリケーション、およびシステム運用に必要な機能を担う Kubernetes アプリケーションをデプロイします。

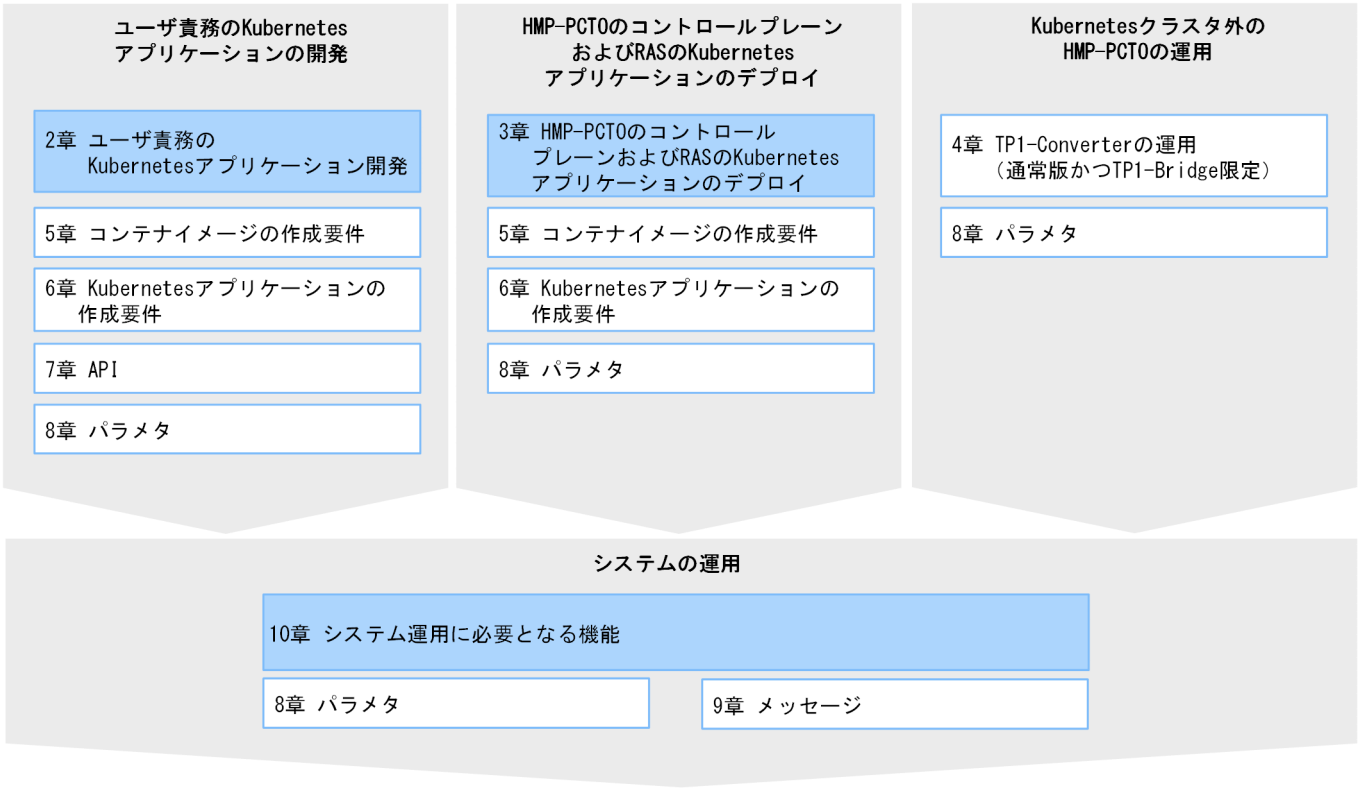
     各 Kubernetes アプリケーションは、コンテナイメージを作成し、Kubernetes マニフェストを作成、または Helm チャートをカスタマイズしてデプロイします。
2. HMP-PCTO を適用したシステムを運用する（10. システム運用に必要となる機能）

   システムの運用を開始します。通信障害、連携製品の障害などを検知し原因を調査するために、必要な情報を取得します。

なお、Kubernetes クラスタ外の他製品と連携する場合は、2.の前に、連携に必要なアプリケーションの運用、各種定義の設定などを実施します（4. TP1-Converter の運用（通常版かつ TP1-Bridge 限定））。

実施する内容に応じて、次の図で示す「必ず読む章」をお読みください。また、API、パラメタ、メッセージなどのリファレンス情報が必要な際に「必要に応じて読む章」で示す個所を参照してください。章・節・項タイトル末尾に「(～限定)」とある個所については、使用する製品や機能に該当する場合にだけお読みください。

図 1-12　HMP-PCTO のアプリケーション開発からシステム運用までの流れ



（凡例）

：必ず読む章　　：必要に応じて読む章

2 ユーザ責務の Kubernetes アプリケーション開発

2.1　ユーザ責務の Kubernetes アプリケーションの開発の流れ

HMP-PCTO を用いたユーザ責務の Kubernetes アプリケーションは次の流れで開発します。

表 2-1　ユーザ責務の Kubernetes アプリケーションの開発の流れ

項番	項目	説明	参照先
1	Java アプリケーションの開発	HMP-PCTO が提供する Java ライブラリを使用してユーザの Java アプリケーションを開発します。	2.2　Java アプリケーションの開発
2	SQL-Participant のビルド	HMP-PCTO の提供物を使用して SQL-Participant をビルドします。SQL-Participant を使用する場合だけ実施します。	2.3　SQL-Participant のビルド（SQL-Participant 限定）
3	コンテナイメージ作成	HMP-PCTO が提供する Dockerfile の記述例を基に Dockerfile を作成・ビルドします。	2.4　コンテナイメージ作成
4	共有ファイルシステムでの作業	共有ファイルシステムで、HMP-PCTO で必要なディレクトリを用意します。通常版の場合だけ実施します。	2.5　共有ファイルシステムでの作業（通常版限定）
5	Kubernetes マニフェスト作成	HMP-PCTO が提供する Kubernetes マニフェストの記述例を基に Kubernetes マニフェストを作成します。	2.6　Kubernetes マニフェスト作成
6	デプロイ	Kubernetes アプリケーションを、ユーザ任意の方法で Kubernetes クラスタにデプロイします。	2.7　ユーザ責務の Kubernetes アプリケーションのデプロイとアンデプロイ

重要

トライアル版 HMP-PCTO を使用して構築した環境は、通常版 HMP-PCTO にそのまま移行できません。

トライアル版 HMP-PCTO を使用してデプロイまで完了している場合に、通常版 HMP-PCTO を使用して環境を構築したいときは、「(8)　ユーザ責務の Kubernetes アプリケーションのアンデプロイ」を実施したあと、「ユーザ責務の Kubernetes アプリケーションの開発の流れ」の手順を 1 から実施してください。各手順で通常版、トライアル版で説明が分かれている場合は通常版の手順で実施してください。

2.2 Java アプリケーションの開発

HMP-PCTO ライブラリを使用した Java アプリケーションの開発方法について説明します。

Java アプリケーションの開発環境のビルド自動化システムには Gradle を使用する前提で説明します。説明中の Gradle の構文例は、Gradle7.3.3 を使用した場合を例に記述しています。

2.2.1 作業項目の一覧

Java アプリケーション開発の作業項目の一覧を次の表に示します。

表 2-2 Java アプリケーション開発の作業項目の一覧

分類	作業項目	コンポーネント	備考	参照先
共通	事前準備	全コンポーネント共通	必ず実施します。	2.2.2 事前準備
	ライブラリのセットアップ（通常版限定）	全コンポーネント共通	通常版の場合だけ実施します。	2.2.3 ライブラリのセットアップ（通常版限定）
Java アプリケーション開発	コンポーネントスキャンの追加	• Orchestrator • Entity-Service • TCC-Participant	必ず実施します。	2.2.4 コンポーネントスキャンの追加
	トランザクション境界の設定	Orchestrator	必ず実施します。	2.2.6 トランザクション境界の設定
	トランザクションの伝搬	• Orchestrator • Entity-Service	必ず実施します。	2.2.7 トランザクションの伝搬
	トランザクションブランチの属性の設定	• Entity-Service • TCC-Participant	必ず実施します。	2.2.8 トランザクションブランチの属性の設定
	先行プリペア機能の使用（SQL-Participant 限定）	Entity-Service	SQL-Participant を使用する場合だけ実施します	2.2.9 先行プリペア機能の使用（SQL-Participant 限定）
	JDBC Proxy Driver の使用（SQL-Participant 限定）	Entity-Service	SQL-Participant を使用する場合だけ実施します。	2.2.10 JDBC Proxy Driver の使用（SQL-Participant 限定）
	TccService インタフェースの実装クラスの作成（TCC-Participant 限定）	TCC-Participant	TCC-Participant を使用する場合だけ実施します。	2.2.11 TccService インタフェースの実装

分類	作業項目	コンポーネント	備考	参照先
				クラスの作成（TCC-Participant 限定）
	業務ロジックからのexecuteTry()メソッドの呼び出し（TCC-Participant 限定）	TCC-Participant	TCC-Participant を使用する場合だけ実施します。	2.2.12　業務ロジックからの executeTry メソッドの呼び出し（TCC-Participant 限定）
	TP1-Bridge の使用（通常版かつ TP1-Bridge 限定）	• Orchestrator • Entity-Service	TP1-Bridge を使用する場合だけ実施します。	2.2.13　TP1-Bridge の使用（通常版かつ TP1-Bridge 限定）
	ロギングライブラリの設定ファイルのカスタマイズ	• Orchestrator • Entity-Service	アプリケーションでロギングライブラリ Logback を使用する場合だけ実施します。	2.2.14　ロギングライブラリの設定ファイルのカスタマイズ
	Quartz スケジューラ機能の使用	• SQL-Participan • TCC-Participant	アプリケーションでジョブスケジューリングライブラリ Quartz を使用する場合だけ実施します。	2.2.15　Quartz スケジューラ機能の利用

2.2.2　事前準備

事前準備については「3.2.2　事前準備（通常版限定）」を参照してください。

2.2.3　ライブラリのセットアップ（通常版限定）

ライブラリのセットアップについては「3.2.3　ライブラリのセットアップ（通常版限定）」を参照してください。

2.2.4　コンポーネントスキャンの追加

コンポーネントスキャンの追加は、Orchestrator、Entity-Service、TCC-Participant のアプリケーションで実装してください。

アプリケーションが HMP-PCTO のコンポーネントを DI できるようにするために、@SpringBootApplication アノテーションを付与した main メソッドを持つクラスに、HMP-PCTO のパッケージをコンポーネントスキャンする設定を記述してください。記述する際には、アプリケーションのパッケージと HMP-PCTO のパッケージの両方を記述してください。

アプリケーションのパッケージが"com.example.demo"の場合のコーディング例

```
@ComponentScan({"com.example.demo", "jp.co.Hitachi.soft.hmppcto"})
@SpringBootApplication
```

2.2.5　DI に使用するアノテーションの注意事項

DI に使用するアノテーションは、@Autowired アノテーションを使用してください。@Inject など他のアノテーションで DI した場合の動作は保障しません。

2.2.6　トランザクション境界の設定

トランザクション境界の設定をするアプリケーションの開発方法について説明します。

トランザクション境界の設定は、Orchestrator のアプリケーションで実装してください。

トランザクション境界の設定は、@PctoTransactional アノテーションを使用します。
@PctoTransactional アノテーションの API 仕様の詳細については、「7.　API」を参照してください。
@PctoTransactional アノテーションを付与したメソッドは、別のクラスから呼び出してください。同一クラスの別メソッドから呼び出した場合、HMP-PCTO の機能は正しく動作しません。

(1)　ライブラリの組み込み

@PctoTransactional を使用するには、hmppcto-orchestrator ライブラリをアプリケーションに組み込む必要があります。Gradle で組み込む記載例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-orchestrator:<version>'
}
```

(2)　コーディング例

@PctoTransactional アノテーションを使用するコーディング例を次に示します。@PctoTransactional アノテーションを使用するメソッドのアクセス制御修飾子は、private 以外にしてください。private とした場合は、HMP-PCTO の機能は正しく動作しません。

```
@PctoTransactional(rollbackOn = { java.io.IOException.class}, dontRollbackOn = {})
public void service() {

    // 業務処理

}
```

(3) エラーハンドリング

@PctoTransactional アノテーションを設定したメソッドの呼び出し元には、障害発生時に HMP-PCTO の例外が送出されます。例外によってトランザクションの結果を確認できます。必要に応じて例外を catch して後処理をしてください。

サービスのルートとなるメソッドに@PctoTransactional アノテーションを指定した場合、例外がどのようなエラー処理にマッピングされるかは、使用しているフレームワーク、AP サーバの仕様に依存します。@PctoTransactional アノテーションは、ルートとなるメソッド以外に設定してエラーハンドリングすることを推奨します。

トランザクションの結果と例外の関係を次の表に示します。

表 2-3　トランザクションの結果と例外の関係

項番	トランザクションの結果	例外	備考
1	トランザクション未実行	TransactionSystemException	トランザクション開始時にトランザクションの合意に必要な数の Mediator が起動していない場合など、トランザクションが開始できなかった場合に発生します。 この例外が送出された場合、@PctoTransactional アノテーションを設定したメソッドは実行されていません。
2	コミット決着	なし、または@PctoTransactional アノテーションを設定したメソッドが送出した例外	－
3	ロールバック決着	RollbackTerminationException	@PctoTransactional アノテーションを設定したメソッドが送出した例外は、RollbackTerminationException の getSuppressed()メソッドで受け取ることができます。
4	結果不明	TerminationFailureException	Mediator との通信障害によって結果が受け取れなかった場合に発生します。トランザクションの決着結果の確認方法については、「10.　システム運用に必要となる機能」を参照してください。 @PctoTransactional アノテーションを設定したメソッドが送出した例外は、TerminationFailureException の getSuppressed()メソッドで受け取ることができます。

2.2.7　トランザクションの伝搬

トランザクションの伝搬をするアプリケーションの開発方法について説明します。

トランザクションの伝搬は、Orchestrator、Entity-Service のアプリケーションで実装してください。

トランザクションの伝搬は、Entity-service または TP1-Bridge の呼び出しに使用するクライアントによって設定方法が異なります。使用するクライアントに対応する参照先を参照してください。

表 2-4　トランザクションの伝搬に使用するクライアント

項番	トランザクションの伝搬に使用するクライアント	参照先
1	Spring Framework の RestTemplate	(1)　Spring Framework の RestTemplate を使用する場合
2	HMP-ADIF の ISC コマンド（Imperative かつ REST 通信）	(2)　HMP-ADIF の ISC コマンド（Imperative かつ REST 通信）を使用する場合

(1)　Spring Framework の RestTemplate を使用する場合

TxPropagationRestTemplateInterceptor クラスを org.springframework.web.client.RestTemplate クラスの setInterceptors()メソッド、または org.springframework.boot.web.client.RestTemplateBuilder クラスの interceptors()メソッドで登録して使用します。

(a)　ライブラリの組み込み

TxPropagationRestTemplateInterceptor クラスを登録するには、hmppcto-propagation-rest-template ライブラリをアプリケーションに組み込む必要があります。Gradle で組み込む記載例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-propagation-rest-template:<version>'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-common:<version>'
}
```

(b)　コーディング例

org.springframework.web.client.RestTemplate クラスの setInterceptors()メソッドを使用して登録する例を説明します。アプリケーションが使用するほかのインタセプタとの登録順番に制約はありません。

TxPropagationRestTemplateInterceptor クラスのインスタンスは@Autowired アノテーションで DI して取得してください。DI したインスタンスを使用するメソッドのアクセス制御修飾子は、private 以外にしてください。private とした場合は、HMP-PCTO の機能は正しく動作しません。

```
@Autowired
TxPropagationRestTemplateInterceptor txIc;

RestTemplate restTemplate = new RestTemplate();                    // RestTemplateのインスタンス作成
List<ClientHttpRequestInterceptor> interceptors = new ArrayList<>();   // インタセプタのリストを作成
```

```
interceptors.add(txIc);                                    // HMP-PCTOのインタセプタを
リストに追加
restTemplate.setInterceptors(interceptors);                // RestTemplateにインタセプ
タを登録
```

(c) エラーハンドリング

TxPropagationRestTemplateInterceptor クラスでは、アプリケーションでエラーハンドリングが必要なエラーが発生することはありません。

RestTemplate を使用した通信で例外が発生した場合に、例外の Caused に次のように HMP-PCTO の TxPropagationRestTemplateInterceptor クラスで例外が発生したことを示すスタックトレースが出力されます。これは、例外が発生した直接の原因を示すものではありません。直接の原因は、例外の Caused の前に出力された内容を参照してください。

```
Caused by: java.net.SocketException: Unexpected end of file from server
:
at org.springframework.http.client.AbstractClientHttpResponse.getStatusCode(AbstractClientHttpResponse.java:33)
at jp.co.Hitachi.soft.hmppcto.filter.propagation.TxPropagationRestTemplateInterceptor.intercept(TxPropagationRestTemplateInterceptor.java:93)
:
```

(2) HMP-ADIF の ISC コマンド（Imperative かつ REST 通信）を使用する場合

TxPropagationIscImperativePlugin クラスをアプリケーションに組み込んで使用します。TxPropagationIscImperativePlugin クラスは、HMP-ADIF の org.globallogic.microservice.modules.isc.imperative.IscImperativeCommandExecutionPlugin インタフェースを実装しています。アプリケーションに組み込むとアプリケーション起動時に HMP-ADIF が自動的に登録します。

(a) ライブラリの組み込み

TxPropagationIscImperativePlugin クラスを登録するには、hmppcto-propagation-isc-imperative ライブラリをアプリケーションに組み込む必要があります。Gradle で組み込む記載例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-propagation-isc-imperative:<version>'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-common:<version>'
}
```

(b) コーディング例

アプリケーション起動時に HMP-ADIF が自動的に登録するため、コーディングは不要です。

(c) エラーハンドリング

TxPropagationIscImperativePlugin クラスでは、アプリケーションでエラーハンドリングが必要なエラーが発生することはありません。

2.2.8　トランザクションブランチの属性の設定

トランザクションブランチとして動作するアプリケーションの開発方法について説明します。

トランザクションブランチとして動作するアプリケーションは、Entity-Service、TCC-Participant のアプリケーションで実装してください。

また、@PctoBranchRollbackFor アノテーションを使用してトランザクションブランチの属性を設定する方法を説明します。@PctoBranchRollbackFor アノテーションの API 仕様の詳細については、「7.　API」を参照してください。@PctoBranchRollbackFor アノテーションを付与したメソッドは、別のクラスから呼び出してください。同一クラスの別メソッドから呼び出した場合、HMP-PCTO の機能は正しく動作しません。

@PctoBranchRollbackFor アノテーションを使用しない場合でも、ライブラリの組み込みは必要です。

(1)　ライブラリの組み込み

トランザクションブランチとして動作するアプリケーションには、hmppcto-entity-service ライブラリと javax.ws.rs-api ライブラリをアプリケーションに組み込む必要があります。Gradle で組み込む記載例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-entity-service:<version>'
    implementation 'javax.ws.rs:javax.ws.rs-api:<rs-apiのversion>※'
}
```

注※

*<rs-api の version>*に指定するバージョンは、「表 1-5　HMP-PCTO の前提 OS および前提プログラム」の「Java ライブラリ」の「javax.ws.rs:javax.ws.rs-api」を参照してください。

(2)　コーディング例

@PctoBranchRollbackFor アノテーションを使用してトランザクションブランチの属性を設定するコーディング例を次に示します。@PctoBranchRollbackFor アノテーションを使用するメソッドのアクセス制御修飾子は、private 以外にしてください。private とした場合は、HMP-PCTO の機能は正しく動作しません。

```
@PctoBranchRollbackFor(rollbackOn = {},rollbackOnFamily = {Family.CLIENT_ERROR, Family.SERVE
R_ERROR})
public void service() {

    // 業務処理

}
```

(3) エラーハンドリング

@PctoBranchRollbackFor アノテーションの設定によって、アプリケーションでエラーハンドリングが必要なエラーが発生することはありません。

Entity-Service のルートとなるメソッドが例外を送出した場合、HMP-PCTO が HTTP ステータスを 500 に設定します。Spring Boot の機能によってエラーページ（/error）へ遷移する前に、HMP-PCTO で HTTP ステータスを設定するため、エラーページをカスタマイズしていた場合でも HTTP ステータスは 500 になります。

次の場合はトランザクションをロールバックしません。そのため、トランザクションをロールバックしたい場合は、Orchestrator のアプリケーションでトランザクションがロールバックする例外を送出してください。

- Entity-Service のルートとなるメソッドが例外を送出した場合
- HTTP ステータス 404 の場合
- HTTP ステータス 405 の場合

(4) Spring Boot の機能を使用する場合の制限事項

HTTP ステータスコードの設定する手段として Spring Boot の@ResponseStatus アノテーションは使用しないでください。サーブレットの機能である HttpServletResponse インタフェースを使用して HTTP レスポンスを設定するか、HMP-ADIF の機能を使用して設定してください。

2.2.9 先行プリペア機能の使用（SQL-Participant 限定）

先行プリペア機能を使用するアプリケーションの開発方法について説明します。

先行プリペア機能を使用するアプリケーションは、Orchestrator、Entity-Service で実装してください。

先行プリペア機能を使用する場合、トランザクションを伝搬する際にアプリケーションが HMP-PCTO が提供するインタフェースを使用して、HTTP ヘッダを追加します。

(1) ライブラリの組み込み

先行プリペア機能を使用する場合に組み込むライブラリは、トランザクションの伝搬に使用するライブラリと同じです。ライブラリについては、「2.2.7 トランザクションの伝搬」を参照してください。

(2) コーディング例

先行プリペア機能を使用する際に追加する HTTP ヘッダ名と値は、XidHeaders インタフェースで提供します。ヘッダ名は XidHeaders.IS_LAST_CALL、値は XidHeaders.TRUE です。これを Entity-Service

呼び出し時に追加してください。トランザクションの伝搬に使用するクライアントごとに追加する例を次に示します。

(a) Spring Framework の RestTemplate を使用する場合

```
RequestEntity<MyRequest> request = RequestEntity                    // エンティティ作成
     .post("https://example.com/{foo}", "bar")
     .header(XidHeaders.IS_LAST_CALL, XidHeaders.TRUE)               // 先行プリペアヘッダ追加
     .body(body);
```

(b) HMP-ADIF の ISC コマンド（Imperative かつ REST 通信）を使用する場合

```
IscImperativeCreateCommandBuilder<?,?> request = Isc.create()  // ISCコマンド作成
      .on("sample")
      .using("samplePath")
      .header(XidHeaders.IS_LAST_CALL, XidHeaders.TRUE)        // 先行プリペアヘッダ追加
      .data(payload);
```

(3) エラーハンドリング

先行プリペア機能の使用によって、アプリケーションでエラーハンドリングが必要なエラーが発生することはありません。

2.2.10 JDBC Proxy Driver の使用（SQL-Participant 限定）

JDBC Proxy Driver を使用するアプリケーションの開発方法について説明します。

JDBC Proxy Driver を使用するアプリケーション、Entity-Service のアプリケーションで実装してください。

JDBC Proxy Driver を使用するには、HMP-PCTO のパラメタの設定が必要です。パラメタの設定方法については、「8. パラメタ」を参照してください。

(1) ライブラリの組み込み

JDBC Proxy Driver を使用するには、hmppcto-jdbc-proxy-driver ライブラリをアプリケーションに組み込む必要があります。Gradle で組み込む記載例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto: hmppcto-jdbc-proxy-driver:<version>'
}
```

(2) コーディング例

JDBC Proxy Driver は、各ベンダが提供する JDBC Driver を使用する場合と同様にコーディングして使用します。Datasource を使用して Connection インスタンスを取得して、任意の SQL を実行します。JDBC Proxy Driver は、同一のトランザクションブランチで初めて getConnection()が実行されたときに Participant をトランザクションに参加させます。そのため、アプリケーションで Connection インスタンスをプールして別のトランザクションブランチで Connection インスタンスを再利用しないでください。サービスごとに Datasource を使用して Connection インスタンスを取得してください。

```
@Autowired
DataSource ds;                                  // JDBC Proxy DriverのDataSourceを挿入
Connection con = ds.getConnection();            // Connectionを取得
con.…                                           // 任意のSQLを実行
```

(3) エラーハンドリング

JDBC Proxy Driver は、JDBC Driver の仕様に準拠した例外を送出します。各ベンダの JDBC Driver を直接使用している場合と同様にエラーハンドリングできます。

Participant で JDBC Driver を実行時に発生した例外の SQLState、例外コード、メッセージ、例外チェーンなどの情報は、Participant から Entity-Service に応答し、そのままアプリケーションに送出します。

Entity-Service と Participant の間の通信障害など、JDBC Proxy Driver 固有の例外は、java.sql.SQLException のサブクラスとして実装しているため、各ベンダの JDBC Driver が送出する例外と同様にエラーハンドリングできます。JDBC Proxy Driver 固有の例外の詳細については、「7. API」を参照してください。

取扱説明書「HMP-PCTO テクニカルガイド」の、DB アクセス機能の RDBMS ごとの注意事項も確認してください。

(4) 実行できる SQL

JDBC Proxy Driver を使用して実行できる SQL は、XA 接続時に実行可能な SQL です。これは、SQL-participant が DB との接続に XA 接続を使用しているためです。XA 接続時に実行可能な SQL については、使用する DB のマニュアルを参照してください（なお、DB の多くは、XA 接続時に定義系 SQL は使用できません）。

2.2.11 TccService インタフェースの実装クラスの作成（TCC-Participant 限定）

ここでは、TccService インタフェースの実装クラスの作成に関して説明します。

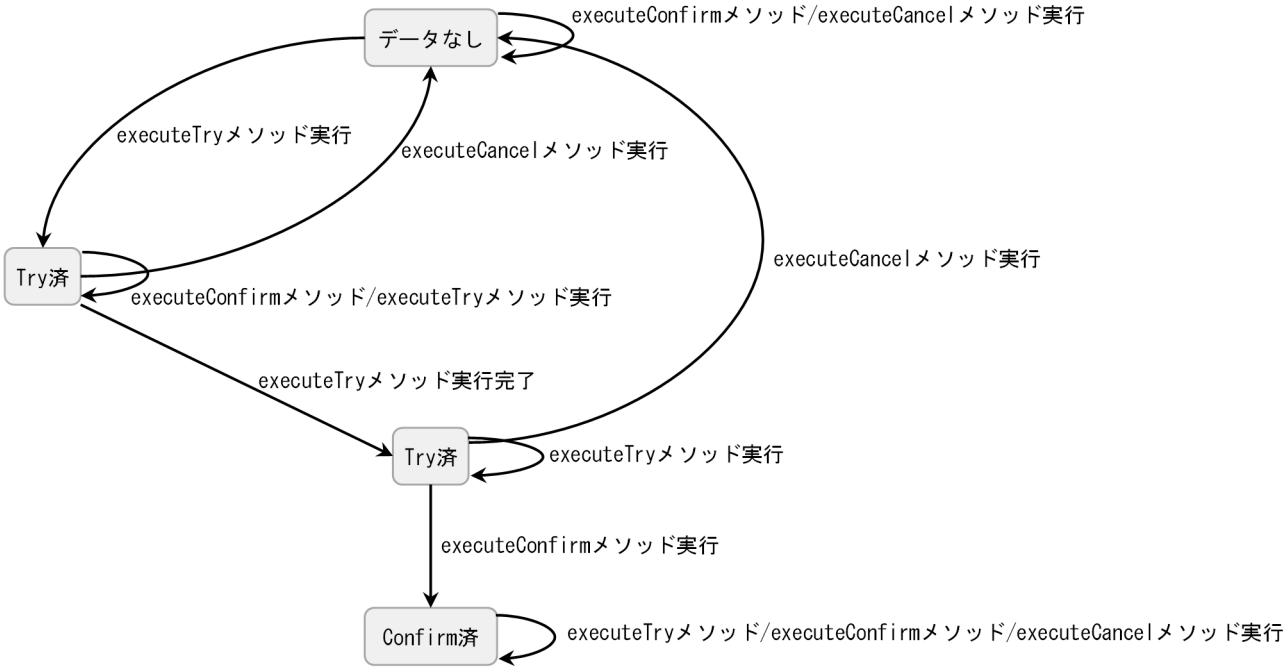
TccService インタフェースの実装クラスの作成は、TCC-Participant のアプリケーションで実装してください。

TccService インタフェースの詳細については、「7.4.5　TccService<I, O extends java.io.Serializable> インタフェース（TCC-Participant 限定）」を参照してください。なお、TCC-Participant では、TccService に実装された executeTry メソッド、executeConfirm メソッド、および executeCancel メソッドの実行によって、API で扱っているデータの状態は、次の表に示すデータの状態遷移、および次の図に示すデータのステートマシン図のように遷移することを想定しています。

表 2-5　データの状態遷移表

イベント	データの状態			
	データなし S0	Try 中 S1	Try 済み S2	Confirm 済み S3
executeTry メソッド実行	S1	S1	S2	S3
executeTry メソッド実行完了	−	S2	−	−
executeConfirm メソッド実行	S0	S1	S3	S3
executeCancel メソッド実行	S0	S0	S0	S3

図 2-1　データのステートマシン図



(1)　@Tcc アノテーションの付与

TccService インタフェースを実装したクラスと、TCC-Participant の機能の紐づけに関して説明します。@Tcc アノテーションを付与することで TccService インタフェースを実装したクラスと TCC-Participant が紐づけられます。これによって、TCC-Participant の処理中に TccService インタフェースを実装したクラスのメソッドが呼び出されて利用されます。

TccService インタフェースを実装したクラスを有効にするには、そのクラスを Bean として Spring IoC Container に登録します。

@Tcc アノテーションを付与したメソッドは、別のクラスから呼び出してください。同一クラスの別メソッドから呼び出した場合、HMP-PCTO の機能は正しく動作しません。

(a) ライブラリの組み込み

@Tcc を使用するには、hmppcto-participant-tcc ライブラリをアプリケーションに組み込む必要があります。Gradle で組み込む記載例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-participant-tcc:<version>'
}
```

また、通常版の場合、hmppcto-participant-tcc ライブラリは、Hitachi Elastic Application Data Store Client for Java の JAR ライブラリを外部読み込みします。このため、実行可能な JAR 形式を作成する際、Launcher クラスとして PropertiesLauncher を使用する必要があります。Gradle で Launcher クラスとして PropertiesLauncher を使用し、実行可能な JAR 形式を作成する記載例を次に示します。

```
bootJar {
    manifest {
        attributes 'Main-Class' : 'org.springframework.boot.loader.PropertiesLauncher'
    }
}
```

なお、トライアル版の場合、Launcher クラスの変更は不要です。

(b) コーディング例

@Tcc アノテーションを利用したコーディング例を次に示します。ここで示すコーディング例では、@Component アノテーションを付与することで、TccService インタフェースを実装したクラスを Spring IoC Container へ登録しています。@Tcc アノテーションを使用するメソッドのアクセス制御修飾子は、private 以外にしてください。private とした場合は、HMP-PCTO の機能は正しく動作しません。

```
@Component
@Tcc
public class TccServiceClass implements TccService<RequestClass, ResponseClass> {
}
```

(2) executeTry メソッドの実装

TCC-Participant では、次に示す機能が executeTry メソッドに実装されていることを想定して動作します。

- データの仮登録
- トランザクションを一意に表す文字列とデータの紐づけの永続化
- TccResponse クラスの返却

(a)　データの仮登録

一般的に TCC パターンの Try フェーズでは、データの仮登録を行います。仮登録であることが分かる状態をデータに持たせる必要があります。

(b)　トランザクションを一意に表す文字列とデータの紐づけの永続化

第二引数としてトランザクションを一意に表す文字列である transactionKey がインタフェースに定義されています。TCC-Participant では、トランザクションを一意に表す文字列をトランザクションで管理する文字列として扱うため、トランザクションを一意に表す文字列と「(a)　データの仮登録」で登録したデータの紐づけが必要となります。

紐づけられた情報は executeRecover メソッドで利用します。executeRecover メソッドは TCC-Participant が再起動時に実行されるため、紐づけられた情報は TCC-Participant が再起動しても情報が失われないよう永続化する必要があります。

トランザクションを一意に表す文字列と(a)で登録したデータの紐づけには、後述の TccResponse クラスを使用してください。TccResponse クラスは java.io.Serializable インタフェースを継承しているため、このクラスのインスタンスそのものを外部 API などを使用して永続化してください。

(c)　TccResponse クラスの返却

インタフェースには戻り値として TccResponse クラスを返す必要があると定義されています。TccResponse クラスの詳細については、「7.4.4　TccResponse<T extends java.io.Serializable>クラス（TCC-Participant 限定）」を参照してください。

TccResponse クラスは string 型の transactionKey と T 型の data を持ちます。第二引数でトランザクションを一意に表す文字列が渡されるため、transactionKey はそのまま設定することを想定しています。data はジェネリクスクラスの T 型のため、任意の型のデータを格納できます。「(a)　データの仮登録」で仮登録したデータのオブジェクトを設定することを想定しています。

HMP-PCTO は、executeTry メソッドの戻り値をチェックし、次のどちらかに該当する場合は KFSG62017-E メッセージを出力し、java.lang.IllegalStateException を送出します。

- 戻り値が null である場合
- 戻り値の TccResponse クラスのインスタンスの transactionKey フィールドが null である場合

(3)　executeConfirm メソッドの実装

TCC-Participant では、次に示す機能が executeConfirm メソッドに実装されていることを想定して動作します。

- executeTry メソッドで登録した仮登録のデータの確定
- 永続化したトランザクションを一意に表す文字列とデータの紐づけ情報の削除

(a) executeTry メソッドで登録した仮登録のデータの確定

一般的に TCC パターンの Confirm フェーズでは、executeTry メソッドで仮登録したデータの確定処理を行います。DB の概念では Commit に該当する処理になります。executeTry メソッドで登録したデータの仮登録の状態を更新し、仮登録ではない状態にする必要があります。executeConfirm メソッドの引数には executeTry メソッドの戻り値の TccResponse クラスが渡されます。TccResponse クラスの data の内容を見て、確定処理を行うデータを特定することを想定しています。

(b) 永続化したトランザクションを一意に表す文字列とデータの紐づけ情報の削除

executeTry メソッドで永続化したトランザクションを一意に表す文字列と、データの紐づけ情報の削除を行う必要があります。executeRecover メソッドでは未決着状態のデータを返す必要があるため、executeConfirm メソッドで決着したデータに関してこの削除処理を行わないと、決着済みのデータを返してしまい、executeRecover メソッドの要件を満たすことができなくなります。

(4) executeCancel メソッドの実装

TCC-Participant では、次に示す機能が executeCancel メソッドに実装されていることを想定して動作します。

- executeTry メソッドで登録した仮登録のデータの削除
- 永続化したトランザクションを一意に表す文字列とデータの紐づけ情報の削除

(a) executeTry メソッドで登録した仮登録のデータの削除

一般的に TCC パターンの Cancel フェーズでは、executeTry メソッドで仮登録したデータの削除処理を行います。

DB の概念では Rollback に該当する処理になります。そのため、executeTry メソッドを実行する前の状態に戻っている必要があります。executeTry メソッドの実行状況によって、次のケースが考えられます。

- executeTry 完了後に executeCancel が実行されるケース
- executeTry 実行中に executeCancel が実行されるケース

executeTry 完了後に executeCancel が実行されるケース

executeCancel メソッドの引数には executeTry メソッドの戻り値の TccResponse クラスが渡されます。TccResponse クラスの data の内容を見て、削除処理を行うデータを特定することを想定しています。

executeTry 実行中に executeCancel が実行されるケース

TCC-Participant では executeTry メソッドの実行がタイムアウトした場合に executeTry メソッドの途中であっても executeCancel メソッドが実行されます。その場合に executeTry メソッドの実行を中止し、executeTry メソッド実行前の状態に戻す必要があります。この場合、executeCancel メソッドの引数に渡される TccResponse クラスには transactionKey だけが設定され、data は null の状態で渡されます。

このケースでは、transactionKey の情報だけで executeTry 実行前の状態に戻す必要があります。

(b) 永続化したトランザクションを一意に表す文字列とデータの紐づけ情報の削除

executeTry メソッドで永続化したトランザクションを一意に表す文字列と、データの紐づけ情報の削除を行う必要があります。executeRecover メソッドでは未決着状態のデータを返す必要があるため、executeCancel メソッドで削除したデータに関してこの削除処理を行わないと、削除済みのデータを返してしまい、executeRecover メソッドの要件を満たすことができなくなります。

(5) executeRecover メソッドの実装

TCC-Participant では、次に示す機能が executeRecover メソッドに実装されていることを想定して動作します。

- 未決着状態かつ自身の TccService が登録した永続化したデータの一覧を返す

(a) 未決着状態かつ自身の TccService が登録した永続化したデータの一覧を返す

executeRecover メソッドは戻り値として TccResponse の一覧を返す必要があります。その際に条件として、未決着状態、かつ、自身の TccService が登録したデータであることが必要です。executeTry メソッド、executeConfirm メソッド、executeCancel メソッドの実装で示したとおり、executeTry メソッドでトランザクションを一意に表す文字列とデータの紐づけの情報が永続化され、executeConfirm メソッド、executeCancel メソッド実行時に削除されます。これによって、永続化されている情報は未決着状態のデータだけとなります。この時点では自身の TccService が登録したデータの絞り込みがされていないため、executeRecover メソッド内で絞り込みを実施する必要があります。

なお、executeRecover メソッドの戻り値に含める TccResponse クラスのインスタンスは、executeTry メソッドのときにシリアライズして永続化した TccResponse クラスのインスタンスを executeRecover メソッド内でデシリアライズしたものである必要があります。このため、executeRecover メソッド内で TccResponse クラスのコンストラクタを呼び出して TccResponse クラスのインスタンスを生成することはできません。もし、executeRecover メソッド内で TccResponse クラスのコンストラクタを呼び出した場合、HMP-PCTO は KFSG62016-E メッセージを出力し、java.lang.IllegalStateException を送出します。

HMP-PCTO は、executeRecover メソッドの戻り値をチェックし、次のどれかに該当する場合は KFSG62018-E メッセージを出力し、java.lang.IllegalStateException を送出します。

- 戻り値が null である場合
- 戻り値の java.util.Set に null が含まれている場合
- 戻り値の java.util.Set に transactionKey フィールドが null である TccResponse クラスのインスタンスが含まれている場合

重要

ユーザが executeRecover メソッド実行時の前提を設けている場合（例えば、トランザクションを一意に表す文字列とデータの紐づけの永続化先として DB を使用しており、その前提として executeRecover メソッド実行時よりも前に DB の初期化を完了させておく必要があるなど）、TCC-Participant の Java アプリケーションプロセス起動開始前にその前提を満たしておく必要があります。

(6) getOperationId メソッドの実装

TCC-Participant では、次に示す機能が getOperationId メソッドに実装されていることを想定して動作します。

- TccService を一意に識別できる文字列を返す

(a) TccService を一意に識別できる文字列を返す

TCC-Participant では getOperationId メソッドの戻り値を利用して TccService を一意に識別しています。ほかの TccService と OperationId が重複した場合には TCC-Participant の動作に影響が出るため、ほかの TccService と重複しない一意に識別できる文字列を返す必要があります。

戻り値の文字列の長さは 1～256 バイトの範囲内である必要があります。

HMP-PCTO は、getOperationId メソッドの戻り値をチェックし、1 に該当する場合は KFSG62019-E メッセージ、2 に該当する場合は KFSG62020-E メッセージ、3 に該当する場合は KFSG62021-E メッセージを出力し、java.lang.IllegalStateException を送出します。

1. 戻り値が null である場合
2. 戻り値の文字列の長さが 1～256 バイトの範囲内ではない場合
3. 同一の TCC-Participant 内の他の TccService と同じ OperationId を返した場合

また、TCC-Participant が再起動した際は再起動前と同じ文字列を返す必要があります。

2.2.12 業務ロジックからの executeTry メソッドの呼び出し（TCC-Participant 限定）

業務ロジックからの executeTry メソッドの呼び出しの実装は、TCC-Participant のアプリケーションで実装してください。

TCC-Participant では Orchestrator から API が呼び出されたあと、executeTry メソッドが実行されることを想定し、executeTry メソッドが実行されたタイミングからトランザクション制御を開始します。API 呼び出し後の業務ロジックの中で executeTry メソッドの呼び出しを実施する必要があります。

なお、executeConfirm メソッド、executeCancel メソッド、executeRecover メソッド、getOperationId メソッドに関しては、TCC-Participant がトランザクション制御を行う過程で TCC-Participant から呼び出すため、業務ロジックの中で呼び出す必要はありません。

executeTry メソッドの呼び出しでの HMP-PCTO のエラー発生条件とその動作を次に示します。

- HMP-PCTO がトランザクションブランチ情報の取得に失敗した場合：

  HMP-PCTO は、KFSG62002-E のエラーメッセージを出力し、executeTry メソッドの呼び出し元の業務ロジックに対して、java.lang.IllegalStateException を送出します。
- @Tcc アノテーションの付与対象が TccService になっていない場合：

  HMP-PCTO は、KFSG62003-E のエラーメッセージを出力し、executeTry メソッドの呼び出し元の業務ロジックに対して、java.lang.IllegalStateException を送出します。
- 1 つのトランザクション内で、同一の OperationId を持つ TccService の executeTry メソッドを複数回実行した場合：

  HMP-PCTO は、KFSG62013-E のエラーメッセージを出力し、executeTry メソッドの呼び出し元の業務ロジックに対して、java.lang.IllegalStateException を送出します。

2.2.13　TP1-Bridge の使用（通常版かつ TP1-Bridge 限定）

TP1-Bridge を使用するアプリケーションの開発方法について説明します。

TP1-Bridge の機能を使用する処理は、Orchestrator および Entity-Service のアプリケーションで実装してください。

TP1-Bridge を使用する場合は、OpenTP1 の SPP に接続するためのサービスグループ名とサービス名、および OpenTP1 の SPP への要求電文となる JSON 形式のデータが必要です。

TP1-Bridge を使用するためには、org.springframework.web.client.RestTemplate クラスを使用して TP1-Bridge を呼び出します。

TP1-Bridge を呼び出す前提として、トランザクションを伝搬する処理の実装が必要です。TP1-Bridge を呼び出すために必要な HMP-PCTO のライブラリは、トランザクションの伝搬で必要なライブラリに含まれています。

(1)　コーディング例

サービスグループ名とサービス名は、TP1-Bridge に通信するための URL のパスに設定します。

次に示す形式で設定してください。

```
http://<TP1-Bridgeのホスト名>:<TP1-Bridgeのポート番号>/tp1bridge/<サービスグループ名>/<サービス名>
```

JSON データは、TP1-Bridge に通信する要求電文の Body に設定してください。また、要求電文は POST で送信してください。

```
String url = "http://xxxx:xxxx/tp1bridge/serviceg1/service1";     // TP1-BridgeのURLとサービスグループ名サービス名を用意

RestTemplate restTemplate = new RestTemplate();                   // RestTemplateのインスタンス作成
String jsonData = "{¥"Data1¥":¥"AAAA¥",¥"Data2¥":¥"BBBB¥"}";     // String型のJsonデータを用意
RequestEntity.BodyBuilder rb = RequestEntity.post(url);           // TP1-BridgeのURL設定
RequestEntity<?> request = rb.body(jsonData);                     // JsonデータをBodyに設定
ResponseEntity<?> response = restTemplate.exchange(request, String.class); // 通信実行
String responseBody = response.getBody();                         // 応答電文のJsonデータをBody部から取り出し
```

注

この例では、インタセプタの登録（トランザクションの伝搬）に関する記述を省略しています。

(2) エラーハンドリング

RestTemplate を使用した通信で例外が発生した場合は、RestTemplate が送出する例外、例外のメッセージ、およびステータスコードを参照してください。

HMP-PCTO が送出する statuscode については、取扱説明書「HMP-PCTO テクニカルガイド」の TP1-Acceptor の機能の HTTP ステータスマッピングに関する説明を参照してください。

(3) JSON の形式に関する注意事項

TP1-Bridge は、JSON 形式のデータをバイナリ形式に変換する際に、uCosminexus Service Platform の JSON-XML 変換 API を使用します。そのため、TP1-Bridge を使用する際の JSON 形式のデータは、uCosminexus Service Platform の JSON-XML 変換 API の仕様が定める形式としてください。

JSON 形式の詳細については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編」の、JSON-XML 変換で扱う JSON と標準仕様の JSON との差異に関する説明を参照してください。

2.2.14 ロギングライブラリの設定ファイルのカスタマイズ

アプリケーションでロギングライブラリ Logback を使用する場合、その設定は logback-spring.xml に定義します。このファイルを記載する場合、HMP-PCTO のメッセージログ機能に関するカスタマイズが必要です。また、HMP-ADIF と連携し、HMP-ADIF のロギングパターンでログを出力する場合も、このファイルのカスタマイズが必要です。詳細については、「(3) メッセージログ機能が参照するファイル」を参照してください。

2.2.15　Quartz スケジューラ機能の利用

アプリケーションでジョブスケジューリングライブラリ Quartz を使用する場合、スケジューラはデフォルトスケジューラを使用するか、または専用のスケジューラを作成して使用してください。

また、HMP-PCTO で定義している Quartz スケジューラ用の設定ファイル「hmppcto_quartz.properties」と同名のファイルは作成、または使用しないでください。

2.3　SQL-Participant のビルド（SQL-Participant 限定）

HMP-PCTO ライブラリを使用した SQL-Participant のビルドについて説明します。

Java アプリケーションの開発環境のビルド自動化システムには Gradle を使用する前提で説明します。説明中の Gradle の構文例は、Gradle7.3.3 を使用した場合を例に記述しています。

2.3.1　作業項目の一覧（SQL-Participant 限定）

SQL-Participant のビルドの作業項目の一覧を次の表に示します。

表 2-6　SQL-Participant のビルドの作業項目の一覧

分類	作業項目	備考	参照先
共通	事前準備（通常版かつ SQL-Participant 限定）	通常版の場合で、SQL-Participant を使用するときだけ実施します。	2.3.2　事前準備（通常版かつ SQL-Participant 限定）
	ライブラリのセットアップ（通常版かつ SQL-Participant 限定）	通常版の場合で、SQL-Participant を使用するときだけ実施します。	2.3.3　ライブラリのセットアップ（通常版かつ SQL-Participant 限定）
SQL-Participant のビルド	JAR ファイルのビルド（SQL-Participant 限定）	SQL-Participant を使用する場合だけ実施します。	2.3.4　JAR ファイルのビルド（SQL-Participant 限定）

2.3.2　事前準備（通常版かつ SQL-Participant 限定）

HMP-PCTO ライブラリを使用したアプリケーションの開発は、Gradle（バージョン 7.3.3 以上）の利用を前提としています。これらのソフトウェアを開発環境にインストールしてください。

詳細については、「3.2.2　事前準備（通常版限定）」を参照してください。

2.3.3　ライブラリのセットアップ（通常版かつ SQL-Participant 限定）

HMP-PCTO のライブラリ、および HMP-PCTO が依存するライブラリのセットアップについては、「3.2.3　ライブラリのセットアップ（通常版限定）」を参照してください。

2.3.4　JAR ファイルのビルド（SQL-Participant 限定）

「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル（sql-participant-*V.R.S*※.jar）」は、HMP-PCTO の提供物を使用してビルドします。

HMP-PCTO が提供する「SQL-Participant 用 Java プロジェクトディレクトリ」（java-projects/sql-participant）で、次に示す Gradle のコマンドを実行して JAR ファイルをビルドしてください。

```
cd java-projects/sql-participant
gradle build
```

ビルドに成功すると、次に示すパスに「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル（sql-participant-*V.R.S*※.jar）」が生成されます。

```
java-projects/sql-participant/build/lib/sql-participant-V.R.S※.jar
```

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

2.4　コンテナイメージ作成

ここでは、ユーザ責務のコンテナイメージ作成方法について説明します。

コンテナイメージビルド環境のコンテナエンジンはユーザ任意のものを使用してください。説明中のコンテナエンジンのコマンド例は、Docker を使用した場合を例に記述しています。コンテナエンジンの各コマンドの詳細については、使用するコンテナエンジンのマニュアルなどを参照してください。

2.4.1　作業項目の一覧

コンテナイメージ作成の作業項目の一覧を次の表に示します。

表 2-7　コンテナイメージ作成の作業項目の一覧

作業項目	備考	参照先
事前準備	必ず実施します。	2.4.2　事前準備
uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成（通常版限定）	通常版の場合で、uCosminexus Application Runtime with Java for Spring Boot を導入するときだけ作成します。	2.4.3　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成（通常版限定）
JDK コンテナベースイメージの作成	uCosminexus Application Runtime with Java for Spring Boot を導入しない場合だけ作成します。	2.4.4　JDK コンテナベースイメージの作成
Orchestrator コンテナイメージの作成	必ず作成します。	2.4.5　Orchestrator コンテナイメージの作成
Entity-Service コンテナイメージの作成	サービス間を中継するサービスとしての Entity-Service がある場合だけ作成します。 詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のトランザクションのモデル（その 2）に関する説明を参照してください。	2.4.6　Entity-Service コンテナイメージの作成
SQL-Participant コンテナイメージの作成（SQL-Participant 限定）	SQL-Participant を使用する場合だけ作成します。	2.4.7　SQL-Participant コンテナイメージの作成（SQL-Participant 限定）
Entity-Module/TCC-Participant コンテナイメージの作成（TCC-Participant 限定）	TCC-Participant を使用する場合だけ作成します。	2.4.8　Entity-Module/TCC-Participant コンテナイメージの作成（TCC-Participant 限定）

2.4.2　事前準備

コンテナイメージを登録するためのコンテナレジストリ、およびコンテナイメージをビルドするための Docker などのコンテナエンジンをインストールしたコンテナイメージビルド環境を用意してください。

詳細については、「3.4.2　事前準備（通常版限定）」を参照してください。

2.4.3　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成（通常版限定）

このコンテナベースイメージは、通常版のユーザ責務のコンテナイメージで uCosminexus Application Runtime with Java for Spring Boot を導入する場合だけ作成します。

uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージは、uCosminexus Application Runtime with Java for Spring Boot を導入するユーザ責務のコンテナイメージのベースイメージとして使用します。uCosminexus Application Runtime with Java for Spring Boot のインストール部分を共通のベースイメージとして作成することで、各種コンテナイメージの Containerfile の記述量の軽減、およびビルドした各種コンテナイメージのサイズを低減できるメリットがあります。

uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成方法について説明します。

(1)　Containerfile の作成

「5.3.1　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（通常版限定）」に記載されている要件を満たすように、uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージをビルドするための Containerfile を作成してください。

Containerfile の記述例を次に示します。

```
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860 AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        ncompress ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN     …3
COPY X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HL0T ./
RUN mkdir patch     …4
WORKDIR patch
COPY PACK_TAR.Z .
RUN tar xzvf PACK_TAR.Z && ./UPDATE -f; export RET=$?; if [ ${RET} -ne 0 -a ${RET} -ne 1 ];
then exit ${RET}; fi

FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860     …5
COPY --from=builder /opt /opt     …6
COPY --from=builder /etc/.java /etc/.java     …7
```

```
RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        dnf ¥
    && dnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        --allowerasing ¥
        curl-7.61.1 ¥
        jq-1.6 ¥
        <uCosminexus Application Runtime with Java for Spring Bootの前提RPMパッケージ>
…8
         :
    && dnf clean all ¥
    && rm -rf /var/cache/dnf/*
ENV PATH=/opt/Cosminexus/jdk/bin:$PATH     …9
USER 1001:0     …10
WORKDIR /deployments     …11
```

記述例の番号は、説明の番号と対応しています。

1. マルチステージビルド用のコンテナベースイメージとして UBI 8 の任意のベースイメージを指定します。この例では、「Red Hat Universal Base Image 8 Minimal 8.8」を使用しています。
2. uCosminexus Application Runtime with Java for Spring Boot のインストーラの前提 RPM パッケージをインストールします。
3. uCosminexus Application Runtime with Java for Spring Boot 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（「-c」オプションに「HL0T」（uCosminexus Application Runtime with Java for Spring Boot）を指定して実行してください）。
4. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot の修正パッチを適用する場合だけ記述します。

   uCosminexus Application Runtime with Java for Spring Boot 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」オプションを指定して実行してください）。
5. コンテナベースイメージとして UBI 8 の任意のベースイメージを指定します。この例では、「Red Hat Universal Base Image 8 Minimal 8.8」を使用しています。
6. マルチステージビルド用のコンテナイメージから uCosminexus Application Runtime with Java for Spring Boot のインストール先ディレクトリ「/opt」をコピーします。これによって、このコンテナイメージに uCosminexus Application Runtime with Java for Spring Boot をインストールした状態となります。
7. HMP-PCTO の前提 RPM パッケージをインストールします。
8. uCosminexus Application Runtime with Java for Spring Boot の前提 RPM パッケージをインストールします。

オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot のリリースノートを参照し、uCosminexus Application Runtime with Java for Spring Boot に必要な前提 RPM パッケージを記述してください。

9. 環境変数 PATH の先頭に、日立 JDK のコマンド群がインストールされているディレクトリを追加します。
10. 任意のコンテナ実行ユーザ・グループを指定します。
11. アプリケーション配置用ディレクトリを作成します。

ヒント

Red Hat OpenShift Container Platform で実行する場合のファイルのパーミッション設定

Red Hat OpenShift Container Platform には、SCC（Security Context Constraints）と呼ばれる、Pod のパーミッションを制御する機能があります。この SCC のデフォルトの設定である「restricted」では、Pod の Kubernetes マニフェストで runAsUser フィールドによるコンテナ起動ユーザが未設定の場合、コンテナイメージビルド時に設定したユーザ・グループを使用しないで、一定範囲内のランダムな UID（非 root）と GID「0」（root）でコンテナを実行します。

このため、Red Hat OpenShift Container Platform の SCC の restricted 設定で動作可能なコンテナイメージを作成する際は、配置するファイルのグループに「0」（root）を設定し、パーミッションにはオーナーの権限に加えてグループの権限も付与することを推奨します。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 2-8 Containerfile と資材の配置構成

ファイル／ディレクトリ構成		作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/		ユーザ	－
	Containerfile	ユーザ	－
	X64LIN (uCosminexus Application Runtime with Java for Spring Boot のインストーラ)	日立	uCosminexus Application Runtime with Java for Spring Boot のインストール CD-ROM に格納されている X64LIN ディレクトリをコピーしてください。
	PACK_TAR.Z (uCosminexus Application Runtime with Java for Spring Boot の修正パッチ)	日立	uCosminexus Application Runtime with Java for Spring Boot の修正パッチを適用する場合だけ格納してください。

（凡例）－：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]ucars:<uCosminexus Application Runtime with Java for Spring Bootバージョン(V.R.S※)>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/ucars:V.R.S※ .
```

この例では、uCosminexus Application Runtime with Java for Spring Boot バージョンを「*V.R.S*※」としています。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

2.4.4 JDK コンテナベースイメージの作成

このコンテナベースイメージは、トライアル版の場合、または通常版のユーザ責務のコンテナイメージで uCosminexus Application Runtime with Java for Spring Boot を導入しない場合だけ作成します。

JDK コンテナベースイメージは、uCosminexus Application Runtime with Java for Spring Boot を導入しないユーザ責務のコンテナイメージのベースイメージとして使用します。JDK のインストール部分を共通のベースイメージとして作成することで、各種コンテナイメージの Containerfile の記述量の軽減、およびビルドした各種コンテナイメージのサイズを低減できるメリットがあります。

JDK コンテナベースイメージの作成方法について説明します。

(1) Containerfile の作成

「5.3.2 JDK コンテナベースイメージ」に記載されている要件を満たすように、JDK コンテナベースイメージをビルドするための Containerfile を作成してください。

Containerfile の記述例を次に示します。

■通常版の場合

```
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        java-1.8.0-openjdk-headless ¥
        curl-7.61.1 ¥
        jq-1.6 ¥
    && microdnf clean all ¥
    && rm -rf /var/cache/dnf/*     …3
USER 1001:0     …4
WORKDIR /deployments
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. コンテナベースイメージとして UBI 8 の任意のベースイメージを指定します。この例では、「Red Hat Universal Base Image 8 Minimal 8.8」を使用しています。
2. 前提 RPM パッケージをインストールします。この例では、Java SE 8 準拠の JDK として「OpenJDK 8」を使用しています。
3. アプリケーション配置用ディレクトリを作成します。
4. 任意のコンテナ実行ユーザ・グループを指定します。

■トライアル版の場合

```
FROM quay.io/almalinuxorg/8-minimal:8.8     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        java-1.8.0-openjdk-headless ¥
        curl-7.61.1 ¥
        jq-1.6 ¥
    && microdnf clean all ¥     …3
    && rm -rf /var/cache/dnf/*     …4
USER 1001:0
WORKDIR /deployments
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. コンテナベースイメージとして Almalinux 8 Minimal 8.8 を指定します。
2. 前提 RPM パッケージをインストールします。この例では、Java SE 8 準拠の JDK として「OpenJDK 8」を使用しています。
3. アプリケーション配置用ディレクトリを作成します。
4. 任意のコンテナ実行ユーザ・グループを指定します。

ヒント

Red Hat OpenShift Container Platform で実行する場合のファイルのパーミッション設定

Red Hat OpenShift Container Platform には、SCC（Security Context Constraints）と呼ばれる、Pod のパーミッションを制御する機能があります。この SCC のデフォルトの設定である「restricted」では、Pod の Kubernetes マニフェストで runAsUser フィールドによるコンテナ起動ユーザが未設定の場合、コンテナイメージビルド時に設定したユーザ・グループを使用しないで、一定範囲内のランダムな UID（非 root）と GID「0」（root）でコンテナを実行します。

このため、Red Hat OpenShift Container Platform の SCC の restricted 設定で動作可能なコンテナイメージを作成する際は、配置するファイルのグループに「0」（root）を設定し、パーミッションにはオーナーの権限に加えてグループの権限も付与することを推奨します。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 2-9 Containerfile と資材の配置構成

ファイル／ディレクトリ構成		作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/		ユーザ	－
	Containerfile	ユーザ	－

（凡例）－：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

JDK コンテナベースイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]<JDKコンテナベースイメージのユーザ任意のイメージ名>:<ユーザ任意のタグ名>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/jdk:V.R.S※ .
```

この例では、JDK コンテナベースイメージのイメージ名・タグ名を「jdk:*V.R.S*※」としています。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4)　コンテナイメージの登録

ビルドした JDK コンテナベースイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

2.4.5　Orchestrator コンテナイメージの作成

Orchestrator コンテナイメージの作成方法について説明します。

(1)　Containerfile の作成

「5.3.3　Orchestrator コンテナイメージ」に記載されている要件を満たすように、Orchestrator コンテナイメージをビルドするための Containerfile を作成してください。

Containerfile の記述例を次に示します。

■通常版・uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合

```
FROM myregistry.example.com/myproject/ucars:V.R.S※      …1
COPY --chown=1001:0 --chmod=0440 ./orchestrator-V.R.S※.jar /deployments/      …2
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/orchest
rator-V.R.S※.jar      …3
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. コンテナベースイメージとして「2.4.3　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成（通常版限定）」で作成した「uCosminexus Application Runtime

with Java for Spring Boot コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/jdk:*V.R.S*※」を指定しています。

2. 「ユーザの Orchestrator アプリケーションの JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「ユーザの Orchestrator アプリケーションの JAR ファイル」の名前を「orchestrator-*V.R.S*※.jar」としています。
3. コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト(starter.sh)」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Orchestrator アプリケーションの JAR ファイル」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

■通常版・uCosminexus Application Runtime with Java for Spring Boot 導入なし、トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:V.R.S※      …1
COPY --chown=1001:0 --chmod=0440 ./orchestrator-V.R.S※.jar /deployments/      …2
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/orchestrator-V.R.S※.jar      …3
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. コンテナベースイメージとして「2.4.4　JDK コンテナベースイメージの作成」で作成した「JDK コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/jdk:*V.R.S*※」を指定しています。
2. 「ユーザの Orchestrator アプリケーションの JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「ユーザの Orchestrator アプリケーションの JAR ファイル」の名前を「orchestrator-*V.R.S*※.jar」としています。
3. コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Orchestrator アプリケーションの JAR ファイル」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

ヒント

Red Hat OpenShift Container Platform で実行する場合のファイルのパーミッション設定

Red Hat OpenShift Container Platform には、SCC（Security Context Constraints）と呼ばれる、Pod のパーミッションを制御する機能があります。この SCC のデフォルトの設定である「restricted」では、Pod の Kubernetes マニフェストで runAsUser フィールドによるコンテナ起動ユーザが未設定の場合、コンテナイメージビルド時に設定したユーザ・グループを

使用しないで、一定範囲内のランダムな UID（非 root）と GID「0」（root）でコンテナを実行します。

このため、Red Hat OpenShift Container Platform の SCC の restricted 設定で動作可能なコンテナイメージを作成する際は、配置するファイルのグループに「0」（root）を設定し、パーミッションにはオーナーの権限に加えてグループの権限も付与することを推奨します。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 2-10 Containerfile と資材の配置構成

ファイル／ディレクトリ構成		作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/		ユーザ	−
	Containerfile	ユーザ	−
	<ユーザの Orchestrator アプリケーションの JAR ファイル>	ユーザ	−

（凡例）−：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

Orchestrator コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]<Orchestratorコンテナイメージのユーザ任意のイメージ名>:<ユーザ任意のタグ名>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/orchestrator:V.R.S※ .
```

この例では、Orchestrator コンテナイメージのイメージ名・タグ名を「orchestrator:*V.R.S*※」としています。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4)　コンテナイメージの登録

ビルドした Orchestrator コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

2.4.6　Entity-Service コンテナイメージの作成

Entity-Service コンテナイメージの作成方法について説明します。

サービス間を中継するサービスとしての Entity-Service がある場合だけ作成します。

詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のトランザクションのモデル（その 2）に関する説明を参照してください。

ユーザ責務の Kubernetes アプリケーションの Entity-Service コンテナの数だけ、このコンテナイメージの Containerfile を作成しビルドしてください。

(1)　Containerfile の作成

「5.3.4　Entity-Service コンテナイメージ」に記載されている要件を満たすように、Entity-Service コンテナイメージをビルドするための Containerfile を作成してください。

Containerfile の記述例を次に示します。

■通常版・uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合

```
FROM myregistry.example.com/myproject/ucars:V.R.S※      …1
COPY --chown=1001:0 --chmod=0440 ./entity-service-V.R.S※.jar /deployments/      …2
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/entity-service-V.R.S※.jar      …3
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. コンテナベースイメージとして「2.4.3　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成（通常版限定）」で作成した「uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/jdk:*V.R.S*※」を指定しています。

2.「ユーザの Entity-Service アプリケーションの JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「ユーザの Entity-Service アプリケーションの JAR ファイル」の名前を「entity-service-*V.R.S*※.jar」としています。

3. コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト(starter.sh)」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Service アプリケーションの JAR ファイル」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

■通常版・uCosminexus Application Runtime with Java for Spring Boot 導入なし、トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:V.R.S※      …1
COPY --chown=1001:0 --chmod=0440 ./entity-service-V.R.S※.jar /deployments/      …2
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/entity-service-V.R.S※.jar      …3
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. コンテナベースイメージとして「2.4.4　JDK コンテナベースイメージの作成」で作成した「JDK コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/jdk:*V.R.S*※」を指定しています。

2.「ユーザの Entity-Service アプリケーションの JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「ユーザの Entity-Service アプリケーションの JAR ファイル」の名前を「entity-service-*V.R.S*※.jar」としています。

3. コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Service アプリケーションの JAR ファイル」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

ヒント

Red Hat OpenShift Container Platform で実行する場合のファイルのパーミッション設定

Red Hat OpenShift Container Platform には、SCC（Security Context Constraints）と呼ばれる、Pod のパーミッションを制御する機能があります。この SCC のデフォルトの設定である「restricted」では、Pod の Kubernetes マニフェストで runAsUser フィールドによるコンテナ起動ユーザが未設定の場合、コンテナイメージビルド時に設定したユーザ・グループを使用しないで、一定範囲内のランダムな UID（非 root）と GID「0」（root）でコンテナを実行します。

このため、Red Hat OpenShift Container Platform の SCC の restricted 設定で動作可能なコンテナイメージを作成する際は、配置するファイルのグループに「0」（root）を設定し、パーミッションにはオーナーの権限に加えてグループの権限も付与することを推奨します。

(2) Containerfile と資材の配置

次の表に示すとおり Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 2-11 Containerfile と資材の配置構成

ファイル／ディレクトリ構成		作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/		ユーザ	−
	Containerfile	ユーザ	−
	<ユーザの Entity-Service アプリケーションの JAR ファイル>	ユーザ	−

（凡例）−：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

Entity-Service コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]<Entity-Serviceコンテナイメージのユーザ任意のイメージ名>:<ユーザ任意のタグ名>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/entity-service:V.R.S※ .
```

この例では、Entity-Service コンテナイメージのイメージ名・タグ名を「entity-service:*V.R.S*※」としています。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした Entity-Service コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

2.4.7　SQL-Participant コンテナイメージの作成（SQL-Participant 限定）

SQL-Participant コンテナイメージの作成方法について説明します。

(1)　Containerfile の作成

「5.3.5　SQL-Participant コンテナイメージ」に記載されている要件を満たすように、SQL-Participant コンテナイメージをビルドするための Containerfile を作成してください。

Containerfile の記述例を次に示します。

■通常版・uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合

```
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        ncompress ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN     …3
COPY X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HC13 ./
RUN mkdir patch     …4
WORKDIR patch
COPY PACK_TAR.Z .
RUN tar xzvf PACK_TAR.Z && ./UPDATE -f; export RET=$?; if [ ${RET} -ne 0 -a ${RET} -ne 1 ];
then exit ${RET}; fi

FROM myregistry.example.com/myproject/ucars:V.R.S※     …5
RUN mkdir /deployments/eads-lib ¥
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log     …6
ENV LOADER_PATH=/deployments/eads-lib     …7
COPY --from=builder --chown=1001:0 --chmod=0440 /opt/hitachi/xeads/javaclient/lib/ /deployme
nts/eads-lib     …8
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/     …9
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/     …10
COPY --chown=1001:0 --chmod=0440 ./sql-participant-V.R.S※.jar /deployments/     …11
COPY --chown=1001:0 --chmod=0440 ./pdjdbc4.jar /deployments/     …12
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/     …13
```

```
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/sql-par
ticipant-V.R.S※.jar      …14
```

記述例の番号は、説明の番号と対応しています。

1. マルチステージビルド用のコンテナベースイメージとして UBI 8 の任意のベースイメージを指定します。この例では、「Red Hat Universal Base Image 8 Minimal 8.8」を使用しています。
2. Hitachi Elastic Application Data Store Client for Java のインストーラの前提 RPM パッケージをインストールします。
3. Hitachi Elastic Application Data Store Client for Java 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（「-c」オプションに「HC13」(Hitachi Elastic Application Data Store Client for Java)を指定して実行してください)。
4. オレンジ色の背景で示す行は、Hitachi Elastic Application Data Store Client for Java の修正パッチを適用する場合だけ記述します。

   Hitachi Elastic Application Data Store Client for Java 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」オプションを指定して実行してください)。
5. コンテナベースイメージとして「2.4.3　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成（通常版限定)」で作成した「uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/ucars:*V.R.S*※」を指定しています。
6. Hitachi Elastic Application Data Store Client for Java のライブラリをコピーするディレクトリを作成します。ディレクトリ名は任意ですが、この例では/deployments/eads-lib としています。

   Hitachi Elastic Application Data Store Client for Java のログを出力するディレクトリを作成します。ディレクトリ名は/deployments/eads-log としてください。
7. 環境変数「LOADER_PATH」に Hitachi Elastic Application Data Store Client for Java のライブラリのディレクトリを設定します。6.で作成したディレクトリ名を指定します。
8. マルチステージビルド用のコンテナイメージから Hitachi Elastic Application Data Store Client for Java のインストール先ディレクトリ「/opt/hitachi/xeads/javaclient/lib/」をコピーし、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。コピー先は 6.で作成したディレクトリ名を指定します。
9. 「HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh)」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
10. 「HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh)」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
11. 「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル（sql-participant-*V.R.S*.jar)」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル（sql-participant-V.R.S.jar)」の名前を「sql-participant-*V.R.S*※.jar」としています。

12. 「JDBC Driver の JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「JDBC Driver の JAR ファイル」の名前を「pdjdbc4.jar」としています。ここで「JDBC Driver の JAR ファイル」を格納したパスが、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.path パラメタに設定する値となります。
13. オレンジ色の背景で示す行は、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.db-type パラメタに設定する値が「HiRDB」の場合だけ記述します。HiRDB への接続に必要な HiRDB のクライアント環境定義ファイル（HiRDB.ini）を配置し、パーミッションにコンテナを実行するユーザおよびグループそれぞれの読み取り権限を付与します。
14. コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト（starter.sh）」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル（sql-participant-*V.R.S*※.jar）」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

■トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:V.R.S※      …1
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/      …3
COPY --chown=1001:0 --chmod=0440 ./sql-participant-V.R.S※.jar /deployments/      …4
COPY --chown=1001:0 --chmod=0440 ./postgresql-42.2.22.jar /deployments/      …5
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/      …6
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/sql-participant-V.R.S※.jar      …7
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. コンテナベースイメージとして「2.4.4　JDK コンテナベースイメージの作成」で作成した「JDK コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/jdk:*V.R.S*※」を指定しています。
2. 「HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh）」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
3. 「HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh）」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
4. 「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル（sql-participant-*V.R.S*※.jar）」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル（sql-participant-*V.R.S*※.jar）」の名前を「sql-participant-*V.R.S*※.jar」としています。
5. 「JDBC Driver の JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「JDBC Driver の JAR ファイル」の名前を

「postgresql-42.2.22.jar」としています。ここで「JDBC Driver の JAR ファイル」を格納したパスが、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.path パラメタに設定する値となります。

6. SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.db-type パラメタに設定する値が「HiRDB」の場合、HiRDB への接続に必要な HiRDB のクライアント環境定義ファイル（HiRDB.ini）を配置し、パーミッションにコンテナ実行ユーザおよびグループそれぞれの読み取り権限を付与します。
7. コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル(sql-participant-*V.R.S*※.jar)」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

ヒント

Red Hat OpenShift Container Platform で実行する場合のファイルのパーミッション設定

Red Hat OpenShift Container Platform には、SCC（Security Context Constraints）と呼ばれる、Pod のパーミッションを制御する機能があります。この SCC のデフォルトの設定である「restricted」では、Pod の Kubernetes マニフェストで runAsUser フィールドによるコンテナ起動ユーザが未設定の場合、コンテナイメージビルド時に設定したユーザ・グループを使用しないで、一定範囲内のランダムな UID（非 root）と GID「0」（root）でコンテナを実行します。

このため、Red Hat OpenShift Container Platform の SCC の restricted 設定で動作可能なコンテナイメージを作成する際は、配置するファイルのグループに「0」（root）を設定し、パーミッションにはオーナーの権限に加えてグループの権限も付与することを推奨します。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 2-12 Containerfile と資材の配置構成

ファイル／ディレクトリ構成	作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/	ユーザ	－
Containerfile	ユーザ	－
sql-participant-V.R.S.jar (HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル)	ユーザ	「2.3 SQL-Participant のビルド（SQL-Participant 限定）」で作成したもの
<JDBC Driver の JAR ファイル>	ユーザ	－

ファイル／ディレクトリ構成	作成・提供元	備考
health_check.sh (HMP-PCTO のトランザクションヘルスチェックスクリプト)	日立	－
wait_transactions.sh (HMP-PCTO の未決着トランザクション待機スクリプト)	日立	－
X64LIN (Hitachi Elastic Application Data Store Client for Java のインストーラ)	日立	通常版の場合だけ、この製品に同梱している Hitachi Elastic Application Data Store Client for Java のインストーラの X64LIN ディレクトリをコピーしてください。
PACK_TAR.Z (Hitachi Elastic Application Data Store Client for Java の修正パッチ)	日立	通常版の場合で、この製品に Hitachi Elastic Application Data Store Client for Java の修正パッチが同梱されている場合だけ格納してください。同梱されていない場合は格納不要です。

（凡例）－：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

SQL-Participant コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]<SQL-Participant コンテナイメージのユーザ任意のイメージ名>:<ユーザ任意のタグ名>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/sql-participant:V.R.S※ .
```

この例では、SQL-Participant コンテナイメージのイメージ名・タグ名を「sql-participant:*V.R.S*※」としています。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした SQL-Participant コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

2.4.8 Entity-Module/TCC-Participant コンテナイメージの作成（TCC-Participant 限定）

Entity-Module/TCC-Participant コンテナイメージの作成方法について説明します。

また、ユーザ責務の Kubernetes アプリケーションの Entity-Module/TCC-Participant コンテナの数だけ、このコンテナイメージの Containerfile を作成しビルドしてください。

(1) Containerfile の作成

「5.3.6 Entity-Module/TCC-Participant コンテナイメージ」に記載されている要件を満たすように、Entity-Module/TCC-Participant コンテナイメージをビルドするための Containerfile を作成してください。

Containerfile の記述例を次に示します。

■通常版・uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合

```
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        ncompress ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN     …3
COPY X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HC13 ./
RUN mkdir patch     …4
WORKDIR patch
COPY PACK_TAR.Z .
RUN tar xzvf PACK_TAR.Z && ./UPDATE -f; export RET=$?; if [ ${RET} -ne 0 -a ${RET} -ne 1 ];
then exit ${RET}; fi

FROM myregistry.example.com/myproject/ucars:V.R.S※     …5
RUN mkdir /deployments/eads-lib ¥
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log     …6
ENV LOADER_PATH=/deployments/eads-lib     …7
COPY --from=builder --chown=1001:0 --chmod=0440 /opt/hitachi/xeads/javaclient/lib/ /deployme
```

```
nts/eads-lib     …8
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/     …9
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/     …10
COPY --chown=1001:0 --chmod=0440 ./entity-module-tcc-participant-V.R.S※.jar /deployments/
   …11
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/entity-
module-tcc-participant-V.R.S※.jar     …12
```

記述例の番号は、説明の番号と対応しています。

1. マルチステージビルド用のコンテナベースイメージとして UBI 8 の任意のベースイメージを指定します。この例では、「Red Hat Universal Base Image 8 Minimal 8.8」を使用しています。
2. Hitachi Elastic Application Data Store Client for Java のインストーラの前提 RPM パッケージをインストールします。
3. Hitachi Elastic Application Data Store Client for Java 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（「-c」オプションに「HC13」(Hitachi Elastic Application Data Store Client for Java)を指定して実行してください)。
4. オレンジ色の背景で示す行は、Hitachi Elastic Application Data Store Client for Java の修正パッチを適用する場合だけ記述します。

   Hitachi Elastic Application Data Store Client for Java 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」オプションを指定して実行してください)。
5. コンテナベースイメージとして「2.4.3　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの作成（通常版限定)」で作成した「uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/ucars:*V.R.S*※」を指定しています。
6. Hitachi Elastic Application Data Store Client for Java のライブラリをコピーするディレクトリを作成します。ディレクトリ名は任意ですが、この例では/deployments/eads-lib としています。

   Hitachi Elastic Application Data Store Client for Java のログを出力するディレクトリを作成します。ディレクトリ名は/deployments/eads-log としてください。
7. 環境変数「LOADER_PATH」に Hitachi Elastic Application Data Store Client for Java のライブラリのディレクトリを設定します。6.で作成したディレクトリ名を指定します。
8. マルチステージビルド用のコンテナイメージから Hitachi Elastic Application Data Store Client for Java のインストール先ディレクトリ「/opt/hitachi/xeads/javaclient/lib/」をコピーし、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。コピー先は 6.で作成したディレクトリ名を指定します。
9. 「HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh)」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
10. 「HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh)」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
11. 「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「ユーザ

の Entity-Module/TCC-Participant アプリケーションの JAR ファイル」の名前を「entity-module-tcc-participant-*V.R.S*※.jar」としています。

12. コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト(starter.sh)」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

■通常版・uCosminexus Application Runtime with Java for Spring Boot 導入なしの場合

```
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        ncompress ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN     …3
COPY X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HC13 ./
RUN mkdir patch     …4
WORKDIR patch
COPY PACK_TAR.Z .
RUN tar xzvf PACK_TAR.Z && ./UPDATE -f; export RET=$?; if [ ${RET} -ne 0 -a ${RET} -ne 1 ];
then exit ${RET}; fi

FROM myregistry.example.com/myproject/jdk:V.R.S※     …5
RUN mkdir /deployments/eads-lib ¥
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log     …6
ENV LOADER_PATH=/deployments/eads-lib     …7
COPY --from=builder --chown=1001:0 --chmod=0440 /opt/hitachi/xeads/javaclient/lib/ /deployme
nts/eads-lib     …8
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/     …9
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/     …10
COPY --chown=1001:0 --chmod=0440 ./entity-module-tcc-participant-V.R.S※.jar /deployments/
   …11
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/entity-module-tcc-participant-V.R.S※.jar
     …12
```

記述例の番号は、説明の番号と対応しています。

1. マルチステージビルド用のコンテナベースイメージとして UBI 8 の任意のベースイメージを指定します。この例では、「Red Hat Universal Base Image 8 Minimal 8.8」を使用しています。

2. Hitachi Elastic Application Data Store Client for Java のインストーラの前提 RPM パッケージをインストールします。
3. Hitachi Elastic Application Data Store Client for Java 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（「-c」オプションに「HC13」(Hitachi Elastic Application Data Store Client for Java)を指定して実行してください）
4. オレンジ色の背景で示す行は、Hitachi Elastic Application Data Store Client for Java の修正パッチを適用する場合だけ記述します。

   Hitachi Elastic Application Data Store Client for Java 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」オプションを指定して実行してください）。
5. コンテナベースイメージとして「2.4.4　JDK コンテナベースイメージの作成」で作成した「JDK コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/jdk:*V.R.S*※」を指定しています。
6. Hitachi Elastic Application Data Store Client for Java のライブラリをコピーするディレクトリを作成します。ディレクトリ名は任意ですが、この例では/deployments/eads-lib としています。

   Hitachi Elastic Application Data Store Client for Java のログを出力するディレクトリを作成します。ディレクトリ名は/deployments/eads-log としてください。
7. 環境変数「LOADER_PATH」に Hitachi Elastic Application Data Store Client for Java のライブラリのディレクトリを設定します。6.で作成したディレクトリ名を指定します。
8. マルチステージビルド用のコンテナイメージから Hitachi Elastic Application Data Store Client for Java のインストール先ディレクトリ「/opt/hitachi/xeads/javaclient/lib/ 」をコピーし、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。コピー先は 6.で作成したディレクトリ名を指定します。
9. 「HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh)」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
10. 「HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh)」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
11. 「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」の名前を「entity-module-tcc-participant-*V.R.S*※.jar」としています。
12. コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

■トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:V.R.S※      …1
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/      …3
COPY --chown=1001:0 --chmod=0440 ./entity-module-tcc-participant-V.R.S※.jar /deployments/
   …4
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/entity-module-tcc-participant-V.R.S※.jar
     …5
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. コンテナベースイメージとして「2.4.4　JDK コンテナベースイメージの作成」で作成した「JDK コンテナベースイメージ」を指定します。この例では、「myregistry.example.com/myproject/jdk:*V.R.S*※」を指定しています。
2. 「HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh）」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
3. 「HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh）」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの実行権限を付与します。
4. 「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」を配置し、パーミッションにコンテナ実行ユーザとグループそれぞれの読み取り権限を付与します。この例では、「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」の名前を「entity-module-tcc-participant-*V.R.S*※.jar」としています。
5. コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」を指定します。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

ヒント

Red Hat OpenShift Container Platform で実行する場合のファイルのパーミッション設定

Red Hat OpenShift Container Platform には、SCC（Security Context Constraints）と呼ばれる、Pod のパーミッションを制御する機能があります。この SCC のデフォルトの設定である「restricted」では、Pod の Kubernetes マニフェストで runAsUser フィールドによるコンテナ起動ユーザが未設定の場合、コンテナイメージビルド時に設定したユーザ・グループを使用しないで、一定範囲内のランダムな UID（非 root）と GID「0」（root）でコンテナを実行します。

このため、Red Hat OpenShift Container Platform の SCC の restricted 設定で動作可能なコンテナイメージを作成する際は、配置するファイルのグループに「0」（root）を設定し、パーミッションにはオーナーの権限に加えてグループの権限も付与することを推奨します。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 2-13 Containerfile と資材の配置構成

ファイル／ディレクトリ構成	作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/	ユーザ	−
Containerfile	ユーザ	−
<ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル>	ユーザ	−
health_check.sh (HMP-PCTO のトランザクションヘルスチェックスクリプト)	日立	−
wait_transactions.sh (HMP-PCTO の未決着トランザクション待機スクリプト)	日立	−
X64LIN (Hitachi Elastic Application Data Store Client for Java のインストーラ)	日立	通常版の場合だけ、この製品に同梱している Hitachi Elastic Application Data Store Client for Java のインストーラの X64LIN ディレクトリをコピーしてください。
PACK_TAR.Z (Hitachi Elastic Application Data Store Client for Java の修正パッチ)	日立	通常版の場合で、この製品に Hitachi Elastic Application Data Store Client for Java の修正パッチが同梱されている場合だけ格納してください。同梱されていない場合は格納不要です。

（凡例）−：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

Entity-Module/TCC-Participant コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]<Entity-Module/TCC-Participantコンテナイメージのユーザ任意のイメージ名>:<ユーザ任意のタグ名>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/entity-module-tcc-participant:V.R.S※ .
```

この例では、Entity-Module/TCC-Participant コンテナイメージのイメージ名・タグ名を「entity-module-tcc-participant:*V.R.S*※」としています。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした Entity-Module/TCC-Participant コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

2.5　共有ファイルシステムでの作業（通常版限定）

HMP-PCTO が前提とする共有ファイルシステムでの作業について説明します。

共有ファイルシステムでの作業の詳細については、OS、各製品のマニュアルなどを参照してください。

2.5.1　作業項目の一覧（通常版限定）

HMP-PCTO が前提とする共有ファイルシステムでの作業項目の一覧を次の表に示します。

表 2-14　HMP-PCTO が前提とする共有ファイルシステムでの作業項目の一覧

作業項目	備考	参照先
uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定）	次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	2.5.2　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定）

2.5.2　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定）

uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリを、次の要領で全ワーカーノードで共有可能な NFS に作成してください。この作業は、次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に実施します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

詳細については、「3.5.4　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定）」を参照してください。

2.6　Kubernetes マニフェスト作成

ユーザ責務の Kubernetes アプリケーションの、Kubernetes マニフェストの作成方法について説明します。

Kubernetes マニフェストの記述仕様の詳細は、Kubernetes のマニュアルなどを参照してください。

SQL-Participant を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名の長さの合計は、40 バイト以下としてください。

TCC-Participant を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名の長さの合計は、40 バイト以下としてください。

Kubernetes マニフェスト作成時のログおよびトラブルシュート情報の取得の設定については、「3.9.1　ログおよびトラブルシュート情報の取得の設定」を参照してください。

2.6.1　作業項目の一覧

Kubernetes マニフェスト作成の作業項目の一覧を次の表に示します。

表 2-15　Kubernetes マニフェスト作成の作業項目の一覧

作業項目	備考	参照先
Orchestrator の Kubernetes マニフェストの作成	必ず作成します。	2.6.2　Orchestrator の Kubernetes マニフェストの作成
Entity-Service（Participant コンテナを含まない Pod とする場合）の Kubernetes マニフェストの作成	サービス間を中継するサービスとしての Entity-Service がある場合だけ作成します。 詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のトランザクションのモデル（その 2）に関する説明を参照してください。	2.6.3　Entity-Service（Participant コンテナを含まない Pod とする場合）の Kubernetes マニフェストの作成
SQL-Participant の Kubernetes マニフェストの作成（SQL-Participant 限定）	SQL-Participant を使用する場合だけ作成します。	2.6.4　SQL-Participant の Kubernetes マニフェストの作成（SQL-Participant 限定）
TCC-Participant の Kubernetes マニフェストの作成（TCC-Participant 限定）	TCC-Participant を使用する場合だけ作成します。	2.6.5　TCC-Participant の Kubernetes マニフェストの作成（TCC-Participant 限定）
Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成	必ず作成します。	2.6.6　Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成
デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成	次のどちらかの Kubernetes アプリケーションの Kubernetes マニフェスト作成時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	2.6.7　デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成

作業項目	備考	参照先
uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）	通常版の場合だけ作成します。次のどちらかの Kubernetes アプリケーションの Kubernetes マニフェスト作成時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	2.6.8　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）

2.6.2　Orchestrator の Kubernetes マニフェストの作成

「6.4.1　Orchestrator の要件」に記載されている要件を満たすように、Orchestrator をデプロイするための Kubernetes マニフェストを作成してください。

Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: apps/v1     …1
kind: Deployment
metadata:
  name: orchestrator
  namespace: my-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: orchestrator
  template:
    metadata:
      labels:
        app.kubernetes.io/name: orchestrator
    spec:
      serviceAccount: orchestrator-service-account     …6
      initContainers:
      - name: dependency-checker     …2
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※2
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/deployments/sql-participant※1
      containers:
      - name: orchestrator     …3
        image: myregistry.example.com/myproject/orchestrator:V.R.S※2
        resources:     …14
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
```

```
        - name: http     …4
          containerPort: 8080
          protocol: TCP
        - name: ucars
          containerPort: 28081     …13
          protocol: TCP
        env:
        - name: TZ     …7
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS  …10
          value: -XX:+UseG1GC
        volumeMounts:
        - name: config     …5
          mountPath: /deployments/config
        - name: ucars-config     …11
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume     …12
          mountPath: /ucars-snapshots
        - name: tomcat     …8
          mountPath: /deployments/tomcat
      - name: filebeat-sidecar     …9
        image: myregistry.example.com/myproject/filebeat:V.R.S※2
        args:
        - "-e"
        - "-E"
        - "http.enabled=true"
        env:     …15
        - name: HMPPCTO_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: HMPPCTO_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: HMPPCTO_NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        livenessProbe:
          exec:
            command:
              - sh
              - -c
              - |
                #!/usr/bin/env bash -e
                curl --fail 127.0.0.1:5066
          initialDelaySeconds: 10
          periodSeconds: 10
          failureThreshold: 3
          timeoutSeconds: 5
        readinessProbe:
          exec:
            command:
              - sh
              - -c
              - |
                #!/usr/bin/env bash -e
```

```
                filebeat test output
          initialDelaySeconds: 10
          periodSeconds: 10
          failureThreshold: 3
          timeoutSeconds: 5
        securityContext:
          runAsUser: 0
        volumeMounts:
         - name: beatconfig
           mountPath: /usr/share/filebeat/filebeat.yml
           readOnly: true
           subPath: filebeat.yml
         - name: tomcat
           mountPath: /deployments/tomcat
      volumes:
      - name: config     …5
        configMap:
          name: orchestrator-config
          items:
          - key: application.properties
            path: application.properties
      - name: ucars-config     …11
        configMap:
          name: orchestrator-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: snapshot-volume     …12
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: beatconfig    …9
        configMap:
          name: orchestrator-filebeat-sidecar-config
      - name: tomcat
---
apiVersion: v1
kind: Service
metadata:
  name: orchestrator
  namespace: my-namespace
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: orchestrator
  ports:
  - name: http     …4
    protocol: TCP
    port: 8080
    targetPort: http
  - name: ucars
    protocol: TCP
    port: 28081    …13
    targetPort: ucars
---
apiVersion: v1
kind: ConfigMap    …5
metadata:
  name: orchestrator-config
```

```
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    jp.co.Hitachi.soft.hmppcto.learner.consensus-type=Three-Quarters
    jp.co.Hitachi.soft.hmppcto.mediator.name=mediator
    jp.co.Hitachi.soft.hmppcto.mediator.name-suffix=.mediator.my-namespace.svc.cluster.local
    jp.co.Hitachi.soft.hmppcto.mediator.size=5
    jp.co.Hitachi.soft.hmppcto.instance-id=orchestrator
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger-collector:14268/api/traces
---
apiVersion: v1     …11
kind: ConfigMap
metadata:
  name: orchestrator-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081     …13
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: orchestrator-filebeat-sidecar-config
  namespace: my-namespace
data:     …9
  filebeat.yml: |
    filebeat.inputs:
      - type: filestream
        id: ”tomcat”
        paths:
          - ’/deployments/tomcat/logs/access_log.*.log’
    processors:     …15
      - add_fields:
          target: kubernetes
          fields:
            namespace: ${HMPPCTO_POD_NAMESPACE}
            pod:
              name: ${HMPPCTO_POD_NAME}
            node:
              name: ${HMPPCTO_NODE_NAME}
    output.logstash:
      hosts: [”logstash.my-namespace.svc.cluster.local:5044”]
---
apiVersion: v1     …6
kind: ServiceAccount
metadata:
  name: orchestrator-service-account
  namespace: my-namespace
```

注※1

SQL-Participant を使用する場合の例を記述しています。

- TCC-Participant を使用するときは、次の記述に置き換えてください。

  - my-namespace/deployments/tcc-participant

- TP1 連携機能を使用するときは、次の記述に置き換えてください。<リリース名>は、「(15) Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。

  - my-namespace/deployments/<リリース名>

  (例)

  リリース名が tp1-bridge1 の場合：- my-namespace/deployments/tp1-bridge1

注※2

V.R.S は使用する HMP-PCTO と Filebeat のバージョンに合わせて読み替えてください。

記述例の番号は、説明の番号と対応しています。

＜説明＞

この例では、各 Kubernetes オブジェクトの Namespace 名（.metadata.namespace フィールド）は、「my-namespace」としています。

1. ワークロードリソースを定義します。この例では、ワークロードリソースを「Deployment」としています。
2. 初期化コンテナ構成に Dependency-Checker コンテナを定義します。「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成したコンテナイメージを指定します。
3. コンテナ構成に Orchestrator コンテナを定義します。「2.4.6　Entity-Service コンテナイメージの作成」で作成したコンテナイメージを指定します。
4. ユーザ責務の Kubernetes アプリケーション（および Prometheus 向け）で使用する各種ポート番号を Kubernetes クラスタ内で公開するように定義します。この例では、アプリケーションが使用するポートを TCP プロトコルの 8080 ポートとしています。
5. Orchestrator コンテナのパラメタを、Spring Boot のコンフィグソースに設定するように定義します。この例では、設定するパラメタは最小限の例としています。
6. Pod に対するロールベースアクセス制御（RBAC）に使用する ServiceAccount を定義します。
7. Orchestrator コンテナのタイムゾーンを設定します。設定するタイムゾーンはユーザ任意です。この例では、タイムゾーンを日本標準時（Asia/Tokyo、UTC+09:00）としています。
8. Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。
9. Filebeat の Sidecar コンテナを定義します。 /usr/share/filebeat/filebeat.yml に Filebeat の設定ファイルをマウントし、 /deployments/tomcat に Tomcat のログディレクトリをマウントします。この設定によって Tomcat からのログを Filebeat が取得します。なお、output.logstash フィールドの Namespace 名には Logstash アプリケーションをデプロイする Namespace を指定します。

10. Orchestrator の Java プロセス起動時に Java VM に渡すオプションを設定します。設定するオプションはユーザ任意です（ただし、「-XX:+UseG1GC」オプションは必ず含めてください)。この例では、オプションを「-XX:+UseG1GC」だけとしています。

11. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot の設定ファイル(config.properties)を Orchestrator コンテナ内にマウントします。この例では、次のパラメタを設定しています

    - monitor.rest.bindaddress=0.0.0.0

      プロセスモニタの HTTP 機能に割り当てる IP アドレスを定義します。コンテナ外部から通信を受け付けることができるようにするために、設定を行います。

    - monitor.rest.port=28081

      プロセスモニタの HTTP 機能の受付ポート番号を定義します。

    - snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot

      異常検知時のスナップショットログ出力先を、永続ボリューム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。

      スナップショットログの収集対象を追加したい場合は、「10.6.2　情報の収集方法」を参照して設定してください。

12. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを Orchestrator コンテナ内にマウントします。

    .spec.template.spec.volumes[].persistentVolumeClaim.claimName フィールドには、「2.6.8 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定)」で作成する uCosminexus Application Runtime with Java for Spring Boot スナップショットログの PersistentVolumeClaim リソースの名称(.metadata.name フィールドに指定した値)を設定してください。この例では、PersistentVolumeClaim リソースの名称を「snapshot-volume」としています。

13. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot のスナップショットログの手動取得のため、HTTP 機能として公開する受付ポート番号を定義します。

14. Orchestrator コンテナのリソースを設定します。実稼働を通して、環境に合わせた設定を実施してください。

15. Kibana などの可視化ツールで検索する際、条件として使用するフィールドを付与するために環境変数を設定します。また、環境変数を使用してフィールドを付与する設定を ConfigMap に記載します。

重要

Kubernetes マニフェスト作成時の注意事項

Kubernetes マニフェストは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して作成してください。

2.6.3 Entity-Service（Participant コンテナを含まない Pod とする場合）の Kubernetes マニフェストの作成

この Kubernetes マニフェストは、サービス間を中継するサービスとしての Entity-Service がある場合だけ作成します。詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のトランザクションのモデル（その 2）に関する説明を参照してください。

「6.4.2 Entity-Service（Participant コンテナを含まない Pod とする場合）の要件」に記載されている要件を満たすように、Entity-Service をデプロイするための Kubernetes マニフェストを作成してください。

Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: apps/v1     …1
kind: Deployment
metadata:
  name: entity-service
  namespace: my-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: entity-service
  template:
    metadata:
      labels:
        app.kubernetes.io/name: entity-service
    spec:
      serviceAccount: entity-service-service-account     …6
      initContainers:
      - name: dependency-checker     …2
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※2
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/deployments/sql-participant※1
      containers:
      - name: entity-service     …3
        image: myregistry.example.com/myproject/entity-service:V.R.S※2
        resources:     …14
```

```
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http     …4
          containerPort: 8080
          protocol: TCP
        - name: ucars
          containerPort: 28081     …13
          protocol: TCP
        env:
        - name: TZ     …7
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS  …10
          value: -XX:+UseG1GC
        volumeMounts:
        - name: entity-service-config     …5
          mountPath: /deployments/config
        - name: ucars-config     …11
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume     …12
          mountPath: /ucars-snapshots
        - name: tomcat                …8
          mountPath: /deployments/tomcat
      - name: filebeat-sidecar
        image: myregistry.example.com/myproject/filebeat:V.R.S※2
        args:
        - "-e"
        - "-E"
        - "http.enabled=true"
        env:     …15
        - name: HMPPCTO_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: HMPPCTO_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: HMPPCTO_NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        livenessProbe:
          exec:
            command:
              - sh
              - -c
              - |
                #!/usr/bin/env bash -e
                curl --fail 127.0.0.1:5066
          initialDelaySeconds: 10
          periodSeconds: 10
          failureThreshold: 3
          timeoutSeconds: 5
```

```
        readinessProbe:
          exec:
            command:
              - sh
              - -c
              - |
                #!/usr/bin/env bash -e
                filebeat test output
          initialDelaySeconds: 10
          periodSeconds: 10
          failureThreshold: 3
          timeoutSeconds: 5     …9
        securityContext:
          runAsUser: 0
        volumeMounts:
         - name: beatconfig
           mountPath: /usr/share/filebeat/filebeat.yml
           readOnly: true
           subPath: filebeat.yml
         - name: tomcat
           mountPath: /deployments/tomcat
      volumes:
      - name: entity-service-config     …5
        configMap:
          name: entity-service-config
          items:
          - key: application.properties
            path: application.properties
      - name: ucars-config     …11
        configMap:
          name: entity-service-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: snapshot-volume     …12
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: beatconfig  …9
        configMap:
          name: es-filebeat-sidecar-config
      - name: tomcat
---
apiVersion: v1
kind: Service
metadata:
  name: entity-service
  namespace: my-namespace
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: entity-service
  ports:
  - name: http     …4
    protocol: TCP
    port: 8080
    targetPort: http
  - name: grpc     …7
    protocol: TCP
```

```
    port: 20552
    targetPort: grpc
  - name: ucars
    protocol: TCP
    port: 28081     …13
    targetPort: ucars
---
apiVersion: v1     …5
kind: ConfigMap
metadata:
  name: entity-service-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    jp.co.Hitachi.soft.hmppcto.client.proposer.type=SQL
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger-collector:14268/api/traces
    spring.datasource.type=jp.co.Hitachi.soft.hmppcto.jdbc.client.PctoDataSource
    spring.datasource.driver-class-name=jp.co.Hitachi.soft.hmppcto.jdbc.client.PctoDriver
    spring.datasource.url=http://localhost:20553
    spring.datasource.username=dummy
    spring.datasource.password=dummy
---
apiVersion: v1  …11
kind: ConfigMap
metadata:
  name: entity-service-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081     …13
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: es-filebeat-sidecar-config
  namespace: my-namespace  …9
data:
  filebeat.yml: |
    filebeat.inputs:
      - type: filestream
        id: "tomcat"
        paths:
          - '/deployments/tomcat/logs/access_log.*.log'
    processors:     …15
      - add_fields:
          target: kubernetes
          fields:
            namespace: ${HMPPCTO_POD_NAMESPACE}
            pod:
              name: ${HMPPCTO_POD_NAME}
            node:
              name: ${HMPPCTO_NODE_NAME}
    output.logstash:
```

```
      hosts: ["logstash.my-namespace.svc.cluster.local:5044"]
---
apiVersion: v1      …6
kind: ServiceAccount
metadata:
  name: entity-service-service-account
  namespace: my-namespace
```

注※1

SQL-Participant を使用する場合の例を記述しています。

- TCC-Participant を使用するときは、次の記述に置き換えてください。

  - my-namespace/deployments/tcc-participant

- TP1 連携機能を使用するときは、次の記述に置き換えてください。<リリース名>は、「(15) Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。

  - my-namespace/deployments/<リリース名>

  （例）

  リリース名が tp1-bridge1 の場合：- my-namespace/deployments/tp1-bridge1

注※2

V.R.S は使用する HMP-PCTO と Filebeat のバージョンに合わせて読み替えてください。

記述例の番号は、説明の番号と対応しています。

＜説明＞

この例では、各 Kubernetes オブジェクトの Namespace 名（.metadata.namespace フィールド）は、「my-namespace」としています。

1. ワークロードリソースを定義します。この例では、ワークロードリソースを「Deployment」としています。
2. 初期化コンテナ構成に Dependency-Checker コンテナを定義します。「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成したコンテナイメージを指定します。
3. コンテナ構成に Entity-Service コンテナを定義します。「2.4.6　Entity-Service コンテナイメージの作成」で作成したコンテナイメージを指定します。
4. ユーザ責務の Kubernetes アプリケーション（および Prometheus 向け）で使用する各種ポート番号を Kubernetes クラスタ内で公開するように定義します。この例では、アプリケーションが使用するポートを TCP プロトコルの 8080 ポートとしています。
5. Entity-Service コンテナのパラメタを、Spring Boot のコンフィグソースに設定するように定義します。この例では、必須パラメタだけ指定しています。
6. Pod に対するロールベースアクセス制御（RBAC）に使用する ServiceAccount を定義します。
7. Entity-Service コンテナのタイムゾーンを設定します。設定するタイムゾーンはユーザ任意です。この例では、タイムゾーンを日本標準時（Asia/Tokyo、UTC+09:00）としています。

8. Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。
9. Filebeat の Sidecar コンテナを定義します。 /usr/share/filebeat/filebeat.yml に Filebeat の設定ファイルをマウントし、 /deployments/tomcat に Tomcat のログディレクトリをマウントします。この設定によって Tomcat からのログを Filebeat が取得します。なお、output.logstash フィールドの Namespace 名には Logstash アプリケーションをデプロイする Namespace を指定します。
10. Entity-Service の Java プロセス起動時に Java VM に渡すオプションを設定します。設定するオプションはユーザ任意です（ただし、「-XX:+UseG1GC」オプションは必ず含めてください）。この例では、オプションを「-XX:+UseG1GC」だけとしています。
11. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot の設定ファイル(config.properties)を Entity-Service コンテナ内にマウントします。この例では、次のパラメタを設定しています。

    - monitor.rest.bindaddress=0.0.0.0

      プロセスモニタの HTTP 機能に割り当てる IP アドレスを定義します。コンテナ外部から通信を受け付けることができるようにするために、設定を行います。

    - monitor.rest.port=28081

      プロセスモニタの HTTP 機能の受付ポート番号を定義します。

    - snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot

      異常検知時のスナップショットログ出力先を、永続ボリューム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。

      スナップショットログの収集対象を追加したい場合は、「10.6.2　情報の収集方法」を参照して設定してください。
12. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを Entity-Service コンテナ内にマウントします。

    .spec.template.spec.volumes[].persistentVolumeClaim.claimName フィールドには、「2.6.8 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」で作成する uCosminexus Application Runtime with Java for Spring Boot スナップショットログの PersistentVolumeClaim リソースの名称(.metadata.name フィールドに指定した値)を設定してください。この例では、PersistentVolumeClaim リソースの名称を「snapshot-volume」としています。
13. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot のスナップショットログの手動取得のため、HTTP 機能として公開する受付ポート番号を定義します。

14. Entity-Service コンテナのリソースを設定します。実稼働を通して、環境に合わせた設定を実施してください。

15. Kibana などの可視化ツールで検索する際、条件として使用するフィールドを付与するために環境変数を設定します。また、環境変数を使用してフィールドを付与する設定を ConfigMap に記載します。

> **重要**
>
> Kubernetes マニフェスト作成時の注意事項
>
> Kubernetes マニフェストは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して作成してください。

2.6.4 SQL-Participant の Kubernetes マニフェストの作成（SQL-Participant 限定）

「6.4.3 SQL-Participant の要件」に記載されている要件を満たすように、Participant をデプロイするための Kubernetes マニフェストを作成してください。

Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: apps/v1     …1
kind: Deployment
metadata:
  name: sql-participant
  namespace: my-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: sql-participant
  template:
    metadata:
      labels:
        app.kubernetes.io/name: sql-participant
    spec:
      serviceAccount: sql-participant-service-account     …13
      initContainers:
      - name: dependency-checker     …2
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/statefulsets/eads-server     …21
        - my-namespace/statefulsets/postgresql
      containers:
      - name: entity-service     …3
```

```
        image: myregistry.example.com/myproject/entity-service:V.R.S※
        resources:     …29
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http     …4
          containerPort: 8080
          protocol: TCP
        - name: ucars-entity
          containerPort: 28082 …28
          protocol: TCP
        env:
        - name: TZ     …15
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS
          value: -XX:+UseG1GC
        volumeMounts:
        - name: entity-service-config     …5
          mountPath: /deployments/config
        - name: entity-service-ucars-config  …22
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume  …23
          mountPath: /ucars-snapshots
        - name: tomcat  …19
          mountPath: /deployments/tomcat
      - name: sql-participant     …6
        image: myregistry.example.com/myproject/sql-participant:V.R.S※
        resources:     …30
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: grpc     …7
          containerPort: 20552
          protocol: TCP
        - name: http     …17
          containerPort: 8080
          protocol: TCP
        - name: ucars-sql
          containerPort: 28081  …28
          protocol: TCP
        env:
        - name: TZ     …16
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS
          value: -XX:+UseG1GC
        volumeMounts:
        - name: sql-participant-config     …7
          mountPath: /deployments/config
        - name: sql-participant-ucars-config  …24
          mountPath: /opt/hitachi/ucars/conf
```

```
        - name: snapshot-volume  …25
          mountPath: /ucars-snapshots
        - name: eads-config     …14
          mountPath: /eads-properties
        - name: tomcat     …19
          mountPath: /deployments/tomcat
        - name: eads
          mountPath: /deployments/eads-log   …20
        livenessProbe:     …9
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
            - "10"
            - "20555"
          initialDelaySeconds: 180
          periodSeconds: 30
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
          postStart:     …10
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "20555"
          preStop:     …11
            exec:
              command:
              - /deployments/wait_transactions.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "20555"
      - name: filebeat-sidecar     …18
        image: myregistry.example.com/myproject/filebeat:V.R.S※
        args:
        - "-e"
        - "-E"
        - "http.enabled=true"
        env:     …31
        - name: HMPPCTO_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: HMPPCTO_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: HMPPCTO_NODE_NAME
          valueFrom:
            fieldRef:
```

```
            fieldPath: spec.nodeName
      livenessProbe:
        exec:
          command:
            - sh
            - -c
            - |
              #!/usr/bin/env bash -e
              curl --fail 127.0.0.1:5066
        initialDelaySeconds: 10
        periodSeconds: 10
        failureThreshold: 3
        timeoutSeconds: 5
      readinessProbe:
        exec:
          command:
            - sh
            - -c
            - |
              #!/usr/bin/env bash -e
              filebeat test output
        initialDelaySeconds: 10
        periodSeconds: 10
        failureThreshold: 3
        timeoutSeconds: 5     …18
      securityContext:
        runAsUser: 0
      volumeMounts:
        - name: beatconfig
          mountPath: /usr/share/filebeat/filebeat.yml
          readOnly: true
          subPath: filebeat.yml
        - name: tomcat
          mountPath: /deployments/tomcat
        - name: eads
          mountPath: /deployments/eads-log
    terminationGracePeriodSeconds: 330     …12
    volumes:
    - name: entity-service-config     …5
      configMap:
        name: entity-service-config
        items:
        - key: application.properties
          path: application.properties
    - name: entity-service-ucars-config  …22
      configMap:
        name: entity-service-ucars-config
        items:
          - key: config.properties
            path: config.properties
    - name: snapshot-volume  …23,25
      persistentVolumeClaim:
        claimName: snapshot-volume
    - name: sql-participant-config
      configMap:
        name: sql-participant-config     …8
        items:
        - key: application.properties
```

```
            path: application.properties
      - name: sql-participant-ucars-config …24
        configMap:
          name: sql-participant-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: eads-config     …14
        configMap:
          name: eads-config
          items:
          - key: hmp-pcto.eads-client.properties
            path: hmp-pcto.eads-client.properties
      - name: beatconfig     …18
        configMap:
          name: sql-p-filebeat-sidecar-config
      - name: tomcat
      - name: eads
---
apiVersion: v1
kind: Service
metadata:
  name: sp
  namespace: my-namespace
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: sql-participant
  ports:
  - name: http     …4
    protocol: TCP
    port: 8080
    targetPort: http
  - name: grpc     …7
    protocol: TCP
    port: 20552
    targetPort: grpc
  - name: ucars-sql
    protocol: TCP
    port: 28081 …28
    targetPort: ucars-sql
  - name: ucars-entity
    protocol: TCP
    port: 28082 …28
    targetPort: ucars-entity
---
apiVersion: v1     …5
kind: ConfigMap
metadata:
  name: entity-service-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    jp.co.Hitachi.soft.hmppcto.client.proposer.type=SQL
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger-collector:14268/api/traces
```

```
    spring.datasource.type=jp.co.Hitachi.soft.hmppcto.jdbc.client.PctoDataSource
    spring.datasource.driver-class-name=jp.co.Hitachi.soft.hmppcto.jdbc.client.PctoDriver
    spring.datasource.url=http://localhost:20553
    spring.datasource.username=dummy
    spring.datasource.password=dummy
    spring.sql.init.mode=never
---
apiVersion: v1  …22
kind: ConfigMap
metadata:
  name: entity-service-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28082  …28
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
---
apiVersion: v1     …8
kind: ConfigMap
metadata:
  name: sql-participant-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=20555
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    jp.co.Hitachi.soft.hmppcto.sql.pod.service-name=sp.my-namespace
    jp.co.Hitachi.soft.hmppcto.sql.xa-server-name=sql-participant
    jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.path=/deployments/postgresql-42.2.22.jar
    jp.co.Hitachi.soft.hmppcto.sql.datasource.class-name=org.postgresql.xa.PGXADataSource
    jp.co.Hitachi.soft.hmppcto.sql.datasource.property.values=serverName=postgresql;portNumb
er=5432;databaseName=appuser1;user=appuser1;password=appuser1
    jp.co.Hitachi.soft.hmppcto.sql.pool.max-pool-size=10
    jp.co.Hitachi.soft.hmppcto.mediator.name=mediator
    jp.co.Hitachi.soft.hmppcto.mediator.name-suffix=.mediator.my-namespace.svc.cluster.local
    jp.co.Hitachi.soft.hmppcto.mediator.port=20551
    jp.co.Hitachi.soft.hmppcto.sql.db-type=PostgreSQL     …32
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger-collector:14268/api/traces
---
apiVersion: v1 …24
kind: ConfigMap
metadata:
  name: sql-participant-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081 …28
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
---
apiVersion: v1     …14
kind: ConfigMap
metadata:
  name: eads-config
  namespace: my-namespace
data:
```

```
  hmp-pcto.eads-client.properties: |
    eads.client.connect.servers=sv1,sv2,sv3
    eads.client.sv1.address=eads-service-eads-service-0.my-namespace
    eads.client.sv1.port=24600
    eads.client.sv2.address=eads-service-eads-service-1.my-namespace
    eads.client.sv2.port=24600
    eads.client.sv3.address=eads-service-eads-service-2.my-namespace
    eads.client.sv3.port=24600
    eads.client.logger.dir=eads-log
    eads.client.connectionPool.poolsize=1
---
apiVersion: v1     …13
kind: ServiceAccount
metadata:
  name: sql-participant-service-account
  namespace: my-namespace
---
apiVersion: v1     …18
kind: ConfigMap
metadata:
  name: sql-p-filebeat-sidecar-config
  namespace: my-namespace
data:
  filebeat.yml: |
    filebeat.inputs:
      - type: filestream
        id: "eads"
        paths:
          - '/deployments/eads-log/eads_client_message*.log'
          - '/deployments/eads-log/maintenance/eads_client_maintenance*.log'
      - type: filestream
        id: "tomcat"
        paths:
          - '/deployments/tomcat/logs/access_log.*.log'
    processors:     …31
      - add_fields:
          target: kubernetes
          fields:
            namespace: ${HMPPCTO_POD_NAMESPACE}
            pod:
              name: ${HMPPCTO_POD_NAME}
            node:
              name: ${HMPPCTO_NODE_NAME}
    output.logstash:
      hosts: ["logstash.my-namespace.svc.cluster.local:5044"]
---
```

注※

V.R.S は使用する HMP-PCTO と Filebeat のバージョンに合わせて読み替えてください。

記述例の番号は、説明の番号と対応しています。

＜説明＞

この例では、各 Kubernetes オブジェクトの Namespace 名（.metadata.namespace フィールド）は、「my-namespace」としています。

1. ワークロードリソースを定義します。この例では、ワークロードリソースを「Deployment」としています。
2. 初期化コンテナ構成に Dependency-Checker コンテナを定義します。「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成したコンテナイメージを指定します。
3. コンテナ構成に Entity-Service コンテナを定義します。「2.4.6　Entity-Service コンテナイメージの作成」で作成したコンテナイメージを指定します。
4. ユーザ責務の Kubernetes アプリケーション（および Prometheus 向け）で使用する各種ポート番号を Kubernetes クラスタ内で公開するように定義します。この例では、アプリケーションが使用するポートを TCP プロトコルの 8080 ポートとしています。
5. Entity-Service コンテナのパラメタを、Spring Boot のコンフィグソースに設定するように定義します。この例では、必須パラメタだけ指定しています。
6. コンテナ構成に SQL-Participant コンテナを定義します。「2.4.7　SQL-Participant コンテナイメージの作成（SQL-Participant 限定）」で作成したコンテナイメージを指定します。
7. SQL-Participant の gRPC ポート番号（Mediator 向け）を Kubernetes クラスタ内で公開するように定義します。SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.mediator.port パラメタの設定値と同じポート番号を指定します。
8. SQL-Participant のパラメタを、Spring Boot のコンフィグソースに設定するように定義します。この例では、必須パラメタだけ指定しています。
9. コンテナライフサイクル設定の Liveness Probe を定義します。
10. コンテナライフサイクル設定の postStart フックを定義します。
11. コンテナライフサイクル設定の preStop フックを定義します。
12. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。
13. Pod に対するロールベースアクセス制御（RBAC）に使用する ServiceAccount を定義します。
14. EADS クライアント定義のパラメタを、EADS クライアント定義ファイルのコンフィグソースに設定するように定義します。この例では、必須パラメタだけ指定しています。EADS クライアント定義ファイルのディレクトリおよび名称は「8.2.4　SQL-Participant」のパラメタ jp.co.Hitachi.soft.hmppcto.external-consensus.eads.properties に指定した値と合わせる必要があります。

    トライアル版の場合、この定義は必要ありません。
15. Entity-Service コンテナのタイムゾーンを設定します。設定するタイムゾーンはユーザ任意です。この例では、タイムゾーンを日本標準時（Asia/Tokyo、UTC+09:00）としています。
16. SQL-Participant コンテナのタイムゾーンを設定します。設定するタイムゾーンはユーザ任意です。この例では、タイムゾーンを日本標準時（Asia/Tokyo、UTC+09:00）としています。

17. Prometheus 向けで使用する各種ポート番号を Kubernetes クラスタ内で公開するように定義します。この例では、アプリケーションが使用するポートを TCP プロトコルの 8080 ポートとしています。

18. Filebeat の Sidecar コンテナを定義します。 /usr/share/filebeat/filebeat.yml に Filebeat の設定ファイルをマウントします。この設定によって EADS からのログを Filebeat が取得します。なお、output.logstash フィールドの Namespace 名には Logstash アプリケーションをデプロイする Namespace を指定します。

19. Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。

20. EADS のログ出力ディレクトリとして/deployments/eads-log をマウントします。

21. 通常版の場合は my-namespace/statefulsets/eads-server を記載してください。トライアル版の場合は my-namespace/deployments/ext-cons を記載してください。

22. Entity-Service の Java プロセス起動時に Java VM に渡すオプションを設定します。設定するオプションはユーザ任意です（ただし、「-XX:+UseG1GC」オプションは必ず含めてください)。この例では、オプションを「-XX:+UseG1GC」だけとしています。

23. SQL-Participant の Java プロセス起動時に Java VM に渡すオプションを設定します。設定するオプションはユーザ任意です（ただし、「-XX:+UseG1GC」オプションは必ず含めてください)。この例では、オプションを「-XX:+UseG1GC」だけとしています。

24. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot の設定ファイル(config.properties)を Entity-Service コンテナ内にマウントします。この例では、次のパラメタを設定しています。

    - monitor.rest.bindaddress=0.0.0.0

      プロセスモニタの HTTP 機能に割り当てる IP アドレスを定義します。コンテナ外部から通信を受け付けることができるようにするために、設定を行います。

    - monitor.rest.port=28082

      プロセスモニタの HTTP 機能の受付ポート番号を定義します。設定値はデフォルト値(28081)とは異なるポート番号に変更します。これは、同一 Pod 内に起動する Entity-Service コンテナと SQL-Participant コンテナでポート番号の重複を避けるためです。

    - snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot

      異常検知時のスナップショットログ出力先を、永続ボリューム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。

      スナップショットログの収集対象を追加したい場合は、「10.6.2　情報の収集方法」を参照して設定してください。

25. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを Entity-Service コンテナ内にマウントします。

.spec.template.spec.volumes[].persistentVolumeClaim.claimName フィールドには、「2.6.8 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」で作成する uCosminexus Application Runtime with Java for Spring Boot スナップショットログの PersistentVolumeClaim リソースの名称(.metadata.name フィールドに指定した値)を設定してください。この例では、PersistentVolumeClaim リソースの名称を「snapshot-volume」としています。

26. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot の設定ファイル(config.properties)を SQL-Participant コンテナ内にマウントします。この例では、次のパラメタを設定しています。

    - monitor.rest.bindaddress=0.0.0.0

      プロセスモニタの HTTP 機能に割り当てる IP アドレスを定義します。コンテナ外部から通信を受け付けることができるようにするために、設定を行います。

    - monitor.rest.port=28081

      プロセスモニタの HTTP 機能の受付ポート番号を定義します。

    - snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot

      異常検知時のスナップショットログ出力先を、永続ボリューム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。

      スナップショットログの収集対象を追加したい場合は、「10.6.2　情報の収集方法」を参照して設定してください。

27. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを SQL-Participant コンテナ内にマウントします。.spec.template.spec.volumes[].persistentVolumeClaim.claimName フィールドには、「2.6.8 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」で作成する uCosminexus Application Runtime with Java for Spring Boot スナップショットログの PersistentVolumeClaim リソースの名称(.metadata.name フィールドに指定した値)を設定してください。この例では、PersistentVolumeClaim リソースの名称を「snapshot-volume」としています。

28. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot のスナップショットログの手動取得のため、HTTP 機能として公開する受付ポート番号を定義します。

29. Entity-Service コンテナのリソースを設定します。実稼働を通して、環境に合わせた設定を実施してください。

30. SQL-Participant コンテナのリソースを設定します。実稼働を通して、環境に合わせた設定を実施してください。

31. Kibana などの可視化ツールで検索する際、条件として使用するフィールドを付与するために環境変数を設定します。また、環境変数を使用してフィールドを付与する設定を ConfigMap に記載します。

32. SQL-Participant が接続する RDBMS を指定します。PostgreSQL を指定する場合、「8.4.1 PostgreSQL（SQL-Participant 限定）」のパラメタも確認してください。

重要

Kubernetes マニフェスト作成時の注意事項

- Kubernetes マニフェストは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して作成してください。
- SQL-Participant を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名の長さの合計は、40 バイト以下としてください。

2.6.5 TCC-Participant の Kubernetes マニフェストの作成（TCC-Participant 限定）

「6.4.4 TCC-Participant の要件」に記載されている要件を満たすように、Participant をデプロイするための Kubernetes マニフェストを作成してください。

Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: apps/v1     …1
kind: Deployment
metadata:
  name: tcc-participant
  namespace: my-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: tcc-participant
  template:
    metadata:
      labels:
        app.kubernetes.io/name: tcc-participant
    spec:
      serviceAccount: tcc-participant-service-account     …11
      initContainers:
      - name: dependency-checker     …2
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/statefulsets/eads-server …21
```

```
      containers:
      - name: entity-module-tcc-participant     …3
        image: myregistry.example.com/myproject/entity-module-tcc-participant:V.R.S※
        resources:     …22
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http     …4
          containerPort: 8080
          protocol: TCP
        - name: grpc     …5
          containerPort: 20552
          protocol: TCP
        - name: ucars
          containerPort: 28081     …20
          protocol: TCP
        env:
        - name: TZ     …13
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS     …17
          value: -XX:+UseG1GC
        volumeMounts:
        - name: entity-module-tcc-participant-config     …6
          mountPath: /deployments/config
        - name: ucars-config     …18
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume     …19
          mountPath: /ucars-snapshots
        - name: eads-config     …12
          mountPath: /eads-properties
        - name: tomcat     …15
          mountPath: /deployments/tomcat
        - name: eads     …16
          mountPath: /deployments/eads-log
        livenessProbe:     …7
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
            - "10"
            - "8080"
          initialDelaySeconds: 180
          periodSeconds: 30
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
          postStart:     …8
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
```

```
        - "10"
        - "10"
        - "8080"
    preStop:     …9
      exec:
        command:
        - /deployments/wait_transactions.sh
        - "30"
        - "10"
        - "10"
        - "10"
        - "8080"
- name: filebeat-sidecar
  image: myregistry.example.com/myproject/filebeat:V.R.S※
  args:
  - "-e"
  - "-E"
  - "http.enabled=true"
  env:     …23
  - name: HMPPCTO_POD_NAMESPACE
    valueFrom:
      fieldRef:
        fieldPath: metadata.namespace
  - name: HMPPCTO_POD_NAME
    valueFrom:
      fieldRef:
        fieldPath: metadata.name
  - name: HMPPCTO_NODE_NAME
    valueFrom:
      fieldRef:
        fieldPath: spec.nodeName
  livenessProbe:
    exec:
      command:
        - sh
        - -c
        - |
          #!/usr/bin/env bash -e
          curl --fail 127.0.0.1:5066
    initialDelaySeconds: 10
    periodSeconds: 10
    failureThreshold: 3
    timeoutSeconds: 5
  readinessProbe:
    exec:
      command:
        - sh
        - -c
        - |
          #!/usr/bin/env bash -e
          filebeat test output
    initialDelaySeconds: 10
    periodSeconds: 10
    failureThreshold: 3
    timeoutSeconds: 5     …14
  securityContext:
    runAsUser: 0
  volumeMounts:
```

```
          - name: beatconfig
            mountPath: /usr/share/filebeat/filebeat.yml
            readOnly: true
            subPath: filebeat.yml
          - name: tomcat
            mountPath: /deployments/tomcat
          - name: eads
            mountPath: /deployments/eads-log
      terminationGracePeriodSeconds: 330     …10
      volumes:
      - name: entity-module-tcc-participant-config     …6
        configMap:
          name: entity-module-tcc-participant-config
          items:
          - key: application.properties
            path: application.properties
      - name: ucars-config     …18
        configMap:
          name: entity-module-tcc-participant-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: snapshot-volume     …19
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: eads-config     …12
        configMap:
          name: eads-config
          items:
          - key: hmp-pcto.eads-client.properties
            path: hmp-pcto.eads-client.properties
      - name: beatconfig
        configMap:
          name: tcc-p-filebeat-sidecar-config
      - name: tomcat
      - name: eads
---
apiVersion: v1
kind: Service
metadata:
  name: tp
  namespace: my-namespace
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: tcc-participant
  ports:
  - name: http     …4
    protocol: TCP
    port: 8080
    targetPort: http
  - name: grpc     …5
    protocol: TCP
    port: 20552
    targetPort: grpc
  - name: ucars
    protocol: TCP
    port: 28081     …20
```

```
    targetPort: ucars
---
apiVersion: v1     …6
kind: ConfigMap
metadata:
  name: entity-module-tcc-participant-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    jp.co.Hitachi.soft.hmppcto.tcc.pod.service-name=tp.my-namespace
    jp.co.Hitachi.soft.hmppcto.client.proposer.type=TCC
    jp.co.Hitachi.soft.hmppcto.mediator.name=mediator
    jp.co.Hitachi.soft.hmppcto.mediator.name-suffix=.mediator.my-namespace.svc.cluster.local
    jp.co.Hitachi.soft.hmppcto.mediator.port=20551
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger-collector:14268/api/traces
---
apiVersion: v1     …18
kind: ConfigMap
metadata:
  name: entity-module-tcc-participant-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081     …20
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
---
apiVersion: v1     …12
kind: ConfigMap
metadata:
  name: eads-config
  namespace: my-namespace
data:
  hmp-pcto.eads-client.properties: |
    eads.client.connect.servers=sv1,sv2,sv3
    eads.client.sv1.address=eads-service-eads-service-0.my-namespace
    eads.client.sv1.port=24600
    eads.client.sv2.address=eads-service-eads-service-1.my-namespace
    eads.client.sv2.port=24600
    eads.client.sv3.address=eads-service-eads-service-2.my-namespace
    eads.client.sv3.port=24600
    eads.client.logger.dir=eads-log
    eads.client.connectionPool.poolsize=1
---
apiVersion: v1     …11
kind: ServiceAccount
metadata:
  name: tcc-participant-service-account
  namespace: my-namespace
---
apiVersion: v1     …14
kind: ConfigMap
metadata:
  name: tcc-p-filebeat-sidecar-config
  namespace: my-namespace
```

```
data:
  filebeat.yml: |
    filebeat.inputs:
      - type: filestream
        id: "eads"
        paths:
          - '/deployments/eads-log/eads_client_message*.log'
          - '/deployments/eads-log/maintenance/eads_client_maintenance*.log'
      - type: filestream
        id: "tomcat"
        paths:
          - '/deployments/tomcat/logs/access_log.*.log'
    processors:     …23
      - add_fields:
          target: kubernetes
          fields:
            namespace: ${HMPPCTO_POD_NAMESPACE}
            pod:
              name: ${HMPPCTO_POD_NAME}
            node:
              name: ${HMPPCTO_NODE_NAME}
    output.logstash:
      hosts: ["logstash.my-namespace.svc.cluster.local:5044"]
```

注※

V.R.S は使用する HMP-PCTO と Filebeat のバージョンに合わせて読み替えてください。

記述例の番号は、説明の番号と対応しています。

＜説明＞

この例では、各 Kubernetes オブジェクトの Namespace 名（.metadata.namespace フィールド）は、「my-namespace」としています。

1. ワークロードリソースを定義します。この例では、ワークロードリソースを「Deployment」としています。
2. 初期化コンテナ構成に Dependency-Checker コンテナを定義します。「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成したコンテナイメージを指定します。
3. コンテナ構成に Entity-Module/TCC-Participant コンテナを定義します。「2.4.8　Entity-Module/TCC-Participant コンテナイメージの作成（TCC-Participant 限定）」で作成したコンテナイメージを指定します。
4. ユーザ責務の Kubernetes アプリケーション（および Prometheus 向け）で使用する各種ポート番号を Kubernetes クラスタ内で公開するように定義します。この例では、アプリケーションが使用するポートを TCP プロトコルの 8080 ポートとしています。
5. TCC-Participant の gRPC ポート番号（Mediator 向け）を Kubernetes クラスタ内で公開するように定義します。TCC-Participant の jp.co.Hitachi.soft.hmppcto.tcc.mediator.port パラメタの設定値と同じポート番号を指定します。

6. Entity-Module/TCC-Participant コンテナのパラメタを、Spring Boot のコンフィグソースに設定するように定義します。この例では、必須パラメタだけ指定しています。
7. コンテナライフサイクル設定の Liveness Probe を定義します。
8. コンテナライフサイクル設定の postStart フックを定義します。
9. コンテナライフサイクル設定の preStop フックを定義します。
10. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。
11. Pod に対するロールベースアクセス制御（RBAC）に使用する ServiceAccount を定義します。
12. EADS クライアント定義のパラメタを、EADS クライアント定義ファイルのコンフィグソースに設定するように定義します。この例では、必須パラメタだけ指定しています。EADS クライアント定義ファイルのディレクトリおよび名称は「8.2.5　TCC-Participant」のパラメタ jp.co.Hitachi.soft.hmppcto.external-consensus.eads.properties に指定した値と合わせる必要があります。

    トライアル版の場合、この定義は必要ありません。
13. Entity-Module/TCC-Participant コンテナのタイムゾーンを設定します。設定するタイムゾーンはユーザ任意です。この例では、タイムゾーンを日本標準時（Asia/Tokyo、UTC+09:00）としています。
14. Filebeat の Sidecar コンテナを定義します。 /usr/share/filebeat/filebeat.yml に Filebeat の設定ファイルをマウントします。この設定によって EADS からのログを Filebeat が取得します。なお、output.logstash フィールドの Namespace 名には Logstash アプリケーションをデプロイする Namespace を指定します。
15. Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。
16. EADS のログ出力ディレクトリとして/deployments/eads-log をマウントします。
17. Entity-Module/TCC-Participant の Java プロセス起動時に Java VM に渡すオプションを設定します。設定するオプションはユーザ任意です（ただし、「-XX:+UseG1GC」オプションは必ず含めてください）。この例では、オプションを「-XX:+UseG1GC」だけとしています。
18. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot の設定ファイル(config.properties)を Entity-Module/TCC-Participant コンテナ内にマウントします。この例では、次のパラメタを設定しています。

    - monitor.rest.bindaddress=0.0.0.0

      プロセスモニタの HTTP 機能に割り当てる IP アドレスを定義します。コンテナ外部から通信を受け付けることができるようにするために、設定を行います。
    - monitor.rest.port=28081

      プロセスモニタの HTTP 機能の受付ポート番号を定義します。
    - snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot

異常検知時のスナップショットログ出力先を、永続ボリューム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。

スナップショットログの収集対象を追加したい場合は、「10.6.2　情報の収集方法」を参照して設定してください。

19. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを Entity-Module/TCC-Participant コンテナ内にマウントします。.spec.template.spec.volumes[].persistentVolumeClaim.claimName フィールドには、「2.6.8 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」で作成する uCosminexus Application Runtime with Java for Spring Boot スナップショットログの PersistentVolumeClaim リソースの名称(.metadata.name フィールドに指定した値)を設定してください。この例では、PersistentVolumeClaim リソースの名称を「snapshot-volume」としています。

20. オレンジ色の背景で示す行は、uCosminexus Application Runtime with Java for Spring Boot 導入ありの場合だけ記述します。

    uCosminexus Application Runtime with Java for Spring Boot のスナップショットログの手動取得のため、HTTP 機能として公開する受付ポート番号を定義します。

21. 通常版の場合は my-namespace/statefulsets/eads-server を記載してください。トライアル版の場合は my-namespace/deployments/ext-cons を記載してください。

22. Entity-Module/TCC-Participant コンテナのリソースを設定します。実稼働を通して、環境に合わせた設定を実施してください。

23. Kibana などの可視化ツールで検索する際、条件として使用するフィールドを付与するために環境変数を設定します。また、環境変数を使用してフィールドを付与する設定を ConfigMap に記載します。

重要

Kubernetes マニフェスト作成時の注意事項

- Kubernetes マニフェストは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して作成してください。
- TCC-Participant を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名の長さの合計は、40 バイト以下としてください。

2.6.6 Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成

次に示す要件を満たすように、Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストを作成してください。

このロールおよびロールバインディングを使用することで、対象コンテナが Mediator 選択機能に必要な Kubernetes オブジェクトにアクセスできるようにします。

Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成要領を次に示します。

＜ロールの作成要領＞

- リソースの種類(.kind フィールド)は、ClusterRole リソースを指定してください。
- ロールの名称（.metadata.name フィールド）に、ユーザ任意の名称を指定してください。
- ロールのルール（.rules フィールド）に、次のとおり指定してください。
  - API グループ（apiGroups）
    空（コアグループ）
  - リソース（resources）
    services、endpoints
  - アクセス許可（verbs）
    get、list、watch

＜ロールバインディングの作成要領＞

- リソースの種類(.kind フィールド)は、ClusterRoleBinding リソースを指定してください。
- ロールバインディングの名称（.metadata.name フィールド）に、ユーザ任意の名称を指定してください。
- バインドするロール（.roleRef フィールド）は、次のとおり指定してください。
  - .roleRef.apiGroup フィールドに、「rbac.authorization.k8s.io」を指定してください。
  - .roleRef.kind フィールドに、「ClusterRole」を指定してください。
  - .roleRef.name フィールドに、ロールで指定したロールの名称を指定してください。
- バインドする対象物（.subjects フィールド）は、次のとおり指定してください。
  - リストの要素数は Orchestrator、Entity-Service、Participant と同じ数にします。
  - .subjects[].kind フィールドに、「ServiceAccount」を指定してください。
  - .subjects[].name フィールドに、Orchestrator、Entity-Service、Participant の ServiceAccount リソースの.metadata.name フィールドに指定した名称を指定してください。

- .subjects[].namespace フィールドに、Orchestrator、Entity-Service、Participant の ServiceAccount リソースの.metadata.namespace フィールドに指定した Namespace を指定してください。

Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: lb-role
rules:
- apiGroups:
  - ""
  resources:
  - services
  - endpoints
  verbs:
  - get
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: lb-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: lb-role
subjects:
- kind: ServiceAccount
  name: orchestrator-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: entity-service-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: sql-participant-service-account※
  namespace: my-namespace
```

注※

SQL-Participant を使用する場合の例を記述しています。

- TCC-Participant を使用する場合は、「name: tcc-participant-service-account」に置き換えてください。
- TP1-Bridge を使用する場合は、「name: <リリース名>-service-account」に置き換えてください。<リリース名>は、「(15)　Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。

  例：リリース名が tp1-bridge1 の場合

  name: tp1-bridge1-service-account

2.6.7 デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成

デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストは、次のどちらかの Kubernetes アプリケーションの Kubernetes マニフェスト作成時に作成します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

詳細については、「3.6.6 デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成」を参照してください。

2.6.8 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）

uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストは、次のどちらかの Kubernetes アプリケーションの Kubernetes マニフェスト作成時に作成します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

詳細については、「3.6.10 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」を参照してください。

2.7　ユーザ責務の Kubernetes アプリケーションのデプロイとアンデプロイ

ユーザ責務の Kubernetes アプリケーションのデプロイ、アンデプロイ方法を説明します。

2.7.1　作業項目の一覧

ユーザ責務の Kubernetes アプリケーションをデプロイするための作業項目の一覧を次に示します。作業対象は Kubernetes コントロールプレーンです。表の上から順に作業を実行してください。

表 2-16　ユーザ責務の Kubernetes アプリケーションをデプロイするための作業項目の一覧

作業項目	備考	参照先
各種資材の配置	必ず実施します。	(1)　各種資材の配置
Namespace の作成	次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	(2)　Namespace の作成
Orchestrator、Participant の gRPC 通信機能の暗号化通信用シークレットの作成	gRPC 通信機能で暗号化通信をする場合だけ作成します。	(3)　Orchestrator、Participant の gRPC 通信機能の暗号化通信用シークレットの作成
Mediator 選択機能用ロールおよびロールバインディングのデプロイ	必ず実施します。	(4)　Mediator 選択機能用ロールおよびロールバインディングのデプロイ
デプロイ依存関係チェック機能用ロールおよびロールバインディングのデプロイ	次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に実施します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	(5)　デプロイ依存関係チェック機能用ロールおよびロールバインディングのデプロイ
uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求のデプロイ（通常版限定）	通常版の場合だけ作成します。次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に実施します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	(6)　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求のデプロイ（通常版限定）
Kubernetes アプリケーションのデプロイ	必ず実施します。	(7)　ユーザ責務の Kubernetes アプリケーションのデプロイ
Kubernetes アプリケーションのアンデプロイ	必ず実施します。	(8)　ユーザ責務の Kubernetes アプリケーションのアンデプロイ

2.7.2　Kubernetes コントロールプレーンでの作業

(1)　各種資材の配置

Kubernetes コントロールプレーンの任意のパスへ、次の各種資材を配置してください。

表 2-17　Kubernetes コントロールプレーンに配置する資材の一覧

資材	備考
Orchestrator の Kubernetes マニフェスト	「2.6.2　Orchestrator の Kubernetes マニフェストの作成」で作成したもの
Entity-Service の Kubernetes マニフェスト	「2.6.3　Entity-Service（Participant コンテナを含まない Pod とする場合）の Kubernetes マニフェストの作成」で作成したもの
Participant の Kubernetes マニフェスト	「2.6.4　SQL-Participant の Kubernetes マニフェストの作成（SQL-Participant 限定）」、「2.6.5　TCC-Participant の Kubernetes マニフェストの作成（TCC-Participant 限定）」で作成したもの
Participant が使用する DB/外部サービスの Kubernetes マニフェスト（Participant を使用する場合）	ユーザが適宜作成したもの
Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェスト	「2.6.6　Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成」で作成したもの
デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェスト	「2.6.7　デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成」で作成したもの 次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に配置します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション
uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェスト（通常版限定）	「2.6.8　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」で作成したもの 通常版の場合だけ実施します。次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に配置します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

(2) Namespace の作成

HMP-PCTO の Kubernetes アプリケーションをデプロイする Namespace を作成してください。この作業は、次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に実施します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

詳細については、「(2) Namespace の作成」を参照してください。

(3) Orchestrator、Participant の gRPC 通信機能の暗号化通信用シークレットの作成

HMP-PCTO の gRPC 通信機能で暗号化通信をする場合だけ実施します。

Kubernetes コントロールプレーンの任意のパスへ、次のとおり HMP-PCTO の gRPC 通信機能の暗号化通信に必要なファイルを準備してください。

表 2-18 HMP-PCTO の gRPC 通信機能の暗号化通信に必要なファイルの一覧

対象 Kubernetes アプリケーション	ファイル	備考
Orchestrator	接続先サーバのサーバ証明書を発行した CA の証明書（PEM 形式）ファイル	必ず準備します。
	自クライアント証明書ファイル（PEM 形式）	クライアント認証を行う場合だけ準備します。
	自クライアント証明書を作成した秘密鍵ファイル	クライアント認証を行う場合だけ準備します。
Participant（SQL-Participant、TCC-Participant）※	接続先サーバのサーバ証明書を発行した CA の証明書（PEM 形式）ファイル	必ず準備します。
	自サーバ証明書（PEM 形式）ファイル	必ず準備します。
	自サーバ証明書を作成した秘密鍵ファイル	必ず準備します。
	自クライアント証明書ファイル（PEM 形式）	クライアント認証を行う場合だけ準備します。
	自クライアント証明書を作成した秘密鍵ファイル	クライアント認証を行う場合だけ準備します。
	接続元クライアント（Orchestrator、Participant）の秘密鍵が格納されているキーストア（pkcs12、または JKS）ファイル	クライアント認証を行う場合だけ準備します。

注※

Participant アプリケーションの数だけ準備してください。

HMP-PCTO の gRPC 通信機能の暗号化通信に必要なファイルを準備したディレクトリ/ファイル構成の例を次に示します。

クライアント認証を行わない場合

```
<任意のパス>/
    ca.crt              # 接続先サーバのサーバ証明書を発行したCAの証明書(PEM形式)ファイル
    participant/
        server.crt      # Participantの自サーバ証明書(PEM形式)ファイル
        server.key      # Participantの自サーバ証明書を作成した秘密鍵ファイル
```

クライアント認証を行う場合

```
<任意のパス>/
    ca.crt              # 接続先サーバのサーバ証明書を発行したCAの証明書(PEM形式)ファイル
    orchestrator/
        client.crt      # Orchestratorの自クライアント証明書ファイル(PEM形式)
        client.key      # Orchestratorの自クライアント証明書を作成した秘密鍵ファイル
    participant/
        server.crt      # Participantの自サーバ証明書(PEM形式)ファイル
        server.key      # Participantの自サーバ証明書を作成した秘密鍵ファイル
        client.crt      # Participantの自クライアント証明書ファイル(PEM形式)
        client.key      # Participantの自クライアント証明書を作成した秘密鍵ファイル
        keystore.jks    # Participantの接続元クライアント(Orchestrator,Participant)の秘密鍵
が格納されている
                        # キーストア(pkcs12またはJKS)ファイル
```

準備したファイルを使用して Orchestrator、Participant の gRPC 通信機能の暗号化通信用シークレットを作成します。

ユーザ責務の Kubernetes アプリケーションである Orchestrator、SQL-Participant、TCC-Participant の gRPC 通信機能の暗号化通信用シークレットは、ユーザ要件に合わせて作成してください。

Orchestrator、SQL-Participant、TCC-Participant の gRPC 通信機能の暗号化通信用シークレットを作成するコマンド例を次に示します。

ユーザ責務の Kubernetes アプリケーションである、Orchestrator、SQL-Participant、TCC-Participant の gRPC 通信機能の暗号化通信用シークレットの作成コマンドの一例です。ユーザ要件に合わせて作成してください。

- Orchestrator 用シークレット名：「orchestrator-secret」
- SQL-Participant、TCC-Participant 用シークレット名：「participant-secret」
- HMP-PCTO の Kubernetes アプリケーションをデプロイする Namespace：「my-namespace」

クライアント認証を行わない場合

```
kubectl create secret generic --save-config orchestrator-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  -n my-namespace
```

```
kubectl create secret generic --save-config participant-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  --from-file=server.crt=./participant/server.crt ¥
  --from-file=server.key=./participant/server.key ¥
  -n my-namespace
```

クライアント認証を行う場合

```
kubectl create secret generic --save-config orchestrator-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  --from-file=server.crt=./orchestrator/server.crt ¥
  --from-file=server.key=./orchestrator/server.key ¥
  -n my-namespace

kubectl create secret generic --save-config participant-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  --from-file=server.crt=./participant/server.crt ¥
  --from-file=server.key=./participant/server.key ¥
  --from-file=client.crt=./participant/client.crt ¥
  --from-file=client.key=./participant/client.key ¥
  --from-file=keystore=./participant/keystore.jks ¥
  --from-literal=keystore_password=changeit ¥
  -n my-namespace
```

＜コマンドの実行結果の例＞

```
secret/orchestrator-secret created

secret/participant-secret created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

(4) Mediator 選択機能用ロールおよびロールバインディングのデプロイ

「(1) 各種資材の配置」で配置した Mediator 選択機能用ロールおよびロールバインディングの Kubernetes マニフェストをデプロイしてください。

コマンド例を次に示します。

```
kubectl apply -f ./lb-role.yaml
```

この例では、Kubernetes マニフェストのファイル名を「lb-role.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
clusterrole.rbac.authorization.k8s.io/lb-role created
clusterrolebinding.rbac.authorization.k8s.io/lb-role-binding created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

(5) デプロイ依存関係チェック機能用ロールおよびロールバインディングのデプロイ

「(1) 各種資材の配置」で配置したデプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストをデプロイしてください。この作業は、次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に実施します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

詳細については、「(10) デプロイ依存関係チェック機能用ロールおよびロールバインディングのデプロイ」を参照してください。

(6) uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求のデプロイ（通常版限定）

「(1) 各種資材の配置」で配置した uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストをデプロイしてください。この作業は、次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に実施します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

詳細については、「(14) uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求のデプロイ（通常版限定）」を参照してください。

(7) ユーザ責務の Kubernetes アプリケーションのデプロイ

「(1) 各種資材の配置」で配置した各 Kubernetes アプリケーションの Kubernetes マニフェストをデプロイしてください。

なお、HMP-PCTO の各 Kubernetes アプリケーションは次の順序でデプロイする必要があります。

1. Elasticsearch
2. Logstash、Jaeger-query、Jaeger-collector、Prometheus、Metricbeat、Filebeat

3. Mediator、EADS（通常版だけ）、Ext-Cons（トライアル版だけ）、Participant が使用する DB/外部サービス
4. Participant
5. Orchestrator

上記以外の順序でデプロイした場合

各 Kubernetes アプリケーションの初期化コンテナとして構成している Dependency-Checker コンテナで、上記のとおり依存する Kubernetes アプリケーションが先にデプロイされるまで待機します。待機処理がリトライオーバーし、初期化コンテナが異常終了した場合、Pod の状態は CrashLoopBackOff になります。その場合は、Dependency-Checker コンテナが実行しているデプロイ依存関係チェックスクリプトのエラーメッセージに従い対処してください。

ユーザ責務の Kubernetes アプリケーションは、ユーザ任意の方法でデプロイしてください。

kubectl コマンドでデプロイするコマンド例を次に示します。

```
kubectl apply -f ./orchestrator.yaml
```

この例では、Kubernetes マニフェストのファイル名を「orchestrator.yaml」としています。また、各ファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
service/orchestrator created
serviceaccount/orchestrator-service-account created
deployment.apps/orchestrator created
configmap/orchestrator-config created
configmap/orchestrator-ucars-config created
configmap/orchestrator-filebeat-sidecar-config created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

HMP-PCTO の各 Kubernetes アプリケーションをデプロイしたあと、次のコマンド例のように、各 Kubernetes アプリケーションが期待どおり起動していること（STATUS 列が"Running"になっていることや、NODE 列で Pod が意図したワーカーノードに割り当てられていることなど）を確認してください。

```
kubectl get pods -n my-namespace -o wide
```

＜コマンドの実行結果の例（SQL-Participant を使用する場合）＞

```
NAMESPACE      NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
my-namespace   sql-participant-94497bdcf-7btxm     1/1     Running   0          60s   192.168.24.40   worker-1   <none>           <none>
my-namespace   orchestrator-7599cdd5c4-tk44v       1/1     Running   0          60s   192.168.24.41   worker-1   <none>           <none>
my-namespace   mediator-0                          1/1     Running   0          60s   192.168.24.42   worker-1   <none>           <none>
my-namespace   mediator-1                          1/1     Running   0          60s   192.168.24.43   worker-2   <none>           <none>
my-namespace   mediator-2                          1/1     Running   0          60s   192.168.24.44   worker-3   <none>           <none>
my-namespace   mediator-3                          1/1     Running   0          60s   192.168.24.45   worker-4   <none>           <none>
my-namespace   mediator-4                          1/1     Running   0          60s   192.168.24.46   worker-5   <none>           <none>
my-namespace   eads-server-0                       1/1     Running   0          60s   192.168.24.47   worker-1   <none>           <none>
my-namespace   eads-server-1                       1/1     Running   0          60s   192.168.24.48   worker-2   <none>           <none>
my-namespace   eads-server-2                       1/1     Running   0          60s   192.168.24.49   worker-3   <none>           <none>
my-namespace   jaeger-5cf9b67bfc-8dp7d             1/1     Running   0          60s   192.168.24.50   worker-1   <none>           <none>
my-namespace   prometheus-5fc6ddd87b-lnvgf         1/1     Running   0          60s   192.168.24.51   worker-2   <none>           <none>
my-namespace   elasticsearch-56cdb7c587-rssj2      1/1     Running   0          60s   192.168.24.52   worker-3   <none>           <none>
my-namespace   kibana-6d4b75cb6d-mj4fg             1/1     Running   0          60s   192.168.24.53   worker-4   <none>           <none>
my-namespace   filebeat-6fb89c8f89-2ncl2           1/1     Running   0          60s   192.168.24.54   worker-1   <none>           <none>
my-namespace   filebeat-6b4d774475-5q2t2           1/1     Running   0          60s   192.168.24.55   worker-2   <none>           <none>
my-namespace   filebeat-6fb89c8f89-52dgm           1/1     Running   0          60s   192.168.24.56   worker-3   <none>           <none>
my-namespace   filebeat-79989457d9-7d9kf           1/1     Running   0          60s   192.168.24.57   worker-4   <none>           <none>
my-namespace   filebeat-79989457d9-t558h           1/1     Running   0          60s   192.168.24.58   worker-5   <none>           <none>
```

＜コマンドの実行結果の例（TCC-Participant を使用する場合）＞

```
NAMESPACE      NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
my-namespace   tcc-participant-94497bdcf-7btxm     1/1     Running   0          60s   192.168.24.40   worker-1   <none>           <none>
my-namespace   orchestrator-7599cdd5c4-tk44v       1/1     Running   0          60s   192.168.24.41   worker-1   <none>           <none>
my-namespace   mediator-0                          1/1     Running   0          60s   192.168.24.42   worker-1   <none>           <none>
my-namespace   mediator-1                          1/1     Running   0          60s   192.168.24.43   worker-2   <none>           <none>
my-namespace   mediator-2                          1/1     Running   0          60s   192.168.24.44   worker-3   <none>           <none>
my-namespace   mediator-3                          1/1     Running   0          60s   192.168.24.45   worker-4   <none>           <none>
my-namespace   mediator-4                          1/1     Running   0          60s   192.168.24.46   worker-5   <none>           <none>
my-namespace   eads-server-0                       1/1     Running   0          60s   192.168.24.47   worker-1   <none>           <none>
my-namespace   eads-server-1                       1/1     Running   0          60s   192.168.24.48   worker-2   <none>           <none>
my-namespace   eads-server-2                       1/1     Running   0          60s   192.168.24.49   worker-3   <none>           <none>
my-namespace   jaeger-5cf9b67bfc-8dp7d             1/1     Running   0          60s   192.168.24.50   worker-1   <none>           <none>
my-namespace   prometheus-5fc6ddd87b-lnvgf         1/1     Running   0          60s   192.168.24.51   worker-2   <none>           <none>
my-namespace   elasticsearch-56cdb7c587-rssj2      1/1     Running   0          60s   192.168.24.52   worker-3   <none>           <none>
my-namespace   kibana-6d4b75cb6d-mj4fg             1/1     Running   0          60s   192.168.24.53   worker-4   <none>           <none>
my-namespace   filebeat-6fb89c8f89-2ncl2           1/1     Running   0          60s   192.168.24.54   worker-1   <none>           <none>
my-namespace   filebeat-6b4d774475-5q2t2           1/1     Running   0          60s   192.168.24.55   worker-2   <none>           <none>
my-namespace   filebeat-6fb89c8f89-52dgm           1/1     Running   0          60s   192.168.24.56   worker-3   <none>           <none>
my-namespace   filebeat-79989457d9-7d9kf           1/1     Running   0          60s   192.168.24.57   worker-4   <none>           <none>
my-namespace   filebeat-79989457d9-t558h           1/1     Running   0          60s   192.168.24.58   worker-5   <none>           <none>
```

なお、各列に表示される値は実行環境によって異なります。また、kubectl の describe サブコマンドを使用すると、各 Kubernetes オブジェクトの詳細な情報を参照できます。

(8) ユーザ責務の Kubernetes アプリケーションのアンデプロイ

デプロイした HMP-PCTO の各 Kubernetes アプリケーションをアンデプロイする方法について説明します。

なお、HMP-PCTO の各 Kubernetes アプリケーションは、次の順序でアンデプロイする必要があります。

1. Orchestrator
2. Participant
3. Mediator、EADS（通常版だけ）、Ext-Cons（トライアル版だけ）、Participant が使用する DB/外部サービス
4. Logstash、Jaeger-query、Jaeger-collector、Prometheus、Metricbeat、Filebeat

5. Elasticsearch

> **重要**
>
> 上記以外の順序でアンデプロイした場合に発生する問題
>
> 上記以外の順序でアンデプロイした場合、トラブルシュートに必要な情報が収集できなかったり、処理中のトランザクションが適切に終了しなかったりなどの問題が発生するおそれがあります。

ユーザ責務の Kubernetes アプリケーションは、ユーザ任意の方法でアンデプロイしてください。

kubectl コマンドでアンデプロイするコマンド例を次に示します。

```
kubectl delete -f ./orchestrator.yaml
```

> **重要**
>
> 「--grace-period=0 --force」オプションの指定について
>
> Kubernetes オブジェクトをすぐに削除する「--grace-period=0 --force」オプションは指定しないでください。このオプションを指定して Kubernetes オブジェクトを削除した場合、トランザクション結果が不整合になるおそれがあります。

この例では、Kubernetes マニフェストのファイル名を「orchestrator.yaml」としています。また、各ファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけアンデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
service/orchestrator deleted
serviceaccount/orchestrator-service-account deleted
deployment.apps/orchestrator deleted
configmap/orchestrator-config deleted
configmap/orchestrator-ucars-config deleted
configmap/orchestrator-filebeat-sidecar-config deleted
```

このように、各 Kubernetes オブジェクトの削除に成功したメッセージが出力されていることを確認してください。

HMP-PCTO の各 Kubernetes アプリケーションをアンデプロイしたあと、次のコマンド例のように、各 Kubernetes アプリケーションが期待どおり停止していること（一覧に出力されないこと）を確認してください。

```
kubectl get pods -n my-namespace -o wide
```

＜コマンドの実行結果の例＞

```
No resources found in my-namespace namespace.
```

3 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイ

3.1 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイの流れ

HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションは次の流れでデプロイします。

表 3-1 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイの流れ

項番	項目	説明	参照先
1	Mediator のビルド	HMP-PCTO の提供物を使用して Mediator をビルドします。通常版の場合だけ実施します。	3.2 Mediator のビルド（通常版限定）
2	TP1-Bridge のビルド（通常版かつ TP1-Bridge 限定）	HMP-PCTO の提供物を使用して TP1-Bridge をビルドします。通常版の場合で TP1-Bridge を使用するときだけ実施します。	3.3 TP1-Bridge のビルド（通常版かつ TP1-Bridge 限定）
3	HMP-PCTO のコントロールプレーンのコンテナイメージ作成	コンテナイメージは、HMP-PCTO が提供する Dockerfile を基にビルドします。通常版の場合だけ実施します。	3.4 コンテナイメージ作成（通常版限定）
4	共有ファイルシステムでの作業	共有ファイルシステムで、HMP-PCTO で必要なディレクトリを用意します。	3.5 共有ファイルシステムでの作業
5	Kubernetes マニフェスト作成	HMP-PCTO が提供する Kubernetes マニフェストの記述例を基に Kubernetes マニフェストを作成します。	3.6 Kubernetes マニフェスト作成
6	Helm チャートのカスタマイズ	HMP-PCTO が提供する Helm チャートのカスタマイズ例を基に Helm チャートをカスタマイズします。	3.7 Helm チャートのカスタマイズ
7	デプロイ	Kubernetes マニフェストまたは Helm を使用して Kubernetes クラスタにデプロイします。	3.8 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションの運用

重要

トライアル版 HMP-PCTO を使用して構築した環境は、通常版 HMP-PCTO にそのまま移行できません。

トライアル版 HMP-PCTO を使用してデプロイまで完了している場合に、通常版 HMP-PCTO を使用して環境を構築したいときは、「(20) Kubernetes アプリケーションのアンデプロイ」を実施したあと、「HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイの流れ」の手順を 1 から実施してください。各手順で通常版、トライアル版で説明が分かれている場合は通常版の手順で実施してください。

3.2　Mediator のビルド（通常版限定）

HMP-PCTO ライブラリを使用した Mediator のビルドについて説明します。

Mediator のビルドは、通常版の場合だけ実施します。

Java アプリケーションの開発環境のビルド自動化システムには Gradle を使用する前提で説明します。説明中の Gradle の構文例は、Gradle7.3.3 を使用した場合を例に記述しています。

3.2.1　作業項目の一覧（通常版限定）

Mediator のビルドの作業項目の一覧を次の表に示します。

表 3-2　Mediator のビルドの作業項目の一覧

分類	作業項目	備考	参照先
共通	事前準備（通常版限定）	通常版の場合だけ実施します。	3.2.2　事前準備（通常版限定）
	ライブラリのセットアップ（通常版限定）	通常版の場合だけ実施します。	3.2.3　ライブラリのセットアップ（通常版限定）
Mediator のビルド	JAR ファイルのビルド（通常版限定）	通常版の場合だけ実施します。	3.2.4　JAR ファイルのビルド（通常版限定）

3.2.2　事前準備（通常版限定）

HMP-PCTO ライブラリを使用したアプリケーションの開発は、Gradle（バージョン 7.3.3 以上）の利用を前提としています。これらのソフトウェアを開発環境にインストールしてください。

3.2.3　ライブラリのセットアップ（通常版限定）

(1)　HMP-PCTO ライブラリ

HMP-PCTO の提供媒体の HMP-PCTO ライブラリ格納フォルダを、アプリケーション開発環境のローカル Maven リポジトリにコピーします。

- HMP-PCTO ライブラリ格納フォルダ（コピー元）：
  <HMP-PCTO 提供媒体のルートフォルダ>¥libs の下のフォルダ
- ローカル Maven リポジトリ（コピー先）：

%USERPROFILE%¥.m2¥repository

これはデフォルトのローカル Maven リポジトリです。デフォルトからパスを変更している場合は、変更先のローカル Maven リポジトリをコピー先としてください。

build.gradle には、Maven ローカルリポジトリを参照するための記述を追加してください。

コーディング例

```
repositories {
    mavenLocal()
    mavenCentral()
}
```

HMP-PCTO ライブラリを格納する Maven リポジトリとして、ローカル Maven リポジトリ以外の任意の Maven リポジトリを利用することもできます。

(2) 依存ライブラリ

HMP-PCTO が依存するライブラリのうち HMP-ADIF および TP1/Client/J のライブラリ以外は、HMP-PCTO ライブラリを組み込むと Gradle がダウンロードします。ダウンロードはインターネット接続環境で行う必要があります。もし、インターネット接続のない閉塞環境で開発を行う場合は、一度インターネット接続環境でそれらのダウンロードを実行したあと、ダウンロード先であるローカル Maven リポジトリおよび Gradle キャッシュを、フォルダごと閉塞環境の同じ場所にコピーしてください。

HMP-ADIF と連携する場合は、HMP-ADIF のライブラリのセットアップが必要です。HMP-ADIF のライブラリのセットアップ方法は、HMP-ADIF の取扱説明書を参照してください。

また、TP1 連携機能を使用する場合は、TP1/Client/J のインストールが必要です。TP1/Client/J のインストール方法は、マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」を参照してください。

3.2.4 JAR ファイルのビルド（通常版限定）

「HMP-PCTO の Mediator アプリケーションの JAR ファイル（mediator-*V.R.S*※.jar)」は、HMP-PCTO の提供物を使用してビルドします。

HMP-PCTO が提供する「Mediator 用 Java プロジェクトディレクトリ」(java-projects/mediator）で、次に示す Gradle のコマンドを実行して JAR ファイルをビルドしてください。

```
cd java-projects/mediator
gradle build
```

ビルドに成功すると、次に示すパスに「HMP-PCTO の Mediator アプリケーションの JAR ファイル (mediator-*V.R.S*※.jar)」が生成されます。

```
java-projects/mediator/build/lib/mediator-V.R.S※.jar
```

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

3.3　TP1-Bridge のビルド（通常版かつ TP1-Bridge 限定）

HMP-PCTO ライブラリを使用した TP1-Bridge のビルドについて説明します。

TP1-Bridge のビルドは、通常版の場合で、TP1-Bridge を使用するときだけ実施します。

Java アプリケーションの開発環境のビルド自動化システムには Gradle を使用する前提で説明します。説明中の Gradle の構文例は、Gradle7.3.3 を使用した場合を例に記述しています。

3.3.1　作業項目の一覧（通常版かつ TP1-Bridge 限定）

TP1-Bridge のビルドの作業項目の一覧を次の表に示します。

表 3-3　TP1-Bridge のビルドの作業項目の一覧

分類	作業項目	備考	参照先
共通	事前準備（通常版かつ TP1-Bridge 限定）	通常版の場合だけ実施します。	3.3.2　事前準備（通常版かつ TP1-Bridge 限定）
	ライブラリのセットアップ（通常版かつ TP1-Bridge 限定）	通常版の場合だけ実施します。	3.3.3　ライブラリのセットアップ（通常版かつ TP1-Bridge 限定）
TP1-Bridge のビルド	JAR ファイルのビルド（通常版かつ TP1-Bridge 限定）	通常版の場合だけ実施します。	3.3.4　JAR ファイルのビルド（通常版かつ TP1-Bridge 限定）

3.3.2　事前準備（通常版かつ TP1-Bridge 限定）

HMP-PCTO ライブラリを使用したアプリケーションの開発は、Gradle（バージョン 7.3.3 以上）の利用を前提としています。これらのソフトウェアを開発環境にインストールしてください。

3.3.3　ライブラリのセットアップ（通常版かつ TP1-Bridge 限定）

ライブラリのセットアップ方法は、Mediator の場合と同じです。「3.2.3　ライブラリのセットアップ（通常版限定）」を参照してください。

3.3.4 JAR ファイルのビルド（通常版かつ TP1-Bridge 限定）

「HMP-PCTO の TP1-Bridge アプリケーションの JAR ファイル（tp1-bridge-*V.R.S*※.jar）」は、HMP-PCTO の提供物を使用してビルドします。

HMP-PCTO が提供する「TP1-Bridge 用外部ライブラリ格納ディレクトリ」（java-projects/tp1-bridge/libs）に、TP1/Client/J をインストールした環境から TP1Client.jar をコピーしてください。そのあと、「TP1-Bridge 用 Java プロジェクトディレクトリ」（java-projects/tp1-bridge）で、次に示す Gradle のコマンドを実行して JAR ファイルをビルドしてください。

```
cp TP1Client.jar java-projects/tp1-bridge/libs
cd java-projects/tp1-bridge
gradle build
```

ビルドに成功すると、次に示すパスに「HMP-PCTO の TP1-Bridge アプリケーションの JAR ファイル（tp1-brdige-*V.R.S*※.jar）」が生成されます。

```
java-projects/tp1-bridge/build/lib/tp1-bridge-V.R.S※.jar
```

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

3.4　コンテナイメージ作成（通常版限定）

ここでは、HMP-PCTO のコントロールプレーンのコンテナイメージ作成方法について説明します。HMP-PCTO のコントロールプレーンのコンテナイメージは、通常版の場合だけ作成します。

コンテナイメージビルド環境のコンテナエンジンはユーザ任意のものを使用してください。説明中のコンテナエンジンのコマンド例は、Docker を使用した場合を例に記述しています。コンテナエンジンの各コマンドの詳細については、使用するコンテナエンジンのマニュアルなどを参照してください。

3.4.1　作業項目の一覧（通常版限定）

コンテナイメージ作成の作業項目の一覧を次の表に示します。

表 3-4　コンテナイメージ作成の作業項目の一覧

作業項目	備考	参照先
事前準備（通常版限定）	通常版の場合だけ実施します。	3.4.2　事前準備（通常版限定）
Mediator コンテナイメージの作成（通常版限定）	通常版の場合だけ作成します。トライアル版の場合は提供ファイルに含まれるコンテナイメージを使用します。	3.4.3　Mediator コンテナイメージの作成（通常版限定）
TP1-Bridge コンテナイメージの作成（通常版かつ TP1-Bridge 限定）	通常版の場合で、TP1-Bridge を使用するときだけ作成します。	3.4.4　TP1-Bridge コンテナイメージの作成（通常版かつ TP1-Bridge 限定）
Dependency-Checker コンテナイメージの作成（通常版限定）	通常版の場合だけ作成します。トライアル版の場合は提供ファイルに含まれるコンテナイメージを使用します。	3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）
EADS サーバ用コンテナイメージの作成（通常版限定）	通常版の場合だけ作成します。	3.4.6　EADS サーバ用コンテナイメージの作成（通常版限定）
EADS コマンド実行用コンテナイメージの作成（通常版限定）	通常版の場合だけ作成します。	3.4.7　EADS コマンド実行用コンテナイメージの作成（通常版限定）

3.4.2　事前準備（通常版限定）

コンテナイメージを登録するためのコンテナレジストリ、およびコンテナイメージをビルドするための Docker などのコンテナエンジンをインストールしたコンテナイメージビルド環境を用意してください。

また、以降に示す Containerfile の記述例、および HMP-PCTO が提供する Containerfile のテンプレートは、次のすべての設定が完了していることを前提としています。

- ホストは RHEL を想定※

- コンテナ内のインターネット接続
- プロキシ設定
- RPM リポジトリ設定
- コンテナレジストリ設定

注※

ベースイメージに UBI を使用するとき、ホスト RHEL のサブスクリプション情報を共有されることで、UBI コンテナ内から RHEL のリポジトリが利用できるようになります。なお、RHEL のバージョンは UBI と一致している必要はありません。

これらの設定が完了している場合は、コンテナベースイメージのプル、およびコンテナ内で dnf コマンドなどによる RPM パッケージのオンラインインストールができます。設定が完了していない場合は、先に設定を完了させてください。または、Containerfile を作成する際に dnf コマンドなどによるインストールの設定を組み込んでください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、一度インターネット接続環境でダウンロードした資材をコンテナイメージビルド環境にコピーしたりするなどの方法で対応してください。

3.4.3 Mediator コンテナイメージの作成（通常版限定）

Mediator コンテナイメージの作成方法について説明します。

このコンテナイメージは通常版の場合だけ作成します。トライアル版の場合は提供ファイルに含まれるコンテナイメージを使用します。

(1) Containerfile のテンプレートの修正

HMP-PCTO が提供する Containerfile のテンプレートについて、次の修正をしてください。

- 「3.4.2 事前準備（通常版限定）」に示すとおり、ユーザの開発環境に合わせて dnf コマンドなどによるオンラインインストールの設定を組み込むなどの方法で修正してください。
- オレンジ色の背景で示す個所に、uCosminexus Application Runtime with Java for Spring Boot のリリースノートを参照し、uCosminexus Application Runtime with Java for Spring Boot に必要な前提 RPM パッケージを記述してください。※1

uCosminexus Application Runtime with Java for Spring Boot の修正パッチがない場合の Containerfile のテンプレートの内容を次に記載しています。Containerfile のテンプレートに対する修正内容は、uCosminexus Application Runtime with Java for Spring Boot の修正パッチがない場合と修正パッチがある場合のどちらも同じです。

```
# All Rights Reserved. Copyright (C) 2023, Hitachi, Ltd.
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860 AS builder
```

```
RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        ncompress ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN
COPY X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HL0T ./

FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860
COPY --from=builder /opt /opt
COPY --from=builder /etc/.java /etc/.java
RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        dnf ¥
    && dnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        --allowerasing ¥
        curl-7.61.1 ¥
        jq-1.6 ¥
        glibc-2.28 ¥
        <uCosminexus Application Runtime with Java for Spring Bootの前提RPMパッケージ>※1
        :
    && dnf clean all ¥
    && rm -rf /var/cache/dnf/*
ENV PATH=/opt/Cosminexus/jdk/bin:$PATH
USER 1001:0
WORKDIR /deployments
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/
COPY --chown=1001:0 --chmod=0440 ./libhmppcto-mediator-V.R.S※2.so /deployments/
COPY --chown=1001:0 --chmod=0440 ./mediator-V.R.S※2.jar /deployments/
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/mediato
r-V.R.S※2.jar
```

注※1

オレンジ色の背景で示す個所以外の内容は、HMP-PCTO が提供する Containerfile のテンプレートの実際の内容と多少異なる場合があります。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 3-5 Containerfile と資材の配置構成

ファイル／ディレクトリ構成	作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/	ユーザ	－
Containerfile	ユーザ	■uCosminexus Application Runtime with Java for Spring Boot の修正パッチがない場合 HMP-PCTO の提供ファイル「templates/container-files/mediator/normal/Containerfile」を使用してください。 ■uCosminexus Application Runtime with Java for Spring Boot の修正パッチがある場合 HMP-PCTO の提供ファイル「templates/container-files/mediator/patch/Containerfile」を使用してください。
mediator-V.R.S.jar (HMP-PCTO の Mediator アプリケーションの JAR ファイル)	ユーザ	「3.2.4 JAR ファイルのビルド（通常版限定）」で作成したもの
libhmppcto-mediator-V.R.S.so (HMP-PCTO の Mediator アプリケーションのダイレクト IO 用ライブラリファイル)	日立	－
health_check.sh (HMP-PCTO のトランザクションヘルスチェックスクリプト)	日立	－
wait_transactions.sh (HMP-PCTO の未決着トランザクション待機スクリプト)	日立	－
X64LIN (uCosminexus Application Runtime with Java for Spring Boot のインストーラ)	日立	uCosminexus Application Runtime with Java for Spring Boot のインストール CD-ROM に格納されている X64LIN ディレクトリをコピーしてください。
PACK_TAR.Z (uCosminexus Application Runtime with Java for Spring Boot の修正パッチ)	日立	uCosminexus Application Runtime with Java for Spring Boot の修正パッチを適用する場合だけ格納してください。

（凡例）－：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

Mediator コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]mediator:<HMP-PCTOバージョン(V.R.S※)>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/mediator:V.R.S※ .
```

この例では、HMP-PCTO バージョンを「*V.R.S*※」としています。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした Mediator コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

3.4.4 TP1-Bridge コンテナイメージの作成（通常版かつ TP1-Bridge 限定）

TP1-Bridge コンテナイメージの作成方法について説明します。

このコンテナイメージは通常版の場合で、TP1-Bridge を使用するときだけ作成します。

(1) Containerfile のテンプレートの修正

uCosminexus Application Runtime with Java for Spring Boot の修正パッチがあるかどうかを確認して、使用する HMP-PCTO が提供する TP1-Bridge 用 Containerfile のテンプレートを選択してください。選択したテンプレートに、次の修正をしてください。

- 「3.4.2 事前準備（通常版限定）」に示すとおり、ユーザの開発環境に合わせて dnf コマンドなどによるオンラインインストールの設定を組み込むなどの方法で修正してください。

uCosminexus Application Runtime with Java for Spring Boot の修正パッチがある場合の Containerfile のテンプレートの例を、次に示します。

```
FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860 AS builder
RUN microdnf install ¥
```

```
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        ncompress ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN
COPY ucars/X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HL0T ./
RUN mkdir patch
WORKDIR patch
COPY PACK_TAR.Z .
RUN tar xzvf PACK_TAR.Z && ./UPDATE -f; export RET=$?; if [ ${RET} -ne 0 -a ${RET} -ne 1 ];
then exit ${RET}; fi

FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860 AS ucars
COPY --from=builder /opt /opt
COPY --from=builder /etc/.java /etc/.java
RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        dnf ¥
    && dnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        --allowerasing ¥
        curl-7.61.1 ¥
        jq-1.6 ¥
        glibc-2.28 ¥
    && dnf clean all ¥
    && rm -rf /var/cache/dnf/*
ENV PATH=/opt/Cosminexus/jdk/bin:$PATH
USER 1001:0

FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860 AS ucars_eads
RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        dnf ¥
    && dnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        --allowerasing ¥
        curl-7.61.1 ¥
        jq-1.6 ¥
        glibc-2.28 ¥
    && dnf clean all ¥
    && rm -rf /var/cache/dnf/*

RUN microdnf install ¥
```

```
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        ncompress ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN
COPY /X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HC13 ./
RUN mkdir patch
WORKDIR patch
COPY PACK_TAR.Z .
RUN tar xzvf PACK_TAR.Z && ./UPDATE -f; export RET=$?; if [ ${RET} -ne 0 -a ${RET} -ne 1 ];
then exit ${RET}; fi

FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8-860
RUN mkdir -p /deployments/eads-lib ¥
    && mkdir -p /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log ¥
    && mkdir -p /deployments/tp1-log ¥
    && chown 1001:0 /deployments/tp1-log ¥
    && chmod 770 /deployments/tp1-log
ENV LOADER_PATH=/deployments/eads-lib
ENV PATH=/opt/Cosminexus/jdk/bin:$PATH
COPY --from=ucars /opt /opt
COPY --from=ucars /etc/.java /etc/.java
RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        dnf ¥
    && dnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        --allowerasing ¥
        curl-7.61.1 ¥
        jq-1.6 ¥
        glibc-2.28 ¥
    && dnf clean all ¥
    && rm -rf /var/cache/dnf/*
WORKDIR /deployments
COPY --from=ucars_eads --chown=1001:0 --chmod=0440 /opt/hitachi/xeads/javaclient/lib/ /deplo
yments/eads-lib
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/
COPY --chown=1001:0 --chmod=0440 ./tp1-bridge-V.R.S※.jar /deployments/
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/tp1-bri
dge-V.R.S※.jar
```

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 3-6 Containerfile と資材の配置構成

ファイル／ディレクトリ構成	作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/	ユーザ	－
Containerfile	ユーザ	－
tp1-bridgeV.R.S.jar (HMP-PCTO の TP1-Bridge アプリケーションの JAR ファイル)	ユーザ	「3.3.4 JAR ファイルのビルド（通常版かつ TP1-Bridge 限定)」で作成したもの
health_check.sh (HMP-PCTO のトランザクションヘルスチェックスクリプト)	日立	－
wait_transactions.sh (HMP-PCTO の未決着トランザクション待機スクリプト)	日立	－
X64LIN (Hitachi Elastic Application Data Store Client for Java のインストーラ)	日立	HMP-PCTO に同梱されている Hitachi Elastic Application Data Store Client for Java のインストーラの X64LIN ディレクトリをコピーしてください。
PACK_TAR.Z (Hitachi Elastic Application Data Store Client for Java の修正パッチ)	日立	HMP-PCTO に Hitachi Elastic Application Data Store Client for Java の修正パッチが同梱されている場合だけ格納してください。

（凡例）－：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

TP1-Bridge コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]<TP1-Bridgeコンテナイメージのユーザ任意のイメージ名>:<ユーザ任意のタグ名>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

この例では、「TP1-Bridge コンテナイメージのユーザ任意のイメージ名:ユーザ任意のタグ名」を「tp1-bridge:*V.R.S*※」としています。

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/tp1-bridge:V.R.S※ .
```

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした TP1-Bridge コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

3.4.5 Dependency-Checker コンテナイメージの作成（通常版限定）

Dependency-Checker コンテナイメージの作成方法について説明します。

このコンテナイメージは通常版の場合だけ作成します。トライアル版の場合は提供ファイルに含まれるコンテナイメージを使用します。

(1) Containerfile のテンプレートの修正

HMP-PCTO が提供する Containerfile のテンプレートについて、「3.4.2 事前準備（通常版限定）」に示すとおり、ユーザの開発環境に合わせて dnf コマンドなどによるオンラインインストールの設定を組み込むなどの方法で修正してください。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 3-7 Containerfile と資材の配置構成

ファイル／ディレクトリ構成		作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/		ユーザ	−
	Containerfile	ユーザ	−
	dependency_check.sh (HMP-PCTO のデプロイ依存関係チェックスクリプト)	日立	−

（凡例）－：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

Dependency-Checker コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]dependency-checke
r:<HMP-PCTOバージョン(V.R.S※)>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/dependency-checker:V.R.S
※ .
```

この例では、HMP-PCTO バージョンを「*V.R.S※*」としています。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした Dependency-Checker コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

3.4.6 EADS サーバ用コンテナイメージの作成（通常版限定）

EADS サーバ用コンテナイメージの作成方法について説明します。

このコンテナイメージは通常版の場合だけ作成します。

(1) Containerfile のテンプレートの修正

Containerfile のテンプレートについて、「3.4.2 事前準備（通常版限定）」に示すとおり、ユーザの開発環境に合わせて dnf コマンドなどによるオンラインインストールの設定を組み込むなどの方法で修正してください。また、後述する<コンテナイメージビルド作業ディレクトリ>/main-image ディレクトリに配

置した Containerfile のテンプレート内の FROM 句に記載された内容をコンテナビルド時に指定する文字列へ変更してください。

- 変更前：FROM localhost/eads-server-base-image:latest
- 変更後：FROM <コンテナレジストリ>/[<namespace>/]eads-server-base-image:< EADS バージョン(VV-RR(-SS)※)>

注※

VV-RR(-SS)は使用する EADS のバージョンに合わせて読み替えてください。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 3-8 Containerfile と資材の配置構成

ファイル／ディレクトリ構成	作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>/base-image	ユーザ	－
Containerfile	ユーザ	EADS サーバをインストールした環境から下記ディレクトリをコピーして、base-image にリネームします。 EADS サーバの修正パッチが発行されていない場合： /opt/hitachi/xeads/container/normal EADS サーバの修正パッチが発行されている場合： /opt/hitachi/xeads/container/patch
X64LIN (Hitachi Elastic Application Data Store のインストーラ)	日立	Hitachi Elastic Application Data Store をインストールした環境からインストール CD-ROM に格納されている X64LIN ディレクトリをコピーしてください。
PACK_TAR.Z (Hitachi Elastic Application Data Store の修正パッチ)	日立	Hitachi Elastic Application Data Store の修正パッチを適用する場合だけ格納してください。
<コンテナイメージビルド作業ディレクトリ>/main-image	ユーザ	－
Containerfile	ユーザ	EADS サーバをインストールした環境から次のディレクトリをコピーして、main-image にリネームします。 /opt/hitachi/xeads/orchestration/container/server
payload.tgz	日立	EADS サーバをインストールした環境から次のディレクトリをコピーして、main-image にリネームします。 /opt/hitachi/xeads/orchestration/container/server

（凡例）－：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

EADS サーバ用コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/base-image
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]eads-server-base-image:<EADSバージョン(VV-RR(-SS)※)>△.

cd <コンテナイメージビルド作業ディレクトリ>/main-image
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]eads-server-image:<EADSバージョン(VV-RR(-SS)※)>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/base-image
docker build -f ./Containerfile -t myregistry.example.com/myproject/eads-server-base-image:VV-RR(-SS)※ .

cd build/main-image
docker build -f ./Containerfile -t myregistry.example.com/myproject/eads-server-image:VV-RR(-SS)※ .
```

この例では、EADS バージョンを「VV-RR(-SS)※」としています。

注※

VV-RR(-SS)は使用する EADS のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした EADS サーバ用コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

3.4.7 EADS コマンド実行用コンテナイメージの作成（通常版限定）

EADS コマンド実行用コンテナイメージの作成方法について説明します。

このコンテナイメージは通常版の場合だけ作成します。

(1) Containerfile のテンプレートの修正

Containerfile のテンプレートについて、「3.4.2 事前準備（通常版限定）」に示すとおり、ユーザの開発環境に合わせて dnf コマンドなどによるオンラインインストールの設定を組み込むなどの方法で修正してください。

(2) Containerfile と資材の配置

次の表に示すとおり、Containerfile と資材をコンテナイメージビルド環境に配置してください。

表 3-9 Containerfile と資材の配置構成

ファイル／ディレクトリ構成	作成・提供元	備考
<コンテナイメージビルド作業ディレクトリ>	ユーザ	－
Containerfile	ユーザ	EADS サーバをインストールした環境から次のディレクトリをコピーします。 /opt/hitachi/xeads/orchestration/container/command
payload.tgz	日立	EADS サーバをインストールした環境から次のディレクトリをコピーします。 /opt/hitachi/xeads/orchestration/container/command
oc コマンドおよび kubectl コマンド	ユーザ	OpenShift ダウンロードページから"OpenShift command-line interface(oc)"でダウンロードできる oc コマンドおよび kubectl コマンドを配置してください。

（凡例）－：該当しません。

(3) コンテナイメージのビルド

Containerfile を使用して、コンテナイメージをビルドします。

EADS コマンド実行用コンテナイメージのビルドのコマンドを次に示します。

```
cd <コンテナイメージビルド作業ディレクトリ>/
docker△build△-f△./Containerfile△-t△<コンテナレジストリ>/[<namespace>/]eads-command-image:<EADSバージョン(VV-RR(-SS)※)>△.
```

（凡例）△：半角スペース 1 文字

＜実行例＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/eads-command-image:VV-RR(-SS)※ .
```

この例では、EADS バージョンを「VV-RR(-SS)※」としています。

注※

VV-RR(-SS)は使用する EADS のバージョンに合わせて読み替えてください。

(4) コンテナイメージの登録

ビルドした EADS コマンド実行用コンテナイメージを、Kubernetes クラスタで使用するコンテナレジストリに docker push コマンドなどで登録してください。

インターネット接続のない閉塞環境の場合は、ローカルのリポジトリ・レジストリなどを使用したり、コンテナイメージビルド環境でエクスポートしたコンテナイメージを Kubernetes クラスタの各ワーカーノードにインポートしたりするなどの方法で対応してください。

3.5　共有ファイルシステムでの作業

HMP-PCTO が前提とする共有ファイルシステムでの作業について説明します。

共有ファイルシステムでの作業の詳細については、OS、各製品のマニュアルなどを参照してください。

3.5.1　作業項目の一覧

HMP-PCTO が前提とする共有ファイルシステムでの作業項目の一覧を次の表に示します。

表 3-10　HMP-PCTO が前提とする共有ファイルシステムでの作業項目の一覧

作業項目	備考	参照先
Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（NFS ストレージとする場合）	Mediator の ConsensusLog のマウント用ディレクトリを NFS ストレージとする場合は、必ず作成します。	3.5.2　Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（NFS ストレージとする場合）
Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリの作成	必ず作成します。	3.5.3　Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリの作成
uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定）	通常版の場合だけ作成します。	3.5.4　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定）

3.5.2　Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（NFS ストレージとする場合）

Mediator の ConsensusLog のマウント用ディレクトリを NFS ストレージとする場合は、次の作業を実施してください。

Mediator が存在する各ワーカーノードでアクセスできる NFS に、Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリを次の要領で作成してください。

Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成要領

- ディレクトリのグループに GID「0」（root）を設定してください。
- ディレクトリのパーミッションにグループの書き込み権限を付与してください。
- ディレクトリ名に制限はありません。

なお、「3.6.2　Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」の例では、ディレクトリ名を「/mediator-consensus-log-volume」としています。

- GID「0」（root）で書き込みできる状態で、NFS でディレクトリを公開してください。

ボリュームマウント用ディレクトリ内に、すべての Mediator の ConsensusLog ファイルが格納されます。

1 つの NFS サーバ上に、すべての Mediator の ConsensusLog ファイルを配置する場合、NFS サーバが単一障害点となるため、必要な稼働率を確保する必要があります。

パブリッククラウドのサービス（Azure Blob Storage、Amazon EFS など）を利用した冗長化を推奨します。パブリッククラウドのサービスを利用しない場合は、システムごとに独自に NFS サーバを冗長化することを検討してください。

3.5.3　Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリの作成

Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリを、次の要領で NFS やクラウドストレージなどの全ワーカーノードで共有可能な場所に作成してください。

Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリの作成要領

- ディレクトリのグループに GID「0」（root）を設定してください。
- ディレクトリのパーミッションにグループの書き込み権限を付与してください。
- GID「0」（root）で書き込みできる状態で、NFS でディレクトリを公開してください。

3.5.4　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定）

uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリを、次の要領で全ワーカーノードで共有可能な NFS に作成してください。この作業は、次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に実施します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリの作成要領

- ディレクトリのグループに GID「0」（root）を設定してください。

- ディレクトリのパーミッションにグループの書き込み権限を付与してください。
- GID「0」（root）で書き込みできる状態で、NFS でディレクトリを公開してください。

3.6 Kubernetes マニフェスト作成

HMP-PCTO の Kubernetes マニフェスト作成方法について説明します。

Kubernetes マニフェストの記述仕様の詳細は、Kubernetes のマニュアルなどを参照してください。

Kubernetes マニフェスト作成時のログおよびトラブルシュート情報の取得の設定については、「3.9.1 ログおよびトラブルシュート情報の取得の設定」を参照してください。

3.6.1 作業項目の一覧

Kubernetes マニフェスト作成の作業項目の一覧を次の表に示します。

表 3-11 Kubernetes マニフェスト作成の作業項目の一覧

作業項目	備考	参照先
Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成	必ず作成します。	3.6.2 Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成
Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成	必ず作成します。	3.6.3 Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成
Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成	必ず作成します。	3.6.4 Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成
EADS の永続ボリュームの Kubernetes マニフェストの作成（通常版限定）	通常版の場合だけ作成します。	3.6.5 EADS の永続ボリュームの Kubernetes マニフェストの作成（通常版限定）
デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成	次のどちらかの Kubernetes アプリケーションの Kubernetes マニフェスト作成時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	3.6.6 デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成
コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成	必ず作成します。	3.6.7 コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成

作業項目	備考	参照先
メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成	必ず作成します。	3.6.8　メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成
メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストの作成	必ず作成します。	3.6.9　メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストの作成
uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）	通常版の場合だけ作成します。次のどちらかの Kubernetes アプリケーションの Kubernetes マニフェスト作成時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	3.6.10　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）

3.6.2　Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成

「6.4.5　Mediator の要件」に記載されている Mediator の ConsensusLog の永続ボリュームの要件を満たすように、Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストを作成してください。

永続ボリュームの種類を選択する判断基準の例を示します。

- Blue/Green アップデートのように、Mediator の Pod をデプロイするノードを固定したくない場合は、NFS ストレージを選択してください。
- NFS サーバを用意しないで、最小限の構成としたい場合は、ローカル永続ボリューム（local）を選択してください。
- ローカル永続ボリューム（local）を選択すると、NFS ストレージよりも永続ボリュームへアクセスする性能が高くなります。
- ローカル永続ボリューム（local）を選択すると、ノードが障害になったときの稼働率が安定します。Mediator の Pod をデプロイするノードを固定することで、Mediator をノードに均等に配置するためです。

永続ボリュームの種類がローカル永続ボリューム（local）の場合は、Mediator で使用する永続ボリュームの種類を「ローカル永続ボリューム（local)」とし、Kubernetes の nodeAffinity で永続ボリュームの割り当てノードを固定することで、その永続ボリュームを使用する Mediator の Pod は Kubernetes スケジューラによって必ず永続ボリュームと同一ノードに割り当てられるようになります。これによって、Mediator の Pod が必ずノードにアタッチされたローカルストレージデバイスを使用できるようにします。

また、Mediator 専用のストレージクラスを作成することで、Mediator の ConsensusLog の永続ボリュームと Mediator の ConsensusLog の永続ボリュームクレームのバインドを必ず一対一になるようにします。

永続ボリュームの種類が NFS ストレージの場合は、Mediator で使用する永続ボリュームの種類を「NFS ストレージ」とし、Kubernetes の nodeAffinity で永続ボリュームの割り当てノードを固定しないことで、Mediator の Pod を不特定のノード上に割り当てます。

Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成要領を次に示します。

＜ストレージクラスの作成要領＞

- ストレージクラスの名称（metadata.name フィールド）に、ユーザ任意の名称を指定してください。ここで指定した名称は、各永続ボリュームの storageClassName フィールド、および Mediator の Helm チャートの consensusLog.storageClassName パラメタの値に指定します。
- ストレージクラスの provisioner フィールドに、「kubernetes.io/no-provisioner」を指定してください。

＜永続ボリュームの作成要領＞

- Kubernetes クラスタに起動する Mediator の数と同じ数だけ、永続ボリュームの Kubernetes マニフェストを作成してください。
- 各永続ボリュームの名称（metadata.name フィールド）に、ユーザ任意の名称を指定してください。
- 永続ボリュームの種類によって、次のどちらかの作業を実施してください。
  - 永続ボリュームの種類がローカル永続ボリューム（local）の場合

    各永続ボリュームの.spec.local フィールドに、「(1)　Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（ローカル永続ボリューム（local）とする場合)」で作成するワーカーノード上のボリュームマウント用ディレクトリパスを指定してください。
  - 永続ボリュームの種類が NFS ストレージの場合

    各永続ボリュームの.spec.nfs フィールドに、「3.5.2　Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（NFS ストレージとする場合)」で作成する NFS のサーバアドレスとディレクトリパスを指定してください。
- 永続ボリュームの種類がローカル永続ボリューム（local）の場合、各永続ボリュームの nodeAffinity フィールドに、Mediator を配置する個々のワーカーノードを決めてノード名を指定してください。

  例として、Kubernetes クラスタに起動する Mediator の数が「5」で、ワーカーノードが 5 台（「ワーカーノード-0」、「ワーカーノード-1」、「ワーカーノード-2」、「ワーカーノード-3」、「ワーカーノード-4」）がある場合、次のように指定します。
  - 「永続ボリューム-0」の nodeAffinity フィールドに、「ワーカーノード-0」を指定する
  - 「永続ボリューム-1」の nodeAffinity フィールドに、「ワーカーノード-1」を指定する
  - 「永続ボリューム-2」の nodeAffinity フィールドに、「ワーカーノード-2」を指定する
  - 「永続ボリューム-3」の nodeAffinity フィールドに、「ワーカーノード-3」を指定する

- 「永続ボリューム-4」の nodeAffinity フィールドに、「ワーカーノード-4」を指定する

- 永続ボリュームの種類が NFS ストレージの場合、各永続ボリュームの.spec.nodeAffinity フィールドで、Mediator を配置しないノード（コントロールプレーンなど）を決め、それ以外のノードに配置されるように指定してください。例として、.spec.nodeAffinity.required.nodeSelectorTerms.matchExpressions フィールドに「operator: NotIn」「values:- control-plane-0」のように指定します。
- 各永続ボリュームの capacity.storage フィールドに、次に示す値以上のサイズを指定してください。

  ```
  指定サイズ＝「(a) × (b) ＋ (c)」以上のサイズ
  (a) MediatorのHelmチャートの.consensusLog.sizeパラメタに指定した値
  (b) 1,024(1つのConsensusLogのサイズ)
  (c) 1,048,576(ボリュームの書き込み容量不足にならないための予備サイズ(1MB))
  ```

  例として、(a)が「256」の場合、「256×1,024 + 1,048,576」＝「1,310,720」以上となるため、capacity.storage フィールドには「2Mi」（2 メビバイト）を指定します。
- 各永続ボリュームの accessModes フィールドに、「ReadWriteOnce」を指定してください。
- 各永続ボリュームの persistentVolumeReclaimPolicy フィールドに、「Retain」を指定してください。
- 各永続ボリュームの storageClassName フィールドに、前述のストレージクラスで指定した名称を指定してください。

ローカル永続ボリューム（local）とする場合と、NFS ストレージとする場合の、Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの記述例を次に示します。

＜永続ボリュームの種類をローカル永続ボリューム（local）とする場合＞

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: mediator-consensus-log
provisioner: kubernetes.io/no-provisioner
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-0
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
  local:
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
```

```
          operator: In
          values:
          - worker-0
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-1
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
  local:
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-1
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-2
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
  local:
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-2
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-3
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
```

```
  storageClassName: mediator-consensus-log
  local:
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-3
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-4
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
  local:
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-4
```

この例では、Kubernetes クラスタに起動する Mediator の数が「5」で、ワーカーノードが 5 台ある場合の例としています。

＜永続ボリュームの種類を NFS ストレージとする場合＞

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: mediator-consensus-log
provisioner: kubernetes.io/no-provisioner
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-0
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
```

```
  nfs:
    server: nfs-server.example.com
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-1
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
  nfs:
    server: nfs-server.example.com
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-2
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
  nfs:
    server: nfs-server.example.com
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
---
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-3
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
  nfs:
    server: nfs-server.example.com
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mediator-consensus-log-4
spec:
  capacity:
    storage: 2Mi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: mediator-consensus-log
  nfs:
    server: nfs-server.example.com
    path: /mediator-consensus-log-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
```

この例では、Kubernetes クラスタに起動する Mediator の数が「5」で、ワーカーノードが 5 台ある場合の例としています。

3.6.3 Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成

「6.4.14 Elasticsearch の要件」に記載されている Elasticsearch データディレクトリの永続ボリュームの要件を満たすように、Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストを作成してください。

また、Elasticsearch データディレクトリ専用のストレージクラスを作成することで、Elasticsearch の永続ボリュームと Elasticsearch データディレクトリの永続ボリュームクレームのバインドを必ず一対一になるようにします。クラスタ構成で構築する場合、マスターノード用、データノード用の 2 つのストレージクラスを作成してください。

Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成要領を次に示します。

＜ストレージクラスの作成要領＞

- ストレージクラスの名称（metadata.name フィールド）に、ユーザ任意の名称を指定してください。ここで指定した名称は、各永続ボリュームの storageClassName フィールド、および Elasticsearch の Helm チャートの data.storage-class-name パラメタの値に指定します。
- ストレージクラスの provisioner フィールドに、「kubernetes.io/no-provisioner」を指定してください。

＜永続ボリュームの作成要領＞

- Kubernetes クラスタに起動する Elasticsearch のノード数と同じ数だけ、永続ボリュームの Kubernetes マニフェストを作成してください。クラスタ構成の場合はマスターノードとデータノードの数の合計の数だけ作成してください。
- 各永続ボリュームの名称（metadata.name フィールド）に、ユーザ任意の名称を指定してください。
- 各永続ボリュームの.spec.nodeAffinity フィールドに、Elasticsearch を配置する個々のワーカーノードを決めてノード名を指定してください。マスターノード内で同じワーカーノードを指定しないでください。また、クラスタ構成で構築する場合、データノード内で同じワーカーノードを指定しないでください。

  例として、Kubernetes クラスタに起動する Elasticsearch のマスターノードの数が「3」、データノードの数が「3」で、ワーカーノードが 3 台（「ワーカーノード-1」、「ワーカーノード-2」、「ワーカーノード-3」）がある場合、次のように指定します。
  - マスターノード向け「永続ボリューム-1」の.spec.nodeAffinity フィールドに、「ワーカーノード-1」を指定する。
  - マスターノード向け「永続ボリューム-2」の.spec.nodeAffinity フィールドに、「ワーカーノード-2」を指定する。
  - マスターノード向け「永続ボリューム-3」の.spec.nodeAffinity フィールドに、「ワーカーノード-3」を指定する。

- データノード向け「永続ボリューム-1」の.spec.nodeAffinity フィールドに、「ワーカーノード-1」を指定する。
- データノード向け「永続ボリューム-2」の.spec.nodeAffinity フィールドに、「ワーカーノード-2」を指定する。
- データノード向け「永続ボリューム-3」の.spec.nodeAffinity フィールドに、「ワーカーノード-3」を指定する。

- 各永続ボリュームの.spec.capacity.storage フィールドに Elasticsearch データディレクトリとして割り当てる永続ボリュームのサイズを指定します。データ量については次の式を目安とし、実測するようお願いします。

```
[必要なデータ量] = ((A) + (B) + (C)) × (D) × (E) 以上のサイズ
(A) 分散トレースで使用する1時間当たりのindexサイズ(MB)
(B) メッセージログで使用する1時間当たりのindexサイズ(MB)
(C) メトリクスで使用する1時間当たりのindexサイズ(MB)
(D) 1日当たりのシステムの想定稼働時間(h)
(E) 保存期間(日)

(A)分散トレースで使用する1時間当たりのindexサイズ
indexサイズ = ((a) + ((b) × (c)) + ((d) × (e)) + ((f) × (g))) × (h)
(a) SQL-Participant、TCC-ParticipantおよびTP1-Bridge以外が出力する分散トレースのサイズ：500MB
(b) SQL-Participantが出力する分散トレースのサイズ：400MB
(c) SQL-ParticipantのPod数
(d) TCC-Participantが出力する分散トレースのサイズ：300MB
(e) TCC-ParticipantのPod合計数
(f) TP1-Bridgeが出力する分散トレースのサイズ：300MB
(g) TP1-BridgeのPod数
(h) 想定する1秒間のトランザクション数

(B)メッセージログで使用する1時間当たりのindexサイズ
indexサイズ = (a)
(a) 1時間当たりの使用量:100MB(エラーなどによってメッセージログが継続して出力される場合を想定した推奨値)

(C)メトリクスで使用する1時間当たりのindexサイズ
indexサイズ = ((a) + ((b) × (c)) + ((d) × (e)) + ((f) × (g))) × 15 / (h)
(a) SQL-Participant、TCC-ParticipantおよびTP1-Bridge以外が出力するメトリクスのサイズ：1000MB
(b) SQL-Participantが出力するメトリクスのサイズ：100MB
(c) SQL-ParticipantのPod数
(b) TCC-Participantが出力するメトリクスのサイズ：100MB
(e) TCC-ParticipantのPod数
(f) TP1-Bridgeが出力するメトリクスのサイズ：100MB
(g) TP1-BridgeのPod数
(h) メトリクスの取得間隔。Prometheusのパラメタglobal.scrape_intervalの指定値の単位を秒に変換した値。
```

  - シングル構成の場合: 「必要なデータ量」の式から計算した値を記載してください。
  - クラスタ構成の場合

    ・マスターノードについては 3GB を指定してください。

・データノードについては、「必要なデータ量」で計算した値を各ノードに配置されるプライマリシャード、レプリカシャードの合計数倍してください。

- 各永続ボリュームの accessModes フィールドに、「ReadWriteOnce」を指定してください。
- 各永続ボリュームの persistentVolumeReclaimPolicy フィールドに、「Retain」を指定してください。
- 各永続ボリュームの storageClassName フィールドに、ストレージクラスで指定した名称を指定してください。

Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: elasticsearch-data-masternode
provisioner: kubernetes.io/no-provisioner
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: elasticsearch-data-datanode
provisioner: kubernetes.io/no-provisioner
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-data-masternode-0
spec:
  capacity:
    storage: 3Gi
  storageClassName: elasticsearch-data-masternode
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /elasticsearch-data-volume-masternode
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
            - worker-1
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-data-masternode-1
spec:
  capacity:
    storage: 3Gi
  storageClassName: elasticsearch-data-masternode
  accessModes:
```

```
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /elasticsearch-data-volume-masternode
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
            - worker-2
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-data-masternode-2
spec:
  capacity:
    storage: 3Gi
  storageClassName: elasticsearch-data-masternode
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /elasticsearch-data-volume-masternode
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
            - worker-3
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-data-datanode-0
spec:
  capacity:
    storage: 3Gi
  storageClassName: elasticsearch-data-datanode
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /elasticsearch-data-volume-datanode
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
            - worker-1
---
apiVersion: v1
```

```
kind: PersistentVolume
metadata:
  name: elasticsearch-data-datanode-1
spec:
  capacity:
    storage: 3Gi
  storageClassName: elasticsearch-data-datanode
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /elasticsearch-data-volume-datanode
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
            - worker-2
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-data-datanode-2
spec:
  capacity:
    storage: 3Gi
  storageClassName: elasticsearch-data-datanode
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /elasticsearch-data-volume-datanode
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
            - worker-3
```

この例では、Kubernetes クラスタに起動する Elasticsearch のマスターノードの数が「3」、データノードの数が「3」で、ワーカーノードが 3 台ある場合の例としています。

3.6.4 Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成

「6.4.14 Elasticsearch の要件」に記載されている Elasticsearch リポジトリディレクトリの永続ボリュームの要件を満たすように、Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストを作成してください。

また、Elasticsearch リポジトリディレクトリ専用のストレージクラスを作成することで、Elasticsearch の永続ボリュームと Elasticsearch リポジトリディレクトリの永続ボリュームクレームのバインドを必ず一対一になるようにします。

Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成要領を次に示します。

＜ストレージクラスの作成要領＞

- ストレージクラスの名称（metadata.name フィールド）に、ユーザ任意の名称を指定してください。ここで指定した名称は、後述の永続ボリュームの storageClassName フィールド、および Elasticsearch の Helm チャートの repo.storage-class-name パラメタの値に指定します。

＜永続ボリュームの作成要領＞

- Kubernetes クラスタに起動する Elasticsearch のノード数と同じ数だけ、永続ボリュームの Kubernetes マニフェストを作成してください。クラスタ構成の場合はマスターノードとデータノードの数の合計の数だけ作成してください。
- 永続ボリュームの名称（metadata.name フィールド）に、ユーザ任意の名称を指定してください。
- 永続ボリュームの.spec.capacity.storage フィールドに Elasticsearch リポジトリディレクトリとして割り当てる永続ボリュームのサイズを指定します。データディレクトリの永続ボリュームで指定した値を目安とし、実測してください。
- 永続ボリュームの accessModes フィールドに、「ReadWriteMany」を指定してください。
- 永続ボリュームの persistentVolumeReclaimPolicy フィールドに、「Retain」を指定してください。
- 永続ボリュームの nfs フィールドに、Kubernetes 各ワーカーノードに「3.5.3　Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリの作成」で作成する NFS のサーバアドレスとディレクトリを指定してください。

  次に示す記述例では、ディレクトリ名を「/elasticsearch-repo-volume」としています。

Elasticsearch リポジトリディレクトリの永続ボリュームの Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: elasticsearch-repo
provisioner: kubernetes.io/no-provisioner
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-repo-master-0
spec:
  capacity:
    storage: 3Gi
  accessModes:
```

```
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: elasticsearch-repo
  nfs:
    server: nfs-server.example.com
    path: /elasticsearch-repo-volume
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-repo-master-1
spec:
  capacity:
    storage: 3Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: elasticsearch-repo
  nfs:
    server: nfs-server.example.com
    path: /elasticsearch-repo-volume
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-repo-master-2
spec:
  capacity:
    storage: 3Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: elasticsearch-repo
  nfs:
    server: nfs-server.example.com
    path: /elasticsearch-repo-volume
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-repo-data-0
spec:
  capacity:
    storage: 3Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: elasticsearch-repo
  nfs:
    server: nfs-server.example.com
    path: /elasticsearch-repo-volume
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-repo-data-1
spec:
  capacity:
```

```
    storage: 3Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: elasticsearch-repo
  nfs:
    server: nfs-server.example.com
    path: /elasticsearch-repo-volume
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-repo-data-2
spec:
  capacity:
    storage: 3Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: elasticsearch-repo
  nfs:
    server: nfs-server.example.com
    path: /elasticsearch-repo-volume
```

3.6.5 EADS の永続ボリュームの Kubernetes マニフェストの作成（通常版限定）

「6.4.7 EADS の要件（通常版限定）」に記載されている EADS の永続ボリュームの要件を満たすように、EADS の永続ボリュームの Kubernetes マニフェストを作成してください。

EADS で使用する永続ボリュームの種類を「ローカル永続ボリューム（local）」とし、Kubernetes の nodeAffinity で永続ボリュームの割り当てノードを固定することで、その永続ボリュームを使用する EADS の Pod は Kubernetes スケジューラによって必ず永続ボリュームと同一ノードに割り当てられます。これによって、EADS の Pod が必ずノードにアタッチされたローカルストレージデバイスを使用できるようにします。

また、永続ボリュームにどの永続ボリュームクレームにバインドさせるかを指定することで、EADS の永続ボリュームと EADS の永続ボリュームクレームのバインドを必ず一対一になるようにします。

EADS の永続ボリュームの Kubernetes マニフェストの作成要領を次に示します。

＜見積もり要領＞

デプロイする EADS の構成（EADS サーバ台数、多重度、キャッシュなど）を決め、マニュアル「Elastic Application Data Store ユーザーズガイド」を参考にして EADS サーバの各種見積もり量（使用メモリ量・ログファイル用ディスク容量・キャッシュデータファイル、およびキャッシュインデックスファイル用ディスク容量・ストアデータファイル用ディスク容量）を見積もってください。

＜ PVC の種類と個数について＞

EADS サーバが定義する PVC（永続ボリュームクレーム）を満たすボリュームを決めて、対応するリソースのマニフェストファイルを作成してデプロイします。

表 3-12　EADS サーバで定義している PVC の種類と個数

ボリューム種類	PVC 命名規則	個数
ログファイル	logs-pvc-<Server リリース名>-eads-server-<複製インデックス>	EADS サーバ台数
ストアデータファイル	store-pvc-<Server リリース名>-eads-server-<複製インデックス>	EADS サーバ台数
キャッシュデータファイルおよびキャッシュインデックスファイル	cache-pvc-<キャッシュ定義インデックス※1>-<Server リリース名>-eads-server-<複製インデックス>	キャッシュ数 ×EADS サーバ台数
追加ボリューム	vol-pvc-<追加ボリューム定義インデックス※2>-<Server リリース名>-eads-server-<複製インデックス>	追加ボリューム数※3×EADS サーバ台数

注※1

「3.7.5　EADS の Helm チャートのカスタマイズ（通常版限定）」で説明する EADS サーバチャート用 values.yaml の caches フィールド下に登録したキャッシュ定義配列のインデックス（0 から始まる）です。キャッシュ定義数分の PVC が追加されます。ただし、name が重複した要素はインデックスから除外されます。

注※2

「3.7.5　EADS の Helm チャートのカスタマイズ（通常版限定）」で説明する EADS サーバチャート用 values.yaml の volumes フィールド下に登録した追加ボリューム定義配列のインデックス（0 から始まる）です。ただし、name が重複した要素はインデックスから除外されます。

注※3

「3.7.5　EADS の Helm チャートのカスタマイズ（通常版限定）」で説明する EADS サーバチャート用 values.yaml の volumes フィールド下に登録した追加ボリューム定義配列の総数です。ただし、name が重複した要素は総数から除外されます。

＜永続ボリュームの作成要領＞

- Kubernetes クラスタに起動する EADS の数×3（ログファイル用ボリューム 1 個、ストアデータファイル用ボリューム 1 個、キャッシュデータファイルおよびキャッシュインデックスファイル用ボリューム 1 個、追加ボリューム用ボリューム 0 個の計 3 個）と同じ数だけ、永続ボリュームの Kubernetes マニフェストを作成してください。
- 各永続ボリュームの名称（metadata.name フィールド）に、ユーザ任意の名称を指定してください。
- 各永続ボリュームは、ボリュームの種類を「ローカル永続ボリューム（local）」とし、「(3)　EADS の永続ボリュームマウント用ディレクトリの作成（通常版限定）」で作成するノード上のボリュームマウント用ディレクトリパスを指定してください。
- 各永続ボリュームの nodeAffinity フィールドに、EADS を配置する個々のノードを決めてノード名を指定してください。

  例として、Kubernetes クラスタに起動する EADS の数が「3」で、ノードが 3 台（「ノード-1」、「ノード-2」、「ノード-3」）がある場合、次のように指定します。

- 「永続ボリューム-1」の nodeAffinity フィールドに、「ノード-1」を指定する
- 「永続ボリューム-2」の nodeAffinity フィールドに、「ノード-2」を指定する
- 「永続ボリューム-3」の nodeAffinity フィールドに、「ノード-3」を指定する

- 各永続ボリュームの capacity.storage フィールドに、次に示す値以上のサイズを指定してください。

  ```
  (a)EADSのHelmチャートのvalues.yamlのchaches-pvc-capacityパラメタに指定した値
  (b)EADSのHelmチャートのvalues.yamlのlogs-capacityパラメタに指定した値
  (c)EADSのHelmチャートのvalues.yamlのstore-capacityパラメタに指定した値
  ```

- 各永続ボリュームの accessModes フィールドに、「ReadWriteOnce」を指定してください。
- 各永続ボリュームの persistentVolumeReclaimPolicy フィールドに、「Retain」を指定してください。
- 各永続ボリュームの claimRef フィールドに、EADS を配置する個々のノードを決めて EADS の PVC（「表 3-12　EADS サーバで定義している PVC の種類と個数」の「PVC 命名規則」列参照）を指定してください。

  例として、Kubernetes クラスタに起動する EADS が 3 台（「Pod-0」、「Pod-1」、「Pod-2」）の数が「3」で、ノードが 3 台（「ノード-1」、「ノード-2」、「ノード-3」）がある場合、次のように指定します。

  - nodeAffinity フィールドが「ノード-1」に指定してある永続ボリュームの claimRef フィールドに、「Pod-0」の PVC を指定する
  - nodeAffinity フィールドが「ノード-2」に指定してある永続ボリュームの claimRef フィールドに、「Pod-1」の PVC を指定する
  - nodeAffinity フィールドが「ノード-3」に指定してある永続ボリュームの claimRef フィールドに、「Pod-2」の PVC を指定する

EADS の永続ボリュームの Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-cache-0
spec:
  storageClassName: ""
  capacity:
    storage: "514Mi"
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-cache-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-1
  claimRef:
    name: cache-pvc-0-eads-server-eads-server-0
```

```
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-logs-0
spec:
  storageClassName: ""
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-logs-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-1
  claimRef:
    name: logs-pvc-eads-server-eads-server-0
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-store-0
spec:
  storageClassName: ""
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-store-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-1
  claimRef:
    name: store-pvc-eads-server-eads-server-0
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-cache-1
spec:
  storageClassName: ""
```

```
  capacity:
    storage: ”514Mi”
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-cache-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-2
  claimRef:
    name: cache-pvc-0-eads-server-eads-server-1
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-logs-1
spec:
  storageClassName: ””
  capacity:
    storage: ”1Gi”
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-logs-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-2
  claimRef:
    name: logs-pvc-eads-server-eads-server-1
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-store-1
spec:
  storageClassName: ””
  capacity:
    storage: ”1Gi”
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-store-volume
  nodeAffinity:
```

```
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-2
  claimRef:
    name: store-pvc-eads-server-eads-server-1
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-cache-2
spec:
  storageClassName: ""
  capacity:
    storage: "514Mi"
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-cache-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-3
  claimRef:
    name: cache-pvc-0-eads-server-eads-server-2
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-logs-2
spec:
  storageClassName: ""
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-logs-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-3
  claimRef:
```

```
    name: logs-pvc-eads-server-eads-server-2
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-store-2
spec:
  storageClassName: ""
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  local:
    path: /eads-store-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker-3
  claimRef:
    name: store-pvc-eads-server-eads-server-2
    namespace: my-namespace
```

3.6.6 デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成

次に示す要件を満たすように、デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストを作成してください。この作業は、次のどちらかの Kubernetes アプリケーションの Kubernetes マニフェスト作成時に実施します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

このロールおよびロールバインディングを使用することで、各 Kubernetes アプリケーションの Dependency-Checker コンテナがデプロイ依存関係チェック機能に必要な Kubernetes オブジェクトにアクセスできるようにします。

デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成要領を次に示します。

＜ロールの作成要領＞

- リソースの種類（.kind フィールド）は、次のとおり指定してください。

- Dependency-Checker コンテナを含む Kubernetes アプリケーションと Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※の Namespace がすべて同一の場合：
  Role リソースを指定
- Dependency-Checker コンテナを含む Kubernetes アプリケーションと Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※の Namespace が 1 つでも異なる場合：
  ClusterRole リソースを指定

注※

Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「6.4　各 Kubernetes アプリケーションの作成要件」を参照してください。

- ロールの名称（.metadata.name フィールド）に、ユーザ任意の名称を指定してください。
- Role リソースの場合、ロールの Namespace（.metadata.namespace フィールド）に、Dependency-Checker コンテナを含む Kubernetes アプリケーションと同一の Namespace を指定してください。
  ClusterRole リソースの場合、Namespace は指定しません。
- ロールのルール（.rules フィールド）に、次のとおり指定してください。
  - API グループ（apiGroups）
    空（コアグループ）、apps
  - リソース（resources）
    pods、replicasets、deployments、statefulsets、daemonsets
  - アクセス許可（verbs）
    get、list

＜ロールバインディングの作成要領＞

- リソースの種類（.kind フィールド）は、次のとおり指定してください。
  - Dependency-Checker コンテナを含む Kubernetes アプリケーションと Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※の Namespace がすべて同一の場合：
    RoleBinding リソースを指定
  - Dependency-Checker コンテナを含む Kubernetes アプリケーションと Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※の Namespace が 1 つでも異なる場合：
    ClusterRoleBinding リソースを指定

注※

Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「6.4　各 Kubernetes アプリケーションの作成要件」を参照してください。

- ロールバインディングの名称（.metadata.name フィールド）に、ユーザ任意の名称を指定してください。
- RoleBinding リソースの場合、ロールバインディングの Namespace（.metadata.namespace フィールド）に、Dependency-Checker コンテナを含む Kubernetes アプリケーションと同一の Namespace を指定してください。

  ClusterRoleBinding リソースの場合、Namespace は指定しません。
- バインドするロール（.roleRef フィールド）は、次のとおり指定してください。
  - .roleRef.apiGroup フィールドに、「rbac.authorization.k8s.io」を指定してください。
  - .roleRef.kind フィールドに、「Role」を指定してください。
  - .roleRef.name フィールドに、ロールで指定したロールの名称を指定してください。
- バインドする対象物（.subjects フィールド）は、次のとおり指定してください。
  - リストの要素数は Dependency-Checker コンテナを含む Kubernetes アプリケーションと同じ数にします。
  - .subjects[].kind フィールドに、「ServiceAccount」を指定してください。
  - .subjects[].name フィールドは、Dependency-Checker コンテナを含む Kubernetes アプリケーションの ServiceAccount リソースの.metadata.name フィールドに指定した名称を指定してください。HMP-PCTO 責務の Kubernetes アプリケーションおよびユーザと HMP-PCTO 責務の Kubernetes アプリケーションの ServiceAccount の名称は次のとおりです。

Kubernetes アプリケーション	ServiceAccount の名称
TP1-Bridge	<リリース名>-service-account※
Mediator	mediator-service-account
Ext-Cons	ext-cons-service-account
Filebeat	filebeat-service-account
Jaeger-query	jaeger-query-service-account
Jaeger-collector	jaeger-collector-service-account
Prometheus	prometheus-service-account
Metricbeat	metricbeat-service-account
Logstash	logstash-service-account

注※

<リリース名>は「(15)　Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。

例：リリース名が tp1-bridge1 の場合

tp1-bridge1-service-account

- .subjects[].namespace フィールドに、Dependency-Checker コンテナを含む Kubernetes アプリケーションの ServiceAccount リソースの.metadata.namespace フィールドに指定した名称を指定してください。

デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの記述例を次に示します。

■Dependency-Checker コンテナを含む Kubernetes アプリケーションと Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※1 の Namespace がすべて同一の場合

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: dependency-checker-role
  namespace: my-namespace
rules:
- apiGroups:
  - ""
  - apps
  resources:
  - pods
  - replicasets
  - deployments
  - statefulsets
  - daemonsets
  verbs:
  - get
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: dependency-checker-role-binding
  namespace: my-namespace
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: dependency-checker-role
subjects:
- kind: ServiceAccount
  name: orchestrator-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: entity-service-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: sql-participant-service-account※2
  namespace: my-namespace
- kind: ServiceAccount
  name: mediator-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: filebeat-service-account
  namespace: my-namespace
- kind: ServiceAccount
```

```
  name: jaeger-query-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: jaeger-collector-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: prometheus-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: metricbeat-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: logstash-service-account
  namespace: my-namespace
```

注※1

Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「6.4 各 Kubernetes アプリケーションの作成要件」を参照してください。

注※2

SQL-Participant を使用する場合の例を記述しています。

- TCC-Participant を使用する場合は、「name: tcc-participant-service-account」に置き換えてください。
- TP1-Bridge を使用する場合は、「name: <リリース名>-service-account」に置き換えてください。<リリース名>は、「(15)　Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。

  例：リリース名が tp1-bridge1 の場合

  name: tp1-bridge1-service-account

■Dependency-Checker コンテナを含む Kubernetes アプリケーションと Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※1 の Namespace が 1 つでも異なる場合

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: dependency-checker-role
rules:
- apiGroups:
  - ""
  - apps
  resources:
  - pods
  - replicasets
  - deployments
  - statefulsets
  - daemonsets
  verbs:
  - get
  - list
---
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dependency-checker-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: dependency-checker-role
subjects:
- kind: ServiceAccount
  name: orchestrator-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: entity-service-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: sql-participant-service-account※2
  namespace: my-namespace
- kind: ServiceAccount
  name: mediator-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: filebeat-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: jaeger-query-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: jaeger-collector-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: prometheus-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: metricbeat-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: logstash-service-account
  namespace: my-namespace
```

注※1

Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「6.4 各 Kubernetes アプリケーションの作成要件」を参照してください。

注※2

SQL-Participant を使用する場合の例を記述しています。

- TCC-Participant を使用する場合は、「name: tcc-participant-service-account」に置き換えてください。
- TP1-Bridge を使用する場合は、「name: <リリース名>-service-account」に置き換えてください。
  <リリース名>は、「(15)　Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。
  例：リリース名が tp1-bridge1 の場合

name: tp1-bridge1-service-account

3.6.7 コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成

次に示す要件を満たすように、コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストを作成してください。

このロールおよびロールバインディングを使用することで、Filebeat コンテナがコンテナログ収集に必要な Kubernetes オブジェクトにアクセスできるようにします。

コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成要領を次に示します。

＜ロールの作成要領＞

- リソースの種類(.kind フィールド)は、ClusterRole リソースを指定してください。
- ロールの名称(.metadata.name フィールド)に、ユーザ任意の名称を指定してください。
- ロールのルール（.rules フィールド）に、次のとおり指定してください。

API グループ(apiGroups)	リソース(resources)	アクセス許可(verbs)
空(コアグループ)	pods、nodes、namespaces	get、watch、list

＜ロールバインディングの作成要領＞

- リソースの種類(.kind フィールド)は、ClusterRoleBinding リソースを指定してください。
- ロールバインディングの名称(.metadata.name フィールド)に、ユーザ任意の名称を指定してください。
- バインドするロール(.roleRef フィールド)は、次のとおりとしてください。
  - .roleRef.apiGroup フィールドに、「rbac.authorization.k8s.io」を指定してください。
  - .roleRef.kind フィールドに、「ClusterRole」を指定してください。
  - .roleRef.name フィールドに、前述のロールで指定したロールの名称を指定してください。
- バインドする対象物(.subjects フィールド)は、次のとおりとしてください。
  - リストの要素数は 1 つにします。
  - .subjects[].kind フィールドに、「ServiceAccount」を指定してください。
  - .subjects[].name フィールドは、「filebeat-service-account」を指定してください。
  - .subjects[].namespace フィールドに、Filebeat の ServiceAccount リソースの.metadata.namespace で指定した値を指定してください。

コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストの記述例を示します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: filebeat-role
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - nodes
  - namespaces
  verbs:
  - get
  - watch
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: filebeat-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: filebeat-role
subjects:
- kind: ServiceAccount
  name: filebeat-service-account
  namespace: my-namespace
```

3.6.8 メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成

次に示す要件を満たすように、メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストを作成してください。

このロールおよびロールバインディングを使用することで、Prometheus コンテナがメトリクス収集に必要な Kubernetes オブジェクトにアクセスできるようにします。

メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成要領を次に示します。

＜ロールの作成要領＞

- リソースの種類(.kind フィールド)は、ClusterRole リソースを指定してください。
- ロールの名称(.metadata.name フィールド)に、ユーザ任意の名称を指定してください。
- ロールのルール（.rules フィールド）に、次のとおり指定してください。

API グループ(apiGroups)	リソース(resources)	アクセス許可(verbs)
空(コアグループ)	pods、nodes、nodes/proxy、services、endpoints	get、watch、list

＜ロールバインディングの作成要領＞

- リソースの種類(.kind フィールド)は、ClusterRoleBinding リソースを指定してください。
- ロールバインディングの名称(.metadata.name フィールド)に、ユーザ任意の名称を指定してください。
- バインドするロール(.roleRef フィールド)は、次のとおりとしてください。
  - .roleRef.apiGroup フィールドに、「rbac.authorization.k8s.io」を指定してください。
  - .roleRef.kind フィールドに、「ClusterRole」を指定してください。
  - .roleRef.name フィールドに、前述のロールで指定したロールの名称を指定してください。
- バインドする対象物(.subjects フィールド)は、次のとおりとしてください。
  - リストの要素数は 1 つにします。
  - .subjects[].kind フィールドに、「ServiceAccount」を指定してください。
  - .subjects[].name フィールドは、「prometheus-service-account」を指定してください。
  - .subjects[].namespace フィールドに、Prometheus の ServiceAccount リソースの.metadata.namespace で指定した値を指定してください。

メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストの記述例を示します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus-role
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - nodes
  - nodes/proxy
  - services
  - endpoints
  verbs:
  - get
  - watch
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
```

```
  name: prometheus-role
subjects:
- kind: ServiceAccount
  name: prometheus-service-account
  namespace: my-namespace
```

3.6.9 メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストの作成

次に示す要件を満たすように、メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストを作成してください。

このロールおよびロールバインディングを使用することで、Metricbeat コンテナが Kubernetes メタデータ付与に必要な Kubernetes オブジェクトにアクセスできるようにします。

メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストの作成要領を次に示します。

＜ロールの作成要領＞

- リソースの種類(.kind フィールド)は、ClusterRole リソースを指定してください。
- ロールの名称(.metadata.name フィールド)に、ユーザ任意の名称を指定してください。
- ロールのルール（.rules フィールド）に、次のとおり指定してください。

API グループ(apiGroups)	リソース(resources)	アクセス許可(verbs)
空(コアグループ)	pods、nodes、namespaces、events	get、watch、list

＜ロールバインディングの作成要領＞

- リソースの種類(.kind フィールド)は、ClusterRoleBinding リソースを指定してください。
- ロールバインディングの名称(.metadata.name フィールド)に、ユーザ任意の名称を指定してください。
- バインドするロール(.roleRef フィールド)は、次のとおりとしてください。
  - .roleRef.apiGroup フィールドに、「rbac.authorization.k8s.io」を指定してください。
  - .roleRef.kind フィールドに、「ClusterRole」を指定してください。
  - .roleRef.name フィールドに、前述のロールで指定したロールの名称を指定してください。
- バインドする対象物(.subjects フィールド)は、次のとおりとしてください。
  - リストの要素数は 1 つにします。
  - .subjects[].kind フィールドに、「ServiceAccount」を指定してください。
  - .subjects[].name フィールドは、「metricbeat-service-account」を指定してください。

- .subjects[].namespace フィールドに、Metricbeat の ServiceAccount リソースの.metadata.namespace で指定した値を指定してください。

メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストの記述例を示します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: metricbeat-role
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - nodes
  - namespaces
  - events
  verbs:
  - get
  - watch
  - list
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: metricbeat-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: metricbeat-role
subjects:
- kind: ServiceAccount
  name: metricbeat-service-account
  namespace: my-namespace
```

3.6.10 uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）

以降に示す要件を満たすように、uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストを作成してください。この作業は、次のどちらかの Kubernetes アプリケーションの Kubernetes マニフェスト作成時に実施します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームの種類を「NFS」とし、アクセスモード(accessModes)を「ReadWriteMany」とすることで、uCosminexus Application Runtime with Java for Spring Boot をインストールした各 Pod がどのノードからでも永続ボリュームにスナップショットログを出力できるようにします。

uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成要領を次に示します。

＜永続ボリュームの作成要領＞

- uCosminexus Application Runtime with Java for Spring Boot をインストールした Pod で使用している Namespace 名の数と同じ数だけ、永続ボリュームの Kubernetes マニフェストを作成してください。

  例 1）

  Mediator/Orchestrator/Entity-Service/Participant の Namespace 名が「my-namespace」の場合、Namespace 名の数は 1 つであるため、Kubernetes マニフェストを 1 つ作成します。

  例 2）

  Mediator の Namespace 名が「my-namespace-0」、Orchestrator/Entity-Service/Participant の Namespace 名が「my-namespace-1」の場合、Namespace 名の数は 2 つであるため、Kubernetes マニフェストを 2 つ作成します。
- 各永続ボリュームの名称(.metadata.name フィールド)に、ユーザ任意の名称を指定してください。なお、名称は Kubernetes マニフェストごとに一意にする必要があります。ここで指定した名称は、後述の各永続ボリューム要求の.spec.volumeName フィールドの値に指定します。
- 各永続ボリュームの.spec.capacity.storage フィールドに、スナップショットログの出力先となる永続ボリュームのサイズを指定してください。
- 各永続ボリュームの.spec.accessModes フィールドに、「ReadWriteMany」を指定してください。
- 各永続ボリュームの.persistentVolumeReclaimPolicy フィールドに、「Retain」を指定してください。
- 各永続ボリュームは、ボリュームの種類を「NFS」とし、「3.5.4　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームマウント用ディレクトリ作成（通常版限定)」で作成する NFS 上のボリュームマウント用ディレクトリパスを指定してください。

  以降に示す記述例では、ディレクトリ名を「/ucars-snapshots」としています。

＜永続ボリューム要求の作成要領＞

- ＜永続ボリュームの作成要領＞と同様に、永続ボリューム要求についても、uCosminexus Application Runtime with Java for Spring Boot をインストールした Pod で使用している Namespace 名の数と同じ数だけ、永続ボリューム要求の Kubernetes マニフェストを作成してください。
- 各永続ボリューム要求の名称(.metadata.name フィールド)に、ユーザ任意の名称を指定してください。なお、名称は Kubernetes マニフェストごとに一意にする必要はありません（Kubernetes マニフェストごとに Namespace が異なり同一 Namespace 内で名称が重複しないため)。ここで指定した名称は、uCosminexus Application Runtime with Java for Spring Boot をインストールしたユーザ

責務の Kubernetes アプリケーションの Kubernetes マニフェストの.spec.template.spec.volumes[].persistentVolumeClaim.claimName フィールド、および、Mediator の Helm チャートの.ucars.snapshots.claimName パラメタの値に指定します。

- 各永続ボリューム要求の Namespace(.metadata.namespace フィールド)に、uCosminexus Application Runtime with Java for Spring Boot をインストールした Pod で使用している Namespace 名を指定します。
- 各永続ボリューム要求の.spec.accessModes フィールドに、「ReadWriteMany」を指定してください。
- 各永続ボリューム要求の.spec.volumeName フィールドに、前述の永続ボリュームで指定した名称を指定してください。
- 各永続ボリューム要求の.spec.resources.requests.storage フィールドに、スナップショットログの出力に必要なサイズを指定してください。

Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの記述例を次に示します。

■uCosminexus Application Runtime with Java for Spring Boot をインストールした Pod で使用している Namespace 名の数が 1 つである場合

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: snapshot-volume
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: nfs-server.example.com
    path: /ucars-snapshots
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: snapshot-volume
  namespace: my-namespace
spec:
  accessModes:
  - ReadWriteMany
  volumeName: snapshot-volume
  resources:
    requests:
      storage: 500Mi
```

■uCosminexus Application Runtime with Java for Spring Boot をインストールした Pod で使用している Namespace 名の数が 2 つである場合

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: snapshot-volume-0
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: nfs-server.example.com
    path: /ucars-snapshots
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: snapshot-volume
  namespace: my-namespace-0
spec:
  accessModes:
  - ReadWriteMany
  volumeName: snapshot-volume-0
  resources:
    requests:
      storage: 500Mi
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: snapshot-volume-1
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: nfs-server.example.com
    path: /ucars-snapshots
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: snapshot-volume
  namespace: my-namespace-1
spec:
  accessModes:
  - ReadWriteMany
  volumeName: snapshot-volume-1
  resources:
    requests:
      storage: 500Mi
```

3.7　Helm チャートのカスタマイズ

「6.1　Kubernetes アプリケーションのデプロイ方法」に示すとおり、HMP-PCTO の Kubernetes アプリケーションのうち、HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションについては、HMP-PCTO が Helm チャートを提供し、ユーザが Helm チャートをカスタマイズします。

ここでは、Helm チャートのカスタマイズ方法について説明します。

なお、TP1-Bridge を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名の長さの合計は、40 バイト以下としてください。TP1-Bridge の Service 名は、「(15)　Kubernetes アプリケーションのデプロイ」の「表 3-20　helm コマンドに指定するコマンド引数」のリリース名を参照してください。

3.7.1　作業項目の一覧

Helm チャートのカスタマイズの作業項目の一覧を次の表に示します。

表 3-13　Helm チャートのカスタマイズの作業項目の一覧

作業項目	参照先
Helm チャートへのパラメタ追加	3.7.2　Helm チャートへのパラメタ追加
Mediator の Helm チャートのカスタマイズ	3.7.3　Mediator の Helm チャートのカスタマイズ
TP1-Bridge の Helm チャートのカスタマイズ（通常版かつ TP1-Bridge 限定）	3.7.4　TP1-Bridge の Helm チャートのカスタマイズ（通常版かつ TP1-Bridge 限定）
EADS の Helm チャートのカスタマイズ（通常版限定）	3.7.5　EADS の Helm チャートのカスタマイズ（通常版限定）
Ext-Cons の Helm チャートのカスタマイズ（トライアル版限定）	3.7.6　Ext-Cons の Helm チャートのカスタマイズ（トライアル版限定）
Filebeat の Helm チャートのカスタマイズ	3.7.7　Filebeat の Helm チャートのカスタマイズ
Jaeger-query の Helm チャートのカスタマイズ	3.7.8　Jaeger-query の Helm チャートのカスタマイズ
Jaeger-collector の Helm チャートのカスタマイズ	3.7.9　Jaeger-collector の Helm チャートのカスタマイズ
Prometheus の Helm チャートのカスタマイズ	3.7.10　Prometheus の Helm チャートのカスタマイズ
Metricbeat の Helm チャートのカスタマイズ	3.7.11　Metricbeat の Helm チャートのカスタマイズ

作業項目	参照先
Elasticsearch の Helm チャートのカスタマイズ	3.7.12　Elasticsearch の Helm チャートのカスタマイズ
Logstash の Helm チャートのカスタマイズ	3.7.13　Logstash の Helm チャートのカスタマイズ

3.7.2　Helm チャートへのパラメタ追加

HMP-PCTO が提供する Helm チャートには、HMP-PCTO のトラブルシュート機能で使用する次の OSS を動作させるために必要な最低限のパラメタだけが記載されています。

- Jaeger-collector
- Jaeger-query
- Prometheus
- Elasticsearch
- Filebeat
- Logstash
- Metricbeat

HMP-PCTO のトラブルシュート機能で使用する OSS に対して、Helm チャートに記載のない OSS パラメタを使用したい場合は、次の手順に従ってパラメタを追加してください。なお、この手順では、Jaeger-collector に collector.queue-size-memory パラメタを新規追加し、values.yaml でパラメタの値を設定する場合の例を示します。

1. Helm チャートを展開する

次のコマンドを実行し、tgz ファイルから Helm チャートを展開します。

```
tar -xzf jaeger-collector-V.R.S.tgz
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コマンドを実行すると、次に示す構成で Helm チャートが展開されます。編集対象のファイルを太字で示します。

ファイルまたはディレクトリ構成
<作業ディレクトリ>/
jaeger-collector
.helmignore
Chart.yaml
values.schema.json

ファイルまたはディレクトリ構成
values.yaml
templates
deployment.yaml
service.yaml
serviceaccount.yaml

2. deployment.yaml ファイルを修正する

templates フォルダ内の deployment.yaml ファイルを開き、新規パラメタを追加します。

次に示す例では、新規に collector.queue-size-memory パラメタを作成し、パラメタの値を Jaeger-collector の values.yaml で設定できるようにしています。オレンジ色の背景で示す行が追加した行です。

```
      containers:
      - name: jaeger-collector
        image: '{{ .Values.image }}'
        imagePullPolicy: {{ .Values.imagePullPolicy }}
        args:
        - "--collector.queue-size={{ .Values.collector.queueSize }}"
        - "--collector.num-workers={{ .Values.collector.numWorkers }}"
        - "--es.num-replicas={{ .Values.replicas }}"
        - "--es.num-shards={{ .Values.collector.numShards }}"
        - "--collector.queue-size-memory={{ .Values.collector.queueSizeMemory }}"
```

3. values.yaml を修正する

values.yaml ファイルを開き、手順 2 で追加したパラメタの設定値を追加します。

次に示す例では collector セクション内に、追加したパラメタとその設定値を追記しています。オレンジ色の背景で示す行が追記した行です。

```
collector:
  queueSize: "10000"
  numWorkers: "50"
  numShards: "5"
  queueSizeMemory: "設定値"
```

4. パッケージを再作成する

<作業ディレクトリ>（jaeger-collector フォルダの上の階層）へ移動し、次のコマンドを実行してください。修正した Helm チャートが再度パッケージ化されます。

```
helm package ./jaeger-collector
```

上記の手順を実施することで、Jaeger-collector に新規パラメタが追加され、その設定値が設定されます。このあと、「3.7.9　Jaeger-collector の Helm チャートのカスタマイズ」※を参照して、values.yaml の抽出およびカスタマイズを実施してください。なお、デプロイ手順については「(15)　Kubernetes アプリケーションのデプロイ」を参照してください。

注※

ほかの OSS の Helm チャートに OSS パラメタを追加した場合は、その OSS の Helm チャートのカスタマイズを実施してください。

3.7.3 Mediator の Helm チャートのカスタマイズ

Mediator の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Mediator の Helm チャートパッケージファイル（mediator-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./mediator-V.R.S.tgz > ./mediator-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.1　Mediator」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace
replicas: 5
image: myregistry.example.com/myproject/mediator:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent
dependencyCheck:
  enabled: true
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※
  imagePullPolicy: IfNotPresent
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10
  targets:
  - my-namespace/deployments/logstash
  - my-namespace/deployments/jaeger-query
  - my-namespace/deployments/jaeger-collector
  - my-namespace/deployments/prometheus
  - my-namespace/deployments/metricbeat
  - my-namespace/daemonsets/filebeat
healthCheck:
  enabled: true
livenessProbe:
  initialDelaySeconds: 180
  periodSeconds: 30
  failureThreshold: 3
  timeoutSeconds: 10
  healthCheck:
    retryCount: 0
    retryIntervalSeconds: 0
```

```
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
postStart:
  healthCheck:
    retryCount: 30
    retryIntervalSeconds: 10
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
preStop:
  waitTransactions:
    retryCount: 30
    retryIntervalSeconds: 10
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
terminationGracePeriodSeconds: 330
affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: app.kubernetes.io/name
          operator: In
          values:
          - mediator
      topologyKey: kubernetes.io/hostname
resources:
  limits:
    cpu: 2
    memory: 1Gi
  requests:
    cpu: 1
    memory: 512Mi
participant:
  sender:
    threadPoolSize: 30
    queueSize: 25
consensusLog:
  size: 256
  io:
    threadPoolSize: 12
    queueSize: 25
    watchInSeconds: 240
    blockSize: 4096
  storageClassName: mediator-consensus-log
http:
  port: 20554
grpc:
  port: 20551
  server:
    threadPoolSize: 30
    queueSize: 25
  useTls: false
  authSkip: false
  tls:
    secretName: ""
    clientAuth:
      enabled: false
    session:
```

```
      cacheSize: 0
      timeout: 0
  keepalive:
    interval: 0
    timeout: 60
  watchtime: 5
  dedup: true
  reuse: true
mediator:
  judge:
    threadPoolSize: 30
    queueSize: 25
    proposalLogLimitNum: 512
    tranTerminatedLimitNum: 512
    tranTerminatedConfirmNotifyInterval: 60000
tracing:
  instanceId: ""
  url: http://jaeger-collector:14268/api/traces
  traceUse: true
  queueSize: 20000
  delayTime: 1000
  batchSize: 2000
  grpc:
    traceSize: 256
metrics:
  collection:
    intervalInMillis: 1000
timezone: Asia/Tokyo
java:
  options: -XX:+UseG1GC
logging:
  level:
    pctoStd: INFO
ucars:
  enabled: true
  snapshots:
    claimName: snapshot-volume
```

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。
- 特定の条件でだけ指定するパラメタについては、省略しています（(例）暗号化通信を有効にした場合だけ指定する grpc.tls 接頭辞のパラメタなど)。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.7.4 TP1-Bridge の Helm チャートのカスタマイズ（通常版かつ TP1-Bridge 限定）

TP1-Bridge を使用する場合、TP1-Bridge の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、TP1-Bridge の Helm チャートパッケージファイル（tp1-bridge-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./tp1-bridge-V.R.S.tgz > ./tp1-bridge-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace
image: myregistry.example.com/myproject/tp1-bridge:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent
dependencyCheck:
  enabled: true
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※
  imagePullPolicy: IfNotPresent
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10
  targets:
  - my-namespace/statefulsets/mediator
  - my-namespace/statefulsets/eads-server
filebeat:
  image: myregistry.example.com/myproject/filebeat:V.R.S※
  imagePullPolicy: IfNotPresent
  output:
    logstash:
      namespace: my-namespace
  livenessProbe:
    initialDelaySeconds: 10
    periodSeconds: 10
    failureThreshold: 3
    timeoutSeconds: 5
  readinessProbe:
```

```
    initialDelaySeconds: 10
    periodSeconds: 10
    failureThreshold: 3
    timeoutSeconds: 5
healthCheck:
  enabled: true
livenessProbe:
  initialDelaySeconds: 180
  periodSeconds: 30
  failureThreshold: 3
  timeoutSeconds: 10
  healthCheck:
    retryCount: 0
    retryIntervalSeconds: 0
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
postStart:
  healthCheck:
    retryCount: 30
    retryIntervalSeconds: 10
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
preStop:
  waitTransactions:
    retryCount: 30
    retryIntervalSeconds: 10
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
terminationGracePeriodSeconds: 330
affinity: {}
resources:
  limits:
    cpu: 2
    memory: 1Gi
  requests:
    cpu: 1
    memory: 512Mi
server:
  tomcat:
    threads:
      max: 200
    maxSwallowSize: "2097152"
converter:
  host: converter
  port: 8008
  connectTime: 10
  watchTime: 10
tp1:
  cltj:
    dchost: tp1
    dcrapport: 5001
    dcwatchtim: 180
    dcwatchtiminherit: "N"
    dcwatchtimrpcinherit: "N"
    dccltdelay: 0
    dcuaptrace: "N"
    dcuaptracepath: /deployments/tp1-log
    dcuaptracesize: 4096
```

```
    dcdatatrace: "N"
    dcdatatracepath: /deployments/tp1-log
    dcdatatracesize: 4096
    dcdatatracemaxsize: 128
    dcerrtrace: "N"
    dcerrtracepath: /deployments/tp1-log
    dcerrtracesize: 4096
    dcmethodtrace: "N"
    dcmethodtracepath: /deployments/tp1-log
    dcmethodtracesize: 4096
    dccltrpcmaxmsgsize: 1
    dccltsendbuffsize: 8192
    dccltrecvbuffsize: 8192
  commit:
    retryLimit: -1
    retryIntervalInMillis: 1000
  rollback:
    retryLimit: -1
    retryIntervalInMillis: 1000
  xaServerName: sample
  pool:
    allocationWaitTimeInMillis: 61000
    maxPoolSize: 10
  mediator:
    port: 20551
  pod:
    serviceName: tp1-bridge1.my-namespace
  transactionTermination:
    timeoutInMillis: 60000
    consensus:
      replayIntervalInMillis: 1000
  participantProcessing:
    threadPoolSize: 12
    queueSize: 25
externalConsensus:
  intervalInMillis: 1000
  eads:
    properties: /eads-properties/hmp-pcto.eads-client.properties
    cache: hmppctoCache
mediator:
  name: "mediator"
  nameSuffix: ".mediator.my-namespace.svc.cluster.local"
  port: 20552
mediatorProcessManagement:
  threadPoolSize: 30
mediatorCommunication:
  threadPoolSize: 30
  queueSize: 25
http:
  port: 20554
grpc:
  port: 20552
  useTls: false
  authSkip: false
  tls:
    secretName: ""
    clientAuth:
      enabled: false
```

```
    session:
      cacheSize: 0
      timeout: 0
  keepalive:
    interval: 0
    timeout: 60
  watchtime: 5
tracing:
  instanceId: ""
  url: http://jaeger-collector:14268/api/traces
  traceUse: true
  queueSize: 20000
  delayTime: 1000
  batchSize: 2000
  grpc:
    traceSize: 256
metrics:
  collection:
    intervalInMillis: 1000
timezone: Asia/Tokyo
java:
  options: -XX:+UseG1GC
logging:
  level:
    pctoStd: INFO
ucars:
  enabled: true
  snapshots:
    claimName: snapshot-volume
eads:
  client:
    properties: |
      eads.client.connect.servers=sv1,sv2,sv3
      eads.client.sv1.address=eads-service-eads-service-0.my-namespace
      eads.client.sv1.port=24600
      eads.client.sv2.address=eads-service-eads-service-1.my-namespace
      eads.client.sv2.port=24600
      eads.client.sv3.address=eads-service-eads-service-2.my-namespace
      eads.client.sv3.port=24600
      eads.client.logger.dir=eads-log
      eads.client.connectionPool.poolsize=1
```

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。
- 特定の条件でだけ指定するパラメタについては、省略しています（(例）暗号化通信を有効にした場合だけ指定する grpc.tls 接頭辞のパラメタなど)。

重要

Helm チャートのカスタマイズ時の注意事項

- Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。
- TP1-Bridge を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名の長さの合計は、40 バイト以下としてください。TP1-Bridge の Service 名は、「(15)　Kubernetes アプリケーションのデプロイ」の「表 3-20　helm コマンドに指定するコマンド引数」のリリース名を参照してください。

3.7.5　EADS の Helm チャートのカスタマイズ（通常版限定）

EADS の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、EADS の Helm チャートパッケージファイル（eads-server-chart.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./eads-server-chart.tgz > ./eads-values.yaml
```

注　EADS サーバをインストールした環境から次のディレクトリをコピーします。

/opt/hitachi/xeads/orchestration/chart

「8.3.3　EADS（通常版限定）」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例については、「8.3.3　EADS（通常版限定）」を参照してください。

EADS サーバのログ監視のための設定については、「(1)　EADS サーバのログ監視のための設定」を参照してください。

3.7.6　Ext-Cons の Helm チャートのカスタマイズ（トライアル版限定）

Ext-Cons の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Ext-Cons の Helm チャートパッケージファイル（ext-cons-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./ext-cons-V.R.S.tgz > ./ext-cons-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.4　Ext-Cons（トライアル版限定）」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace
image: myregistry.example.com/myproject/ext-cons:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent
dependencyCheck:
  enabled: true
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※
  imagePullPolicy: IfNotPresent
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10
  targets:
  - my-namespace/deployments/logstash
  - my-namespace/deployments/jaeger-query
  - my-namespace/deployments/jaeger-collector
  - my-namespace/deployments/prometheus
  - my-namespace/deployments/metricbeat
  - my-namespace/daemonsets/filebeat
affinity: {}
http:
  port: 20556
timezone: Asia/Tokyo
java:
  options: -XX:+UseG1GC
logging:
  level:
    pctoStd: INFO
```

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.7.7　Filebeat の Helm チャートのカスタマイズ

Filebeat の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Filebeat の Helm チャートパッケージファイル（filebeat-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./filebeat-V.R.S.tgz > ./filebeat-values.yaml
```

注　*V.R.S* は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.5　Filebeat」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace
image: myregistry.example.com/myproject/filebeat:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent

requests:
  cpu: ”1000m”
  memory: ”2Gi”

limits:
  cpu: ”1000m”
  memory: ”2Gi”

# How long to wait for Filebeat pods to stop gracefully
terminationGracePeriodSeconds: 30

livenessProbe:
  initialDelaySeconds: 10
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 10
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 1
hmpPcto:
  namespaces: [ ”my-namespace” ]

nodes:
  hostnames:
  - worker-1

filebeat:
  queue:
    maxSize: ”10GB”
```

```
    readAhead: "512"
  output:
    maxRetries: "3"

dependencyCheck:
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※
  imagePullPolicy: IfNotPresent
  enabled: true
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10
```

注※

V.R.S は使用する HMP-PCTO と Filebeat のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.7.8 Jaeger-query の Helm チャートのカスタマイズ

Jaeger-query の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Jaeger-query の Helm チャートパッケージファイル（jaeger-query-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./jaeger-query-V.R.S.tgz > ./jaeger-query-values.yaml
```

注　*V.R.S* は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.6　Jaeger-query」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace

requests:
  cpu: "500m"
```

```
  memory: "1Gi"

limits:
  cpu: "500m"
  memory: "1Gi"

image: myregistry.example.com/myproject/jaeger-query:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent
terminationGracePeriodSeconds: 30
http:
  port: 30015
jaegerQuery:
  maxSpanAge: 168h0m0s
livenessProbe:
  initialDelaySeconds: 30
  periodSeconds: 15
  failureThreshold: 3
  timeoutSeconds: 10
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 30
  periodSeconds: 5
  failureThreshold: 3
  timeoutSeconds: 4
  successThreshold: 1
dependencyCheck:
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※
  imagePullPolicy: IfNotPresent
  enabled: true
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10
tls:
  secretName: ""
  enabled: false

elasticUserSecretName: ""
```

注※

V.R.S は使用する HMP-PCTO と Jaeger-query のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.7.9　Jaeger-collector の Helm チャートのカスタマイズ

Jaeger-collector の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Jaeger-collector の Helm チャートパッケージファイル（Jaeger-collector-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./jaeger-collector-V.R.S.tgz > ./jaeger-collector-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.7　Jaeger-collector」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace

replicas: 1

requests:
  cpu: "1000m"
  memory: "500Mi"

limits:
  cpu: "1000m"
  memory: "500Mi"

image: myregistry.example.com/myproject/jaeger-collector:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent
terminationGracePeriodSeconds: 30
livenessProbe:
  initialDelaySeconds: 300
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 60
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 3

affinity:
  podAntiAffinity:
```

```
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app.kubernetes.io/name
            operator: In
            values:
              - jaeger-collector
        topologyKey: "kubernetes.io/hostname"

collector:
  queueSize: "10000"
  numWorkers: "50"
  numShards: "5"

dependencyCheck:
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※
  imagePullPolicy: IfNotPresent
  enabled: true
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10
tls:
  secretName: ""
  enabled: false

elasticUserSecretName: ""
```

注※

V.R.S は使用する HMP-PCTO と Jaeger-collector のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.7.10 Prometheus の Helm チャートのカスタマイズ

Prometheus の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Prometheus の Helm チャートパッケージファイル（prometheus-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./prometheus-V.R.S.tgz > ./prometheus-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.8　Prometheus」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace

image: myregistry.example.com/myproject/prometheus:V.R.S※3
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent

requests:
  cpu: "500m"
  memory: "500Mi"

limits:
  cpu: "500m"
  memory: "500Mi"

http:
  port: 30014

scrapeNamespace:
  mediator: my-namespace
  sqlparticipant: my-namespace※1
  entity: my-namespace
  orchestrator: my-namespace
  jaegercollector: my-namespace

scrapePort:
  mediator: 20554
  sqlparticipant: 20555※2
  entity: 8080
  orchestrator: 8080

terminationGracePeriodSeconds: 300

livenessProbe:
  initialDelaySeconds: 30
  periodSeconds: 15
  failureThreshold: 3
  timeoutSeconds: 10
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 30
  periodSeconds: 5
  failureThreshold: 3
  timeoutSeconds: 4
  successThreshold: 1
```

```
prometheus:
  scrape:
    interval: 15s
  storage:
    retentionTime: 7d
  remoteWrite:
    queue:
      capacity: ”10000”
      maxShards: ”50”
      maxSamplesPerSend: ”2000”

dependencyCheck:
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※3
  imagePullPolicy: IfNotPresent
  enabled: true
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10
```

注※1

SQL-Participant を使用する場合の例を記述しています。

- TCC-Participant を使用する場合は、「tccparticipant: my-namespace」に置き換えてください。
- TP1-Bridge を使用する場合は、「tp1bridge: my-namespace」に置き換えてください。

注※2

SQL-Participant を使用する場合の例を記述しています。

- TCC-Participant を使用する場合は、「tccparticipant: 8080」に置き換えてください。
- TP1-Bridge を使用する場合は、「tp1bridge: 20554」に置き換えてください。

注※3

V.R.S は使用する HMP-PCTO と Prometheus のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.7.11　Metricbeat の Helm チャートのカスタマイズ

Metricbeat の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Metricbeat の Helm チャートパッケージファイル（metricbeat-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./metricbeat-V.R.S.tgz > ./metricbeat-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.9　Metricbeat」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace
replicas: 1

image: myregistry.example.com/myproject/metricbeat:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent

requests:
  cpu: "1000m"
  memory: "2Gi"

limits:
  cpu: "1000m"
  memory: "2Gi"

livenessProbe:
  initialDelaySeconds: 10
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 10
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 1
terminationGracePeriodSeconds: 30
affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app.kubernetes.io/name
            operator: In
            values:
              - metricbeat
        topologyKey: "kubernetes.io/hostname"
```

```
metricbeat:
  queue:
    maxSize: "10GB"
    readAhead: "512"
  output:
    maxRetries: "3"
dependencyCheck:
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※
  imagePullPolicy: IfNotPresent
  enabled: true
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10
tls:
  secretName: ""
  enabled: false

elasticUserSecretName: ""
```

注※

V.R.S は使用する HMP-PCTO と Metricbeat のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.7.12 Elasticsearch の Helm チャートのカスタマイズ

Elasticsearch の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Elasticsearch の Helm チャートパッケージファイル（elasticsearch-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./elasticsearch-V.R.S.tgz > ./elasticsearch-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.10　Elasticsearch」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

クラスタ構成で構築する場合、抽出した values.yaml を基にマスターノード用 values.yaml とデータノード用 values.yaml の 2 つを用意してください。

クラスタ構成ではない場合の Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace

role: ""

requests:
  cpu: "2000m"
  memory: "4Gi"

limits:
  cpu: "2000m"
  memory: "4Gi"

replicas: 1

data:
  storageClassName: elasticsearch-data
  storageSize: 3Gi

repo:
  storageClassName: elasticsearch-repo
  storageSize: 3Gi

image: myregistry.example.com/myproject/elasticsearch:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent

http:
  port: 32580

# How long to wait for elasticsearch to stop gracefully
terminationGracePeriodSeconds: 120

sysctlVmMaxMapCount: 262144

affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app.kubernetes.io/name
            operator: In
            values:
              - elasticsearch-master
        topologyKey: "kubernetes.io/hostname"

livenessProbe:
  initialDelaySeconds: 300
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
```

```
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 60
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 3

tls:
  secretName: ""
  enabled: false

elasticUserSecretName: ""
```

注※

V.R.S は使用する Elasticsearch のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

クラスタ構成の場合のマスターノード用 Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace

role: master

requests:
  cpu: "2000m"
  memory: "4Gi"

limits:
  cpu: "2000m"
  memory: "4Gi"

replicas: 3

data:
  storageClassName: elasticsearch-data-masternode
  storageSize: 3Gi

repo:
  storageClassName: elasticsearch-repo
  storageSize: 3Gi

image: myregistry.example.com/myproject/elasticsearch:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent

http:
  port: 32580
```

```
# How long to wait for elasticsearch to stop gracefully
terminationGracePeriodSeconds: 120

sysctlVmMaxMapCount: 262144

affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app.kubernetes.io/name
            operator: In
            values:
              - elasticsearch-master
        topologyKey: "kubernetes.io/hostname"

livenessProbe:
  initialDelaySeconds: 300
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 60
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 3

tls:
  secretName: ""
  enabled: false

elasticUserSecretName: ""
```

注※

V.R.S は使用する Elasticsearch のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

クラスタ構成の場合のデータノード用 Helm チャートのカスタマイズ例を示します。

```
namespace: my-namespace

role: data

requests:
  cpu: "2000m"
  memory: "4Gi"

limits:
  cpu: "2000m"
```

```
  memory: "4Gi"

replicas: 3

data:
  storageClassName: elasticsearch-data-datanode
  storageSize: 3Gi

repo:
  storageClassName: elasticsearch-repo
  storageSize: 3Gi

image: myregistry.example.com/myproject/elasticsearch:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent

http:
  port: 32580

# How long to wait for elasticsearch to stop gracefully
terminationGracePeriodSeconds: 120

sysctlVmMaxMapCount: 262144

affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app.kubernetes.io/name
            operator: In
            values:
              - elasticsearch-data
        topologyKey: "kubernetes.io/hostname"

livenessProbe:
  initialDelaySeconds: 300
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 60
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 3

tls:
  secretName: ""
  enabled: false

elasticUserSecretName: ""
```

注※

V.R.S は使用する Elasticsearch のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.7.13 Logstash の Helm チャートのカスタマイズ

Logstash の Helm チャートをカスタマイズするために、helm show values コマンドを実行して、Logstash の Helm チャートパッケージファイル（logstash-*V.R.S*.tgz）からデフォルト設定の values.yaml を抽出してください。コマンド例を次に示します。

```
helm show values ./logstash-V.R.S.tgz > ./logstash-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

「8.3.11　Logstash」に記載されている Helm チャートの values.yaml の各パラメタについて、必須パラメタの定義、および任意パラメタを必要に応じてデフォルト値から変更してください。

Helm チャートのカスタマイズ例を次に示します。

```
namespace: my-namespace

replicas: 1

requests:
  cpu: "1000m"
  memory: "3Gi"

limits:
  cpu: "1000m"
  memory: "3Gi"

image: myregistry.example.com/myproject/logstash:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent
```

```
# How long to wait for logstash to stop gracefully
terminationGracePeriodSeconds: 120

livenessProbe:
  initialDelaySeconds: 300
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 1
readinessProbe:
  initialDelaySeconds: 60
  periodSeconds: 10
  failureThreshold: 3
  timeoutSeconds: 5
  successThreshold: 3

affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app.kubernetes.io/name
            operator: In
            values:
              - logstash
        topologyKey: "kubernetes.io/hostname"

logstash:
  pipeline:
    batchSize: "125"
  queue:
    event:
      enabled: true
      maxEvents: "0"
    byte:
      enabled: false
      maxBytes: ""

dependencyCheck:
  image: myregistry.example.com/myproject/dependency-checker:V.R.S※
  imagePullPolicy: IfNotPresent
  enabled: true
  retryCount: 30
  retryIntervalSeconds: 10
  connectTimeoutSeconds: 10
  readTimeoutSeconds: 10

tls:
  secretName: ""
  enabled: false

elasticUserSecretName: ""
```

注※

V.R.S は使用する HMP-PCTO と Logstash のバージョンに合わせて読み替えてください。

この例は、次のとおりパラメタの値を指定しています。

- 必須パラメタについては、サンプルの値を指定しています。
- 任意パラメタについては、デフォルト値を指定しています。

重要

Helm チャートのカスタマイズ時の注意事項

Helm チャートは、行頭のインデント（文字の開始位置）が厳密に評価されます。注意して編集してください。

3.8 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションの運用

Kubernetes や Helm チャートのコマンドを使用して、HMP-PCTO の Kubernetes アプリケーションを運用するための作業項目について説明します。HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションは、HMP-PCTO 主体で運用設計します。

説明中の Kubernetes のコマンド例は、kubectl コマンドを使用した場合を例に記述しています。Red Hat OpenShift Container Platform を使用している場合は、oc コマンドに読み替えてください。

kubectl コマンドや oc コマンドの使い方については、それぞれのマニュアルを参照してください。

3.8.1 作業項目の一覧

HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションを運用するための作業項目の一覧を次に示します。表の上から順に作業を実行してください。

表 3-14 HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションを運用するための作業項目の一覧

作業対象	作業項目	備考	参照先
Kubernetes 各ワーカーノード	Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（ローカル永続ボリューム（local）とする場合）	Mediator の ConsensusLog のマウント用ディレクトリをローカル永続ボリューム（local）とする場合は、必ず作成します。	(1) Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（ローカル永続ボリューム（local）とする場合）
	Elasticsearch データディレクトリの永続ボリュームマウント用ディレクトリの作成	必ず作成します。	(2) Elasticsearch データディレクトリの永続ボリュームマウント用ディレクトリの作成
	EADS の永続ボリュームマウント用ディレクトリの作成（通常版限定）	通常版の場合だけ作成します。	(3) EADS の永続ボ

作業対象	作業項目	備考	参照先
			リュームマウント用ディレクトリの作成（通常版限定）
	Elasticsearch を使用するために必要な追加手順	必ず実施します。	(4) Elasticsearch を使用するために必要な追加手順
Kubernetes コントロールプレーン	各種資材の配置	必ず実施します。	(1)　各種資材の配置
	Namespace の作成	次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	(2) Namespace の作成
	TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成	gRPC 通信機能で暗号化通信をする場合だけ実施します。	(3)　TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成
	Mediator の ConsensusLog の永続ボリュームおよびストレージクラスのデプロイ	必ず実施します。	(4) Mediator の ConsensusLog の永続ボリュームおよびストレージクラスのデプロイ
	EADS の永続ボリュームのデプロイ（通常版限定）	通常版の場合だけ実施します。	(5)　EADS の永続ボリュームのデプロイ

作業対象	作業項目	備考	参照先
			(通常版限定)
	Elasticsearch の elastic ユーザパスワード用シークレットの作成	Elasticsearch の TLS 通信を行う場合だけ作成します。	(6) Elasticsearch の elastic ユーザパスワード用シークレットの作成
	Elasticsearch の TLS 通信用シークレットの作成	Elasticsearch の TLS 通信を行う場合だけ作成します。	(7) Elasticsearch の TLS 通信用シークレットの作成
	Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスのデプロイ	必ず実施します。	(8) Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスのデプロイ
	Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスのデプロイ	必ず実施します。	(9) Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスのデプロイ
	デプロイ依存関係チェック機能用ロールおよびロールバインディングのデプロイ	次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	(10) デプロイ依存関係チェック機能用ロールおよびロールバインディングのデプロイ

作業対象	作業項目	備考	参照先
	コンテナログ収集用ロールおよびロールバインディングのデプロイ	必ず実施します。	(11) コンテナログ収集用ロールおよびロールバインディングのデプロイ
	メトリクス収集用ロールおよびロールバインディングのデプロイ	必ず実施します。	(12) メトリクス収集用ロールおよびロールバインディングのデプロイ
	メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングのデプロイ	必ず実施します。	(13) メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングのデプロイ
	uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求のデプロイ（通常版限定）	通常版の場合だけ実施します。次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に作成します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション	(14) uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求のデプロイ（通常版限定）
	Kubernetes アプリケーションのデプロイ	必ず実施します。	(15) Kubernetes アプリケーションのデプロイ

作業対象	作業項目	備考	参照先
	Elasticsearch のスナップショットリポジトリの作成およびスナップショットライフサイクルポリシーの作成	必ず実施します。	(16) Elasticsearch のスナップショットリポジトリの作成およびスナップショットライフサイクルポリシーの作成
	Elasticsearch の index 設定変更および削除	必ず実施します。	(17) Elasticsearch の index 設定変更および削除
	EADS のセットアップ（通常版限定）	通常版の場合だけ実施します。	(18) EADS のセットアップ（通常版限定）
	EADS のアンセットアップ（通常版限定）	通常版の場合だけ実施します。	(19) EADS のアンセットアップ（通常版限定）
	Kubernetes アプリケーションのアンデプロイ	必ず実施します。	(20) Kubernetes アプリケーションのアンデプロイ

3.8.2 Kubernetes 各ワーカーノードでの作業

(1) Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成（ローカル永続ボリューム（local）とする場合）

Mediator の ConsensusLog のマウント用ディレクトリをローカル永続ボリューム（local）とする場合は、次の作業を実施してください。

「3.6.2 Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で Mediator を配置する対象として決定した各ワーカーノードについて、Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリを次の要領で作成してください。

Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの作成要領

- ディレクトリパスは、「3.6.2　Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で永続ボリュームの.spec.local.path フィールドに指定したディレクトリパスとしてください。
- ディレクトリのグループに GID「0」（root）を設定してください。
- ディレクトリのパーミッションにグループの書き込み権限を付与してください。

Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリを作成するコマンド例を次に示します。

```
mkdir -p /mediator-consensus-log-volume
chgrp 0 /mediator-consensus-log-volume
chmod g+w /mediator-consensus-log-volume
```

次のコマンド例のように、Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリが期待どおり作成されていることを確認してください。

```
ls -ld /mediator-consensus-log-volume
```

＜コマンドの実行結果の例＞

```
drwxrwxr-x 2 root root 6  5月 10 12:34 /mediator-consensus-log-volume
```

(2)　Elasticsearch データディレクトリの永続ボリュームマウント用ディレクトリの作成

「3.6.3　Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で Elasticsearch を配置する対象として決定した各ワーカーノードについて、Elasticsearch データディレクトリの永続ボリュームマウント用ディレクトリを次の要領で作成してください。

Elasticsearch データディレクトリの永続ボリュームマウント用ディレクトリの作成要領

- Elasticsearch の永続ボリューム専用パーティションを作成してください。
- ディレクトリに、Elasticsearch の永続ボリューム専用パーティションの割り当てを実施してください。ディレクトリパスは、「3.6.3　Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で永続ボリュームの.spec.local.path フィールドに指定したディレクトリパスとしてください。
- ディレクトリのグループに GID「0」(root)を設定してください。
- ディレクトリのパーミッションにグループの書き込み権限を付与してください。

Elasticsearch データディレクトリの永続ボリュームマウント用ディレクトリを作成するコマンド例を次に示します。

```
mkdir -p /elasticsearch-data-volume-masternode
mkdir -p /elasticsearch-data-volume-datanode
chown 1000:0 /elasticsearch-data-volume-masternode
chown 1000:0 /elasticsearch-data-volume-datanode
chmod g+w /elasticsearch-data-volume-masternode
chmod g+w /elasticsearch-data-volume-datanode
```

次のコマンド例のように、Elasticsearch データディレクトリの永続ボリュームマウント用ディレクトリが期待どおり作成されていることを確認してください。

```
ls -ld /elasticsearch-data-volume-masternode /elasticsearch-data-volume-datanode
```

＜コマンドの実行結果の例＞

```
drwxrwxr-x 2 user root 6  5月 10 12:34 /elasticsearch-data-volume-masternode
drwxrwxr-x 2 user root 6  5月 10 12:35 /elasticsearch-data-volume-datanode
```

(3)　EADS の永続ボリュームマウント用ディレクトリの作成（通常版限定）

「3.6.5　EADS の永続ボリュームの Kubernetes マニフェストの作成（通常版限定）」で EADS を配置する対象として決定した各ノードについて、EADS の永続ボリュームマウント用ディレクトリを次の要領で作成してください。

EADS の永続ボリュームマウント用ディレクトリの作成要領

- ディレクトリパスは、「3.6.5　EADS の永続ボリュームの Kubernetes マニフェストの作成（通常版限定）」で永続ボリュームの.spec.local.path フィールドに指定したディレクトリパスとしてください。
- ディレクトリのグループに GID「0」(root)を設定してください。
- ディレクトリのパーミッションにグループの書き込み権限を付与してください。

EADS の永続ボリュームマウント用ディレクトリを作成するコマンド例を次に示します。

```
mkdir -p /eads-cache-volume
mkdir -p /eads-logs-volume
mkdir -p /eads-store-volume
chgrp 0 /eads-cache-volume
chgrp 0 /eads-logs-volume
chgrp 0 /eads-store-volume
chmod g+w /eads-cache-volume
chmod g+w /eads-logs-volume
chmod g+w /eads-store-volume
```

次のコマンド例のように、EADS の永続ボリュームマウント用ディレクトリが期待どおり作成されていることを確認してください。

```
ls -ld /eads-cache-volume /eads-logs-volume /eads-store-volume
```

＜コマンドの実行結果の例＞

```
drwxrwxr-x. 2 root root 6  6月 27 07:25 /eads-cache-volume
drwxrwxr-x. 2 root root 6  6月 27 07:25 /eads-logs-volume
drwxrwxr-x. 2 root root 6  6月 27 07:25 /eads-store-volume
```

(4)　Elasticsearch を使用するために必要な追加手順

Elasticsearch を使用するためには、コンテナランタイム containerd に systemd cgroup ドライバを構成する必要があります。

containerd に systemd cgroup ドライバを構成するための作業要領

- containerd のデフォルト設定を、ファイル「/etc/containerd/config.toml」に出力してください。

```
containerd config default > /etc/containerd/config.toml
```

- config.toml の内容で、[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options] の SystemdCgroup を true にしてください。

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
SystemdCgroup = true
```

- 「systemctl daemon-reload」を実行したあと、containerd を再開始してください。

```
systemctl daemon-reload
systemctl restart containerd
```

3.8.3　Kubernetes コントロールプレーンでの作業

(1)　各種資材の配置

Kubernetes コントロールプレーンの任意のパスへ、次の各種資材を配置してください。

表 3-15　Kubernetes コントロールプレーンに配置する資材の一覧

資材	備考
Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェスト	「3.6.2　Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で作成したもの
Mediator の Helm チャートパッケージファイル(mediator-*V.R.S*.tgz)	HMP-PCTO 提供物
Mediator の Helm チャートの values.yaml	「3.7.3　Mediator の Helm チャートのカスタマイズ」でカスタマイズしたもの
TP1-Bridge の Helm チャートパッケージファイル（tp1-bridge-*V.R.S*.tgz）	HMP-PCTO 提供物

資材	備考
	通常版の場合で、TP1-Bridge を使用するときだけ配置します。
TP1-Bridge の Helm チャートの values.yaml	「3.7.4　TP1-Bridge の Helm チャートのカスタマイズ（通常版かつ TP1-Bridge 限定）」でカスタマイズしたもの 通常版の場合で、TP1-Bridge を使用するときだけ配置します。
EADS の永続ボリュームの Kubernetes マニフェスト	「3.6.5　EADS の永続ボリュームの Kubernetes マニフェストの作成（通常版限定）」で作成したもの 通常版の場合だけ配置します。
EADS の Helm チャートパッケージファイル(eads-service-chart.tgz、eads-server-chart.tgz、eads-command-job-chart.tgz、eads-command-cronjob-chart.tgz)	EADS 提供物
EADS の Helm チャートの values.yaml	「3.7.5　EADS の Helm チャートのカスタマイズ（通常版限定）」でカスタマイズしたもの
Ext-Cons の Helm チャートパッケージファイル(ext-cons-*V.R.S*.tgz)	HMP-PCTO 提供物
Ext-Cons の Helm チャートの values.yaml	「3.7.6　Ext-Cons の Helm チャートのカスタマイズ（トライアル版限定）」でカスタマイズしたもの
Filebeat の Helm チャートのパッケージファイル(filebeat-*V.R.S*.tgz)	HMP-PCTO 提供物
Filebeat の Helm チャートの values.yaml	「3.7.7　Filebeat の Helm チャートのカスタマイズ」でカスタマイズしたもの
Jaeger-query の Helm チャートのパッケージファイル(jaeger-query-*V.R.S*.tgz)	HMP-PCTO 提供物
Jaeger-query の Helm チャートの values.yaml	「3.7.8　Jaeger-query の Helm チャートのカスタマイズ」でカスタマイズしたもの
Jaeger-collector の Helm チャートのパッケージファイル(jaeger-collector-*V.R.S*.tgz)	HMP-PCTO 提供物
Jaeger-collector の Helm チャートの values.yaml	「3.7.9　Jaeger-collector の Helm チャートのカスタマイズ」でカスタマイズしたもの
Prometheus の Helm チャートのパッケージファイル(prometheus-*V.R.S*.tgz)	HMP-PCTO 提供物
Prometheus の Helm チャートの values.yaml	「3.7.10　Prometheus の Helm チャートのカスタマイズ」でカスタマイズしたもの
Metricbeat の Helm チャートのパッケージファイル(metricbeat-*V.R.S*.tgz)	HMP-PCTO 提供物
Metricbeat の Helm チャートの values.yaml	「3.7.11　Metricbeat の Helm チャートのカスタマイズ」でカスタマイズしたもの

資材	備考
Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェスト	「3.6.3　Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で作成したもの
Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェスト	「3.6.4　Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で作成したもの
Elasticsearch の Helm チャートのパッケージファイル(elasticsearch-*V.R.S*.tgz)	HMP-PCTO 提供物
Elasticsearch の Helm チャートの values.yaml	「3.7.12　Elasticsearch の Helm チャートのカスタマイズ」でカスタマイズしたもの
Logstash の Helm チャートのパッケージファイル(logstash-*V.R.S*.tgz)	HMP-PCTO 提供物
Logstash の Helm チャートの values.yaml	「3.7.13　Logstash の Helm チャートのカスタマイズ」でカスタマイズしたもの
デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェスト	「3.6.6　デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの作成」で作成したもの 次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に配置します。 • ユーザ責務の Kubernetes アプリケーション • HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション
コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェスト	「3.6.7　コンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成」で作成したもの
メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェスト	「3.6.8　メトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストの作成」で作成したもの
メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェスト	「3.6.9　メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストの作成」で作成したもの
uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェスト	「3.6.10　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」で作成したもの 次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に配置します。 • ユーザ責務の Kubernetes アプリケーション

資材	備考
	• HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

(2) Namespace の作成

HMP-PCTO の Kubernetes アプリケーションをデプロイする Namespace を作成してください。この作業は、次のどちらかの Kubernetes アプリケーションをデプロイするための作業時に実施します。

- ユーザ責務の Kubernetes アプリケーション
- HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーション

Namespace の名称は任意です。また、Namespace は、1 つでも、複数の Namespace に分けても、各 Kubernetes アプリケーションをデプロイできます。Namespace の設計・運用はユーザの要件に応じて対応してください。

ただし、RAS の Kubernetes アプリケーションの場合、デプロイする Namespace は同一にする必要があります。RAS の Kubernetes アプリケーションについては、「1.1　HMP-PCTO の Kubernetes アプリケーションの一覧と分類」の「表 1-1　HMP-PCTO の Kubernetes アプリケーションの一覧と分類」を参照してください。

なお、ここで示すコマンド実行例や Kubernetes マニフェスト記述例は、1 つの Namespace で運用するケースで説明しています。

HMP-PCTO の Kubernetes アプリケーションをデプロイする Namespace を「my-namespace」としたときのコマンド例を示します。

```
kubectl create namespace my-namespace
```

＜コマンドの実行結果の例＞

```
namespace/my-namespace created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

(3) TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成

HMP-PCTO の gRPC 通信機能で暗号化通信をする場合だけ実施します。

Kubernetes コントロールプレーンの任意のパスへ、次のとおり HMP-PCTO の gRPC 通信機能の暗号化通信に必要なファイルを準備してください。なお、Participant（TP1-Bridge）の場合、TP1-Bridge のアプリケーションの数だけ準備してください。

表 3-16　HMP-PCTO の gRPC 通信機能の暗号化通信に必要なファイルの一覧

ファイル	備考
接続先サーバのサーバ証明書を発行した CA の証明書(PEM 形式)ファイル	必ず準備します。
自サーバ証明書（PEM 形式）ファイル	必ず準備します。
自サーバ証明書を作成した秘密鍵ファイル	必ず準備します。
自クライアント証明書ファイル（PEM 形式）	クライアント認証を行う場合だけ準備します。
自クライアント証明書を作成した秘密鍵ファイル	クライアント認証を行う場合だけ準備します。
接続元クライアント（Orchestrator、Participant）の秘密鍵が格納されているキーストア（pkcs12、または JKS）ファイル	クライアント認証を行う場合だけ準備します。

HMP-PCTO の gRPC 通信機能の暗号化通信に必要なファイルを準備したディレクトリ/ファイル構成の例を次に示します。

クライアント認証を行わない場合

```
<任意のパス>/
    ca.crt              # 接続先サーバのサーバ証明書を発行したCAの証明書(PEM形式)ファイル
    participant/
        server.crt      # Participantの自サーバ証明書(PEM形式)ファイル
        server.key      # Participantの自サーバ証明書を作成した秘密鍵ファイル
    mediator/
        server.crt      # Mediatorの自サーバ証明書(PEM形式)ファイル
        server.key      # Mediatorの自サーバ証明書を作成した秘密鍵ファイル
```

クライアント認証を行う場合

```
<任意のパス>/
    ca.crt              # 接続先サーバのサーバ証明書を発行したCAの証明書(PEM形式)ファイル
    participant/
        server.crt      # Participantの自サーバ証明書(PEM形式)ファイル
        server.key      # Participantの自サーバ証明書を作成した秘密鍵ファイル
        client.crt      # Participantの自クライアント証明書ファイル(PEM形式)
        client.key      # Participantの自クライアント証明書を作成した秘密鍵ファイル
        keystore.jks    # Participantの接続元クライアント(orchestrator,participant)の秘密鍵
が格納されている
                        # キーストア(pkcs12またはJKS)ファイル
    mediator/
        server.crt      # Mediatorの自サーバ証明書(PEM形式)ファイル
        server.key      # Mediatorの自サーバ証明書を作成した秘密鍵ファイル
        client.crt      # Mediatorの自クライアント証明書ファイル(PEM形式)
        client.key      # Mediatorの自クライアント証明書を作成した秘密鍵ファイル
        keystore.jks    # Mediatorの接続元クライアント(Orchestrator,Participant)の秘密鍵が格
納されている
                        # キーストア(pkcs12またはJKS)ファイル
```

準備したファイルを使用して TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットを作成します。

TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成要領を次に示します。

表 3-17　TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成要領

キー名	ファイル
ca.crt	接続先サーバのサーバ証明書を発行した CA の証明書（PEM 形式）ファイル
server.crt	自サーバ証明書（PEM 形式）ファイル
server.key	自サーバ証明書を作成した秘密鍵ファイル
client.crt	自クライアント証明書ファイル（PEM 形式）
client.key	自クライアント証明書を作成した秘密鍵ファイル
keystore	接続元クライアント（Orchestrator、Participant）の秘密鍵が格納されているキーストア（pkcs12、または JKS）ファイル
keystore_password	キーストアファイルのパスワード

TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットを作成するコマンド例を次に示します。

- TP1-Bridge シークレット名：「participant-secret」
- Mediator 用シークレット名：「mediator-secret」
- HMP-PCTO の Kubernetes アプリケーションをデプロイする Namespace：「my-namespace」

クライアント認証を行わない場合

```
kubectl create secret generic --save-config participant-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  --from-file=server.crt=./participant/server.crt ¥
  --from-file=server.key=./participant/server.key ¥
  -n my-namespace

kubectl create secret generic --save-config mediator-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  --from-file=server.crt=./mediator/server.crt ¥
  --from-file=server.key=./mediator/server.key ¥
  -n my-namespace
```

クライアント認証を行う場合

```
kubectl create secret generic --save-config participant-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  --from-file=server.crt=./participant/server.crt ¥
  --from-file=server.key=./participant/server.key ¥
  --from-file=client.crt=./participant/client.crt ¥
  --from-file=client.key=./participant/client.key ¥
```

```
  --from-file=keystore=./participant/keystore.jks ¥
  --from-literal=keystore_password=changeit ¥
  -n my-namespace

kubectl create secret generic --save-config mediator-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  --from-file=server.crt=./mediator/server.crt ¥
  --from-file=server.key=./mediator/server.key ¥
  --from-file=client.crt=./mediator/client.crt ¥
  --from-file=client.key=./mediator/client.key ¥
  --from-file=keystore=./mediator/keystore.jks ¥
  --from-literal=keystore_password=changeit ¥
  -n my-namespace
```

＜コマンドの実行結果の例＞

```
secret/participant-secret created

secret/mediator-secret created
```

このように、Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

(4) Mediator の ConsensusLog の永続ボリュームおよびストレージクラスのデプロイ

「(1) 各種資材の配置」で配置した Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストをデプロイしてください。

コマンド例を次に示します。

```
kubectl apply -f ./mediator-consensus-log-volume.yaml
```

この例では、Kubernetes マニフェストのファイル名を「mediator-consensus-log-volume.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
storageclass.storage.k8s.io/mediator-consensus-log created
persistentvolume/mediator-consensus-log-0 created
persistentvolume/mediator-consensus-log-1 created
persistentvolume/mediator-consensus-log-2 created
persistentvolume/mediator-consensus-log-3 created
persistentvolume/mediator-consensus-log-4 created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

Mediator の ConsensusLog の永続ボリュームをデプロイしたあと、次のコマンド例のように、Mediator の ConsensusLog の永続ボリュームのデプロイが期待どおり成功していること（STATUS 列が"Available"になっていること）を確認してください。

```
kubectl get pv
```

＜コマンドの実行結果の例＞

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
mediator-consensus-log-0	2Mi	RWO	Retain	Available		mediator-consensus-log		9s
mediator-consensus-log-1	2Mi	RWO	Retain	Available		mediator-consensus-log		9s
mediator-consensus-log-2	2Mi	RWO	Retain	Available		mediator-consensus-log		9s
mediator-consensus-log-3	2Mi	RWO	Retain	Available		mediator-consensus-log		9s
mediator-consensus-log-4	2Mi	RWO	Retain	Available		mediator-consensus-log		9s

上記の例では、Mediator の ConsensusLog の永続ボリュームだけを記載しています。実行環境によっては、そのほかに存在する永続ボリュームも一覧に表示されます。

なお、各列に表示される値は実行環境によって異なります。また、kubectl の describe サブコマンドを使用すると、各 Kubernetes オブジェクトのさらに詳細な情報を参照できます。

(5)　EADS の永続ボリュームのデプロイ（通常版限定）

「(1)　各種資材の配置」で配置した EADS の永続ボリュームの Kubernetes マニフェストをデプロイしてください。

コマンド例を次に示します。

```
kubectl apply -f ./eads-volume.yaml
```

この例では、EADS の永続ボリュームの Kubernetes マニフェストのファイル名を「eads-volume.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
persistentvolume/eads-cache-1 created
persistentvolume/eads-logs-1 created
persistentvolume/eads-store-1 created
persistentvolume/eads-cache-2 created
persistentvolume/eads-logs-2 created
persistentvolume/eads-store-2 created
persistentvolume/eads-cache-3 created
```

```
persistentvolume/eads-logs-3 created
persistentvolume/eads-store-3 created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

(6) Elasticsearch の elastic ユーザパスワード用シークレットの作成

Elasticsearch の elastic ユーザのパスワードを設定するためのシークレットを作成してください。この作業は、Elasticsearch の TLS 通信を行う場合だけ実施します。

次の例で Elasticsearch の elastic ユーザパスワード用シークレットを作成するコマンド例を示します。

- Elasticsearch の elastic ユーザパスワード用シークレット名：「user-secret」
- elastic ユーザパスワード：「password」
- HMP-PCTO の Kubernetes アプリケーションをデプロイする Namespace：「my-namespace」

```
echo -n password > ./password
kubectl create secret generic --save-config user-secret --from-file=./password -n my-namespa
ce
```

(7) Elasticsearch の TLS 通信用シークレットの作成

Kubernetes コントロールプレーンの任意のパスへ、次のとおり Elasticsearch の TLS 通信に必要なファイルを準備してください。この作業は、Elasticsearch の TLS 通信を行う場合だけ実施します。

表 3-18 HMP-PCTO の Elasticsearch の TLS 通信に必要なファイルの一覧

対象 Kubernetes アプリケーション	ファイル	備考
Elasticsearch	Elasticsearch のサーバ証明書を発行した CA の証明書（PEM 形式）ファイル	-
	自サーバ証明書（PEM 形式）ファイル	拡張属性の Extended Key Usage を使用して、クライアント認証も可能な証明書として作成してください。
	自サーバ証明書を作成した秘密鍵ファイル	-

HMP-PCTO の Elasticsearch の TLS 通信に必要なファイルを準備したディレクトリ/ファイル構成の例を次に示します。

```
<任意のパス>/
    ca.crt              # 接続先サーバのサーバ証明書を発行したCAの証明書(PEM形式)ファイル
    elasticsearch/
```

```
elastic.crt       # 自サーバ証明書(PEM形式)ファイル
elastic.key       # 自サーバ証明書を作成した秘密鍵ファイル
```

準備したファイルを使用して Elasticsearch の TLS 通信用シークレットを作成します。

Elasticsearch 用シークレット名は elasticsearch-secret で作成してください。

Elasticsearch の TLS 通信用シークレットの作成要領を次に示します。

表 3-19　Elasticsearch の TLS 通信用シークレットの作成要領

キー名	ファイル
ca.crt	接続先サーバのサーバ証明書を発行した CA の証明書(PEM 形式)ファイル
elastic.crt	自サーバ証明書(PEM 形式)ファイル
elastic.key	自サーバ証明書を作成した秘密鍵ファイル

Elasticsearch の TLS 通信用シークレットを作成するコマンド例を次に示します。

- Elasticsearch 用シークレット名：「elasticsearch-secret」
- HMP-PCTO の Kubernetes アプリケーションをデプロイする Namespace：「my-namespace」

```
kubectl create secret generic --save-config elasticsearch-secret ¥
  --from-file=ca.crt=./ca.crt ¥
  --from-file=elastic.crt=./elasticsearch/elastic.crt ¥
  --from-file=elastic.key=./elasticsearch/elastic.key ¥
  -n my-namespace
```

＜コマンドの実行結果の例＞

```
secret/elasticsearch-secret created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

(8)　Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスのデプロイ

「(1)　各種資材の配置」で配置した Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストをデプロイしてください。

コマンド例を次に示します。

```
kubectl apply -f ./elasticsearch-data-volume.yaml
```

この例では、Kubernetes マニフェストのファイル名を「elasticsearch-data-volume.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
storageclass.storage.k8s.io/elasticsearch-data-masternode created
storageclass.storage.k8s.io/elasticsearch-data-datanode created
persistentvolume/elasticsearch-data-masternode-0 created
persistentvolume/elasticsearch-data-masternode-1 created
persistentvolume/elasticsearch-data-masternode-2 created
persistentvolume/elasticsearch-data-datanode-0 created
persistentvolume/elasticsearch-data-datanode-1 created
persistentvolume/elasticsearch-data-datanode-2 created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

Elasticsearch データディレクトリの永続ボリュームをデプロイしたあと、次のコマンド例のように、Elasticsearch データディレクトリの永続ボリュームのデプロイが期待どおり成功していること（STATUS 列が"Available"になっていること）を確認してください。

```
kubectl get pv
```

＜コマンドの実行結果の例＞

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
elasticsearch-data-masternode-0	3Gi	RWO	Retain	Available		elasticsearch-data-masternode		9s
elasticsearch-data-masternode-1	3Gi	RWO	Retain	Available		elasticsearch-data-masternode		9s
elasticsearch-data-masternode-2	3Gi	RWO	Retain	Available		elasticsearch-data-masternode		9s
elasticsearch-data-datanode-0	3Gi	RWO	Retain	Available		elasticsearch-data-datanode		9s
elasticsearch-data-datanode-1	3Gi	RWO	Retain	Available		elasticsearch-data-datanode		9s
elasticsearch-data-datanode-2	3Gi	RWO	Retain	Available		elasticsearch-data-datanode		9s

上記の例では、Elasticsearch データディレクトリの永続ボリュームだけを記載しています。実行環境によっては、そのほかに存在する永続ボリュームも一覧に表示されます。

なお、各列に表示される値は実行環境によって異なります。また、kubectl の describe サブコマンドを使用すると、各 Kubernetes オブジェクトのさらに詳細な情報を参照できます。

(9) Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスのデプロイ

「(1) 各種資材の配置」で配置した Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストをデプロイしてください。

コマンド例を次に示します。

```
kubectl apply -f ./elasticsearch-repo-volume.yaml
```

この例では、Kubernetes マニフェストのファイル名を「elasticsearch-repo-volume.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
storageclass.storage.k8s.io/elasticsearch-repo created
persistentvolume/elasticsearch-repo-master-0 created
persistentvolume/elasticsearch-repo-master-1 created
persistentvolume/elasticsearch-repo-master-2 created
persistentvolume/elasticsearch-repo-data-0 created
persistentvolume/elasticsearch-repo-data-1 created
persistentvolume/elasticsearch-repo-data-2 created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

Elasticsearch リポジトリディレクトリの永続ボリュームをデプロイしたあと、次のコマンド例のように、Elasticsearch リポジトリディレクトリの永続ボリュームのデプロイが期待どおり成功していること（STATUS 列が"Available"になっていること）を確認してください。

```
kubectl get pv
```

＜コマンドの実行結果の例＞

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
elasticsearch-repo-master-0	3Gi	RWX	Retain	Available		elasticsearch-repo		9s
elasticsearch-repo-master-1	3Gi	RWX	Retain	Available		elasticsearch-repo		9s
elasticsearch-repo-master-2	3Gi	RWX	Retain	Available		elasticsearch-repo		9s
elasticsearch-repo-data-0	3Gi	RWX	Retain	Available		elasticsearch-repo		9s
elasticsearch-repo-data-1	3Gi	RWX	Retain	Available		elasticsearch-repo		9s
elasticsearch-repo-data-2	3Gi	RWX	Retain	Available		elasticsearch-repo		9s

上記の例では、Elasticsearch リポジトリディレクトリの永続ボリュームだけを記載しています。実行環境によっては、そのほかに存在する永続ボリュームも一覧に表示されます。

なお、各列に表示される値は実行環境によって異なります。また、kubectl の describe サブコマンドを使用すると、各 Kubernetes オブジェクトのさらに詳細な情報を参照できます。

(10) デプロイ依存関係チェック機能用ロールおよびロールバインディングのデプロイ

「(1) 各種資材の配置」で配置したデプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストをデプロイしてください。この作業は、ユーザ責務の Kubernetes アプリケーション、または HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのどちらかをデプロイするための作業時に実施します。

コマンド例を次に示します。

```
kubectl apply -f ./dependency-checker-role.yaml
```

この例では、Kubernetes マニフェストのファイル名を「dependency-checker-role.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
role.rbac.authorization.k8s.io/dependency-checker-role created
rolebinding.rbac.authorization.k8s.io/dependency-checker-role-binding created
```

このように、各 Kubernetes オブジェクトの作成に成功したメッセージが出力されていることを確認してください。

(11) コンテナログ収集用ロールおよびロールバインディングのデプロイ

「(1) 各種資材の配置」で配置したコンテナログ収集用ロールおよびロールバインディングの Kubernetes マニフェストをデプロイしてください。

次にコマンド例を示します。

```
kubectl apply -f ./filebeat-role.yaml
```

この例では、Kubernetes マニフェストのファイル名を「filebeat-role.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
clusterrole.rbac.authorization.k8s.io/filebeat-role created
clusterrolebinding.rbac.authorization.k8s.io/filebeat-role-binding created
```

上記のように各 Kubernetes オブジェクトの作成に成功した旨のメッセージが出力されていることを確認してください。

(12)　メトリクス収集用ロールおよびロールバインディングのデプロイ

「(1)　各種資材の配置」で配置したメトリクス収集用ロールおよびロールバインディングの Kubernetes マニフェストをデプロイしてください。

次にコマンド例を示します。

```
kubectl apply -f ./prometheus-role.yaml
```

この例では、Kubernetes マニフェストのファイル名を「prometheus-role.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
clusterrole.rbac.authorization.k8s.io/prometheus-role created
clusterrolebinding.rbac.authorization.k8s.io/prometheus-role-binding created
```

上記のように各 Kubernetes オブジェクトの作成に成功した旨のメッセージが出力されていることを確認してください。

(13)　メトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングのデプロイ

「(1)　各種資材の配置」で配置したメトリクス向け Kubernetes メタデータ付与用ロールおよびロールバインディングの Kubernetes マニフェストをデプロイしてください。

次にコマンド例を示します。

```
kubectl apply -f ./metricbeat-role.yaml
```

この例では、Kubernetes マニフェストのファイル名を「metricbeat-role.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
clusterrole.rbac.authorization.k8s.io/metricbeat-role created
clusterrolebinding.rbac.authorization.k8s.io/metricbeat-role-binding created
```

上記のように各 Kubernetes オブジェクトの作成に成功した旨のメッセージが出力されていることを確認してください。

(14) uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求のデプロイ（通常版限定）

「(1) 各種資材の配置」で配置した uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストをデプロイしてください。この作業は、ユーザ責務の Kubernetes アプリケーション、または HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのどちらかをデプロイするための作業時に実施します。

次にコマンド例を示します。

```
kubectl apply -f ./ucars-snapshots.yaml
```

この例では、Kubernetes マニフェストのファイル名を「ucars-snapshots.yaml」としています。また、このファイルには、各 Kubernetes マニフェストがすべて 1 つのファイル内に定義されている例としています。

もし、各 Kubernetes マニフェストを別々のファイルで作成している場合は、作成した Kubernetes マニフェストのファイルの分だけデプロイのコマンドを実行してください。

＜コマンドの実行結果の例＞

```
persistentvolume/snapshot-volume created
persistentvolumeclaim/snapshot-volume created
```

上記のように各 Kubernetes オブジェクトの作成に成功した旨のメッセージが出力されていることを確認してください。

uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームをデプロイしたあと、次のコマンド例のように、uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームのデプロイが期待どおり成功していること（STATUS 列が"Available"になっていること）を確認してください。

```
kubectl get pv
```

＜コマンドの実行結果の例＞

```
NAME                      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS          REASON   AGE
 snapshot-volume           1Gi        RWX            Retain           Available                                          9s
```

上記の例では、uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームだけを記載しています。実行環境によっては、そのほかに存在する永続ボリュームも一覧に表示されます。

なお、各列に表示される値は実行環境によって異なります。また、kubectl の describe サブコマンドを使用すると、各 Kubernetes オブジェクトのさらに詳細な情報を参照できます。

(15)　Kubernetes アプリケーションのデプロイ

「(1)　各種資材の配置」で配置した各 Kubernetes アプリケーションの Kubernetes マニフェストおよび Helm チャートをデプロイしてください。

なお、HMP-PCTO の各 Kubernetes アプリケーションは次の順序でデプロイする必要があります。

1. Elasticsearch
2. Logstash、Jaeger-query、Jaeger-collector、Prometheus、Metricbeat、Filebeat
3. Mediator、EADS、Ext-Cons（トライアル版だけ）、Participant が使用する DB/外部サービス
4. Participant※
5. Orchestrator

注※

TP1 連携機能を使用する場合、TP1-Bridge をデプロイする前に、OpenTP1 の rap リスナーと rap サーバを起動してください。

上記以外の順序でデプロイした場合

各 Kubernetes アプリケーションの初期化コンテナとして構成している Dependency-Checker コンテナで、上記のとおり依存する Kubernetes アプリケーションが先にデプロイされるまで待機します。待機処理がリトライオーバーし、初期化コンテナが異常終了した場合、Pod の状態は CrashLoopBackOff になります。その場合は、Dependency-Checker コンテナが実行しているデプロイ依存関係チェックスクリプトのエラーメッセージに従い対処してください。

HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションは、次のとおり helm コマンドでデプロイしてください。

```
helm△install△<リリース名>△<Helmチャートパッケージファイル名>△-f△<values.yamlファイル名>
```

（凡例）△：半角スペース 1 文字

<values.yaml ファイル名>は、ユーザがカスタマイズした values.yaml ファイル名を指定してください。

<リリース名>および<Helm チャートパッケージファイル名>は、次のとおり指定してください。

Elasticsearch をクラスタ構成で構築する場合は、マスターノード用 values.yaml を使用して helm コマンドを実行し、その後データノード用 values.yaml を使用して helm コマンドを実行してください。

表 3-20　helm コマンドに指定するコマンド引数

Kubernetes アプリケーション	リリース名	Helm チャートパッケージファイル名
TP1-Bridge	tp1-bridge<通番> <通番>には、デプロイする tp1-bridge ごとに一意な番号を付与してください。 例 tp1-bridge を 10 台起動する場合 1 台目：tp1-bridge1 2 台目：tp1-bridge2 3 台目：tp1-bridge3 ： 10 台目：tp1-bridge10	tp1-bridge-*V.R.S*※2.tgz
Mediator	Mediator	mediator-*V.R.S*※2.tgz
Ext-Cons	ext-cons	ext-cons-*V.R.S*※2.tgz
Filebeat	filebeat	filebeat-*V.R.S*※2.tgz
Jaeger-query	jaeger-query	jaeger-query-*V.R.S*※2.tgz
Jaeger-collector	jaeger-collector	jaeger-collector-*V.R.S*※2.tgz
Prometheus	prometheus	prometheus-*V.R.S*※2.tgz
Metricbeat	metricbeat	metricbeat-*V.R.S*※2.tgz
Elasticsearch	クラスタ構成ではない場合： elasticsearch クラスタ構成の場合： elasticsearch-master(マスターノード) elasticsearch-data(データノード)	elasticsearch-*V.R.S*※2.tgz
Logstash	logstash	logstash-*V.R.S*※2.tgz
eads※1	eads-service	eads-service-chart.tgz
	eads-server	eads-server-chart.tgz

注※1

eads-service-chart.tgz→eads-server-chart.tgz の順番でデプロイしてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

Mediator をデプロイするコマンド例を次に示します。

```
helm install mediator ./mediator-V.R.S.tgz -f ./mediator-values.yaml
```

注　V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

＜コマンドの実行結果の例＞

```
NAME: mediator
LAST DEPLOYED: Thu May 11 12:34:56 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

このように、Helm チャートのインストールに成功したメッセージが出力されていることを確認してください。なお、コマンドの実行結果の中で「NAMESPACE:」部分に出力される Namespace 名は、values.yaml ファイルに記述した Namespace 名ではなく、このコマンドを実行した Kubernetes コントロールプレーンのデフォルト Namespace 名が表示されます(上記例では"default")。実際には、values.yaml ファイルに記述した Namespace 名で Kubernetes アプリケーションがデプロイされています。

RAS の Kubernetes アプリケーションをデプロイしたあと、次のコマンド例のように、各 Kubernetes アプリケーションが Elasticsearch と接続できることを確認してください。

- Logstash の確認

  ```
  kubectl -n my-namespace exec -it logstash-7765767b94-25wrc -- curl --cacert /usr/share/logstash/config/certs/ca.crt --key /usr/share/logstash/config/certs/elastic.key --cert /usr/share/logstash/config/certs/elastic.crt -u elastic:password -XGET 'https://elasticsearch-master:9200/_cat/health?v'
  ```

  Namespace 名が my-namespace、Logstash の Pod 名が logstash-7765767b94-25wrc、Elasticsearch の elastic ユーザパスワードが password の場合の例です。

  ＜コマンドの実行結果の例＞

```
Defaulted container "logstash" out of: logstash, dependency-checker (init)
epoch      timestamp cluster       status node.total node.data shards pri relo init unassign pending_tasks max_task_wait_time active_shards_percent
1694842610 05:36:50  elasticsearch green           6         3     22  11    0    0        0             0                  -                100.0%
```

  上記のように Elasticsearch の状態が示されていることを確認してください。

- Metricbeat の確認

  ```
  kubectl -n my-namespace exec -it metricbeat-dffb85bc6-w5jlc -- metricbeat test output
  ```

  Namespace 名が my-namespace、Logstash の Pod 名が metricbeat-dffb85bc6-w5jlc の場合の例です。

＜コマンドの実行結果の例＞

```
Defaulted container "metricbeat" out of: metricbeat, dependency-checker (init)
elasticsearch: https://elasticsearch-master:9200...
  parse url... OK
  connection...
    parse host... OK
    dns lookup... OK
    addresses: 10.98.47.178
    dial up... OK
  TLS...
    security: server's certificate chain verification is enabled
    handshake... OK
    TLS version: TLSv1.3
    dial up... OK
  talk to server... OK
  version: V.R.S※
```

注※

V.R.S は使用する Metricbeat のバージョンに合わせて読み替えてください。

上記のように「talk to server」が OK になっていることを確認してください。

- Jaeger-collector および Jaeger-query

  Jaeger-collector および Jaeger-query の起動の際に Elasticsearch との接続確認が行われます。次のコマンド例のように、各 Kubernetes アプリケーションの STATUS が"Running"になっていることを確認してください。

  ```
  kubectl get pods -n my-namespace -o wide
  ```

  ＜コマンドの実行結果の例＞

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
jaeger-collector-68bfc6fb87-c6tlm	1/1	Running	0	95s	192.168.125.136	worker-1	<none>	<none>
jaeger-query-6fb78c549f-tqgw2	1/1	Running	0	72s	192.168.125.145	worker-1	<none>	<none>

  なお、各列に表示される値は実行環境によって異なります。

HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションをデプロイしたあと、次のコマンド例のように、各 Kubernetes アプリケーションが期待どおり起動していること（STATUS 列が"Running"になっていることや、NODE 列で Pod が意図したワーカーノードに割り当てられていることなど）を確認してください。

```
kubectl get pods -n my-namespace -o wide
```

＜コマンドの実行結果の例（SQL-Participant を使用する場合）＞

```
NAMESPACE      NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
my-namespace   sql-participant-94497bdcf-7btxm     1/1     Running   0          60s   192.168.24.40   worker-1   <none>           <none>
my-namespace   orchestrator-7599cdd5c4-tk44v       1/1     Running   0          60s   192.168.24.41   worker-1   <none>           <none>
my-namespace   mediator-0                          1/1     Running   0          60s   192.168.24.42   worker-1   <none>           <none>
my-namespace   mediator-1                          1/1     Running   0          60s   192.168.24.43   worker-2   <none>           <none>
my-namespace   mediator-2                          1/1     Running   0          60s   192.168.24.44   worker-3   <none>           <none>
my-namespace   mediator-3                          1/1     Running   0          60s   192.168.24.45   worker-4   <none>           <none>
my-namespace   mediator-4                          1/1     Running   0          60s   192.168.24.46   worker-5   <none>           <none>
my-namespace   eads-server-0                       1/1     Running   0          60s   192.168.24.47   worker-1   <none>           <none>
my-namespace   eads-server-1                       1/1     Running   0          60s   192.168.24.48   worker-2   <none>           <none>
my-namespace   eads-server-2                       1/1     Running   0          60s   192.168.24.49   worker-3   <none>           <none>
my-namespace   jaeger-5cf9b67bfc-8dp7d             1/1     Running   0          60s   192.168.24.50   worker-1   <none>           <none>
my-namespace   prometheus-5fc6ddd87b-lnvgf         1/1     Running   0          60s   192.168.24.51   worker-2   <none>           <none>
my-namespace   elasticsearch-56cdb7c587-rssj2      1/1     Running   0          60s   192.168.24.52   worker-3   <none>           <none>
my-namespace   kibana-6d4b75cb6d-mj4fg             1/1     Running   0          60s   192.168.24.53   worker-4   <none>           <none>
my-namespace   filebeat-6fb89c8f89-2ncl2           1/1     Running   0          60s   192.168.24.54   worker-1   <none>           <none>
my-namespace   filebeat-6b4d774475-5q2t2           1/1     Running   0          60s   192.168.24.55   worker-2   <none>           <none>
my-namespace   filebeat-6fb89c8f89-52dgm           1/1     Running   0          60s   192.168.24.56   worker-3   <none>           <none>
my-namespace   filebeat-79989457d9-7d9kf           1/1     Running   0          60s   192.168.24.57   worker-4   <none>           <none>
my-namespace   filebeat-79989457d9-t558h           1/1     Running   0          60s   192.168.24.58   worker-5   <none>           <none>
```

＜コマンドの実行結果の例（TCC-Participant を使用する場合）＞

```
NAMESPACE      NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
my-namespace   tcc-participant-94497bdcf-7btxm     1/1     Running   0          60s   192.168.24.40   worker-1   <none>           <none>
my-namespace   orchestrator-7599cdd5c4-tk44v       1/1     Running   0          60s   192.168.24.41   worker-1   <none>           <none>
my-namespace   mediator-0                          1/1     Running   0          60s   192.168.24.42   worker-1   <none>           <none>
my-namespace   mediator-1                          1/1     Running   0          60s   192.168.24.43   worker-2   <none>           <none>
my-namespace   mediator-2                          1/1     Running   0          60s   192.168.24.44   worker-3   <none>           <none>
my-namespace   mediator-3                          1/1     Running   0          60s   192.168.24.45   worker-4   <none>           <none>
my-namespace   mediator-4                          1/1     Running   0          60s   192.168.24.46   worker-5   <none>           <none>
my-namespace   eads-server-0                       1/1     Running   0          60s   192.168.24.47   worker-1   <none>           <none>
my-namespace   eads-server-1                       1/1     Running   0          60s   192.168.24.48   worker-2   <none>           <none>
my-namespace   eads-server-2                       1/1     Running   0          60s   192.168.24.49   worker-3   <none>           <none>
my-namespace   jaeger-5cf9b67bfc-8dp7d             1/1     Running   0          60s   192.168.24.50   worker-1   <none>           <none>
my-namespace   prometheus-5fc6ddd87b-lnvgf         1/1     Running   0          60s   192.168.24.51   worker-2   <none>           <none>
my-namespace   elasticsearch-56cdb7c587-rssj2      1/1     Running   0          60s   192.168.24.52   worker-3   <none>           <none>
my-namespace   kibana-6d4b75cb6d-mj4fg             1/1     Running   0          60s   192.168.24.53   worker-4   <none>           <none>
my-namespace   filebeat-6fb89c8f89-2ncl2           1/1     Running   0          60s   192.168.24.54   worker-1   <none>           <none>
my-namespace   filebeat-6b4d774475-5q2t2           1/1     Running   0          60s   192.168.24.55   worker-2   <none>           <none>
my-namespace   filebeat-6fb89c8f89-52dgm           1/1     Running   0          60s   192.168.24.56   worker-3   <none>           <none>
my-namespace   filebeat-79989457d9-7d9kf           1/1     Running   0          60s   192.168.24.57   worker-4   <none>           <none>
my-namespace   filebeat-79989457d9-t558h           1/1     Running   0          60s   192.168.24.58   worker-5   <none>           <none>
```

なお、各列に表示される値は実行環境によって異なります。また、kubectl の describe サブコマンドを使用すると、各 Kubernetes オブジェクトの詳細な情報を参照できます。

(16) Elasticsearch のスナップショットリポジトリの作成およびスナップショットライフサイクルポリシーの作成

Elasticsearch にスナップショットを保存するためのスナップショットリポジトリを設定してください。

- REST API リクエスト
  PUT /_snapshot/<repository>
- パスパラメタ
  <repository>：(必須）リポジトリ名（string）
- リクエストボディ
  type：リポジトリタイプ。"fs"を指定してください。

settings.location：リポジトリの場所。/usr/share/elasticsearch/repository/<リポジトリ名>を指定してください。

コマンド例を次に示します（Elasticsearch の http.port が 30203 の場合）。

```
curl --cacert ca.crt --key elastic.key --cert elastic.crt -u elastic:password -H "Content-Type: application/json" -XPUT 'https://localhost:30203/_snapshot/backup' -d '{
    "type": "fs",
    "settings": {
        "location": "/usr/share/elasticsearch/repository/backup"
    }
}'
```

＜コマンドの実行結果の例＞

```
{"acknowledged":true}
```

次に示すコマンド例を基にリポジトリが作成できたことを確認してください（Elasticsearch の http.port が 30203 の場合）。

```
curl --cacert ca.crt --key elastic.key --cert elastic.crt -u elastic:password -XGET 'https://localhost:30203/_snapshot/?pretty'
```

＜コマンドの実行結果の例＞

```
{
  "backup" : {
    "type" : "fs",
    "settings" : {
      "location" : "/usr/share/elasticsearch/repository/backup"
    }
  }
}
```

Elasticsearch のスナップショットライフサイクルポリシーを作成して、リポジトリにスナップショットを定期的に取得するように設定してください。

- REST API リクエスト

  PUT /_slm/policy/<snapshot-lifecycle-policy-id>
- パスパラメタ

  <snapshot-lifecycle-policy-id>：スナップショットライフサイクルポリシー ID(string)
- リクエストボディ

  config.ignore_unavailable：データが破損していた場合の無視の有無。"true"を指定してください。

  name：自動的に付与されるスナップショット名（string）。Date math support in index and index alias names | Elasticsearch Guide [7.17] | Elastic を使用して名前を指定できます。

  repository：スナップショットの作成先のリポジトリ名。作成したリポジトリ名を指定してください。

schedule：スナップショットを取得するタイミング（cron 形式）

コマンド例を次に示します（Elasticsearch の http.port が 30203 の場合）。

```
curl --cacert ca.crt --key elastic.key --cert elastic.crt -u elastic:password -H "Content-Ty
pe: application/json" -XPUT 'https://localhost:30203/_slm/policy/my-policy' -d '{
  "schedule": "0 0 13 * * ?",
  "name": "<snapshot-{now/d}>",
  "repository": "backup",
  "config": {
    "ignore_unavailable": true
  }
}'
```

＜コマンドの実行結果の例＞

```
{"acknowledged":true}
```

次に示すコマンド例を基にスナップショットライフサイクルポリシーが作成できたことを確認してください（Elasticsearch の http.port が 30203 の場合）。

```
curl --cacert ca.crt --key elastic.key --cert elastic.crt -u elastic:password -XGET 'https:/
/localhost:30203/_slm/policy/?pretty'
```

＜コマンドの実行結果の例＞

```
{
  "my-policy" : {
    "version" : 1,
    "modified_date_millis" : 1689140931238,
    "policy" : {
      "name" : "<snapshot-{now/d}>",
      "schedule" : "0 0 13 * * ?",
      "repository" : "backup",
      "config" : {
        "ignore_unavailable" : true
      }
    },
    "next_execution_millis" : 1689166800000,
    "stats" : {
      "policy" : "my-policy",
      "snapshots_taken" : 0,
      "snapshots_failed" : 0,
      "snapshots_deleted" : 0,
      "snapshot_deletion_failures" : 0
    }
  }
}
```

(17) Elasticsearch の index 設定変更および削除

HMP-PCTO が Elasticsearch に出力する情報は index として格納されます。これらの index には、次のライフサイクルが設定されています。

表 3-21 Elasticsearch の index のライフサイクルポリシー

項番	Elasticsearch の index 名のプレフィックス	index のライフサイクルポリシー	
		Elasticsearch ILM ポリシー名	ポリシーの内容
1	「logstash」	「logstash-policy」	Elasticsearch ILM によって、次のポリシーが設定されます。 • 作成されてから 30 日が経過したとき、または 50G バイトを超えたときに、index がロールオーバーされ、新たな index が作成されて使用される。index は自動的に削除されない。
2	「jaeger」	（なし）	作成されてから 1 日が経過したときに、index がロールオーバーされ、新たな index が作成されて使用される。index は自動的に削除されない。
3	「metricbeat」	「metricbeat」	Elasticsearch ILM によって、次のポリシーが設定されます。 • 作成されてから 30 日が経過したとき、または 50G バイトを超えたときに、index がロールオーバーされ、新たな index が作成されて使用される。index は自動的に削除されない。

(a) 情報の確認方法

index の一覧とボリューム使用量を確認する方法を次に示します。

Kubernetes コントロールプレーン上で、REST API の GET メソッドを使用して、「IP アドレス:ポート番号/」のあとに「_cat/indices」を指定することで index の一覧とボリューム使用量が表示されます。

次にコマンド例を示します（Elasticsearch の http.port が 30202 の場合）。

```
# curl -XGET 'localhost:30202/_cat/indices?v&s=index'
```

＜コマンドの実行結果の例＞

```
health status index                                uuid                   pri rep docs.count docs.deleted store.size pri.store.size
yellow open   jaeger-jaeger-service-XXXX-XX-XX     XXXXXXXXXXXXXXXXXXXXX   X   X        252           18     89.1kb         89.1kb
yellow open   jaeger-jaeger-span-XXXX-XX-XX        XXXXXXXXXXXXXXXXXXXXX   X   X  180828498            0       7gb            7gb
yellow open   logstash-XXXX.XX.XX-XXXXXX           XXXXXXXXXXXXXXXXXXXXX   X   X          0            0     226b           226b
yellow open   metricbeat-X.XX.X-XXXX.XX.XX-XXXXXX XXXXXXXXXXXXXXXXXXXXX   X   X   31880255            0       5gb            5gb
```

(b) 情報のロールオーバー設定の変更方法

標準出力の情報およびメトリクスの情報は、それぞれ「logstash-policy」「metricbeat」という名前の Elasticsearch ILM ポリシーが設定されています。これらの ILM ポリシーにはロールオーバーする日数・上限サイズがあらかじめ設定されています。

まず、現在の ILM ポリシーの内容を確認します。

Kubernetes コントロールプレーン上で、REST API の GET メソッドを使用して、「IP アドレス:ポート番号/_ilm/policy/」のあとに ILM ポリシー名を指定することで指定した ILM ポリシーを表示できます。

次にコマンド例を示します（Elasticsearch の http.port が 30202 の場合）。

```
# curl -XGET 'localhost:30202/_ilm/policy/logstash-policy?pretty'
```

＜コマンドの実行結果の例＞

```
{
  "logstash-policy" : {
(略)
    "policy" : {
      "phases" : {
        "hot" : {
          "min_age" : "0ms",
          "actions" : {
            "rollover" : {
              "max_size" : "50gb", …1
              "max_age" : "30d"
            }
          }
        }
      }
    },
(略)
  }
}
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. ロールオーバーする上限の index のサイズ(max_size)および日数(max_age)

   上記の「policy」で表示されたポリシーのうち、変更したい部分だけを置き換えて、他の部分は表示された値のままで設定します。Kubernetes コントロールプレーン上で、REST API の PUT メソッドを使用して、「IP アドレス:ポート番号/_ilm/policy/」のあとに ILM ポリシー名を指定し、後続に「policy」配下の設定をすべて記載することで、ILM ポリシーの値を変更できます。

重要

- ILM ポリシー変更時の注意事項

  必ず ILM ポリシーの「policy」以下のうち、変更したい部分だけを置き換えて、他の部分は表示された値をそのまま設定してください。もし他の部分を記載しなかった場合、記載しなかった部分は以前の設定から消えてしまいます。

- バージョンアップ時の注意事項

  HMP-PCTO の関連プログラムの Elasticsearch、Jaeger、または Prometheus をバージョンアップした際、インデックステンプレートが新規に作成されることがあります。カスタマイズした ILM ポリシー設定を使用している場合、カスタマイズした設定内容は、新しいインデックステンプレートには反映されません。ILM ポリシーを再設定してください。

次に、ロールオーバーする上限の index のサイズ(max_size)を 10G バイトに、日数(max_age)を 1 日に変更する場合のコマンド例を示します（Elasticsearch の http.port が 30202 の場合)。

```
# curl -XPUT 'localhost:30202/_ilm/policy/logstash-policy' ¥
-H 'Content-Type: application/json; charset=utf-8' ¥
--data-binary @- << EOF
{
    "policy" : {
      "phases" : {
        "hot" : {
          "min_age" : "0ms",
          "actions" : {
            "rollover" : {
              "max_size" : "10gb",
              "max_age" : "1d"
            }
          }
        }
      }
    }
}
EOF
```

＜コマンドの実行結果の例＞

```
{"acknowledged":true}
```

(c) 情報の削除方法

デフォルトのポリシーでは、Elasticsearch のボリューム使用量が増加し続けるため、不要な index は削除してください。

index を削除する方法を次に示します。

■index を自動的に削除する方法

標準出力の情報およびメトリクスの情報は、それぞれ「logstash-policy」「metricbeat」という名前の Elasticsearch ILM ポリシーが設定されています。これらの ILM ポリシーに対して、削除のポリシーを追加します。

分散トレースの情報は Elasticsearch ILM ポリシーが設定されていないため、削除を実施する ILM ポリシーの設定と、ILM ポリシーを適用するための設定を行います。

Logstash、Metricbeat の index を自動的に削除する方法

まず、現在の ILM ポリシーの内容を確認します（確認の方法は「(2)情報のロールオーバー設定の変更方法」をご覧ください)。

自動的に削除するためには、ILM ポリシーの「policy.phases」に「delete」配下を追加し、delete の「min_age」に、削除するまでに保持する日数を記載します。そのほかの部分は上記の確認結果で表示された値のままで設定します。

Kubernetes コントロールプレーン上で、REST API の PUT メソッドを使用して、「IP アドレス:ポート番号/_ilm/policy/」のあとに ILM ポリシー名を指定し、後続に「policy」配下の設定をすべて記載することで、ILM ポリシーの値を変更できます。

重要

- ILM ポリシー変更時の注意事項

  必ず ILM ポリシーの「policy」以下のうち、変更したい部分だけを置き換えて、他の部分は表示された値をそのまま設定してください。もし他の部分を記載しなかった場合、記載しなかった部分は以前の設定から消えてしまいます。

- バージョンアップ時の注意事項

  HMP-PCTO の関連プログラムの Elasticsearch、Jaeger、または Prometheus をバージョンアップした際、インデックステンプレートが新規に作成されることがあります。カスタマイズした ILM ポリシー設定を使用している場合、カスタマイズした設定内容は、新しいインデックステンプレートには反映されません。ILM ポリシーを再設定してください。

次に、「policy.phases」に 7 日経過したら index を削除する場合のコマンド例を示します (Elasticsearch の http.port が 30202 の場合)。

```
# curl -XPUT 'localhost:30202/_ilm/policy/logstash-policy' ¥
-H 'Content-Type: application/json; charset=utf-8' ¥
--data-binary @- << EOF
{
    "policy" : {
      "phases" : {
        "hot" : {
          "min_age" : "0ms",
          "actions" : {
            "rollover" : {
              "max_size" : "10gb",
              "max_age" : "1d"
            }
          }
```

```
        },
        "delete" : {
          "min_age" : "7d",
          "actions" : {
            "delete" : {}
          }
        }
      }
    }
}
EOF
```

＜コマンドの実行結果の例＞

```
{"acknowledged":true}
```

jaeger の index を自動的に削除する方法

分散トレースの情報は Elasticsearch ILM ポリシーが設定されていません。そのため、次の(i)(ii)の操作が必要になります。

(i)分散トレースの ILM ポリシーの追加

自動的に削除するためには、ILM ポリシーの「policy.phases」配下に「hot」と「delete」配下を追加し、delete の「min_age」に、削除するまでに保持する日数を記載します（「hot」の「min_age」には即時反映を示す 0ms を指定します)。

Kubernetes コントロールプレーン上で、REST API の PUT メソッドを使用して、「IP アドレス:ポート番号/_ilm/policy/」のあとに ILM ポリシー名「jaeger-ilm-policy」を指定し、後続に「policy」配下の設定をすべて記載することで、ILM ポリシーを作成できます。

次に、「policy.phases」に 7 日経過したら index を削除する場合のコマンド例を示します(Elasticsearch の http.port が 30202 の場合)。

```
# curl -XPUT 'localhost:30202/_ilm/policy/jaeger-ilm-policy' ¥
-H 'Content-Type: application/json; charset=utf-8' ¥
--data-binary @- << EOF
{
    "policy" : {
      "phases" : {
        "hot" : {
          "min_age" : "0ms",
          "actions" : {
          }
        },
        "delete" : {
          "min_age" : "7d",
          "actions" : {
            "delete" : {}
          }
        }
      }
    }
}
EOF
```

＜コマンドの実行結果の例＞

```
{"acknowledged":true}
```

(ii)分散トレースの index のテンプレートの更新

新規に作成される分散トレースの index に上記(i)で作成した ILM ポリシーを反映させるため、分散トレースの 2 つの index テンプレート「jaeger-jaeger-service」「jaeger-jaeger-span」それぞれの設定に、(i)の ILM ポリシーを追加します。

まず、現在の index テンプレートの内容を確認します。

Kubernetes コントロールプレーン上で、REST API の GET メソッドを使用して、「IP アドレス:ポート番号/_template/」のあとにテンプレート名を指定することで指定した index テンプレートを表示できます。

次にテンプレート名「jaeger-jaeger-service」を指定した場合のコマンド例を示します(Elasticsearch の http.port が 30202 の場合)。

```
# curl -XGET 'localhost:30202/_template/jaeger-jaeger-service?pretty'
```

＜コマンドの実行結果の例＞

```
{
  "jaeger-jaeger-service" : {
    "order" : 0,
    "index_patterns" : [
      "jaeger-jaeger-service-*"
    ],
    "settings" : {
      "index" : {
        "mapping" : {
          "nested_fields" : {
            "limit" : "50"
          }
        },
        "requests" : {
          "cache" : {
            "enable" : "true"
          }
        },
        "number_of_shards" : "5",
        "number_of_replicas" : "1"
      }
    },
    "mappings" : {
      "dynamic_templates" : [
        {
          "span_tags_map" : {
            "path_match" : "tag.*",
            "mapping" : {
              "ignore_above" : 256,
              "type" : "keyword"
            }
          }
        },
        {
          "process_tags_map" : {
            "path_match" : "process.tag.*",
```

```
            "mapping" : {
              "ignore_above" : 256,
              "type" : "keyword"
            }
          }
        }
      ],
      "properties" : {
        "operationName" : {
          "ignore_above" : 256,
          "type" : "keyword"
        },
        "serviceName" : {
          "ignore_above" : 256,
          "type" : "keyword"
        }
      }
    },
    "aliases" : { }
  }
}
```

index テンプレートに ILM ポリシーを適用するためには、「settings.index」に「lifecycle」配下を追加し、「lifecycle.name」に ILM ポリシー名を記載します。そのほかの部分は上記の確認結果で表示された値のままで設定します。

Kubernetes コントロールプレーン上で、REST API の PUT メソッドを使用して、「IP アドレス:ポート番号/_template/」のあとに index テンプレート名を指定し、後続に「index_patterns」「settings」「mappings」「aliases」の設定をすべて記載することで、ILM ポリシーを適用できます。

重要

index テンプレート変更時の注意事項

必ず「settings.index」に「lifecycle」配下を追加し、他の部分は表示された値をそのまま設定してください。もし他の部分を記載しなかった場合、記載しなかった部分は以前の設定から消えてしまいます。

次に、index テンプレート「jaeger-jaeger-service」に ILM ポリシー「jaeger-ilm-policy」を適用する場合のコマンド例を示します（Elasticsearch の http.port が 30202 の場合）。

```
# curl -XPUT 'localhost:30202/_template/jaeger-jaeger-service' ¥
-H 'Content-Type: application/json; charset=utf-8' ¥
--data-binary @- << EOF
{
    "index_patterns" : [
      "jaeger-jaeger-service-*"
    ],
    "settings" : {
      "index" : {
        "lifecycle" : {
           "name": "jaeger-ilm-policy"
         },
        "mapping" : {
          "nested_fields" : {
```

```
            "limit" : "50"
          }
        },
        "requests" : {
          "cache" : {
            "enable" : "true"
          }
        },
        "number_of_shards" : "5",
        "number_of_replicas" : "1"
      }
    },
    "mappings" : {
      "dynamic_templates" : [
        {
          "span_tags_map" : {
            "path_match" : "tag.*",
            "mapping" : {
              "ignore_above" : 256,
              "type" : "keyword"
            }
          }
        },
        {
          "process_tags_map" : {
            "path_match" : "process.tag.*",
            "mapping" : {
              "ignore_above" : 256,
              "type" : "keyword"
            }
          }
        }
      ],
      "properties" : {
        "operationName" : {
          "ignore_above" : 256,
          "type" : "keyword"
        },
        "serviceName" : {
          "ignore_above" : 256,
          "type" : "keyword"
        }
      }
    },
    "aliases" : { }
}
EOF
```

＜コマンドの実行結果の例＞

```
{"acknowledged":true}
```

分散トレースの 2 つの index テンプレート「jaeger-jaeger-service」「jaeger-jaeger-span」それぞれに対して、ILM ポリシーを適用してください。

■index を手動で削除する方法

Kubernetes コントロールプレーン上で、REST API の DELETE メソッドを使用して、「IP アドレス:ポート番号/」のあとに、index 名を指定することで指定した index を削除できます。

jaeger の index は、同じ日付の名前を持つ「jaeger-jaeger-service XXXX-XX-XX」「jaeger-jaeger-span XXXX-XX-XX」の 2 種類があるため、同じ日付の名前の index をペアで削除してください。

次にコマンド例を示します（Elasticsearch の http.port が 30202 の場合）。

```
# curl -XDELETE 'localhost:30202/metricbeat-X.XX.X-XXXX.XX.XX-XXXXXX'
```

＜コマンドの実行結果の例＞

```
{"acknowledged":true}
```

(18)　EADS のセットアップ（通常版限定）

「(15)　Kubernetes アプリケーションのデプロイ」で EADS をデプロイしたあと、「(1)　各種資材の配置」で配置した EADS の Helm チャートパッケージファイル（eads-command-job-chart.tgz および eads-command-cronjob-chart.tgz）と EADS の Helm チャートの values.yaml を使用して、EADS をセットアップしてください。

コマンド例を次に示します。なお、初デプロイ時と再デプロイ時（＝ eztool createcache コマンドを 1 回実行済みでキャッシュ用 PV 内にファイルが書き込まれている時）で、実行するコマンドは内容が一部異なります。また、コマンド例で強調されている「hmppctoCache」は、EADS の Helm チャートの values.yaml の.caches[].name フィールドに指定した文字列に置き換えてください。

```
（初デプロイ時）
helm -n my-namespace install eads-command-createcache ./eads-command-job-chart.tgz -f ./eads
-values.yaml --set args="createcache hmppctoCache"
kubectl -n my-namespace wait --for=condition=complete job/eads-command-createcache-eads-comm
and-job --timeout=300s
helm -n my-namespace install eads-command-open ./eads-command-job-chart.tgz -f ./eads-values
.yaml --set args="open"
kubectl -n my-namespace wait --for=condition=complete job/eads-command-open-eads-command-job
 --timeout=60s
helm -n my-namespace uninstall eads-command-createcache
helm -n my-namespace uninstall eads-command-open
helm -n my-namespace install eads-command-compaction-0 ./eads-command-cronjob-chart.tgz -f .
/eads-values.yaml --set args="compaction",index=0
helm -n my-namespace install eads-command-compaction-1 ./eads-command-cronjob-chart.tgz -f .
/eads-values.yaml --set args="compaction",index=1
helm -n my-namespace install eads-command-compaction-2 ./eads-command-cronjob-chart.tgz -f .
/eads-values.yaml --set args="compaction",index=2

（再デプロイ時）
helm -n my-namespace install eads-command-resume ./eads-command-job-chart.tgz -f ./eads-valu
es.yaml --set args="resume"
kubectl -n my-namespace wait --for=condition=complete job/eads-command-resume-eads-command-j
ob --timeout=300s
helm -n my-namespace install eads-command-open ./eads-command-job-chart.tgz -f ./eads-values
```

```
.yaml --set args="open"
kubectl -n my-namespace wait --for=condition=complete job/eads-command-open-eads-command-job
 --timeout=60s
helm -n my-namespace uninstall eads-command-resume
helm -n my-namespace uninstall eads-command-open
helm -n my-namespace install eads-command-compaction-0 ./eads-command-cronjob-chart.tgz -f .
/eads-values.yaml --set args="compaction",index=0
helm -n my-namespace install eads-command-compaction-1 ./eads-command-cronjob-chart.tgz -f .
/eads-values.yaml --set args="compaction",index=1
helm -n my-namespace install eads-command-compaction-2 ./eads-command-cronjob-chart.tgz -f .
/eads-values.yaml --set args="compaction",index=2
```

この例では、EADS の Helm チャートの values.yaml のファイル名を「eads-values.yaml」としています。

＜コマンドの実行結果の例＞

```
（初デプロイ時）
NAME: eads-command-createcache
LAST DEPLOYED: Tue Jun 27 17:45:45 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
job.batch/eads-command-createcache-eads-command-job condition met
NAME: eads-command-open
LAST DEPLOYED: Tue Jun 27 17:46:30 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
job.batch/eads-command-open-eads-command-job condition met
release "eads-command-createcache" uninstalled
release "eads-command-open" uninstalled
NAME: eads-command-compaction-0
LAST DEPLOYED: Tue Jun 27 17:46:15 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
NAME: eads-command-compaction-1
LAST DEPLOYED: Tue Jun 27 17:46:20 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
NAME: eads-command-compaction-2
LAST DEPLOYED: Tue Jun 27 17:46:25 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None

（再デプロイ時）
NAME: eads-command-resume
LAST DEPLOYED: Tue Jun 27 17:45:45 2023
NAMESPACE: my-namespace
STATUS: deployed
```

```
REVISION: 1
TEST SUITE: None
job.batch/eads-command-resume-eads-command-job condition met
NAME: eads-command-open
LAST DEPLOYED: Tue Jun 27 17:46:30 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
job.batch/eads-command-open-eads-command-job condition met
release "eads-command-resume" uninstalled
release "eads-command-open" uninstalled
NAME: eads-command-compaction-0
LAST DEPLOYED: Tue Jun 27 17:46:15 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
NAME: eads-command-compaction-1
LAST DEPLOYED: Tue Jun 27 17:46:20 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
NAME: eads-command-compaction-2
LAST DEPLOYED: Tue Jun 27 17:46:25 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

このように、各コマンドが成功したメッセージが出力されていることを確認してください。

(19)　EADS のアンセットアップ（通常版限定）

「(20)　Kubernetes アプリケーションのアンデプロイ」で EADS をアンデプロイする前、「(1)　各種資材の配置」で配置した EADS の Helm チャートパッケージファイル（eads-command-job-chart.tgz）と EADS の Helm チャートの values.yaml を使用して、EADS をアンセットアップする手順を説明します。

コマンド例を次に示します。

```
helm -n my-namespace uninstall eads-command-compaction-0
helm -n my-namespace uninstall eads-command-compaction-1
helm -n my-namespace uninstall eads-command-compaction-2
helm -n my-namespace install eads-command-close ./eads-command-job-chart.tgz -f ./eads-value
s.yaml --set args="close"
kubectl -n my-namespace wait --for=condition=complete job/eads-command-close-eads-command-jo
b --timeout=60s
helm -n my-namespace uninstall eads-command-close
```

この例では、EADS の Helm チャートの values.yaml ファイルのファイル名を「eads-values.yaml」としています。

＜コマンドの実行結果の例＞

```
release ”eads-command-compaction” uninstalled-0
release ”eads-command-compaction” uninstalled-1
release ”eads-command-compaction” uninstalled-2
NAME: eads-command-close
LAST DEPLOYED: Tue Jun 27 18:46:30 2023
NAMESPACE: my-namespace
STATUS: deployed
REVISION: 1
TEST SUITE: None
job.batch/eads-command-close-eads-command-job condition met
release ”eads-command-close” uninstalled
```

このように、各コマンドが成功したメッセージが出力されていることを確認してください。

(20) Kubernetes アプリケーションのアンデプロイ

デプロイした HMP-PCTO の各 Kubernetes アプリケーションをアンデプロイする方法について説明します。

なお、HMP-PCTO の各 Kubernetes アプリケーションは、次の順序でアンデプロイする必要があります。

1. Orchestrator
2. Participant※
3. Mediator、EADS、Ext-Cons（トライアル版だけ）、Participant が使用する DB/外部サービス
4. Logstash、Jaeger-query、Jaeger-collector、Prometheus、Metricbeat、Filebeat
5. Elasticsearch

注※

TP1 連携機能を使用している場合、TP1-Bridge をアンデプロイしたあとに、OpenTP1 の rap リスナーと rap サーバを停止してください。

重要

上記以外の順序でアンデプロイした場合に発生する問題

上記以外の順序でアンデプロイした場合、トラブルシュートに必要な情報が収集できなかったり、処理中のトランザクションが適切に終了しなかったりなどの問題が発生するおそれがあります。

HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションは、次のとおり helm コマンドでアンデプロイしてください。

```
helm△uninstall△<リリース名>
```

（凡例）△：半角スペース 1 文字

<リリース名>は、デプロイ時と同様に指定してください。

Mediator をアンデプロイするコマンド例を次に示します。

```
helm uninstall mediator
```

＜コマンドの実行結果の例＞

```
release ”mediator” uninstalled
```

このように、Helm チャートのアンインストールに成功したメッセージが出力されていることを確認してください。

HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションをアンデプロイしたあと、次のコマンド例のように、各 Kubernetes アプリケーションが期待どおり停止していること（一覧に出力されないこと）を確認してください。

```
kubectl get pods -n my-namespace -o wide
```

＜コマンドの実行結果の例＞

```
No resources found in my-namespace namespace.
```

3.9　ログの取得および監視の設定

ログの取得、および監視の設定について説明します。

3.9.1　ログおよびトラブルシュート情報の取得の設定

連携製品のトラブルシュート情報を取得するため、HMP-PCTO の構築時に次の設定をします。

(1)　Tomcat のアクセスログの取得方法

- ログ取得を有効にする方法

  Orchestrator、Entity-Service、SQL-Participant、および TCC-Participant の Spring Boot のコンフィグソースに server.tomcat.accesslog.enable=true を設定するように定義してください。なお、TP1-Bridge では、ログを取得する設定がされています。

  「2.6.2　Orchestrator の Kubernetes マニフェストの作成」、「2.6.3　Entity-Service（Participant コンテナを含まない Pod とする場合）の Kubernetes マニフェストの作成」、「2.6.4　SQL-Participant の Kubernetes マニフェストの作成（SQL-Participant 限定）」、「2.6.5　TCC-Participant の Kubernetes マニフェストの作成（TCC-Participant 限定）」時に Filebeat コンテナの filebeat.yml に次を設定してください。なお、output.logstash フィールドの Namespace 名には Logstash アプリケーションをデプロイする Namespace を指定します。

```
filebeat.inputs:
- type: filestream
  id: "tomcat"
  paths:
    - '<Tomcatのアクセスログのパス>'

output.logstash:
  hosts: ["logstash.my-namespace.svc.cluster.local:5044"]
```

- ヘッダ情報を取得する方法

  server.tomcat.accesslog.enabled=true を設定した Spring Boot のコンフィグソースに「server.tomcat.accesslog.pattern」の指定値として、「"%{X-transparent}i"」を含めるように定義してください。設定例は次のとおりです。

```
server.tomcat.accesslog.enabled=true
server.tomcat.accesslog.pattern=%h %{X-Forwarded-For}i %u %{dd/MMM/yyyy:HH:mm:ss.SSS Z}t
"%r" %s %b %D %{ucar.rootap}r "%{Referer}i" "%{User-Agent}i" "%{X-transparent}i"
```

(2)　PostgreSQL のログの取得方法（SQL-Participant を使用する場合）

PostgreSQL の Kubernetes マニフェスト作成時に Filebeat をサイドカーコンテナとして立ち上げるようにしてください。

PostgreSQL のログ出力を有効にする設定をしてください。

Filebeat コンテナの filebeat.yml に次を設定してください。なお、output.logstash フィールドの Namespace 名には Logstash アプリケーションをデプロイする Namespace を指定します。

注　ログが 1 回の出力で複数行を出力する場合、Filebeat のマニュアルを参考に Multiline messages の設定を filebeat.yml に対して行い、1 つの行として処理される設定にしてください。

```
filebeat.inputs:
- type: filestream
  id: "postgre"
  paths:
    - '<PostgreSQLログのパス>'

output.logstash:
  hosts: ["logstash.my-namespace.svc.cluster.local:5044"]
```

(3)　HiRDB のトラブルシュート情報の取得方法

トラブルシュートをするためには、次の情報が必要です。

- ログ情報
  メッセージログファイル、システムログファイルを取得してください。
- HiRDB サーバのトランザクションの状態表示の結果
  障害発生時などの場合に、トランザクションの状態表示の結果を取得してください。
- SQL トレース
  SQL トレースの出力は性能に影響があるため、デフォルトでは有効化されていません。SQL トレースの出力を有効化することで、SQL トレースによる詳細な調査ができます。

これらのトラブルシュート情報の取得方法については、マニュアル「HiRDB システム運用ガイド」、マニュアル「HiRDB コマンドリファレンス」、およびマニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4)　SQL Server のトラブルシュート情報の取得方法

トラブルシュートをするためには、次の情報が必要です。

- クエリトレース
- ログ情報
  システムログ、およびエラーログを取得してください。

これらのトラブルシュート情報の出力方法および取得情報については、SQL Server のドキュメントを参照してください。

(5) EADS のトラブルシュート情報の取得方法

EADS のトラブルシュート情報を取得するための設定、および取得方法については、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照してください。

なお、トライアル版では取得できません。

(6) uCosminexus Service Platform のトレース・ログ情報の取得方法

トラブルシュートをするためには、次の情報が必要です。

- J2EE アプリケーションのユーザログ
- NIO HTTP サーバのアクセスログ
- Web サーブレットログ
- JavaVM スタックトレース情報
- JavaVM の GC ログ
- JavaVM ログ（JavaVM ログファイル）

これらのトレース・ログ情報を取得するための設定、および取得方法については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」を参照してください。

(7) OpenTP1 のトレース・ログ情報の取得方法

トラブルシュートをするためには、次の情報が必要です。

- リアルタイム出力機能

このトレース・ログ情報を取得するための設定、および取得方法については、マニュアル「OpenTP1 運用と操作」を参照してください。

(8) TP1/Client/J のトラブルシュート情報の取得方法

TP1-Bridge のトラブルシュート情報は、情報ごとに出力方法が異なります。情報ごとの取得方法を次の表に示します。

表 3-22 TP1/Client/J のトラブルシュート情報の取得方法

項番	トラブルシュート情報	取得方法
1	デバッグトレース	TP1-Bridge の標準出力として出力されるため、Filebeat で収集します。 収集されたデータは Elasticsearch に送信されるため、Elasticsearch のスナップショット機能で取得します。
2	エラートレース	TP1-Bridge の Helm チャートの values.yaml で、トラブルシュート情報の出力を有効化できます。出力を有効化するパラメタについては、「(1) Helm チャートのパラメタ」を参照してください。
3	メソッドトレース	

項番	トラブルシュート情報	取得方法
4	UAP トレース	出力を有効化したトラブルシュート情報は、uCosminexus Application Runtime with Java for Spring Boot のスナップショットログ機能で取得します。
5	データトレース	

3.9.2　監視の設定

EADS サーバのログ監視、Elasticsearch に格納された標準出力の監視、およびメトリクスの監視の設定について説明します。

(1)　EADS サーバのログ監視のための設定

EADS サーバのログ監視を行う場合、そのログを Elasticsearch に格納するために EADS サーバのログを標準出力に出力するよう設定する必要があります。そのため EADS のセットアップ時に使用する Helm チャートの values.yaml の properties.server に eads.logger.message.console.enable=true を追加してください。

```
properties:
  server:
  - eads.logger.message.console.enable=true
```

その後、後述する「Elasticsearch に格納された標準出力を監視するための設定」を実施してください。

この手順で Elasticsearch に格納される EADS サーバのログは監視目的のため、サポート部署に調査を依頼する際には EADS のトラブルシュートファイルを送付してください。

(2)　Elasticsearch に格納された標準出力を監視するための設定

Elasticsearch に格納された標準出力の情報を監視する場合、次の方法があります。

- Kibana の Alert 機能を使用する
  ElasticStack の Kibana にある Alert 機能で、標準出力の監視を行います。
- Watcher を使用する
  ElasticStack の有償の監視機能である Watcher で、標準出力の監視を行います。
- ElastAlert（または ElastAlert2）を使用する
  OSS の ElastAlert で、標準出力の監視を行います。

例として、Kibana の Alert 機能の設定方法を示します。

【Alert 機能の前提条件】

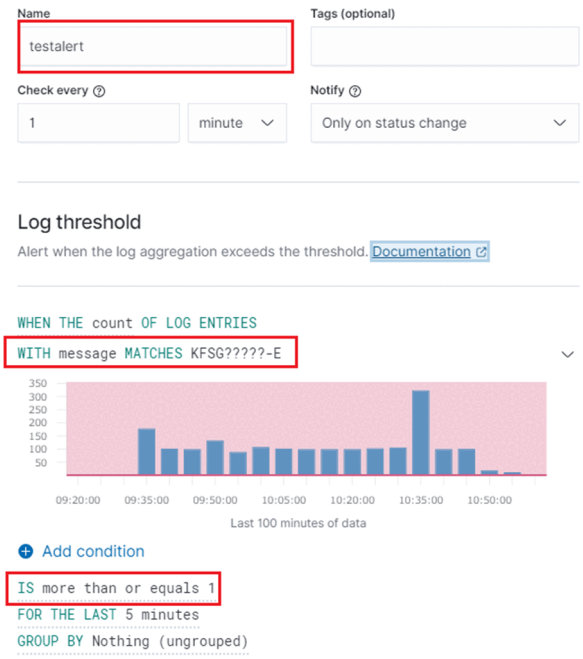
- Kibana と Elasticsearch で通信が暗号化されていること
- Kibana の設定で xpack.encryptedSavedObjects.encryptionKey が設定されていること

【Alert 機能の設定方法】

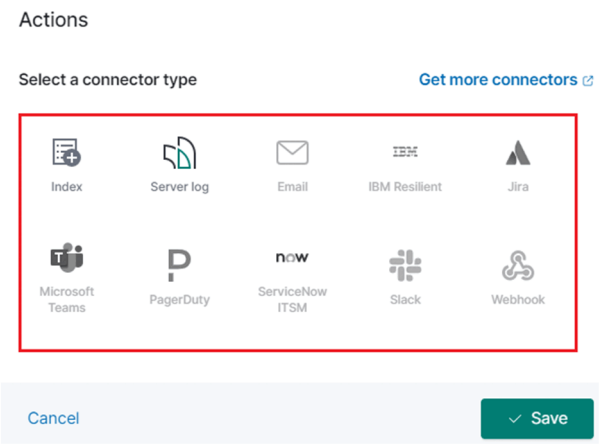
1. Kibana 画面の左上のハンバーガーボタン（三本線）を押し、メニューから Observability を選択する
2. 画面の左カラム内の Logs 内の Stream を選択する
3. 画面の右上の Settings を選択する
4. Log sources 欄内の Log indices に、index 名として「logstash*」をコンマ区切りで追加する
5. Apply を押す
6. 画面の左カラム内の Alerts を選択する
7. 画面の右上の Manage Rules を選択する
8. 画面の右上の Create Alert を選択する

   【注意】Elasticsearch との通信が暗号化されていない場合、画面に「Additional setup required」と出力され、これ以降の設定ができません。
9. ルール設定画面でルールの条件を設定する

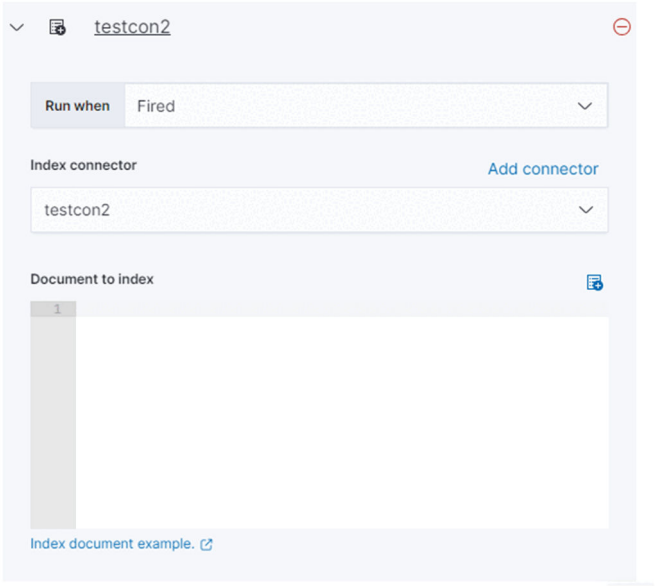
   例として、KFSG の-E メッセージが 1 つ以上出力された場合に、アラートを上げる場合の設定内容を次の図に示します。



10. Action の Select a connector type で通知先への通知の定義を行う



例として Index を選択した場合、Run when に「Fired」を、Index connector では connector（通知先）を指定します。Connector の定義では Write to index に通知先の index 名を指定します。Document to index に、通知したい内容を JSON 形式で記載します。



11. 最後に Save を押して保存します。

(3) メトリクスの監視方法

Prometheus または Elasticsearch 上で収集したメトリクス情報を使用してメトリクスを監視できます。メトリクスの監視方法には、Alertmanager や Kibana などの外部コンポーネントと組み合わせる方法があります。

- Alertmanager

  Prometheus 側で設定したルールに従って、ルールに抵触するメトリクスが発生した場合に、Alertmanager で指定した宛先に対してアラートを送信できます。

- Kibana

  Elasticsearch に収集されたメトリクスに対して、Kibana 上で監視ルールを設定します。ルールに抵触するメトリクスが発生した場合に、指定した宛先に対してアラートを送信できます。

メトリクスの監視についての詳細は、各 OSS のドキュメントを参照してください。

3.10　パラメタの設計とチューニング

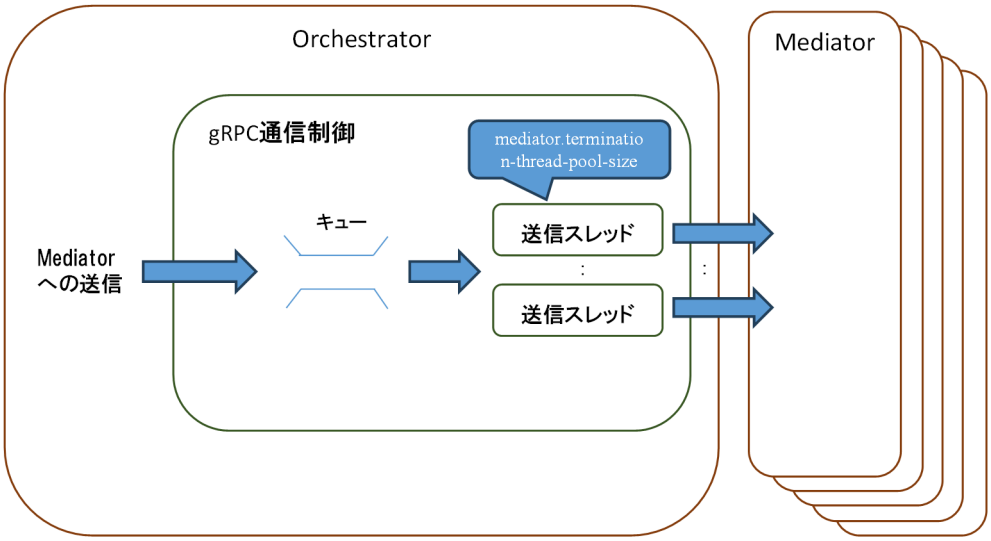
HMP-PCTO のパラメタの設計指針とチューニング方法について説明します。

3.10.1　スレッドプール

(1)　Orchestrator

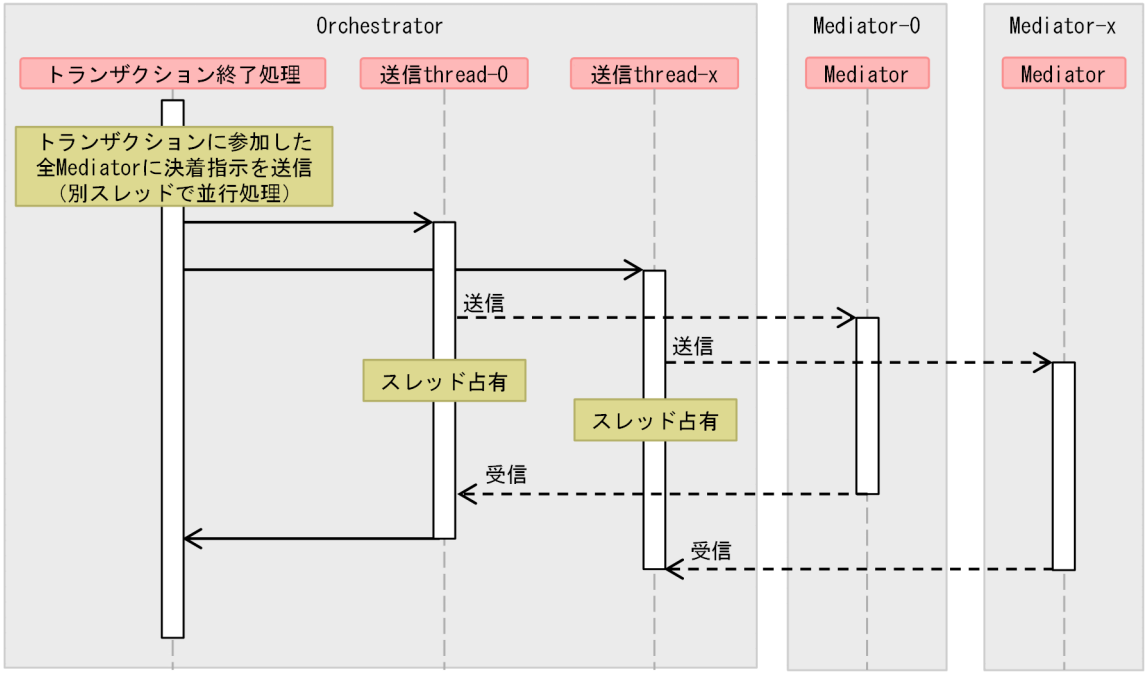
Mediator への送信時（トランザクション決着要求・仕掛中トランザクションのキャンセル要求）の同時実行スレッド数を制御します。同時実行スレッド数は Orchestrator のパラメタ「mediator.termination-thread-pool-size」で指定します。同時実行スレッド数を超えた要求がある場合は、キューに保持します。

図 3-1　同時実行スレッド数制御



1 トランザクションが終了した際に、トランザクションに参加した Mediator の数だけスレッドを使用します。また、スレッドは Mediator へ送信してから応答を受信するまで占有します。

図 3-2　スレッド占有期間



(a)　設計指針

複数のトランザクションを同時に実行する際、極力スレッドの空き待ちが発生しないようにするためにはトランザクションの同時実行数×トランザクションに参加する Mediator の台数分スレッドが必要です。

ただし、トランザクションの終了が重ならないかぎりは、トランザクションの同時実行数×トランザクションに参加する Mediator の台数分のスレッドは必要ありません。

スレッドの占有期間に相当する Mediator からの応答時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために安全率を掛けることを推奨します。

算出したスレッド数がトランザクションに参加する Mediator の台数の 2 倍未満になった場合は、トランザクションに参加する Mediator の台数を 2 倍したスレッド数としてください。

Orchestrator1 台当たりの目標 TPS を基準にした算出方法を次に示します。

X ＝ A × B × C

X:スレッド数

A:目標 TPS

B:トランザクションに参加する Mediator 台数

C:Mediator からの応答時間（秒）

例）

目標 40TPS、トランザクションに参加する Mediator5 台、Mediator からの応答時間 0.1 秒とした場合、必要なスレッド数は 20 となります。

X ＝ 40 × 5 × 0.1

X ＝ 20

安全率を 1.5 とした場合、必要なスレッド数は 30 となります。

X ＝ 20 × 1.5

X ＝ 30

(b)　チューニング対象のパラメタ一覧

表 3-23　Orchestrator のチューニング対象のパラメタ一覧

項番	パラメタ	説明
1	mediator.termination-thread-pool-size	Orchestrator から Mediator への送信に使用するスレッド数を指定します。

パラメタの詳細については、「8.2.2　Orchestrator」を参照してください。

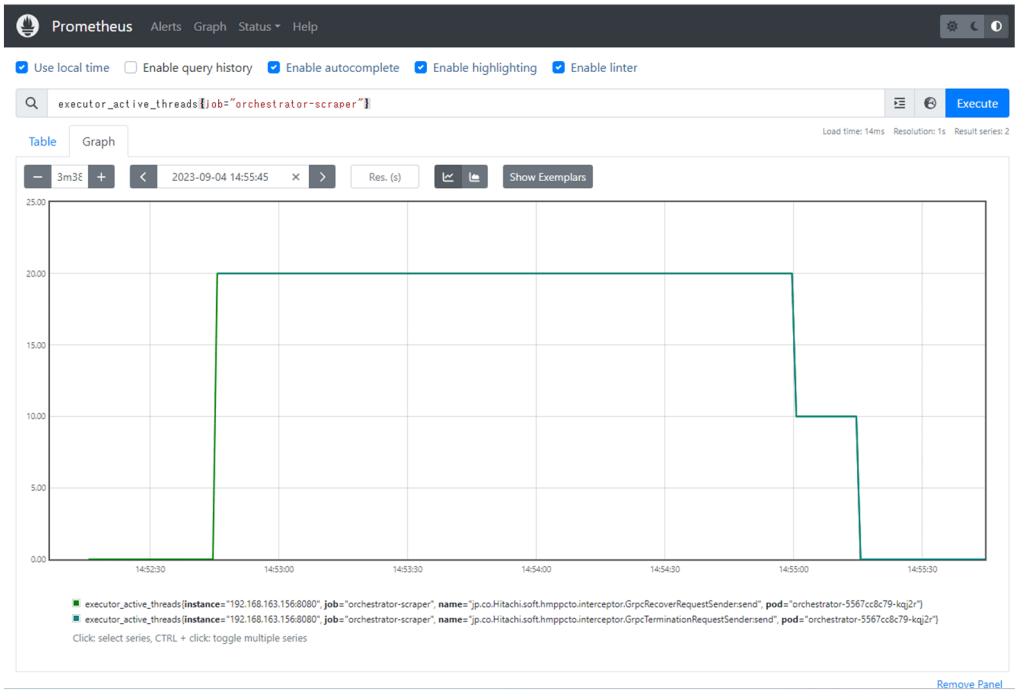
(c)　チューニング方法

mediator.termination-thread-pool-size

このパラメタに関する情報はメトリクスデータによって確認できます。メトリクスデータの確認方法の詳細については、「10.5.3　メトリクス」を参照してください。

Prometheus の Web UI で「executor_active_threads{job="orchestrator-scraper"}」のクエリを入力し、Execute ボタンを押すことで Graph タブに下図のような状態で表示されます。

図 3-3　Prometheus の Web UI



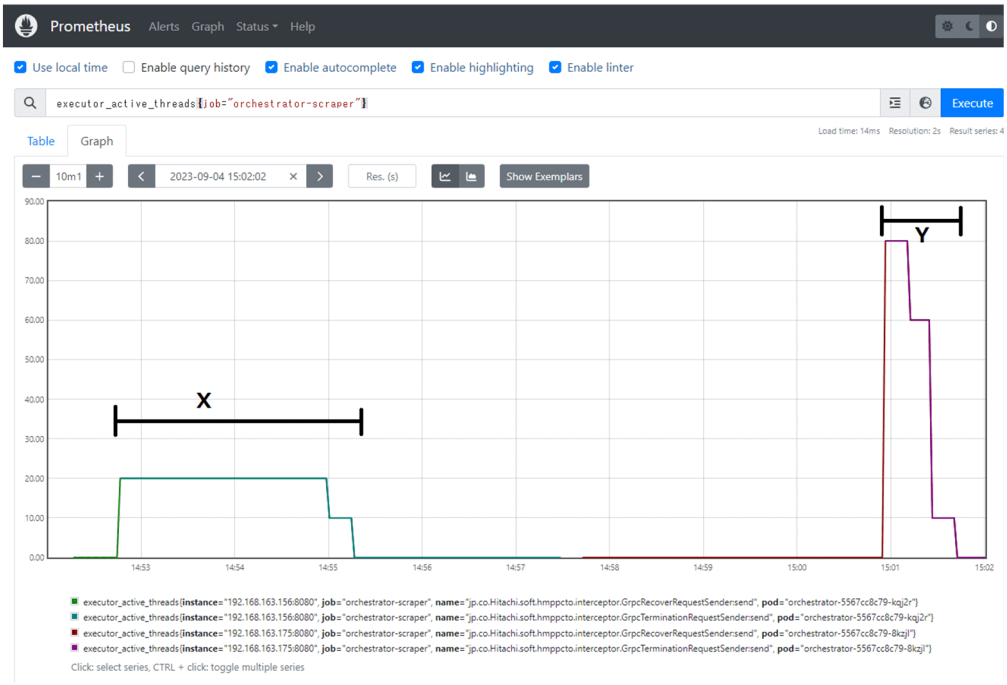
なお、1 つの Prometheus で複数の Orchestrator のメトリクスを収集している場合は、クエリに「instance="TP1-Bridge の IP アドレス:TP1-Bridge の http ポート番号"」を追加してください。

入力例：

executor_active_threads{job="orchestrator-scraper", instance="192.168.125.166:20554"}

長い期間このパラメタで指定した値で推移している場合、スレッド不足によって Mediator への送信待ちが発生しているおそれがあります。このパラメタの値を増やすことで、次の図のように処理時間を改善できる場合があります。図中の X は改善前の処理時間、図中の Y は改善後の処理時間です。

図 3-4　Prometheus の Web UI



また、分散トレースデータでもスレッドの空き待ち状態になっているかどうか確認できます。Jaeger の Web UI で確認したいトレース情報の Trace Page を表示し、次のオペレーション名の span を表示します。

「hmppcto.mediator.TerminationReceiverService/commit」
「hmppcto.mediator.TerminationReceiverService/rollback」

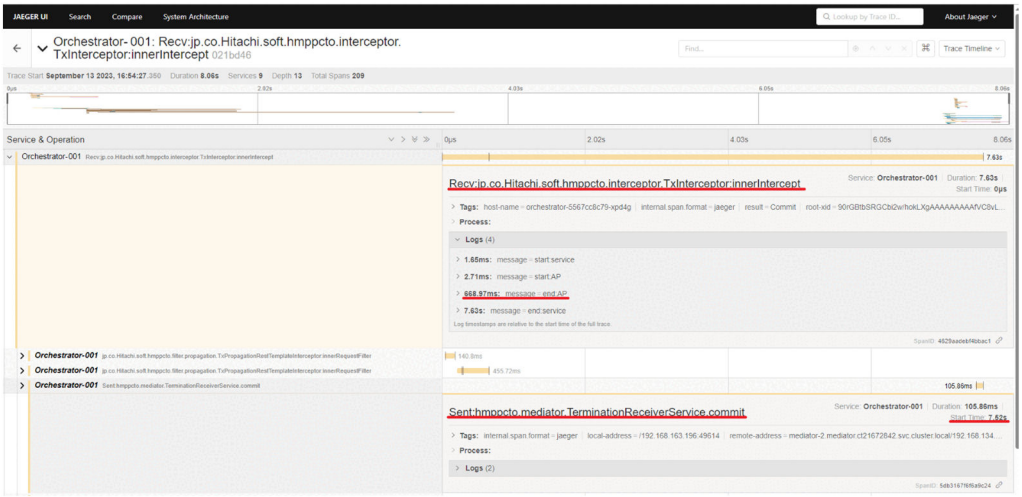
上記 span の Start Time が次の span の Logs「end AP」と差が大きい場合、このパラメタで指定するスレッドが不足してスレッドの空き待ち状態になっているおそれがあります。

オペレーション名「Recv:j.c.H.s.h.i.TxInterceptor:innerIntercept」

分散トレースの確認方法の詳細については、「10.5.2　分散トレース」を参照してください。

スレッドが空き待ちになっている状態の参考データを下図に示します。

図 3-5　スレッドが空き待ち状態



(2)　Entity-Service

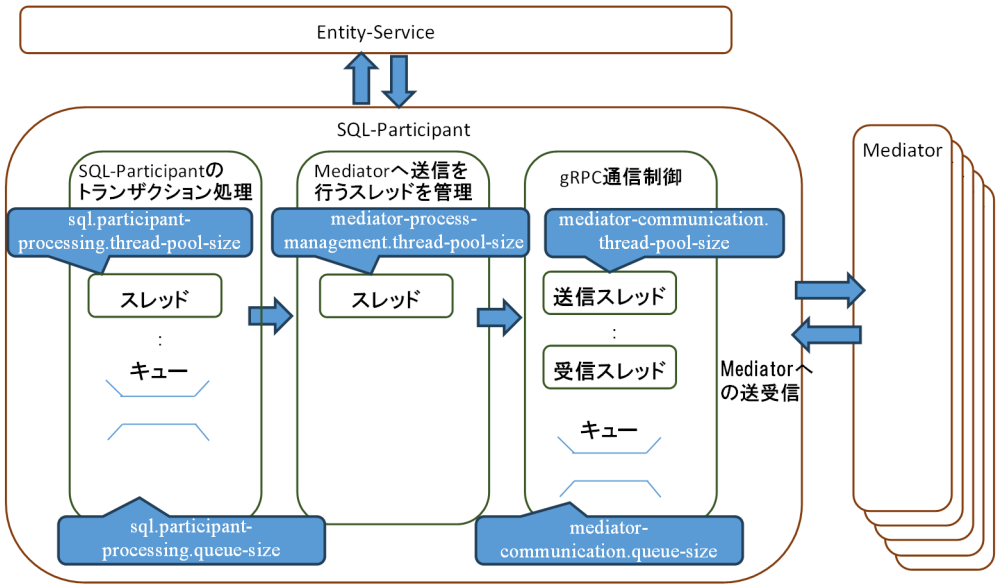
HMP-PCTO が制御するスレッドは存在しないため、HMP-PCTO のスレッドに関するパラメタのチューニングは必要ありません。

(3)　SQL-Participant

SQL-Participant では次に示す 3 つのスレッドプールがあります。

- SQL-Participant のトランザクション処理に使用するスレッドプール
- Mediator へ送信を行うスレッドの管理に使用するスレッドプール
- gRPC 通信制御に使用するスレッドプール

図 3-6　SQL-Participant の同時実行スレッド数制御



それぞれのスレッドプールについて以降に説明します。

SQL-Participant のトランザクション処理に使用するスレッドプールでは、Entity-Service からの Prepare 指示／トランザクションのキャンセル指示・投票処理・分散合意処理・トランザクション完了通知処理の同時実行スレッド数を制御します。同時実行スレッド数は SQL-Participant のパラメタ「sql.participant-processing.thread-pool-size」で指定します。キュー数は SQL-Participant のパラメタ「sql.participant-processing.queue-size」で指定します。

このスレッドプールで使用するスレッドとその占有期間は次のとおりです。

- Entity-Service と通信を行う際、SQL 実行／Prepare 指示／トランザクションのキャンセル指示を受けた数分スレッドを使用します。SQL 実行の場合のスレッドは SQL 実行を開始してから終了するまで占有します。Prepare 指示／トランザクションのキャンセル指示の場合のスレッドは投票処理の開始まで占有します。…1
- 1 トランザクション内で、投票処理の際に、1 スレッドを使用します。スレッドは投票処理を開始してから終了するまで占有します。…2
- 8 でスレッドを生成した際に、トランザクション処理または分散合意処理で 1 スレッドを使用します。スレッドはトランザクション処理または分散合意処理が開始してから終了するまで占有します。…3
- 1 つの分散合意処理のスレッドで、分散合意結果通知で 1 スレッドを使用します。スレッドは分散合意結果通知を開始してから終了するまで占有します。…4
- 1 トランザクション内で、トランザクション完了通知の際に、1 スレッドを使用します。スレッドはトランザクション完了通知を開始してから終了するまで占有します。なお、前述の分散合意結果通知で使用するスレッドの占有期間と、トランザクション完了通知で使用するスレッドの占有期間は、期間が重複することがあります。…5

Mediator へ送信を行うスレッドの管理に使用するスレッドプールでは、gRPC 通信制御に使用するスレッドを生成する処理の同時実行スレッド数を制御します。同時実行スレッド数は SQL-Participant のパラメタ「mediator-process-management.thread-pool-size」で指定します。

このスレッドプールで使用するスレッドとその占有期間は次のとおりです。

- Mediator との通信処理（投票処理、トランザクション完了通知、分散合意結果通知）が発生した際、トランザクションに参加した Mediator 数分スレッドを使用します。6 のスレッドの生成を開始してから終了するまで占有します。…6

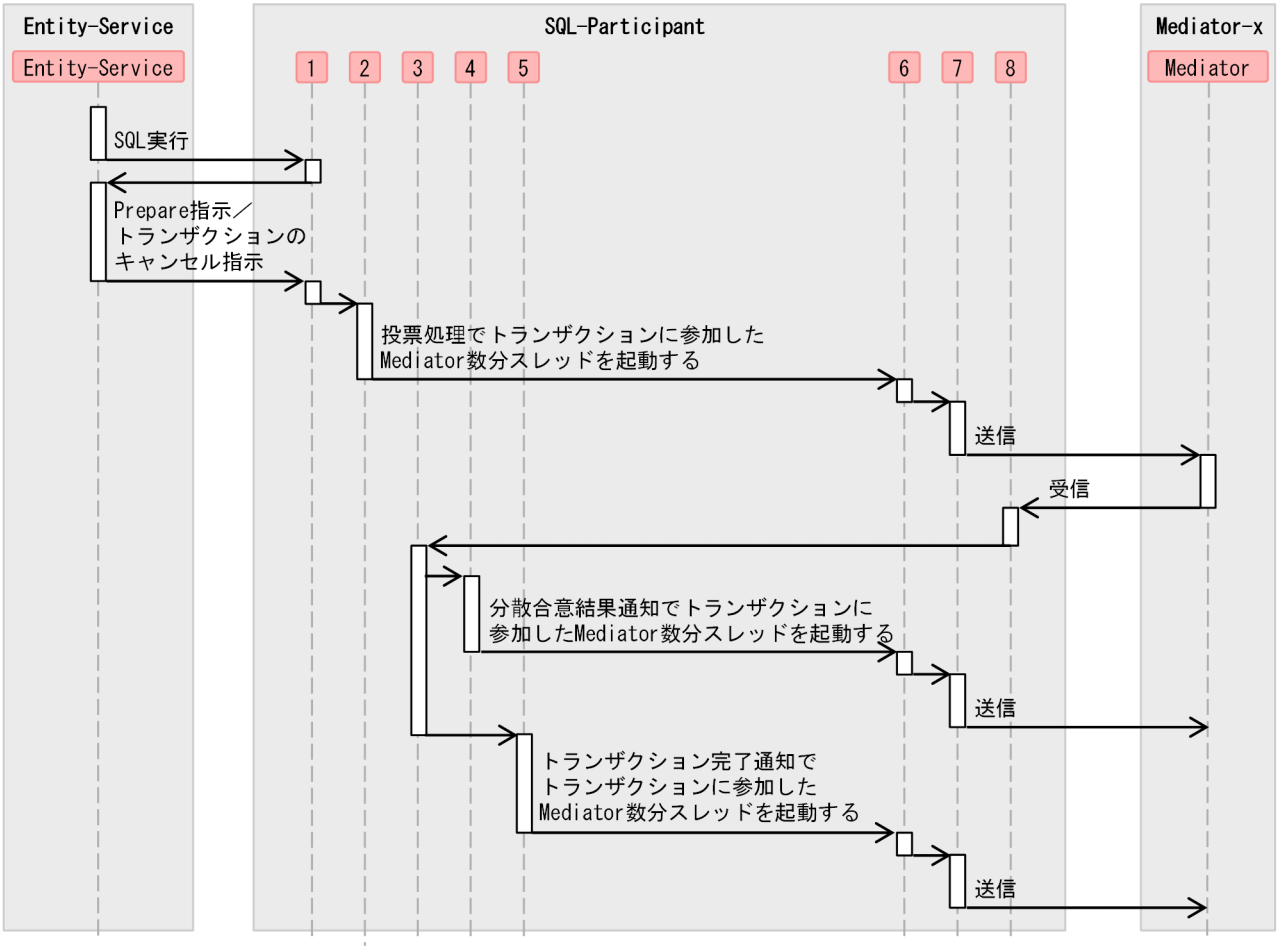
gRPC 通信制御に使用するスレッドプールでは、Participant が Mediator と送受信を行う処理の同時実行スレッド数を制御します。同時実行スレッド数は SQL-Participant のパラメタ「mediator-communication.thread-pool-size」で指定します。キュー数は SQL-Participant のパラメタ「mediator-communication.queue-size」で指定します。

このスレッドプールで使用するスレッドとその占有期間は次のとおりです。

- Mediator と通信を行う際、トランザクションに参加した Mediator 数分スレッドを使用します。Mediator への通信の送信処理が完了するまで占有します。…7

- Mediator からの gRPC 通信を受信した際、1 スレッド使用します。3 のスレッドの生成を開始してから終了するまで占有します。…8

図 3-7　SQL-Participant のスレッド占有期間



(a)　設計指針

■SQL-Participant のトランザクション処理に使用するスレッドプール

複数のトランザクションを同時に実行する際、極力スレッドの空き待ちが発生しないようにするためにはトランザクションの同時実行数分スレッドが必要です。

ただし、トランザクションの実行が重ならないかぎりは、トランザクションの同時実行数分のスレッドは必要ありません。

スレッドの占有期間に相当する処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために安全率を掛けることを推奨します。

SQL-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

X ＝ A × B

X:スレッド数

A:目標 TPS

B:処理時間（秒）

例）

目標 40TPS、処理時間を 0.1 秒とした場合、必要なスレッド数は 4 となります。

X ＝ 40 × 0.1

X ＝ 4

安全率を 1.5 とした場合、必要なスレッド数は 6 となります。

X ＝ 4 × 1.5

X ＝ 6

■SQL-Participant のトランザクション処理に使用するキュー

複数のトランザクションを同時に実行する際、極力スレッドプールのキューの枯渇が発生しないようにするためには十分なキューサイズが必要です。

目標 TPS に応じて、キューサイズを算出します。

SQL-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

B の係数には実稼働を通して環境に合わせた数値を設定してください。

Y ＝ A × B

Y:キュー数

A:目標 TPS

B:係数

例）

目標 40TPS、係数を 0.25 とした場合、必要なキューサイズは 10 となります。

Y ＝ 40 × 0.25

Y ＝ 10

■Mediator へ送信を行うスレッドの管理に使用するスレッドプール

複数のトランザクションを同時に実行する際、極力スレッドの空き待ちが発生しないようにするためにはトランザクションの同時実行数×トランザクションに参加する Mediator の台数分スレッドが必要です。

ただし、トランザクションの実行が重ならないかぎりは、トランザクションの同時実行数×トランザクションに参加する Mediator の台数分のスレッドは必要ありません。

スレッドの占有期間に相当する処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために安全率を掛けることを推奨します。

算出したスレッド数がトランザクションに参加する Mediator の台数の 2 倍未満になった場合は、トランザクションに参加する Mediator の台数を 2 倍したスレッド数としてください。

SQL-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

X ＝ A × B × C × 2※

X:スレッド数

A:目標 TPS

B:トランザクションに参加する Mediator 台数

C:処理時間（秒）

注※

分散合意結果通知とトランザクション完了通知は同時に実行される可能性があるため、2 倍します。

例）

目標 40TPS、トランザクションに参加する Mediator5 台、処理時間 0.1 秒とした場合、必要なスレッド数は 40 となります。

X ＝ 40 × 5 × 0.1 × 2

X ＝ 40

安全率を 1.5 とした場合、必要なスレッド数は 60 となります。

X ＝ 40 × 1.5

X ＝ 60

■gRPC 通信制御に使用するスレッドプール

複数のトランザクションを同時に実行する際、極力スレッドの空き待ちが発生しないようにするためにはトランザクションの同時実行数×トランザクションに参加する Mediator の台数分スレッドが必要です。

ただし、トランザクションの実行が重ならないかぎりは、トランザクションの同時実行数×トランザクションに参加する Mediator の台数分のスレッドは必要ありません。

スレッドの占有期間に相当する処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために安全率を掛けることを推奨します。

算出したスレッド数がトランザクションに参加する Mediator の台数の 2 倍未満になった場合は、トランザクションに参加する Mediator の台数を 2 倍したスレッド数としてください。

SQL-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

X = A × B × C × 2※

X:スレッド数

A:目標 TPS

B:トランザクションに参加する Mediator 台数

C: 処理時間（秒）

注※

分散合意結果通知とトランザクション完了通知は同時に実行される可能性があるため、2 倍します。

例）

目標 40TPS、トランザクションに参加する Mediator5 台、処理時間 0.1 秒とした場合、必要なスレッド数は 40 となります。

X = 40 × 5 × 0.1 × 2

X = 40

安全率を 1.5 とした場合、必要なスレッド数は 60 となります。

X = 40 × 1.5

X = 60

■gRPC 通信制御に使用するキュー

複数のトランザクションを同時に実行する際、極力スレッドプールのキューの枯渇が発生しないようにするためには十分なキューサイズが必要です。

目標 TPS に応じて、キューサイズを算出します。

SQL-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

B の係数には実稼働を通して環境に合わせた数値を設定してください。

Y = A × B × 2※

Y:キュー数

A:目標 TPS

B:係数

注※

分散合意結果通知とトランザクション完了通知は同時に実行される可能性があるため、2 倍します。

例）

目標 40TPS、係数を 0.25 とした場合、必要なキューサイズは 20 となります。

Y = 40 × 0.25 × 2

Y = 20

(b) チューニング対象のパラメタ一覧

表 3-24 SQL-Participant のチューニング対象のパラメタ一覧

項番	パラメタ	説明
1	sql.participant-processing.thread-pool-size	SQL-Participant がトランザクション処理で実行するスレッド数を指定します。
2	sql.participant-processing.queue-size	SQL-Participant がトランザクション処理に使用するスレッドの割り当て待ちをするキューのサイズを指定します。
3	mediator-process-management.thread-pool-size	Mediator へ送信を行うスレッドを管理するスレッド数を指定します。
4	mediator-communication.thread-pool-size	Participant が Mediator と送受信を行うスレッド数を指定します。
5	mediator-communication.queue-size	Participant が Mediator との送受信に使用するスレッドの割り当て待ちをするキューのサイズを指定します。

各パラメタの詳細については、「8.2.4 SQL-Participant」を参照してください。

(c) チューニング方法

各パラメタに関する情報はメトリクスデータによって確認できます。メトリクスデータの確認方法の詳細については、「10.5.3 メトリクス」を参照してください。

Prometheus の Web UI で次に示す表のクエリを入力し、Execute ボタンを押すことで Graph タブに下図のような状態で表示されます。

チューニング対象のパラメタと、下図に表示されたメトリクス名の関連づけは、「表 3-26 チューニング対象のパラメタとメトリクス名の関連づけ」を参照してください。メトリクス名は、クエリを実行した結果によって「name="メトリクス名"」と表示されるため、該当する表示個所からメトリクス名を確認してください。

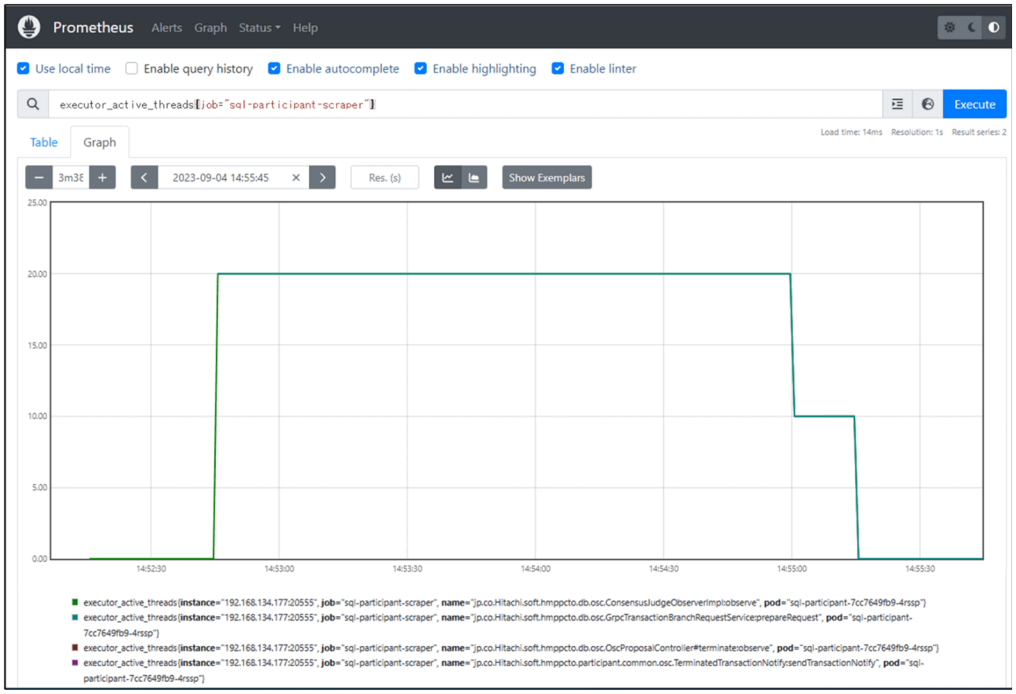
表 3-25 Prometheus で利用するクエリ

チューニング対象のパラメタ	クエリ
sql.participant-processing.thread-pool-size	executor_active_threads{job=" sql-participant-scraper"}
mediator-process-management.thread-pool-size	
mediator-communication.thread-pool-size	
sql.participant-processing.queue-size	executor_queued_tasks{job=" sql-participant-scraper"}
mediator-communication.queue-size	

表 3-26　チューニング対象のパラメタとメトリクス名の関連づけ

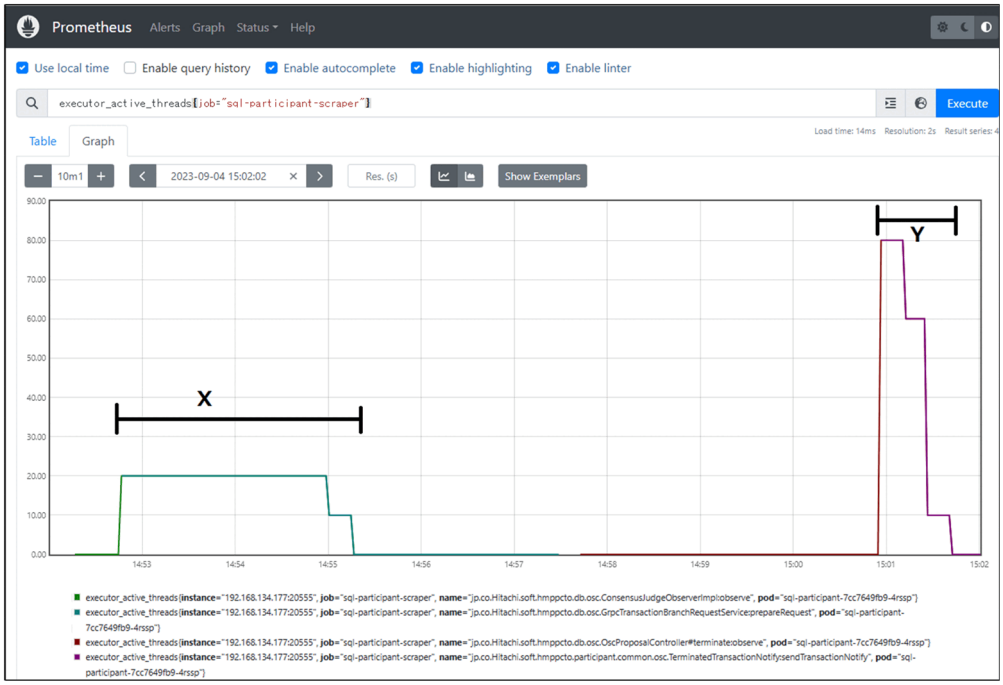
チューニング対象のパラメタ	メトリクス名
sql.participant-processing.thread-pool-size	jp.co.Hitachi.soft.hmppcto.db.osc.GrpcTransactionBranchRequestService:prepareRequest
	jp.co.Hitachi.soft.hmppcto.db.osc.GrpcTransactionBranchRequestService:cancelRequest
	jp.co.Hitachi.soft.hmppcto.txpaxosmanager.osc.OscProposalController#terminate:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.ConsensusJudgeObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionNotify:sendTransactionNotify
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminationProposalObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionVerifyNotifyObserverImpl:observe
sql.participant-processing.queue-size	jp.co.Hitachi.soft.hmppcto.db.osc.GrpcTransactionBranchRequestService:prepareRequest
	jp.co.Hitachi.soft.hmppcto.db.osc.GrpcTransactionBranchRequestService:cancelRequest
	jp.co.Hitachi.soft.hmppcto.txpaxosmanager.osc.OscProposalController#terminate:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.ConsensusJudgeObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionNotify:sendTransactionNotify
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminationProposalObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionVerifyNotifyObserverImpl:observe

図 3-8　Prometheus の Web UI



executor_active_threads{job="sql-participant-scraper"}の結果が、長い期間 sql.participant-processing.thread-pool-size、mediator-process-management.thread-pool-size、mediator-communication.thread-pool-size で指定した値で推移している場合、スレッド不足によって、処理の遅延が発生しているおそれがあります。これらのパラメタの値を増やすことで、次の図のように処理時間を改善できる場合があります。図中の X は改善前の処理時間、図中の Y は改善後の処理時間です。

図 3-9　Prometheus の Web UI



また、executor_queued_tasks{job="sql-participant-scraper"}の結果が、sql.participant-processing.queue-size や mediator-communication.queue-size で指定した値で推移している場合、キューサイズ不足が発生しているおそれがあります。これらのパラメタの値を増やすことでキューサイズ不足を防ぐことができます。

また、分散トレースデータでもスレッドの空き待ち状態になっているかどうか確認できます。Jaeger の Web UI で確認したいトレース情報の Trace Page を表示し、次のオペレーション名の span を表示します。

- sql.participant-processing.thread-pool-size
  オペレーション名「j.c.H.s.h.p.s.GrpcTerminationJudgeRequestSender」
  (Tags が「internal.span.format」と「status.code」だけの span)
  オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」
  (Tags が「internal.span.format」と「status.code」だけの span)
  オペレーション名「j.c.H.s.h.p.c.o.TerminatedTransactionNotify:sendTransactionNotify」
  (Tags が「internal.span.format」と「status.code」だけの span)
- mediator-process-management.thread-pool-size
  オペレーション名「j.c.H.s.h.p.s.GrpcTerminationJudgeRequestSender」
  (Tags に「branch-xid」などが含まれる span)
  オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」
  (Tags に「consensus-value」などが含まれる span)
  オペレーション名「j.c.H.s.h.p.c.o.TerminatedTransactionNotify:sendTransactionNotify」
  (Tags に「host-name」などが含まれる span)
- mediator-communication.thread-pool-size
  オペレーション名「hmppcto.mediator.TerminationReceiverService/propose」
  オペレーション名「hmppcto.mediator.TerminationReceiverService/notifyTerminationResult」
  オペレーション名「hmppcto.mediator.TerminationReceiverService/
  notifyTerminatedTransaction」

上記 span の Duration が大きい場合、SQL-Participant のパラメタで指定するスレッドが不足してスレッドの空き待ち状態になっているおそれがあります。

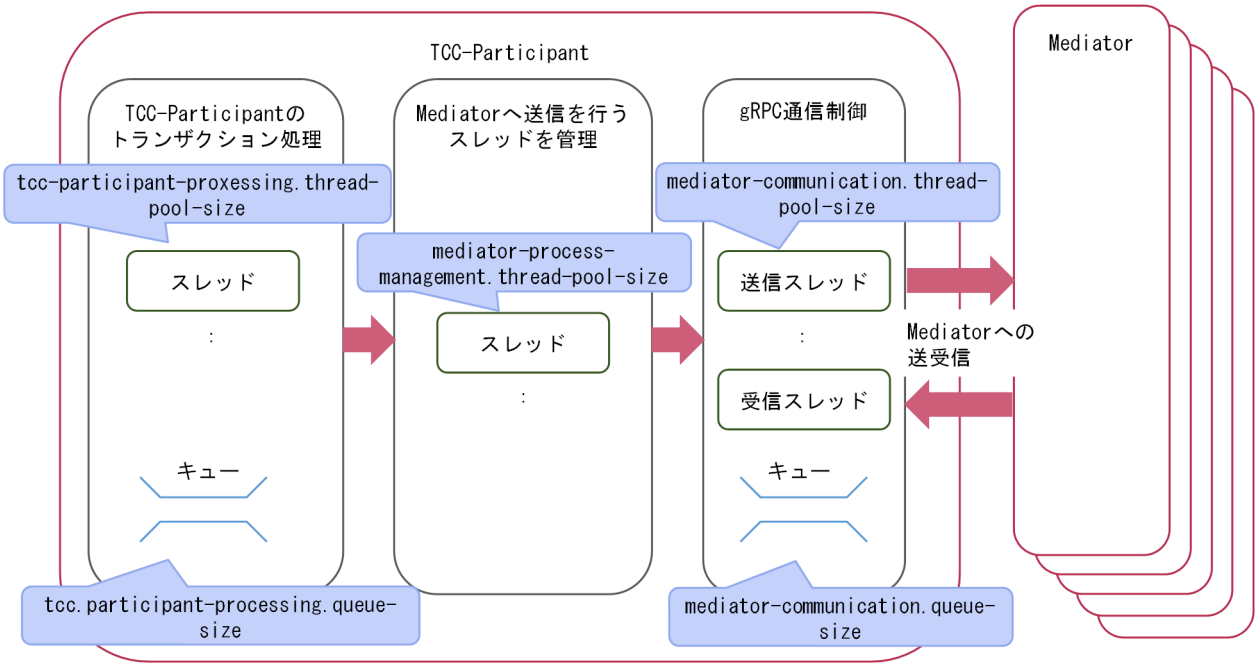
分散トレースの確認方法の詳細については、「10.5.2　分散トレース」を参照してください。

(4)　TCC-Participant

TCC-Participant では次に示す 3 つのスレッドプールがあります。

- TCC-Participant のトランザクション処理に使用するスレッドプール
- Mediator へ送信を行うスレッドの管理に使用するスレッドプール
- gRPC 通信制御に使用するスレッドプール

図 3-10　TCC-Participant の同時実行スレッド数制御



それぞれのスレッドプールについて以降に説明します。

TCC-Participant のトランザクション処理に使用するスレッドプールでは、投票処理・分散合意処理・トランザクション完了通知処理の同時実行スレッド数を制御します。同時実行スレッド数は TCC-Participant のパラメタ「tcc.participant-processing.thread-pool-size」で指定します。キュー数は TCC-Participant のパラメタ「tcc.participant-processing.queue-size」で指定します。

このスレッドプールで使用するスレッドとその占有期間は次のとおりです。

- 1 トランザクション内で、投票処理の際に、1 スレッドを使用します。スレッドは投票処理を開始してから終了するまで占有します。…1
- 7 でスレッドを生成した際に、トランザクション処理または分散合意処理で 1 スレッドを使用します。スレッドはトランザクション処理または分散合意処理が開始してから終了するまで占有します。…2
- 1 つの分散合意処理のスレッドで、分散合意結果通知で 1 スレッドを使用します。スレッドは分散合意結果通知を開始してから終了するまで占有します。…3
- 1 トランザクション内で、トランザクション完了通知の際に、1 スレッドを使用します。スレッドはトランザクション完了通知を開始してから終了するまで占有します。なお、前述の分散合意結果通知で使用するスレッドの占有期間と、トランザクション完了通知で使用するスレッドの占有期間は、期間が重複することがあります。…4

Mediator へ送信を行うスレッドの管理に使用するスレッドプールでは、gRPC 通信制御に使用するスレッドを生成する処理の同時実行スレッド数を制御します。同時実行スレッド数は TCC-Participant のパラメタ「mediator-process-management.thread-pool-size」で指定します。

このスレッドプールで使用するスレッドとその占有期間は次のとおりです。

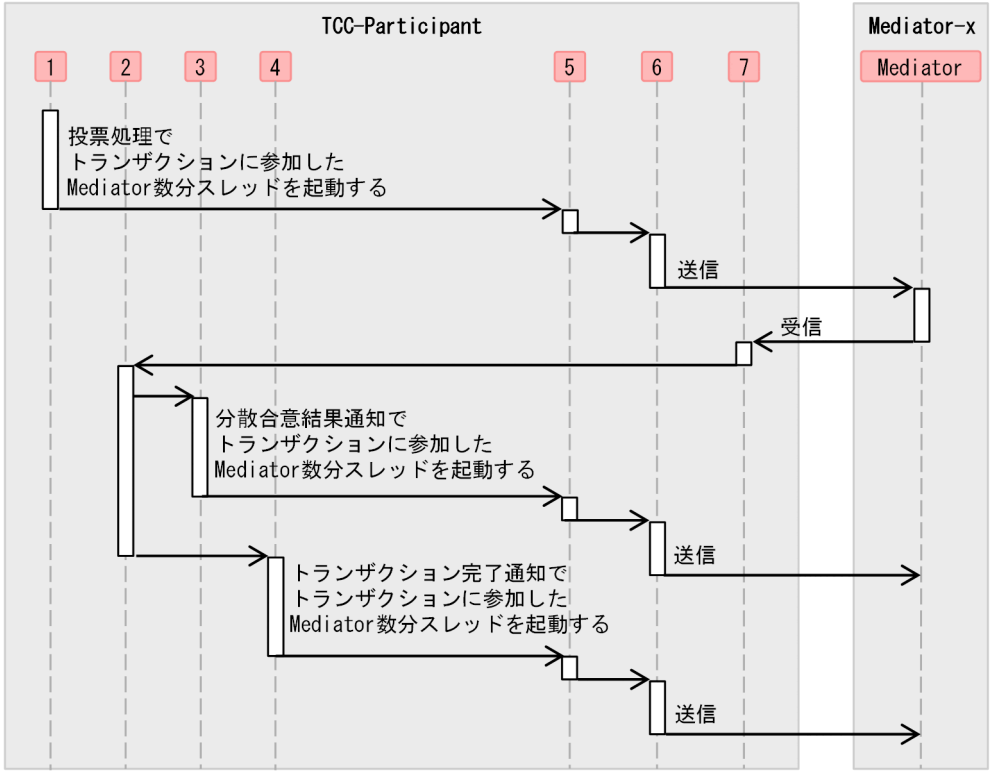
- Mediator との通信処理（投票処理、トランザクション完了通知、分散合意結果通知）が発生した際、トランザクションに参加した Mediator 数分スレッドを使用します。6 のスレッドの生成を開始してから終了するまで占有します。…5

gRPC 通信制御に使用するスレッドプールでは、Participant が Mediator と送受信を行う処理の同時実行スレッド数を制御します。同時実行スレッド数は TCC-Participant のパラメタ「mediator-communication.thread-pool-size」で指定します。キュー数は TCC-Participant のパラメタ「mediator-communication.queue-size」で指定します。

このスレッドプールで使用するスレッドとその占有期間は次のとおりです。

- Mediator と通信を行う際、トランザクションに参加した Mediator 数分スレッドを使用します。Mediator への通信の送信処理が完了するまで占有します。…6
- Mediator からの gRPC 通信を受信した際、1 スレッド使用します。2 のスレッドの生成を開始してから終了するまで占有します。…7

図 3-11　TCC-Participant のスレッド占有期間



(a)　設計指針

■TCC-Participant のトランザクション処理に使用するスレッドプール

複数のトランザクションを同時に実行する際、極力スレッドの空き待ちが発生しないようにするためにはトランザクションの同時実行数分スレッドが必要です。

ただし、トランザクションの実行が重ならないかぎりは、トランザクションの同時実行数分のスレッドは必要ありません。

スレッドの占有期間に相当する処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために安全率を掛けることを推奨します。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

X ＝ A × B

X:スレッド数

A:目標 TPS

B:処理時間（秒）

例）

目標 40TPS、処理時間を 0.1 秒とした場合、必要なスレッド数は 4 となります。

X ＝ 40 × 0.1

X ＝ 4

安全率を 1.5 とした場合、必要なスレッド数は 6 となります。

X ＝ 4 × 1.5

X ＝ 6

■TCC-Participant のトランザクション処理に使用するキュー

複数のトランザクションを同時に実行する際、極力スレッドプールのキューの枯渇が発生しないようにするためには十分なキューサイズが必要です。

目標 TPS に応じて、キューサイズを算出します。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

B の係数には実稼働を通して環境に合わせた数値を設定してください。

Y ＝ A × B

Y:キュー数

A:目標 TPS

B:係数

例）

目標 40TPS、係数を 0.25 とした場合、必要なキューサイズは 10 となります。

Y ＝ 40 × 0.25

Y ＝ 10

■Mediator へ送信を行うスレッドの管理に使用するスレッドプール

複数のトランザクションを同時に実行する際、極力スレッドの空き待ちが発生しないようにするためにはトランザクションの同時実行数×トランザクションに参加する Mediator の台数分スレッドが必要です。

ただし、トランザクションの実行が重ならないかぎりは、トランザクションの同時実行数×トランザクションに参加する Mediator の台数分のスレッドは必要ありません。

スレッドの占有期間に相当する処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために安全率を掛けることを推奨します。

算出したスレッド数がトランザクションに参加する Mediator の台数の 2 倍未満になった場合は、トランザクションに参加する Mediator の台数を 2 倍したスレッド数としてください。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

X ＝ A × B × C × 2※

X:スレッド数

A:目標 TPS

B:トランザクションに参加する Mediator 台数

C:処理時間（秒）

注※

分散合意結果通知とトランザクション完了通知は同時に実行される可能性があるため、2 倍します。

例）

目標 40TPS、トランザクションに参加する Mediator5 台、処理時間 0.1 秒とした場合、必要なスレッド数は 40 となります。

X ＝ 40 × 5 × 0.1 × 2

X ＝ 40

安全率を 1.5 とした場合、必要なスレッド数は 60 となります。

X ＝ 40 × 1.5

X ＝ 60

■gRPC 通信制御に使用するスレッドプール

複数のトランザクションを同時に実行する際、極力スレッドの空き待ちが発生しないようにするためにはトランザクションの同時実行数×トランザクションに参加する Mediator の台数分スレッドが必要です。

ただし、トランザクションの実行が重ならないかぎりは、トランザクションの同時実行数×トランザクションに参加する Mediator の台数分のスレッドは必要ありません。

スレッドの占有期間に相当する処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために安全率を掛けることを推奨します。

算出したスレッド数がトランザクションに参加する Mediator の台数の 2 倍未満になった場合は、トランザクションに参加する Mediator の台数を 2 倍したスレッド数としてください。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

$X = A \times B \times C \times 2$※

X:スレッド数

A:目標 TPS

B:トランザクションに参加する Mediator 台数

C: 処理時間（秒）

注※

分散合意結果通知とトランザクション完了通知は同時に実行される可能性があるため、2 倍します。

例）

目標 40TPS、トランザクションに参加する Mediator5 台、処理時間 0.1 秒とした場合、必要なスレッド数は 40 となります。

$X = 40 \times 5 \times 0.1 \times 2$

$X = 40$

安全率を 1.5 とした場合、必要なスレッド数は 60 となります。

$X = 40 \times 1.5$

$X = 60$

■gRPC 通信制御に使用するキュー

複数のトランザクションを同時に実行する際、極力スレッドプールのキューの枯渇が発生しないようにするためには十分なキューサイズが必要です。

目標 TPS に応じて、キューサイズを算出します。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

B の係数には実稼働を通して環境に合わせた数値を設定してください。

$Y = A \times B \times 2$※

Y:キュー数

A:目標 TPS

B:係数

注※

分散合意結果通知とトランザクション完了通知は同時に実行される可能性があるため、2 倍します。

例）

目標 40TPS、係数を 0.25 とした場合、必要なキューサイズは 20 となります。

$Y = 40 \times 0.25 \times 2$

$Y = 20$

(b)　チューニング対象のパラメター一覧

表 3-27　TCC-Participant のチューニング対象のパラメター一覧

項番	パラメタ	説明
1	tcc.participant-processing.thread-pool-size	TCC-Participant がトランザクション処理で実行するスレッド数を指定します。
2	tcc.participant-processing.queue-size	TCC-Participant がトランザクション処理に使用するスレッドの割り当てを待つキューのサイズを指定します。
3	mediator-process-management.thread-pool-size	Mediator へ送信を行うスレッドを管理するスレッド数を指定します。
4	mediator-communication.thread-pool-size	Participant が Mediator と送受信を行うスレッド数を指定します。
5	mediator-communication.queue-size	Participant が Mediator との送受信に使用するスレッドの割り当てを待つキューのサイズを指定します。

各パラメタの詳細については、「8.2.5　TCC-Participant」を参照してください。

(c)　チューニング方法

各パラメタに関する情報はメトリクスデータによって確認できます。メトリクスデータの確認方法の詳細については、「10.5.3　メトリクス」を参照してください。

Prometheus の Web UI で次に示す表のクエリを入力し、Execute ボタンを押すことで Graph タブに下図のような状態で表示されます。

チューニング対象のパラメタと、下図に表示されたメトリクス名の関連づけは、「表 3-29　チューニング対象のパラメタとメトリクス名の関連づけ」を参照してください。メトリクス名は、クエリを実行した結果によって「name="メトリクス名"」と表示されるため、該当する表示個所からメトリクス名を確認してください。

表 3-28　Prometheus で利用するクエリ

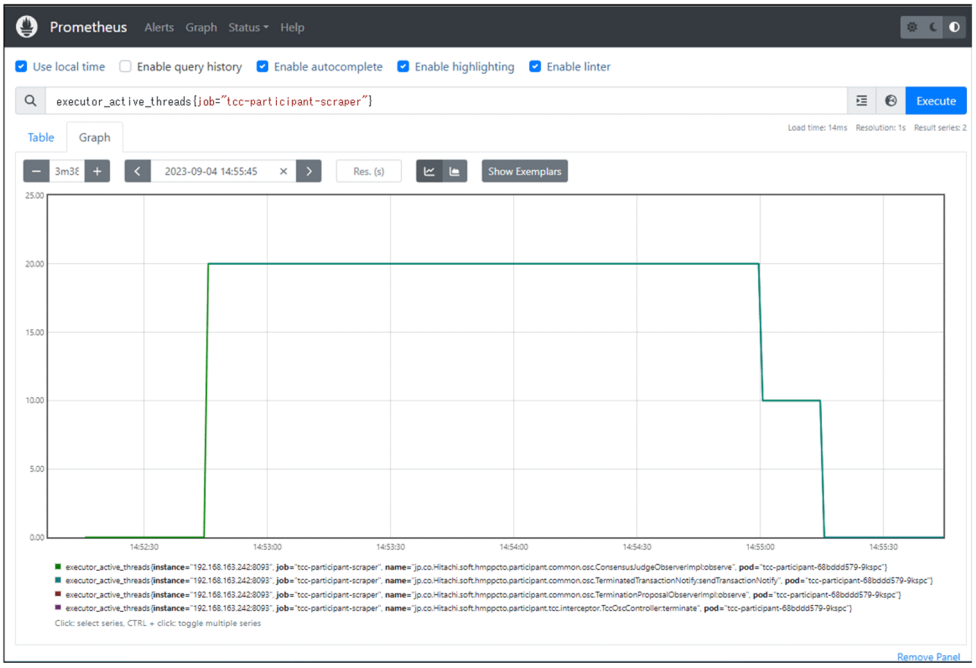
チューニング対象のパラメタ	クエリ
tcc.participant-processing.thread-pool-size	executor_active_threads{job=" tcc-participant-scraper"}
mediator-process-management.thread-pool-size	
mediator-communication.thread-pool-size	

チューニング対象のパラメタ	クエリ
tcc.participant-processing.queue-size	executor_queued_tasks{job=" tcc-participant-scraper"}
mediator-communication.queue-size	

表 3-29　チューニング対象のパラメタとメトリクス名の関連づけ

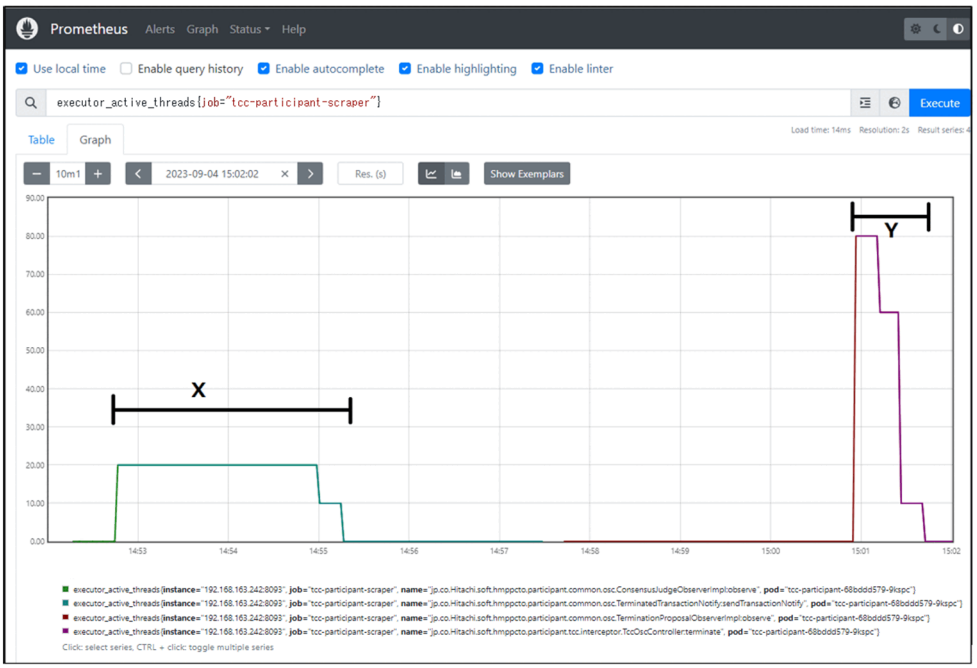
チューニング対象のパラメタ	メトリクス名
tcc.participant-processing.thread-pool-size	jp.co.Hitachi.soft.hmppcto.participant.tcc.interceptor.TccOscController:terminate
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.ConsensusJudgeObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionNotify:sendTransactionNotify
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminationProposalObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionVerifyNotifyObserverImpl:observe
tcc.participant-processing.queue-size	jp.co.Hitachi.soft.hmppcto.participant.tcc.interceptor.TccOscController:terminate
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.ConsensusJudgeObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionNotify:sendTransactionNotify
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminationProposalObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionVerifyNotifyObserverImpl:observe

図 3-12　Prometheus の Web UI



executor_active_threads{job=" tcc-participant-scraper"}の結果が、長い期間 tcc.participant-processing.thread-pool-size、mediator-process-management.thread-pool-size、mediator-communication.thread-pool-size で指定した値で推移している場合、スレッド不足によって、処理の遅延が発生しているおそれがあります。これらのパラメタの値を増やすことで、次の図のように処理時間を改善できる場合があります。図中の X は改善前の処理時間、図中の Y は改善後の処理時間です。

図 3-13　Prometheus の Web UI



また、executor_queued_tasks{job=" tcc-participant-scraper"}の結果が、tcc.participant-processing.queue-size や mediator-communication.queue-size で指定した値で推移している場合、

キューサイズ不足が発生しているおそれがあります。これらのパラメタの値を増やすことでキューサイズ不足を防ぐことができます。

また、分散トレースデータでもスレッドの空き待ち状態になっているかどうか確認できます。Jaeger の Web UI で確認したいトレース情報の Trace Page を表示し、次のオペレーション名の span を表示します。

- tcc.participant-processing.thread-pool-size
  オペレーション名「j.c.H.s.h.p.s.GrpcTerminationJudgeRequestSender」
  （Tags が「internal.span.format」と「status.code」だけの span）
  オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」
  （Tags が「internal.span.format」と「status.code」だけの span）
  オペレーション名「j.c.H.s.h.p.c.o.TerminatedTransactionNotify:sendTransactionNotify」
  （Tags が「internal.span.format」と「status.code」だけの span）
- mediator-process-management.thread-pool-size
  オペレーション名「j.c.H.s.h.p.s.GrpcTerminationJudgeRequestSender」
  （Tags に「branch-xid」などが含まれる span）
  オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」
  （Tags に「consensus-value」などが含まれる span）
  オペレーション名「j.c.H.s.h.p.c.o.TerminatedTransactionNotify:sendTransactionNotify」
  （Tags に「host-name」などが含まれる span）
- mediator-communication.thread-pool-size
  オペレーション名「hmppcto.mediator.TerminationReceiverService/propose」
  オペレーション名「hmppcto.mediator.TerminationReceiverService/notifyTerminationResult」
  オペレーション名「hmppcto.mediator.TerminationReceiverService/
  notifyTerminatedTransaction」

上記 span の Duration が大きい場合、TCC-Participant のパラメタで指定するスレッドが不足してスレッドの空き待ち状態になっているおそれがあります。

分散トレースの確認方法の詳細については、「10.5.2　分散トレース」を参照してください

(5)　Mediator

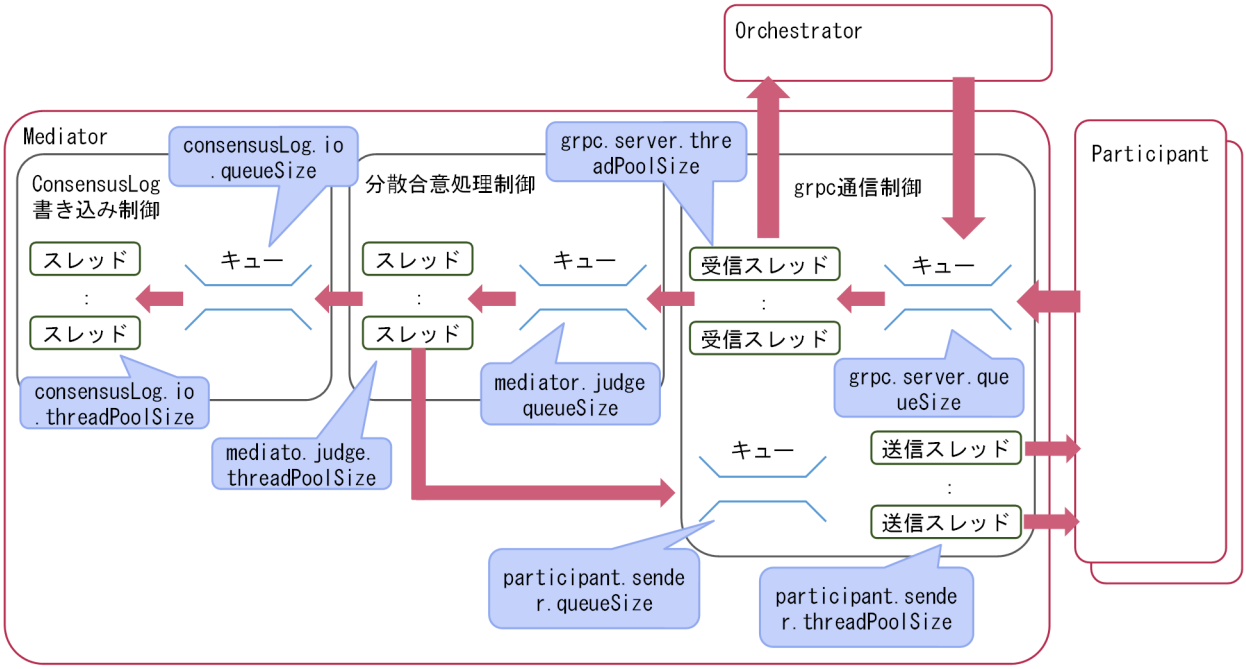
Mediator では次に示す 4 つのスレッドプールがあります。

表 3-30　Mediator のスレッドプール

項番	Mediator のスレッド	説明
1	Mediator の受信に使用するスレッド	Mediator が Participant、Orchestrator から受信する際の同時実行スレッド数を制御します。

項番	Mediator のスレッド	説明
		同時実行スレッド数は Mediator のパラメタ「grpc.server.threadPoolSize」で指定します。キュー数は Mediator のパラメタ「grpc.server.queueSize」で指定します。 同時実行スレッド数を超えた要求がある場合は、キューに保持します。 トランザクションに参加した Participant の数＋ 1 だけスレッドを使用します。また、スレッドは Participant、Orchestrator からの受信完了まで占有します。
2	分散合意処理に使用するスレッド	分散合意処理の同時実行スレッド数を制御します。 同時実行スレッド数は Mediator のパラメタ「mediator.judge.threadPoolSize」で指定します。キュー数は Mediator のパラメタ「mediator.judge.queueSize」で指定します。 同時実行スレッド数を超えた要求がある場合は、キューに保持します。 トランザクションに参加した Participant の数だけスレッドを使用します。また、スレッドは分散合意処理を行い合意結果を ConsensusLog ファイルへの書き込みが完了するまで占有します。
3	ConsensusLog ファイルへの書き込みに使用するスレッド	ConsensusLog ファイルへの書き込みの同時実行スレッド数を制御します。 同時実行スレッド数は Mediator のパラメタ「consensusLog.io.threadPoolSize」で指定します。キュー数は Mediator のパラメタ「consensusLog.io.queueSize」で指定します。 同時実行スレッド数を超えた要求がある場合は、キューに保持します。 トランザクション数だけスレッドを使用します。 スレッドは合意結果を ConsensusLog ファイルへ書き込みが完了するまで占有します。
4	Mediator の送信に使用するスレッド	Participant への送信の同時実行スレッド数を制御します。 同時実行スレッド数は Mediator のパラメタ「participant.sender.threadPoolSize」で指定します。キュー数は Mediator のパラメタ「participant.sender.queueSize」で指定します。 同時実行スレッド数を超えた要求がある場合は、キューに保持します。 トランザクションに参加した Participant の数だけスレッドを使用します。また、スレッドは Participant への送信完了まで占有します。

図 3-14　同時実行スレッド数制御



(a)　設計指針

■Mediator の受信処理に使用するスレッドプール

複数のトランザクションを同時に実行する場合、極力スレッドの空き待ちを発生しないようにするためにはトランザクションの同時実行数×（トランザクションに参加する Participant の台数＋ 1）分スレッドが必要です。

ただし、トランザクションの受信が重ならないかぎりは、理論上の必要なスレッド数は必要ありません。

スレッドの占有期間に相当する受信処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるため安全係数を掛けることを推奨します。

算出したスレッド数がトランザクションに参加する Participant の台数＋ 1 の 2 倍未満になった場合は、トランザクションに参加する Participant の台数＋ 1 を 2 倍したスレッド数としてください。

$X = A \times (B + 1) \times C$

X:スレッド数

A:目標 TPS

B:トランザクションに参加する Participant 台数

D:受信処理時間（秒）

例）

目標 40TPS、トランザクションに参加する Participant2 台、受信処理時間 0.1 秒とした場合、必要なスレッド数は 12 となります。

X = 40 × (2 + 1) × 0.1

X = 12

■Mediator の受信処理に使用するキュー

複数のトランザクションを同時に実行する際、極力スレッドプールのキューの枯渇が発生しないようにするためには十分なキューサイズが必要です。

目標 TPS に応じて、キューサイズを算出します。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

B の係数には実稼働を通して環境に合わせた数値を設定してください。

Y = A × B

Y:キュー数

A:目標 TPS

B:係数

例）

目標 40TPS、係数を 0.25 とした場合、必要なキューサイズは 10 となります。

Y = 40 × 0.25

Y = 10

■分散合意処理に使用するスレッドプール

複数のトランザクションを同時に実行する場合、極力スレッドの空き待ちを発生しないようにするためにはトランザクションの同時実行数×トランザクションに参加する Participant の台数分スレッドが必要です。

ただし、トランザクションの受信が重ならないかぎりは、理論上の必要なスレッド数は必要ありません。

スレッドの占有期間に相当する分散合意処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるため安全係数を掛けることを推奨します。

算出したスレッド数がトランザクションに参加する Participant の台数の 2 倍未満になった場合は、トランザクションに参加する Participant の台数を 2 倍したスレッド数としてください。

X = A × B × C

X:スレッド数

A:TPS

B: トランザクションに参加する Participant 台数

D: 分散合意処理時間（秒）

例）

目標 40TPS、トランザクションに参加する Participant2 台、分散合意処理時間 0.1 秒とした場合、必要なスレッド数は 8 となります。

X = 40 × 2 × 0.1

X =8

■分散合意処理に使用するキュー

複数のトランザクションを同時に実行する際、極力スレッドプールのキューの枯渇が発生しないようにするためには十分なキューサイズが必要です。

目標 TPS に応じて、キューサイズを算出します。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

B の係数には実稼働を通して環境に合わせた数値を設定してください。

Y = A × B

Y:キュー数

A:目標 TPS

B:係数

例）

目標 40TPS、係数を 0.25 とした場合、必要なキューサイズは 10 となります。

Y = 40 × 0.25

Y = 10

■ConsensusLog ファイルへの書き込みに使用するスレッドプール

複数のトランザクションを同時に実行する場合、極力スレッドの空き待ちを発生しないようにするためにはトランザクションの同時実行数分スレッドが必要です。

ただし、ConsensusLog ファイルへの書き込みが重ならないかぎりは、理論上の必要なスレッド数は必要ありません。

スレッドの占有期間に相当する ConsensusLog ファイルへの書き込み処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるため安全係数を掛けることを推奨します。

X = A × C

X:スレッド数

A:TPS

B: ConsensusLog ファイルへの書き込み処理時間（秒）

例）

目標 40TPS、ConsensusLog ファイルへの書き込み処理時間 0.05 秒とした場合、必要なスレッド数は 2 となります。

X ＝ 40 × 0.05

X ＝ 2

■ConsensusLog ファイルへの書き込みに使用するキュー

複数のトランザクションを同時に実行する際、極力スレッドプールのキューの枯渇が発生しないようにするためには十分なキューサイズが必要です。

目標 TPS に応じて、キューサイズを算出します。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

B の係数には実稼働を通して環境に合わせた数値を設定してください。

Y ＝ A × B

Y:キュー数

A:目標 TPS

B:係数

例）

目標 40TPS、係数を 0.25 とした場合、必要なキューサイズは 10 となります。

Y ＝ 40 × 0.25

Y ＝ 10

■Mediator の送信処理に使用するスレッドプール

複数のトランザクションを同時に実行する場合、極力スレッドの空き待ちを発生しないようにするためにはトランザクションの同時実行数×トランザクションに参加する Participant の台数分スレッドが必要です。

ただし、トランザクションの送信が重ならないかぎりは、理論上の必要なスレッド数は必要ありません。

スレッドの占有期間に相当する送信処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるため安全係数を掛けることを推奨します。

算出したスレッド数がトランザクションに参加する Participant の台数の 2 倍未満になった場合は、トランザクションに参加する Participant の台数を 2 倍したスレッド数としてください。

X ＝ A × B × C

X:スレッド数

A:TPS

B: トランザクションに参加する Participant 台数

D:送信処理時間（秒）

例）

目標 40TPS、トランザクションに参加する Participant2 台、送信処理時間 0.05 秒とした場合、必要なスレッド数は 4 となります。

X ＝ 40 × 2 × 0.05

X ＝4

■Mediator の送信処理に使用するキュー

複数のトランザクションを同時に実行する際、極力スレッドプールのキューの枯渇が発生しないようにするためには十分なキューサイズが必要です。

目標 TPS に応じて、キューサイズを算出します。

TCC-Participant1 台当たりの目標 TPS を基準にした算出方法を次に示します。

B の係数には実稼働を通して環境に合わせた数値を設定してください。

Y ＝ A × B

Y:キュー数

A:目標 TPS

B:係数

例）

目標 40TPS、係数を 0.25 とした場合、必要なキューサイズは 10 となります。

Y ＝ 40 × 0.25

Y ＝ 10

(b)　チューニング対象のパラメター一覧

表 3-31　Mediator のチューニング対象のパラメター一覧

項番	パラメタ	説明
1	grpc.server.threadPoolSize	Participant、Orchestrator からの受信に使用するスレッド数を指定します。
2	grpc.server.queueSize	Participant、Orchestrator からの受信に使用するスレッドの割り当てを待つキューのサイズを指定します。

項番	パラメタ	説明
3	mediator.judge.threadPoolSize	分散合意処理に使用するスレッド数を指定します。
4	mediator.judge.queueSize	分散合意処理に使用するスレッドの割り当てを待つキューのサイズを指定します。
5	consensusLog.io.threadPoolSize	ConsensusLog ファイルへの書き込みに使用するスレッド数を指定します。
6	consensusLog.io.queueSize	ConsensusLog ファイルへの書き込みに使用するスレッドの割り当てを待つキューのサイズを指定します。
7	participant.sender.threadPoolSize	Participant への送信に使用するスレッド数を指定します。
8	participant.sender.queueSize	Participant への送信に使用するスレッドの割り当てを待つキューのサイズを指定します。

各パラメタの詳細については、「8.3.1　Mediator」を参照してください。

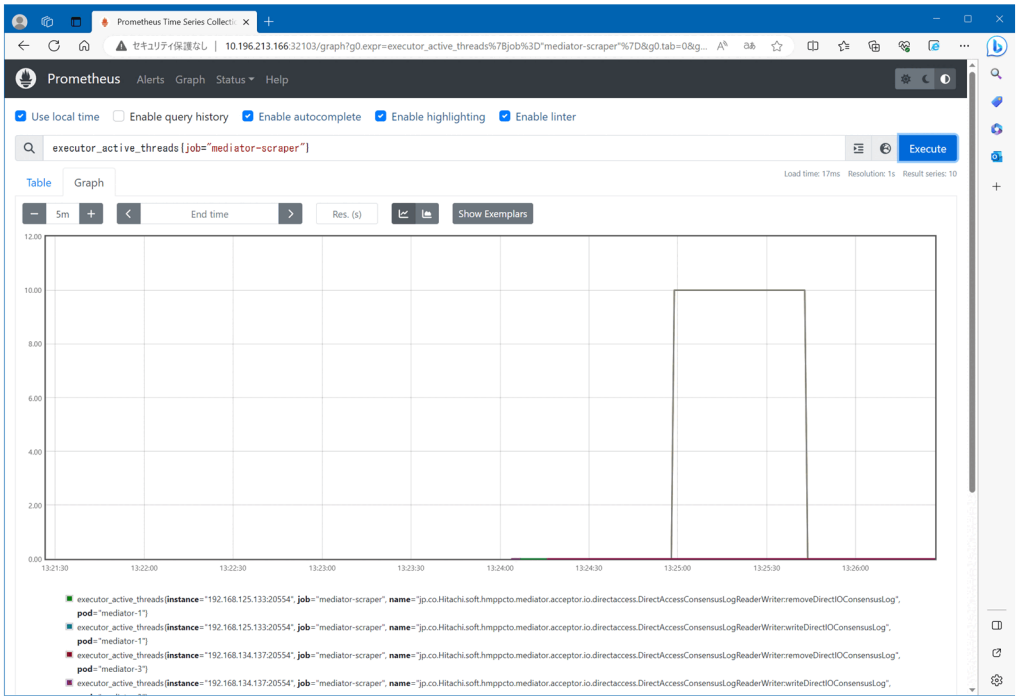
(c)　チューニング方法

consensusLog.io.threadPoolSize

このパラメタに関する情報はメトリクスデータによって確認できます。メトリクスデータの確認方法の詳細については、「10.5.3　メトリクス」を参照してください。

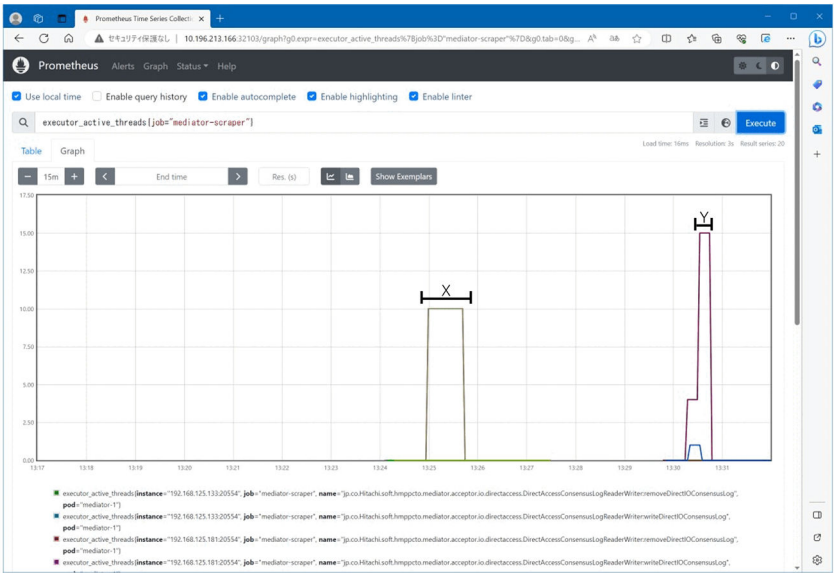
Prometheus の Web UI で「executor_active_threads{job="mediator-scraper"}」のクエリを入力し、Execute ボタンを押すことで Graph タブに下図のような状態で表示されます。

図 3-15　Prometheus の Web UI



長い期間このパラメタで指定した値で推移している場合、スレッド不足によって ConsensusLog の書き込み待ちが発生しているおそれがあります。このパラメタの値を増やすことで、下図のように処理時間を改善できる場合があります。図中の X は改善前の処理時間、図中の Y は改善後の処理時間です。

図 3–16　Prometheus の Web UI



また、分散トレースデータでもスレッドの空き待ち状態になっているかどうか確認できます。Jaeger の Web UI で確認したいトレース情報の Trace Page を表示し、次のオペレーション名の span の例を表示します。

- grpc.server.threadPoolSize
  オペレーション名「hmppcto.mediator.TerminationReceiverService/commit」
- mediator.judge.threadPoolSize
  オペレーション名
  「j.c.H.s.h.m.j.TerminationJudgeImpl$AsyncTerminationRequestTask:runImpl」
- consensusLog.io.threadPoolSize
  オペレーション名
  「j.c.H.s.h.m.a.i.d.DirectAccessConsensusLogReaderWriter:writeDirectIOConsensusLog」
- participant.sender.threadPoolSize
  オペレーション名「hmppcto.learner.ConsensusJudgeReceiverService/receive」

上記 span の Duration が大きい場合、Mediator のパラメタで指定するスレッドが不足してスレッドの空き待ち状態になっているおそれがあります。

分散トレースの確認方法の詳細については、「10.5.2　分散トレース」を参照してください。

(6)　TP1-Bridge

TP1-Bridge でチューニングが必要なスレッドは、次に示す 2 つです。

- HMP-PCTO が使用する Spring Boot の組み込み http サーバの Tomcat が制御するスレッド
- HMP-PCTO が制御するスレッド

それぞれの設計指針、およびチューニング方法を説明します。

(a) Spring Boot の組み込み http サーバの Tomcat が制御するスレッド

TP1-Bridge が AP から受け付けたリクエストは、Spring Boot の組み込み http サーバが制御するワーカースレッドで実行されます。そのため、ワーカースレッドのスレッド数がチューニングの対象となります。TP1-Bridge がワーカースレッドを占有する期間は、AP からのリクエストを受け付けた TP1-Bridge のサービスが開始してから、AP に応答を送信して終了するまでです。これには、TP1-Converter でのデータ変換処理の時間および OpenTP1 の SPP のサービス実行時間が含まれます。

Spring Boot の組み込み http サーバの Tomcat がワーカースレッドを制御する仕組みについては、Tomcat のドキュメントを参照してください。

■設計指針

複数のトランザクションを同時に実行する際、スレッドの空き待ちによるタスクの実行遅延が極力発生しないようにするためには、トランザクションの同時実行数＋ 3 のスレッドが必要です。トランザクション外で実行するメトリクスの収集処理などが使用するスレッドを考慮して、3 を加算しています。

同一の TP1-Bridge を使用する Orchestrator/Entity-Service が複数ある場合は、それぞれのトランザクションの同時実行数の合計数分のスレッドが必要です。

スレッドの占有期間に TP1-Bridge のサービスの処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために、安全率を掛けることを推奨します。なお、算出したスレッド数が 10 未満となる場合は、10 以上の値を設定してください。

目標 TPS を基準にした算出方法を次に示します。

$X = (A \times B) + 3$

X：スレッド数

A：目標 TPS

B：TP1-Bridge のサービスの処理時間（秒）（データ変換および OpenTP1 の SPP の処理時間を含みます）

例）

目標 40TPS、TP1-Bridge のサービスの処理時間 0.2 秒とした場合、必要なスレッド数は 11 となります。

$X = (40 \times 0.2) + 3$

$X = 11$

安全率を 1.5 とした場合、必要なスレッド数は 17 となります。

$X = 11 \times 1.5$

$X \fallingdotseq 17$

■チューニング対象のパラメタ一覧

チューニング対象のパラメタは、Spring Boot のパラメタです。パラメタの詳細については、Spring Boot のドキュメントを参照してください。

表 3-32　Spring Boot が制御するスレッドのチューニング対象のパラメタ一覧

項番	パラメタ	説明
1	server.tomcat.threads.max	TP1-Bridge が AP から受け付けたリクエストの処理を実行するワーカースレッドの最大数を設定します。

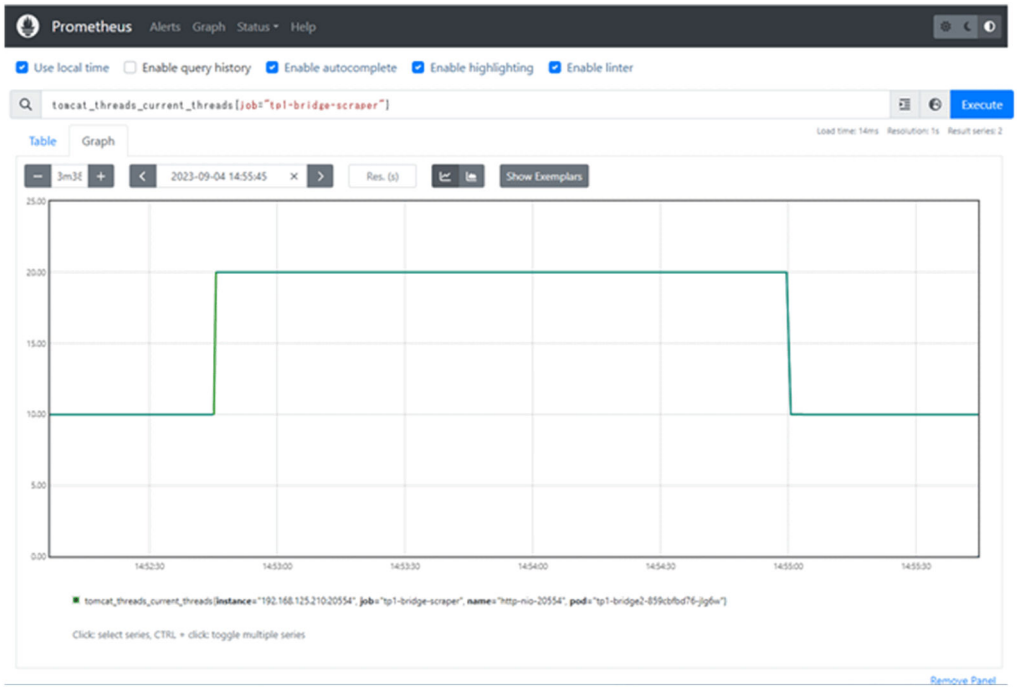
■チューニング方法

server.tomcat.threads.max

このパラメタに関する情報はメトリクスデータによって確認できます。メトリクスデータの確認方法の詳細については、「10.5.3　メトリクス」を参照してください。

Prometheus の Web UI で「tomcat_threads_current_threads{job="tp1-bridge-scraper"}」のクエリを入力し、Execute ボタンを押すことで Graph タブに次の図のような状態で表示されます。

図 3-17　Prometheus の Web UI



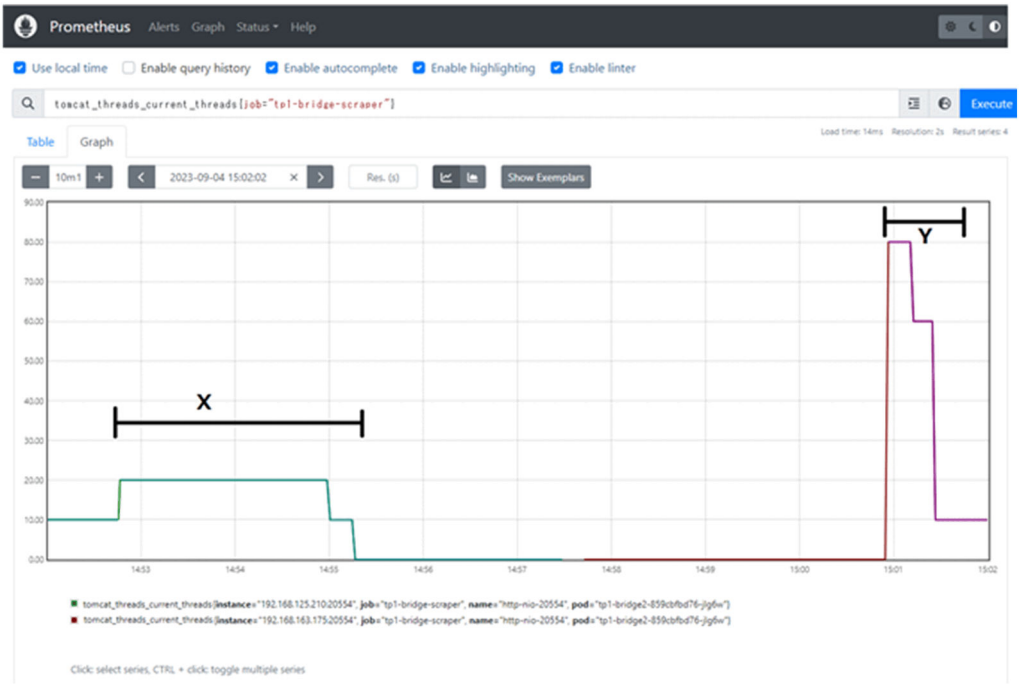
なお、1 つの Prometheus で複数の TP1-Brideg のメトリクスを収集している場合は、クエリに「instance="TP1-Bridge の IP アドレス:TP1-Bridge の http ポート番号"」を追加してください。

入力例：

```
tomcat_threads_current_threads{job="tp1-bridge-scraper",
instance="192.168.125.166:20554"}
```

長い期間このパラメタで指定した値で推移している場合、スレッド不足によって OpenTP1 のサービスの呼び出し処理の実行待ちが発生しているおそれがあります。このパラメタの値を増やすことで、次の図のように処理時間を改善できる場合があります。図中の X は改善前の処理時間、図中の Y は改善後の処理時間です。

図 3–18　Prometheus の Web UI



(b)　HMP-PCTO が制御するスレッド

HMP-PCTO が制御するスレッドは、次に示す 3 つのスレッドプールを使用して制御します。

- トランザクション処理に使用するスレッドプール
- Mediator へ送信を行うスレッドの管理に使用するスレッドプール
- gRPC 通信制御に使用するスレッドプール

それぞれのスレッドプールの役割およびスレッドプールの占有期間は、制御に使用するパラメタ名が異なるだけで、SQL-Participant が制御するスレッドプールと同じです。スレッドプールの役割およびスレッドプールの占有期間の詳細については、「(3)　SQL-Participant」を参照してください。

なお、「(3)　SQL-Participant」で説明している SQL の実行に相当する TP1-Bridge での処理は、OpenTP1 に対する RPC の実行です。RPC の実行は、Spring Boot が制御するスレッドプールで行います。そのため、RPC の実行ではトランザクション処理に使用するスレッドプールは使用しません。

■設計指針

SQL-Participant の設計指針と同じです。「(3)　SQL-Participant」を参照してください。

■チューニング対象のパラメタ一覧

表 3-33　TP1-Bridge のチューニング対象のパラメタ一覧

項番	パラメタ	説明
1	tp1.participant-processing.thread-pool-size	TP1-Bridge がトランザクション処理で実行するスレッド数を指定します。
2	tp1.participant-processing.queue-size	TP1-Bridge がトランザクション処理に使用するスレッドの割り当て待ちをするキューのサイズを指定します。
3	mediator-process-management.thread-pool-size	TP1-Bridge が Mediator へ送信を行うスレッドを管理するスレッド数を指定します。
4	mediator-communication.thread-pool-size	TP1-Bridge が Mediator と送受信を行うスレッド数を指定します。
5	mediator-communication.queue-size	TP1-Bridge が Mediator との送受信に使用するスレッドの割り当て待ちをするキューのサイズを指定します。

各パラメタの詳細については、「8.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）」を参照してください。

■チューニング方法

各パラメタに関する情報はメトリクスデータによって確認できます。メトリクスデータの確認方法については、「10.5.3　メトリクス」を参照してください。

Prometheus の Web UI で次に示す表のクエリを入力し、Execute ボタンを押すことで Graph タブに下図のような状態で表示されます。

チューニング対象のパラメタと、下図に表示されたメトリクス名の関連づけは、「表 3-35　チューニング対象のパラメタとメトリクス名の関連づけ」を参照してください。メトリクス名は、クエリを実行した結果によって「name="メトリクス名"」と表示されるため、該当する表示個所からメトリクス名を確認してください。

表 3-34　Prometheus で利用するクエリ

チューニング対象のパラメタ	クエリ
tp1.participant-processing.thread-pool-size	executor_active_threads{job="tp1-bridge-scraper"}
mediator-process-management.thread-pool-size	
mediator-communication.thread-pool-size	
tp1.participant-processing.queue-size	executor_queued_tasks{job=" tp1-bridge-scraper"}
mediator-communication.queue-size	

表 3-35　チューニング対象のパラメタとメトリクス名の関連づけ

チューニング対象のパラメタ	メトリクス名
tp1.participant-processing.thread-pool-size	jp.co.Hitachi.soft.hmppcto.adapter.TP1BridgeProposalEventObserver:observe
	jp.co.Hitachi.soft.hmppcto.txpaxosmanager.osc.OscProposalController#terminate:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.ConsensusJudgeObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionNotify:sendTransactionNotify
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminationProposalObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionVerifyNotifyObserverImpl:observe
tp1.participant-processing.queue-size	jp.co.Hitachi.soft.hmppcto.adapter.TP1BridgeProposalEventObserver:observe
	jp.co.Hitachi.soft.hmppcto.txpaxosmanager.osc.OscProposalController#terminate:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.ConsensusJudgeObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionNotify:sendTransactionNotify
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminationProposalObserverImpl:observe
	jp.co.Hitachi.soft.hmppcto.participant.common.osc.TerminatedTransactionVerifyNotifyObserverImpl:observe

図 3-19　Prometheus の Web UI

executor_active_threads{job="tp1-bridge-scraper"}の結果が、長い期間 tp1.participant-processing.thread-pool-size、mediator-process-management.thread-pool-size、または mediator-communication.thread-pool-size で指定した値で推移している場合、スレッド不足によって処理の遅延が発生しているおそれがあります。これらのパラメタの値を増やすことで、次の図のように処理時間を改善できる場合があります。図中の X は改善前の処理時間、図中の Y は改善後の処理時間です。

図 3-20　Prometheus の Web UI

また、executor_queued_tasks{job="tp1-bridge-scraper"}の結果が、tp1.participant-processing.queue-size や mediator-communication.queue-size で指定した値で推移している場合、キューサイズ不足が発生しているおそれがあります。これらのパラメタの値を増やすことでキューサイズ不足を防ぐことができます。

なお、分散トレースデータでも、スレッドの空き待ち状態になっているかどうかを確認できます。Jaeger の Web UI で確認したいトレース情報の Trace Page を表示し、次のオペレーション名の span を表示します。

- tp1.participant-processing.thread-pool-size
  オペレーション名「j.c.H.s.h.p.s.GrpcTerminationJudgeRequestSender」
  （Tags が「internal.span.format」と「status.code」だけの span）
  オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」
  （Tags が「internal.span.format」と「status.code」だけの span）
  オペレーション名「j.c.H.s.h.p.c.o.TerminatedTransactionNotify:sendTransactionNotify」
  （Tags が「internal.span.format」と「status.code」だけの span）
- mediator-process-management.thread-pool-size
  オペレーション名「j.c.H.s.h.p.s.GrpcTerminationJudgeRequestSender」
  （Tags に「branch-xid」などが含まれる span）
  オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」
  （Tags に「consensus-value」などが含まれる span）
  オペレーション名「j.c.H.s.h.p.c.o.TerminatedTransactionNotify:sendTransactionNotify」
  （Tags に「host-name」などが含まれる span）
- mediator-communication.thread-pool-size
  オペレーション名「hmppcto.mediator.TerminationReceiverService/propose」
  オペレーション名「hmppcto.mediator.TerminationReceiverService/notifyTerminationResult」
  オペレーション名「hmppcto.mediator.TerminationReceiverService/
  notifyTerminatedTransaction」

上記 span の Duration が大きい場合、TP1-Bridge のパラメタで指定したスレッドの不足によるスレッドの空き待ち状態になっているおそれがあります。

分散トレースの確認方法の詳細については、「10.5.2　分散トレース」を参照してください。

(7)　TP1-Converter

TP1-Converter のデータ変換、キャッシュのリフレッシュ、およびキャッシュの参照の各処理は、uCosminexus Service Platform の J2EE サーバ上の Web コンテナリクエスト処理スレッド（NIO HTTP サーバのリクエスト処理スレッド/ワーカースレッド）で動作します。そのため、リクエスト処理スレッドの最大数は、uCosminexus Application Server の usrconf.properties（J2EE サーバ用ユーザプロパティファイル）のパラメタで設定します。

(a) 設計指針

複数のトランザクションを同時に実行する際、極力スレッドの空き待ちが発生しないようにするためにはトランザクションの同時実行数分＋ 1 のスレッドが必要です。トランザクション外で実行するキャッシュのリフレッシュ、またはキャッシュの参照で使用するスレッドを考慮して 1 を加算しています。

ただし、データ変換処理が重ならないかぎりは、トランザクションの同時実行数分のスレッドは必要ありません。

スレッドの占有期間に相当する処理時間を設定して、目標とする 1 秒間当たりのトランザクション処理数（TPS）を処理するためのスレッド数を算出します。

算出したスレッド数に余裕を持たせるために安全率を掛けることを推奨します。

なお、算出したスレッド数が 10 未満となる場合は、10 以上の値を設定してください。

目標 TPS を基準にした算出方法を次に示します。

X ＝ (A × B) ＋ 1

X：スレッド数

A：目標 TPS

B：データ変換の処理時間（秒）

例）

目標 40TPS、データ変換の処理時間を 0.1 秒とした場合、必要なスレッド数は 5 となります。

X ＝ (40 × 0.1) ＋ 1

X ＝ 5

安全率を 1.5 とした場合、必要なスレッド数は 8 となります。

X ＝ 5 × 1.5

X ≒ 8

10 未満のため、10 を設定します。

(b) チューニング対象のパラメタ一覧

チューニング対象のパラメタは、uCosminexus Service Platform の usrconf.properties（J2EE サーバ用ユーザプロパティファイル）のパラメタです。パラメタの詳細は、マニュアル「Cosminexus アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

表 3-36　TP1-Converter のチューニング対象のパラメタ一覧

項番	パラメタ	説明
1	webserver.connector.nio_http.max_threads	TP1-Bridge のデータ変換機能を実行するリクエスト処理スレッドの最大数を設定します。

項番	パラメタ	説明
2	webserver.connector.nio_http.max_servlet_execute_threads	webserver.connector.nio_http.max_threads と同じ値を設定してください。 webserver.connector.nio_http.max_threads のうち、Web コンテナが同期サーブレットを実行するスレッドの最大数を指定します。 なお、TP1-Converter は、同期サーブレットとして実装しています。

(c) チューニング方法

チューニングに必要な情報は、uCosminexus Service Platform が Web コンテナの稼働情報ファイルに出力されます。

Web コンテナの稼働情報ファイルに出力される NIO HTTP サーバの実行中スレッド数の前回の稼働情報ファイルの出力時からの最大値（WorkerThreadCount.HighWaterMark）や現在値（WorkerThreadCount.Current）が、長時間パラメタ設定値で推移している場合は、データ変換処理がスレッド不足によって実行できないため、トランザクションの実行時間が長くなっている、または TPS が低下していることが考えられます。パラメタの設定値を増やすことで、トランザクションの実行時間や TPS が改善できる場合があります。

Web コンテナの稼働情報ファイルの詳細については、マニュアル「Cosminexus アプリケーションサーバ 機能解説 運用／監視／連携編」を参照してください。

3.10.2 タイマ

トランザクション制御の中で使用するタイマのチューニングパラメタについては次の表の記載のとおりです。

メモ

TP1-Bridge の場合は、次の表中の SQL-Participant を TP1-Bridge に、jp.co.Hitachi.soft.hmppcto.sql を jp.co.Hitachi.soft.hmppcto.tp1 に読み替えてください。

表 3-37 チューニング対象のパラメター一覧

項番	パラメタ設定の主体	パラメタ名	説明
1	Orchestrator	jp.co.Hitachi.soft.hmppcto.grpc.watchtime	＜パラメタの説明＞ Mediator への要求電文に対する応答電文の待ち時間を秒単位で指定します。 このタイマは Orchestrator から Mediator へトランザクション決着指示の要求電文を送信してから、トランザクション合意結果を受け付けるまでの間を監視します。

項番	パラメタ設定の主体	パラメタ名	説明
			このタイマがタイムアウトした時点で Mediator から 1 つも応答がない場合は、トランザクションの決着結果は不明として AP に応答し通信エラーとなります。 パラメタの詳細については、「8.2.2　Orchestrator」を参照してください。 ＜設定方法＞ トランザクション決着結果が確定する前に、このタイマがタイムアウトするのを防ぐためトランザクション決着監視タイマより大きな値を設定してください。その上でユーザ AP の業務要件に従い 1 つのトランザクションで許容できる時間を設定してください。 タイマの値が長いと障害で Mediator がダウンしていた場合に、監視時間分待ち合わせてしまいスレッドを占有してしまいます。 タイマの値が短いと障害によって分散合意ができない場合の遅延を許容できる時間が短くなります。 このタイマは、次のタイマより大きい値を設定してください。 • SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.timeout-in-millis • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.tcc.trn-termination.timeout-in-millis
2	Entity-Service	jp.co.Hitachi.soft.hmppcto.grpc.watchtime	＜パラメタの説明＞ Participant への要求電文に対する応答電文の待ち時間を秒単位で指定します。 このタイマは Entity-Service から Participant への要求電文を送信してから、実行結果を受け付けるまでの間を監視します。対象となるのは次のとおりです。 • Entity-Service から Participant へ SQL 実行を行う場合 • Entity-Service から Participant へ先行 Prepare 要求を行う場合 このタイマがタイムアウトした場合は SQL-Participant の障害と判断し、トランザクションブランチをロールバックします。 パラメタの詳細については、「8.2.3　Entity-Service/Entity-Module」を参照してください。 ＜設定方法＞ このタイマはユーザ AP の業務要件に従い 1 つのトランザクションで許容できる時間を設定してください。 タイマの値が長いと SQL-Participant が SQL 実行を行い応答するまでの遅延を許容できますが、障害が発生した場合にトランザクションブランチをロールバックするまでに時間が掛かります。 タイマの値が短いと障害が起きた際に速やかにトランザクションブランチをロールバックできますが、遅延を障害と判断しロールバックします。 このタイマは、次のタイマより大きい値を設定してください。

項番	パラメタ設定の主体	パラメタ名	説明
			• SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.sql.pool.allocation-wait-time-in-millis
3	SQL-Participant	jp.co.Hitachi.soft.hmppcto.external-consensus.interval-in-millis	＜パラメタの説明＞ 補助合意によってトランザクションの分散合意を行う場合の、補助合意を試行する間隔をミリ秒単位で指定します。 このタイマは Mediator から 1 つ目の合意を受信してから分散合意可能な数の Mediator から合意を受信するまでを監視します。 このタイマがタイムアウトした場合は次の動作を行います。 • Mediator から高速合意できるだけの合意を受信していれば、高速合意を実行します。 • Mediator から高速合意できるだけの合意を受信しておらず、補助合意できるだけの合意を受信していれば、補助合意を実行します。 なお、このタイマは合意できるまで無限にリトライします。 パラメタの詳細については、「8.2.4　SQL-Participant」を参照してください。 ＜設定方法＞ 基本はデフォルト値を設定した上で、補助合意が頻繁に発生するようであれば、長めに調整してください。 タイマの値が長いとタイムアウト時に補助合意できる Mediator がそろう可能性が高まりますが、補助合意実行までの間隔が伸び遅延が大きくなります。 タイマの値が短いと Mediator などに障害が発生した場合のトランザクションの遅延を小さくできますが、通信遅延を障害と誤認し、不要な補助合意が発生します。 このタイマは、次のタイマより小さい値を設定してください。 • SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.timeout-in-millis • SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.consensus.replay-interval-in-millis
4		jp.co.Hitachi.soft.hmppcto.grpc.watchtime	＜パラメタの説明＞ Mediator への要求電文に対する応答電文の待ち時間を秒単位で指定します。 このタイマは Participant から Mediator に要求電文を送信してから、Mediator から要求を受け付けた旨の応答電文を受信するまでの間を監視します。対象となるのは次のとおりです。 • Participant から Mediator へ投票を行う場合 • Participant から Mediator へトランザクション合意結果を通知する場合 • Participant から Mediator へトランザクション完了を通知する場合

項番	パラメタ設定の主体	パラメタ名	説明
			このタイマがタイムアウトした場合は通信エラーとなります。投票が通信エラーとなった場合は、トランザクションの決着が行えずにRollbackとなるおそれがあります。トランザクション合意結果通知および、トランザクション完了通知が通信エラーとなった場合はOrchestratorへの合意結果の通知やログの削除が行えないおそれがあります。 パラメタの詳細については、「8.2.4　SQL-Participant」を参照してください。 ＜設定方法＞ 基本はデフォルト値を設定した上で、Mediatorとの通信環境によってgRPC通信エラーが頻繁に発生するようであれば、長めに調整してください。 タイマの値が長いとMediatorがダウンしたときにスレッドが占有されて他Mediatorへの送信が遅れるおそれがあります。 タイマの値が短いとMediatorなどに障害が発生した場合のトランザクションの遅延を小さくできますが、通信遅延を障害と誤認します。 このタイマは、次のタイマより小さい値を設定してください。 • SQL-Participantのパラメタ jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.timeout-in-millis
5		jp.co.Hitachi.soft.hmppcto.sql.pool.allocation-wait-time-in-millis	＜パラメタの説明＞ SQL-ParticipantがDBアクセスのためのコネクションをコネクションプールから取得する際に、コネクションプールに空きがなかった場合の空き待ち時間をミリ秒単位で指定します。 このタイマはSQL-ParticipantからDBへのコネクション接続要求を送信し、接続確立の応答が帰るまでの間を監視します。 このタイマがタイムアウトした場合はエラーとなり、トランザクションブランチをロールバックします。 パラメタの詳細については、「8.2.4　SQL-Participant」を参照してください。 ＜設定方法＞ 基本はデフォルト値を設定した上で、エラーが頻繁に発生するようであれば、長めに調整してください。 タイマの値が長いとDBとのコネクションが確保できない場合に許容できる時間が増えますが、遅延が大きくなります。 タイマの値が短いと遅延は短縮されますが、許容できる時間が短くなり、エラーリターンします。 このタイマは、次のタイマより小さい値を設定してください。 • Entity-Serviceのパラメタ jp.co.Hitachi.soft.hmppcto.grpc.watchtime
6		jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.timeout-in-millis	＜パラメタの説明＞ トランザクション決着監視タイマのタイマ監視時間をミリ秒単位で指定します。DBの排他ロックが解けない状態となることを許容で

項番	パラメタ設定の主体	パラメタ名	説明
			きる時間をタイマ監視し、設定した時間を超えてもトランザクションが決着しなかった場合に、DB の排他ロック解除のためにトランザクションを決着に向かわせるタイマです。 このタイマは、Participant から DB へ xa_start を行ってから xa_commit または xa_rollback を行うまでの間を監視します。 このタイマがタイムアウトした場合は Participant から Mediator に対して Rollback 投票を行います。 パラメタの詳細については、「8.2.4　SQL-Participant」を参照してください。 ＜設定方法＞ このタイマはユーザ AP の業務要件に従い 1 つのトランザクションで許容できる時間を設定してください。 タイマの値が長いと通信遅延などによって合意が滞っている場合でも Commit 決着できる可能性が高まります。しかし、障害が発生した場合、Rollback するまで、タイマの指定値分の時間が掛かります。 タイマの値が短いと障害が起きた際に速やかに Rollback 決着を行うことができますが、遅延を障害と判断し Rollback 決着します。 このタイマは、次のタイマより大きい値を設定してください。 • SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.external-consensus.interval-in-millis • SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.grpc.watchtime
7		jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.consensus.replay-interval-in-millis	＜パラメタの説明＞ トランザクション分散合意リプレイインターバルタイマのタイマ監視時間をミリ秒単位で指定します。このタイマは、トランザクションをコミットまたはロールバックするためにトランザクション決着処理をする際、トランザクション決着監視タイマのタイムアウトによるトランザクション決着処理をする際、または SQL-Participant ダウンからの再起動時にトランザクション決着処理をする際に設定します。設定した時間を超えてもトランザクションが決着しなかった場合は、トランザクション決着処理を再実行（リプレイ）し、DB の排他ロック解除のためにトランザクションを決着に向かわせます。 このタイマは、Participant から Mediator へ投票を行ってから DB へ xa_commit または xa_rollback を行うまでの間を監視します。 このタイマがタイムアウトした場合は Participant から Mediator に対して Rollback 投票を行います。 パラメタの詳細については、「8.2.4　SQL-Participant」を参照してください。 ＜設定方法＞ このタイマはユーザ AP の業務要件に従い 1 つのトランザクションで許容できる時間を設定してください。

項番	パラメタ設定の主体	パラメタ名	説明
			タイマの値が長いと通信遅延などによって合意が滞っている場合でも Commit 決着できる可能性が高まります。しかし、障害が発生した場合、Rollback するまで、タイマの指定値分の時間が掛かります。 タイマの値が短いと障害が起きた際に速やかに Rollback 決着を行うことができますが、障害が長引いた場合に何度もリトライを行うため通信負荷が掛かります。 このタイマは、次のタイマより大きい値を設定してください。 • SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.external-consensus.interval-in-millis • SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.grpc.watchtime
8	TCC-Participant	jp.co.Hitachi.soft.hmppcto.external-consensus.interval-in-millis	＜パラメタの説明＞ 補助合意によってトランザクションの分散合意を行う場合の、補助合意を試行する間隔をミリ秒単位で指定します。 このタイマは Mediator から 1 つ目の合意を受信してから分散合意可能な数の Mediator から合意を受信するまでを監視します。 このタイマがタイムアウトした場合は次の動作を行います。 • Mediator から高速合意できるだけの合意を受信していれば、高速合意を実行します。 • Mediator から高速合意できるだけの合意を受信しておらず、補助合意できるだけの合意を受信していれば、補助合意を実行します。 なお、このタイマは合意できるまで無限にリトライします。 パラメタの詳細については、「8.2.5　TCC-Participant」を参照してください。 ＜設定方法＞ 基本はデフォルト値を設定した上で、補助合意が頻繁に発生するようであれば、長めに調整してください。 タイマの値が長いとタイムアウト時に補助合意できる Mediator がそろう可能性が高まりますが、補助合意実行までの間隔が伸び遅延が大きくなります。 タイマの値が短いと Mediator などに障害が発生した場合のトランザクションの遅延を小さくできますが、通信遅延を障害と誤認し、不要な補助合意が発生します。 このタイマは、次のタイマより小さい値を設定してください。 • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.tcc.trn-termination.timeout-in-millis • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.tcc.consensus.interval-in-millis
9		jp.co.Hitachi.soft.hmppcto.grpc.watchtime	＜パラメタの説明＞

項番	パラメタ設定の主体	パラメタ名	説明
			Mediator への要求電文に対する応答電文の待ち時間を秒単位で指定します。 このタイマは Participant から Mediator に要求電文を送信してから、Mediator から要求を受け付けた旨の応答電文を受信するまでの間を監視します。対象となるのは次のとおりです。 • Participant から Mediator へ投票を行う場合 • Participant から Mediator へトランザクション合意結果を通知する場合 • Participant から Mediator へトランザクション完了を通知する場合 このタイマがタイムアウトした場合は通信エラーとなります。 投票が通信エラーとなった場合は、トランザクションの決着が行えずに Rollback となるおそれがあります。 トランザクション合意結果通知および、トランザクション完了通知が通信エラーとなった場合は Orchestrator への合意結果の通知やログの削除が行えない可能性があります。 パラメタの詳細については、「8.2.5　TCC-Participant」を参照してください。 ＜設定方法＞ 基本はデフォルト値を設定した上で、Mediator との通信環境によって gRPC 通信エラーが頻繁に発生するようであれば、長めに調整してください。 タイマの値が長いと Mediator がダウンしたときにスレッドが占有されて他 Mediator への送信が遅れるおそれがあります。 タイマの値が短いと Mediator などに障害が発生した場合のトランザクションの遅延を小さくできますが、通信遅延を障害と誤認します。 このタイマは、次のタイマより小さい値を設定してください。 • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.tcc.trn-termination.timeout-in-millis
10		jp.co.Hitachi.soft.hmppcto.tcc.trn-termination.timeout-in-millis	＜パラメタの説明＞ トランザクション決着監視タイマのタイマ監視時間をミリ秒単位で指定します。障害によってトランザクション決着に至らない状態となることを許容できる時間をタイマ監視し、設定時間を超えてもトランザクションが決着しなかった場合に、ユーザが許容できる時間（業務要件）を超過したと判断しトランザクションを決着に向かわせるタイマです。 このタイマは、API アダプタ機能から業務ロジックへ Try を実行する直前から分散合意結果に従って Try 実行に対する Confirm/Cancel の実行が完了するまでの区間をタイマ監視します。 このタイマがタイムアウトした場合は Participant から Mediator に対して Rollback 投票を行います。 パラメタの詳細については、「8.2.5　TCC-Participant」を参照してください。

項番	パラメタ設定の主体	パラメタ名	説明
			＜設定方法＞ このタイマはユーザ AP の業務要件に従い 1 つのトランザクションで許容できる時間を設定してください。 タイマの値が長いと通信遅延などによって合意が滞っている場合でも Commit 決着できる可能性が高まります。しかし、障害が発生した場合に Rollback するまで、タイマの指定値分の時間が掛かります。 タイマの値が短いと障害が起きた際に速やかに Rollback 決着を行うことができますが、遅延を障害と判断し Rollback 決着します。 このタイマは、次のタイマより大きい値を設定してください。 • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.external-consensus.interval-in-millis • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.grpc.watchtime
11		jp.co.Hitachi.soft.hmppcto.tcc.consensus.interval-in-millis	＜パラメタの説明＞ トランザクション分散合意処理をリトライする間隔をミリ秒単位で指定します。障害などによってトランザクションの分散合意ができないとき、この間隔によってトランザクションの分散合意が決着するまで分散合意処理をリトライし、分散合意によるトランザクションの決着が必ず行われるようにするためのリトライ間隔です。 このタイマは、Participant から Mediator へ投票を行ってから分散合意が決着するまでの区間をタイマ監視します。 このタイマがタイムアウトした場合は Participant から Mediator に対して Rollback 投票を行います。 パラメタの詳細については、「8.2.5　TCC-Participant」を参照してください。 ＜設定方法＞ このタイマはユーザ AP の業務要件に従い 1 つのトランザクションで許容できる時間を設定してください。 タイマの値が長いと通信遅延などによって合意が滞っている場合でも Commit 決着できる可能性が高まります。しかし、障害が発生した場合に Rollback するまで、タイマの指定値分の時間が掛かります。 タイマの値が短いと障害が起きた際に速やかに Rollback 決着を行うことができますが、障害が長引いた場合に何度もリトライを行うため通信負荷が掛かります。 このタイマは、次のタイマより大きい値を設定してください。 • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.external-consensus.interval-in-millis • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.grpc.watchtime
12	Mediator	consensusLog.io.watchInSeconds	＜パラメタの説明＞

項番	パラメタ設定の主体	パラメタ名	説明
			ConsensusLog ファイルへの書き込みおよび読み出し時の IO 監視時間を秒で指定します。 このタイマは、トランザクション決着指示を受けた Mediator が ConsensusLog へ永続化結果を書き込んでから完了するまでの間を監視します。 このタイマがタイムアウトした場合は、障害メッセージを出力し、該当する Mediator は故障と判断し分散合意の対象外となります。 パラメタの詳細については、「8.3.1　Mediator」を参照してください。 ＜設定方法＞ 基本はデフォルト値を設定した上で、エラーが頻繁に発生するようであれば、長めに調整してください。 タイマの値が長いとハードウェアやドライバが障害検知した場合でも、タイマの指定値分まで監視を続けてしまいます。 タイマの値が短いと IO ハングアップした場合に早期に検知できますが、監視時間中に容量が空き、復旧できる見込みがあった場合でもエラーとなります。
13		grpc.watchtime	＜パラメタの説明＞ Participant への要求電文に対する応答電文の待ち時間を秒単位で指定します。 このタイマは Mediator から Participant に要求電文を送信してから、Participant から要求を受け付けた旨の応答電文を受信するまでの間を監視します。対象となるのは次のとおりです。 • Mediator から Participant への投票要求を行う場合 • Mediator から Participant へトランザクション決着結果通知を行う場合 • Mediator から Participant へトランザクション完了確認通知を行う場合 このタイマがタイムアウトすると、通信エラーとなります。投票要求および、トランザクション決着結果通知が通信エラーとなった場合は、トランザクションが決着できずに Rollback となるおそれがあります。トランザクション完了確認通知が通信エラーとなった場合は、ログの削除が行えないおそれがあります。 パラメタの詳細については、「8.3.1　Mediator」を参照してください。 ＜設定方法＞ 基本はデフォルト値を設定した上で、Mediator との通信環境によって gRPC 通信エラーが頻繁に発生するようであれば、長めに調整してください。 タイマの値が長いと Mediator がダウンした時にスレッドが占有されて他 Mediator への送信が遅れるおそれがあります。 タイマの値が短いと Mediator などに障害が発生した場合のトランザクションの遅延を小さくできますが、通信遅延を障害と誤認します。 このタイマは、次のタイマより小さい値を設定してください。

項番	パラメタ設定の主体	パラメタ名	説明
			• SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.timeout-in-millis • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.tcc.trn-termination.timeout-in-millis

3.10.3 G1GC

SQL-Participant、Mediator、および TP1-Bridge では、Java のメモリ管理の方式に G1GC を選択します。メモリ領域に対して YoungGC、MixedGC、および FullGC の 3 種類の GC が発生します。このうち、YoungGC と MixedGC については、GC 発生時にアプリケーションが停止する時間を制御できます。G1GC を選択するために、-XX:+UseG1GC オプションのほか、GC 発生時の目標の停止時間、メモリに関するオプションや処理性能のチューニングに関するオプションを指定します。

(1) チューニング対象のパラメター一覧

表 3-38 G1GC チューニング対象のパラメター一覧

項番	パラメタ	説明
1	-XX:+UseG1GC	メモリ管理方式に G1GC を選択します。
2	-XX:MaxGCPauseMillis	希望する最大一時停止時間の目標値を設定します。指定された値はヒープサイズに適合しません。
3	-XX:ParallelGCThreads	STW ワーカースレッドの値を設定します。n の値を論理プロセッサの数に設定します。n の値は、値 8 までは論理プロセッサの数と同じです。 論理プロセッサが 8 つを超える場合は、n の値を論理プロセッサの約 5/8 に設定します。これは、n の値が論理プロセッサの約 5/16 になる大規模な SPARC システムを除き、ほとんどの場合に機能します。
4	-XX:ConcGCThreads	並列マーキングスレッドの数を設定します。n を並列ガベージコレクションスレッド(ParallelGCThreads)の数の約 1/4 に設定します。
5	-Xms	Java ヒープの初期サイズを設定します。※1
6	-Xmx	Java ヒープの最大サイズを設定します。※1
7	-XX:SurvivorRatio	New 領域に対する Survivor 領域で最大に取ることができる領域サイズの割合を設定します。 Survivor 領域の最大で取ることができる領域サイズは次の式で求められます。 (Survivor 領域のサイズ)※2=(New 領域のサイズ)/SurvivorRatio
8	-XX:MetaspaceSize	Metaspace 領域の初期サイズを設定します。※3

項番	パラメタ	説明
9	-XX:MaxMetaspaceSize	Metaspace 領域の最大サイズを設定します。※3
10	-XX:CompressedClassSpaceSize	圧縮オブジェクトポインタ機能が有効な場合に Metaspace 領域内に作成される Compressed Class Space 領域の最大サイズを設定します。このパラメタは、-XX:MaxMetaspaceSize オプションと同じ値の設定を推奨します。

注※1

-Xms と-Xmx の値は同じ値を指定することを推奨します。

注※2

Survivor 領域のサイズとは To 空間または From 空間のサイズを指します。To 空間と From 空間の合計サイズではないことに注意してください。

注※3

-XX:MetaspaceSize と-XX:MaxMetaspaceSize の値は同じ値を指定することを推奨します。

(2) パラメタの設定方法

(a) SQL-Participant

SQL-Participant の Kubernetes マニフェストで、次のとおり指定します。

- 環境変数 HMPPCTO_JAVA_OPTIONS で Entity-Service の Java プロセス起動時に Java VM に渡すオプションを指定します。
- 環境変数 HMPPCTO_JAVA_OPTIONS で SQL-Participant の Java プロセス起動時に Java VM に渡すオプションを指定します。

SQL-Participant の Kubernetes マニフェストの記述例は「2.6.4 SQL-Participant の Kubernetes マニフェストの作成（SQL-Participant 限定）」を参照してください。

(b) Mediator

Mediator の Helm チャートの java.options パラメタで Mediator の Java プロセス起動時に Java VM に渡すオプションを指定します。

java.options パラメタの詳細は、「(1) Helm チャートのパラメタ」を参照してください。

Mediator の Helm チャートのカスタマイズ例は、「3.7.3 Mediator の Helm チャートのカスタマイズ」を参照してください。

(c) TP1-Bridge

TP1-Bridge の Helm チャートの java.options パラメタで、TP1-Bridge の Java プロセス起動時に Java VM に渡すオプションを指定します。

java.options パラメタの詳細は、「(1) Helm チャートのパラメタ」を参照してください。

TP1-Bridge の Helm チャートのカスタマイズ例は、「3.7.4　TP1-Bridge の Helm チャートのカスタマイズ（通常版かつ TP1-Bridge 限定）」を参照してください。

3.10.4　ディスク

(1)　Mediator

「3.6.2　Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」を参照してください。

(2)　EADS（通常版限定）

「3.6.5　EADS の永続ボリュームの Kubernetes マニフェストの作成（通常版限定）」を参照してください。

3.10.5　システム運用に必要となる機能

この項では、冗長化や各コンポーネントで注意する必要がある項目について記載します。

システム運用に必要となる機能で使用する各コンポーネントの CPU、メモリに関しては Elasticsearch を除き、リソースの制限を実施していません。

そのため、実際に RAS コンポーネントを使用したとき消費されるリソース量を確認し、チューニングを実施してください。

(1)　Elasticsearch

(a)　冗長化・レプリケーション

クラスタ構成を実施することで冗長化できます。クラスタ構成を採ることで、Elasticsearch に格納したデータの分散管理、冗長性の確保を実現できます。

Elasticsearch をクラスタ構成にする場合、ノード障害からの保護や、ストレージの圧迫を抑制するため Kubernetes ノード間で分散して Elasticsearch を起動させることを推奨します。

(b)　CPU、メモリのチューニング

Elasticsearch は多量のリソースを使用するため、クラスタ構成を採らない場合での、起動と最低限の利用に必要と思われるリソース値による制限を実施しています。

Elasticsearch のパフォーマンスに応じてチューニングを実施してください。

表 3-39　Elasticsearch のリソース設定

項番	パラメタ	説明
1	requests.cpu	詳細は「8.3.10　Elasticsearch」を参照してください。
2	requests.memory	
3	limits.cpu	
4	limits.memory	

(c)　Elasticsearch データディレクトリ・Elasticsearch リポジトリディレクトリとして使用する永続ボリュームの見積もり

Elasticsearch データディレクトリおよび Elasticsearch リポジトリディレクトリとして使用する永続ボリュームの見積もりの詳細については、「3.6.3　Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」を参照してください。

(2)　Jaeger-collector

(a)　冗長化・レプリケーション

HMP-PCTO の各コンポーネントから出力される Span を分散させるため、また Pod 停止によって分散トレースとして収集する情報に欠落が生じないようにレプリケーションで冗長化を実施できます。

(b)　パラメタ

Jaeger-collector は、HMP-PCTO の各コンポーネントから送信された Span を受信し、Elasticsearch へと送信する動作を行います。受信した Span を格納する内部キューの長さと、キュー内のデータの送信処理を行うワーカに対して、チューニングを行う必要があります。

そのほかのパラメタについては、Jaeger の公式ドキュメントを参照してください。

チューニング対象のパラメタ一覧

表 3-40　チューニング対象のパラメタ一覧

項番	パラメタ	説明
1	collector.queueSize	Jaeger-collector は収集した Span を Elasticsearch に送付するため、受信したデータを内部のキューへと保管します。※ このキューのサイズを指定します。
2	collector.numWorkers	Jaeger-collector 内で動作するワーカの数を指定します。
3	collector.numShards	Elasticsearch に作成するインデックスのシャード数を指定します。 このパラメタは Elasticsearch の性能に影響します。

注※

Jaeger-collector は、内部キューがいっぱいになると古い Span を上書きします。

チューニング方法

表 3-41　チューニング方法

項番	パラメタ	チューニングの考え方	デフォルト値
1	collector.queueSize	Jaeger-collector は、内部キューがいっぱいになると古い Span を破棄します。そのため、「1 トランザクション当たりの Span 数×Transaction per seconds × 時間（分）」で計算し、見積もりを実施します。 チューニングを行う場合は、Jaeger-collector の jaeger_collector_queue_length のメトリクスを参照し、値が増長している かつ、jaeger_collector_spans_dropped_total のメトリクスを参照したとき、Span の drop が発生している場合は、設定値の見直しが必要です。	「(1)　Helm チャートのパラメタ」を参照してください。
2	collector.numWorkers	キュー内のデータ 50 件に対ワーカ 1 つが対応します。 そのため、"キューのサイズ / 50" を設定することを推奨します。	
3	collector.numShards	Elasticsearch は、インデックス内のデータをシャードという単位で分割して管理します。シャード数が多いと、Elasticsearch の検索効率は上昇しますが、消費されるメモリが多くなります。 Elasticsearch のパフォーマンスに問題がある場合にかぎり、このパラメタの数値を変更してください。	

注※

デフォルト値は、Jaeger-collector 1.45.0 におけるデフォルト値です。

(c)　CPU、メモリのチューニング

Jaeger-collector は内部キューの容量、送信処理のチューニングに応じて、必要となるリソースが増加します。Jaeger-collector のパフォーマンスに応じてチューニングを実施してください。

表 3-42　Jaeger-collector のリソース設定

項番	パラメタ	説明
1	requests.cpu	詳細は「(1)　Helm チャートのパラメタ」を参照してください。
2	requests.memory	
3	limits.cpu	
4	limits.memory	

(3) Jaeger-query

(a) パラメタ

Jaeger-query では、分散トレースを参照するための UI を提供します。チューニングとしては、分散トレース参照時の検索範囲に関して設定する必要があります。

チューニング対象のパラメター一覧

Jaeger-query のチューニング対象のパラメター一覧を次に示します。

表 3-43 チューニング対象のパラメター一覧

項番	パラメタ	説明
1	jaegerQuery.maxSpanAge	Jaeger-query が Elasticsearch から分散トレースを取得する際、検索の範囲とする時間の指定を行います。

チューニング方法

表 3-44 チューニング方法

項番	パラメタ	チューニングの考え方	デフォルト値
1	jaegerQuery.maxSpanAge	Elasticsearch ILM ポリシーと合わせた値を設定します。 ILM ポリシーで設定した削除期間内が検索可能となる範囲で設定をしてください。	「8.3.6 Jaeger-query」を参照してください。

(b) CPU、メモリのチューニング

Jaeger-query に対するアクセスが集中するときなど、必要となるリソースが増加する場合があります。Jaeger-query のパフォーマンスに応じてチューニングを実施してください。

表 3-45 Jaeger-query のリソース設定

項番	パラメタ	説明
1	requests.cpu	詳細は「(1) Helm チャートのパラメタ」を参照してください。
2	requests.memory	
3	limits.cpu	
4	limits.memory	

(4) Logstash

(a) 冗長化・レプリケーション

メッセージログが各 Kubernetes ノードに起動した Filebeat より転送されるため、アクセスを分散させます。また、Pod 停止によって収集するメッセージログ情報に欠落が生じないようにレプリケーションで冗長化を実施できます。

ログが大量に出力された際に、内部キューの圧迫を防ぐため、2 台以上の構成で起動することを推奨します。この時、起動させる Kubernetes ノードは、異なるノード上で起動させるようにすることで、可用性が向上します。

(b) パラメタ

Logstash では、Filebeat で収集したメッセージログを Elasticsearch に転送する動作を行います。受信したメッセージログを送信するためにチューニングを行う必要があります。

チューニング対象のパラメタ一覧

Logstash のチューニング対象のパラメタ一覧を次に示します。

表 3-46 チューニング対象のパラメタ一覧

項番	パラメタ	説明
1	logstash.pipeline.batchSize	個々の worker が input 処理によって収集するイベントの最大数を定義します。
2	logstash.queue.event.maxEvents	内部キュー内の未読イベントの最大数を定義します。
3	logstash.queue.byte.maxBytes	内部キューの最大容量（Byte）を定義します。

チューニング方法

表 3-47 チューニング方法

項番	パラメタ	チューニングの考え方	デフォルト値
1	logstash.pipeline.batchSize	数値を大きくするほど、処理効率は向上しますが、メモリのオーバーヘッドが増加します。Filebeat からログの送信に時間が掛かる場合は、この設定を見直してください。	「(1) Helm チャートのパラメタ」を参照してください。
2	logstash.queue.event.maxEvents	queue.max_events および queue.max_bytes が未設定の場合、メモリを無制限に使用します。queue.max_events を設定し、メモリ使用量の上昇を抑制してください。 リソース管理の観点から queue.max_bytes を設定し、永続キューの制限を設定することを推奨します。	

項番	パラメタ	チューニングの考え方	デフォルト値
3	logstash.queue.byte.maxBytes	Elasticsearch のインデックスの見積もり式 と 障害要件（Elasticsearch の復旧までの停止許容時間）を基にキューに格納するログサイズの見積もりを実施してください。 "1TPS（Transactions per seconds）当たりのログ容量×停止時間×安全係数"	

(c) CPU、メモリのチューニング

Logstash では、Filebeat から受信したメッセージログの送信処理に関するチューニングに応じて、必要となるリソースが増加します。

表 3-48 Logstash のリソース設定

項番	パラメタ	説明
1	requests.cpu	詳細は「(1) Helm チャートのパラメタ」を参照してください。
2	requests.memory	
3	limits.cpu	
4	limits.memory	

(5) Filebeat

(a) パラメタ

Filebeat ではコンテナの標準出力として出力されたログ、各 Pod 内のログファイル上に出力されたログを Logstash に転送する動作を行います。メッセージログを送信するためにチューニングを行う必要があります。

チューニング対象のパラメタ一覧

Filebeat のチューニング対象のパラメタ一覧を次に示します。

表 3-49 チューニング対象のパラメタ一覧

項番	パラメタ	説明
1	filebeat.queue.maxSize	内部キューとして使用する容量の最大値を定義します。
2	filebeat.queue.readAhead	内部キューに登録され、ディスク上に出力されたデータから、Logstash への出力のためにメモリに読み取る必要のあるイベントの数を定義します。
3	filebeat.output.maxRetries	Logstash への送信に失敗した際に実施する再送の試行回数を定義します。

チューニング方法

表 3-50　チューニング方法

項番	パラメタ	チューニングの考え方	デフォルト値
1	filebeat.queue.maxSize	• コンテナの標準出力として出力されたログの場合 Elasticsearch のインデックスの見積もり式 と 障害要件(Elasticsearch の復旧までの停止許容時間)をもとにキューに格納するログサイズの見積もりを実施してください。 "1TPS（Transactions per seconds）当たりのログ容量×停止時間×安全係数" • 各 Pod 内のログファイルを参照し、出力されたログの場合 1 トランザクション当たりのログ容量の見積もりを行います。共に障害要件(Elasticsearch の復旧までの停止許容時間)を基にキューに格納するログサイズの見積もりを実施してください。 "1TPS（Transactions per seconds）当たりのログ容量×停止時間×安全係数"	「(1)　Helm チャートのパラメタ」を参照してください。
2	filebeat.queue.readAhead	Logstash への出力が遅くなっている場合、一度に多くのイベントを読み取ることができない状況になるため、この設定値をデフォルト値より高い数値に設定します。このとき、メモリ使用量は増加します。	
3	filebeat.output.maxRetries	無制限とした場合、キューの圧迫につながります。 そのため、機能要件に従った再送回数を定義することを推奨します。	

(b)　CPU、メモリのチューニング

Filebeat では、メッセージログの送信処理に関するチューニングに応じて、必要となるリソースが増加します。

表 3-51　Filebeat のリソース設定

項番	パラメタ	説明
1	requests.cpu	詳細は「(1)　Helm チャートのパラメタ」を参照してください。
2	requests.memory	
3	limits.cpu	
4	limits.memory	

(6) Metricbeat

(a) 冗長化・レプリケーション

Prometheus から送信されてくるメトリクスを Elasticsearch に対して送信を実施します。Prometheus からのアクセスを分散させます。メトリクスの欠落を防止するなどの意図で、レプリケーションで冗長化を実施できます。

Prometheus のレプリケーションを実施した場合や、Metricbeat 停止によるメトリクス欠落を防止する必要がある場合は、2 台以上の構成で起動してください。

このとき、起動させる Kubernetes ノードは、異なるノード上で起動させるようにすることで、可用性が向上します。

(b) パラメタ

Metricbeat では、Prometheus より送信されたメトリクスを Elasticsearch へと送信する動作を行います。送信を実施するために、次の表に示すチューニングを実施してください。

チューニング対象のパラメタ一覧

Metricbeat のチューニング対象のパラメタ一覧を次に示します。

表 3-52 チューニング対象のパラメタ一覧

項番	パラメタ	説明
1	metricbeat.output.maxRetries	Elasticsearch への送信に失敗した際に実施する再送の試行回数を定義します。
2	metricbeat.queue.maxSize	内部キューとして使用する容量の最大値を定義します。
3	metricbeat.queue.readAhead	内部キューに登録され、ディスク上に出力されたデータから、Elasticsearch への出力のためにメモリに読み取る必要のあるイベントの数を定義します。

チューニング方法

表 3-53 チューニング方法

項番	パラメタ	チューニングの考え方	デフォルト値
1	metricbeat.output.maxRetries	無制限とした場合、キューの圧迫につながります。そのため、機能要件に従った再送回数を定義することを推奨します。	「(1) Helm チャートのパラメタ」を参照してください。
2	metricbeat.queue.maxSize	Elasticsearch のインデックスの見積もり式 と 障害要件(Elasticsearch の復旧までの停止許容時間)をもとにキューに格納するログサイズの見積もりを実施してください。 "1TPS（Transactions per seconds）当たりのメトリクス容量×停止時間×安全係数"	

項番	パラメタ	チューニングの考え方	デフォルト値
3	metricbeat.queue.readAhead	Elasticsearch への出力が遅くなっている場合、一度に多くのイベントを読み取ることができない状況になるため、この設定値をデフォルト値より高い数値に設定します。このとき、メモリ使用量は増加します。	

(c) CPU、メモリのチューニング

Metricbeat では、Prometheus から受信したメトリクスの送信処理に関するチューニングに応じて、必要となるリソースが増加します。

表 3-54 Metricbeat のリソース設定

項番	パラメタ	説明
1	requests.cpu	詳細は「(1) Helm チャートのパラメタ」を参照してください。
2	requests.memory	
3	limits.cpu	
4	limits.memory	

(7) Prometheus

(a) 冗長化・レプリケーション

レプリカ数の変更によって、Prometheus をレプリケーションすることはできます。ただし、起動した Prometheus ごとに独立してメトリクスを収集する動作を行います。そのため、Prometheus 間で収集したメトリクスの重複が発生します。

また、Metricbeat に対してのメトリクスの送信も Prometheus ごとに行われるため、Elasticsearch に格納されるメトリクスデータの重複が発生します。

レプリケーションを実施する場合、起動させる Kubernetes ノードは、異なるノード上で起動させるようにすることで、可用性が向上します。

(b) パラメタ

Prometheus では、各コンポーネントよりメトリクス情報の収集、参照と、Metricbeat へ転送する動作を行います。Prometheus のパフォーマンスを調整するため、次の表に示すチューニングを実施してください。

チューニング対象のパラメタ一覧

Prometheus のチューニング対象のパラメタ一覧を次に示します。

表 3-55　チューニング対象のパラメター一覧

項番	パラメタ	説明
1	prometheus.scrape.interval	Prometheus がメトリクスの取得対象から情報を取得する間隔を定義します。
2	prometheus.remoteWrite.queue.capacity	Prometheus が Metricbeat にデータの送信処理をシャードという単位で並列実行してメトリクスを送信します。 データを送信するため、シャードごとにメモリ内でキューを作成します。 このキューのサイズを指定します。
3	prometheus.remoteWrite.queue.maxShards	Metricbeat にデータを送信するシャードのサイズ、または並列処理の最大数を指定します。通常では最大数まで使用しないよう動作するが、送信が遅延した場合などに定義した最大数による送信処理を実施します。
4	prometheus.remoteWrite.queue.maxSamplesPerSend	一度にまとめて送信するデータのサイズを定義します。
5	prometheus.storage.retentionTime	データを保持し続ける期間を定義します。

チューニング方法

表 3-56　チューニング方法

項番	パラメタ	チューニングの考え方	デフォルト値
1	prometheus.scrape.interval	間隔を短くするほど、Prometheus が使用するディスク容量が上昇します。メトリクス取得に関する要件と、Prometheus が使用量を基に、設定値を検討してください。	「(1)　Helm チャートのパラメタ」を参照してください。
2	prometheus.remoteWrite.queue.capacity	Metricbeat に送信する処理のメモリ使用量は、「シャード数 ×（prometheus.remoteWrite.queue.capacity ＋ prometheus.remoteWrite.queue.maxSamplesPerSend）」に比例します。 大きい値に設定するほど、多くのメモリを消費します。Prometheus としては、prometheus.remoteWrite.queue.maxSamplesPerSend の 3～10 倍程度の値に設定することを推奨しています。	
3	prometheus.remoteWrite.queue.maxShards	Prometheus では、Metricbeat への送信に異常に時間が掛かる場合ではないかぎりデフォルト値を超えた値を設定する必要はありません。	
4	prometheus.remoteWrite.queue.maxSamplesPerSend	Prometheus では、ほとんどの利用にかぎり問題ない値としてデフォルト値が定義されています。 Prometheus のメモリ使用量が高い場合にかぎり、prometheus.remoteWrite.queue.capacity の項目に記載したメモリ使用量の見積もり式を基に、チューニングを実施してください。	

項番	パラメタ	チューニングの考え方	デフォルト値
5	prometheus.storage.retentionTime	Prometheus で参照可能なデータの保持期間を定義します。期間が長期になるほど、Prometheus が保持するメトリクスデータは増大します。 Prometheus の利用用途とディスクサイズに合わせて設定してください。	

(c) CPU、メモリのチューニング

Prometheus では、メトリクス収集、および Metricbeat への送信処理に関するチューニングに応じて、必要となるリソースが増加します。

表 3-57 Prometheus のリソース設定

項番	パラメタ	説明
1	requests.cpu	詳細は「(1) Helm チャートのパラメタ」を参照してください。
2	requests.memory	
3	limits.cpu	
4	limits.memory	

4 TP1-Converter の運用（通常版かつ TP1-Bridge 限定）

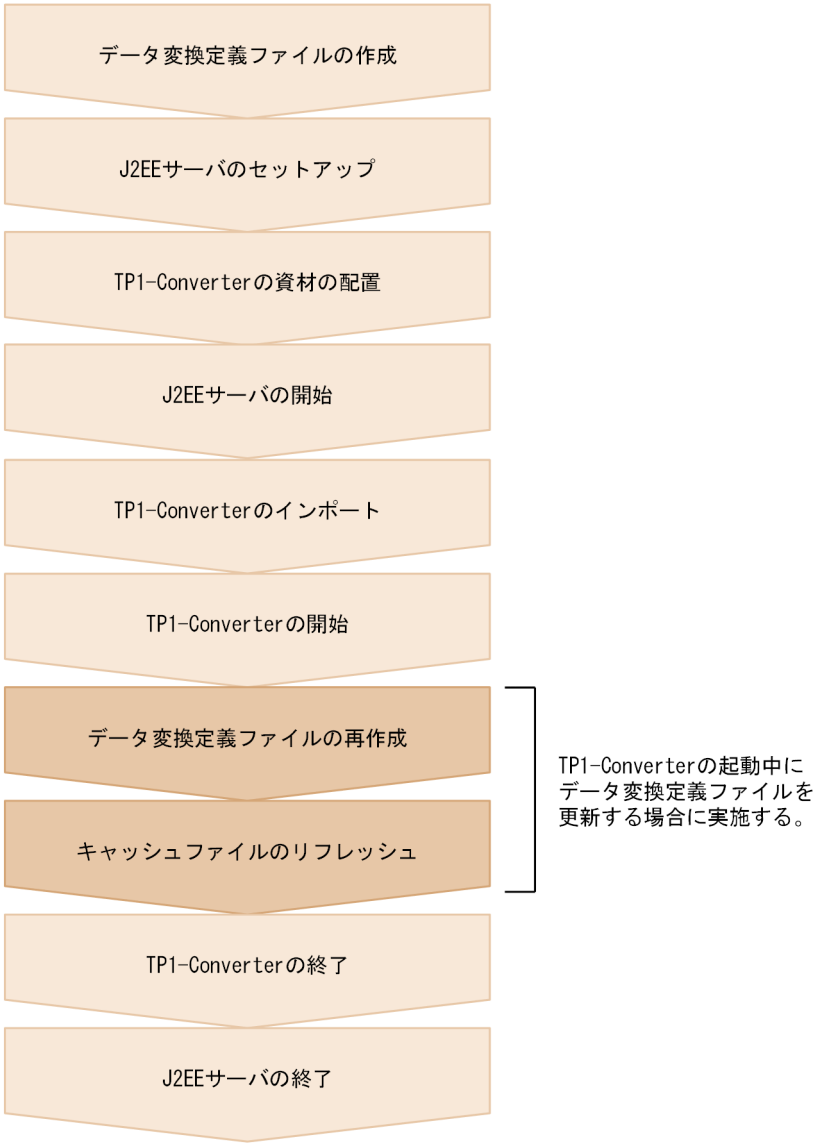
4.1　TP1-Converter の運用の流れ（通常版かつ TP1-Bridge 限定）

TP1 連携機能の中の TP1-Converter は、uCosminexus Service Platform 上で動作する J2EE アプリケーションです。ここでは、TP1-Converter を使用するための運用について説明します。

なお、ここでの説明は、uCosminexus Service Platform のインストールおよび環境設定が完了していることを前提としています。uCosminexus Service Platform のインストールおよび環境設定については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」を参照してください。また、運用方法の中で使用する uCosminexus Service Platform のコマンドの詳細な仕様については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照してください。

TP1-Converter を使用するための作業の流れを、次の図に示します。

図 4-1　TP1-Converter を使用するための作業の流れ



4.2　データ変換定義ファイルの作成（通常版かつ TP1-Bridge 限定）

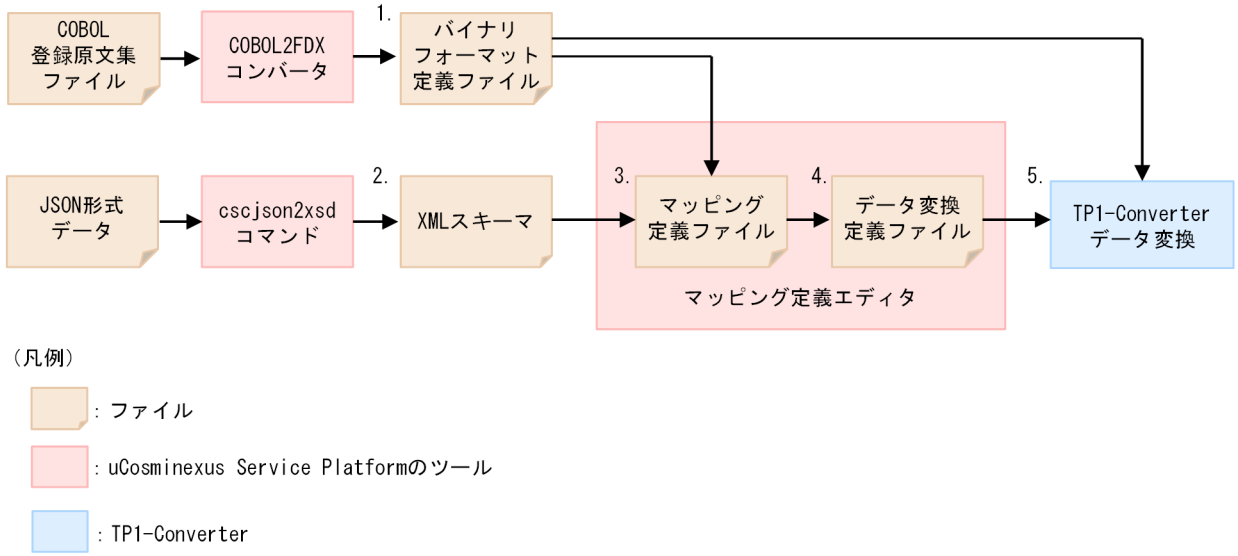
TP1-Converter のデータ変換で使用するデータ変換定義ファイルは、uCosminexus Service Architect の機能を使用して作成します。

ここでは、OpenTP1 の SPP が COBOL で作成されている場合を例に説明します。

4.2.1　データ変換で使用するファイル作成の流れ（通常版かつ TP1-Bridge 限定）

データ変換で使用するファイルを作成する流れを次に示します。

図 4-2　データ変換で使用するファイルを作成する流れ



＜説明＞

1. バイナリフォーマット定義ファイルの作成

   COBOL2FDX コンバータを使用して、COBOL 登録原文集ファイルからバイナリフォーマット定義ファイルを作成します。

2. XML スキーマの作成

   cscjson2xsd コマンドを実行して、JSON 形式のファイルから XML スキーマを作成します。

3. マッピング定義ファイルの作成

   マッピング定義エディタを使用して、1.で作成したバイナリフォーマット定義ファイル、および 2.で作成した XML スキーマからマッピング定義ファイルを作成します。

4. データ変換定義ファイルの作成

   マッピング定義エディタを使用して、3.で作成したマッピング定義ファイルをからデータ変換定義ファイルを作成します。

5. TP1-Converter のデータ変換

TP1-Converter は、1.で作成したバイナリフォーマット定義ファイル、および 4.で作成したデータ変換定義ファイルを使用してデータ変換をします。

表 4-1　データ変換で使用するために作成するファイル

ファイル名	拡張子	作成元になるファイル	作成方法	ファイルの用途	参照先
バイナリフォーマット定義ファイル	fdx	COBOL 登録原文集ファイル※	COBOL2FDX コンバータ※	• マッピング定義ファイルの作成 • データ変換	4.2.2　バイナリフォーマット定義ファイル（通常版かつ TP1-Bridge 限定）
XML スキーマ	xsd	JSON 形式のファイル	cscjson2xsd コマンド	マッピング定義ファイルの作成	4.2.3　XML スキーマ（通常版かつ TP1-Bridge 限定）
マッピング定義ファイル	mdo	• バイナリフォーマット定義ファイル • XML スキーマ	マッピング定義エディタ	データ変換定義ファイルの作成	4.2.4　マッピング定義ファイル（通常版かつ TP1-Bridge 限定）
データ変換定義ファイル	xsl	マッピング定義ファイル	マッピング定義エディタ	データ変換	4.2.5　データ変換定義ファイル（通常版かつ TP1-Bridge 限定）

注※

C 言語から作成する場合は使用しません。

4.2.2　バイナリフォーマット定義ファイル（通常版かつ TP1-Bridge 限定）

バイナリフォーマット定義ファイルは、マッピング定義ファイルの作成時およびデータ変換時に使用します。

バイナリフォーマット定義ファイルは、OpenTP1 への要求電文または応答電文の COBOL 登録原文集ファイルから作成します。なお、要求電文と応答電文それぞれで作成する必要があります。要求電文から作成する場合は、「<サービス名>_jtob.fdx」のファイル名で作成してください。また、応答電文から作成する場合は、「<サービス名>_btoj.fdx」のファイル名で作成してください。

バイナリフォーマット定義ファイルの作成方法については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編」の、COBOL 登録集原文ファイルからのバイナリフォーマット定義ファイルの生成に関する説明を参照してください。

> **メモ**
>
> OpenTP1 の SPP が C 言語で作成されている場合は、バイナリフォーマット定義ファイルを作成する方法が異なります。C 言語の場合のバイナリフォーマット定義ファイルの作成方法については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編」の、バイナリフォーマット定義ファイルの新規作成に関する説明を参照してください。

4.2.3 XML スキーマ（通常版かつ TP1-Bridge 限定）

XML スキーマは、マッピング定義ファイルの作成時に使用します。

XML スキーマは、Orchestrator/Entity-Service から TP1-Bridge への要求電文または応答電文の JSON 形式データから作成します。なお、要求電文と応答電文それぞれで作成する必要があります。

XML スキーマの作成方法については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編」の、JSON 形式データに対応する XML スキーマの生成に関する説明を参照してください。

4.2.4 マッピング定義ファイル（通常版かつ TP1-Bridge 限定）

マッピング定義ファイルは、データ変換定義ファイルの作成時に使用します。

マッピング定義ファイルは、バイナリフォーマット定義ファイルおよび XML スキーマから作成します。なお、要求電文と応答電文それぞれで作成する必要があります。

マッピング定義ファイルの作成方法については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編」の、データ変換の定義手順に関する説明を参照してください。

4.2.5 データ変換定義ファイル（通常版かつ TP1-Bridge 限定）

データ変換定義ファイルは、データ変換時に使用します。

データ変換定義ファイルは、マッピング定義ファイルから作成します。なお、要求電文と応答電文それぞれで作成する必要があります。要求電文から作成する場合は、「<サービス名>_jtob.xsl」のファイル名で作成してください。また、応答電文から作成する場合は、「<サービス名>_btoj.xsl」のファイル名で作成してください。

データ変換定義ファイルの作成方法については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編」の、データ変換の定義手順に関する説明を参照してください。

4.3　J2EE サーバのセットアップ（通常版かつ TP1-Bridge 限定）

uCosminexus Service Platform の cjsetup（J2EE サーバのセットアップとアンセットアップ）コマンドで、J2EE サーバをセットアップします。

次に示すコマンドを実行してください。

```
/opt/Cosminexus/CC/server/bin/cjsetup TP1Converter
```

コマンドの戻り値が 0 の場合は、J2EE サーバのセットアップは正常に終了しています。

コマンドの戻り値が 0 以外の場合は、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照して J2EE サーバのセットアップが正常に終了しなかった原因を取り除いてください。

4.4　TP1-Converter の資材の配置（通常版かつ TP1-Bridge 限定）

uCosminexus Service Platform の環境に TP1-Converter が使用する資材を配置します。

TP1-Converter が使用する資材の配置元と配置先の場所を次に示します。

表 4-2　TP1-Converter が使用する資材の配置元と配置先の場所

項番	ファイル名	配置元の場所	配置先の場所
1	TP1-Converter 用アプリケーションライブラリ（tp1converter.war）	<HMP-PCTO 提供媒体のルートフォルダ>/TP1-Converter/lib/	uCosminexus Service Platform 実行環境の任意のディレクトリ
2	J2EE サーバ用オプション定義ファイル（usrconf.cfg）	<HMP-PCTO 提供媒体のルートフォルダ>/TP1-Converter/usrconf/	/opt/Cosminexus/CC/server/usrconf/ejb/TP1Converter/
3	J2EE サーバ用ユーザプロパティファイル（usrconf.properties）	<HMP-PCTO 提供媒体のルートフォルダ>/TP1-Converter/usrconf/	/opt/Cosminexus/CC/server/usrconf/ejb/TP1Converter/
4	データ変換定義ファイル	「4.2　データ変換定義ファイルの作成（通常版かつ TP1-Bridge 限定）」で作成したファイルの出力先	/opt/Cosminexus/CC/server/usrconf/ejb/TP1Converter/※ サービスグループに対応したディレクトリを追加して配置します。

注※

データ変換定義ファイルの配置先の場所は、usrconf.properties の jp.co.Hitachi.soft.hmppcto.tp1.data-conversion.definition-file.dir の設定で変更できます。

4.5　J2EE サーバの開始（通常版かつ TP1-Bridge 限定）

uCosminexus Service Platform の cjstartsv（J2EE サーバの開始）で、J2EE サーバを開始します。

次に示すコマンドを実行してください。

```
/opt/Cosminexus/CC/server/bin/cjstartsv TP1Converter
```

cjstartsv コマンド実行時の標準出力に KDJE30028-I メッセージが出力された場合は、J2EE サーバは正常に開始しています。

KDJE30028-I メッセージが出力されずエラーメッセージが出力された場合は、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照して J2EE サーバが正常に開始しなかった原因を取り除いてください。

4.6 TP1-Converter のインポート（通常版かつ TP1-Bridge 限定）

uCosminexus Service Platform の cjimportwar（WAR アプリケーションのインポート）コマンドで、TP1-Converter をインポートします。

次に示すコマンドを実行してください。

```
/opt/Cosminexus/CC/admin/bin/cjimportwar TP1Converter -f /tmp/tp1converter.war※
```

注※

TP1-Converter 用アプリケーションライブラリは、uCosminexus Service Platform 実行環境の任意のディレクトリに配置してください。この例では/tmp/としています。

usrconf.properties の ejbserver.naming.host または ejbserver.naming.port の値をデフォルト値から変更している場合は、-nameserver オプションでプロバイダ URL を指定してください。

コマンドの戻り値が 0 の場合は、インポートが正常に終了しています。

コマンドの戻り値が 0 以外の場合は、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照してインポートが正常に終了しなかった原因を取り除いてください。

4.7 TP1-Converter の開始（通常版かつ TP1-Bridge 限定）

uCosminexus Service Platform の cjstartapp（J2EE アプリケーションの開始）コマンドで、TP1-Converter を開始します。

次に示すコマンドを実行してください。

```
/opt/Cosminexus/CC/admin/bin/cjstartapp TP1Converter -name tp1converter
```

usrconf.properties の ejbserver.naming.host または ejbserver.naming.port の値をデフォルト値から変更している場合は、-nameserver オプションでプロバイダ URL を指定してください。

コマンドの戻り値が 0 の場合は、TP1-Converter は正常に開始しています。

コマンドの戻り値が 0 以外の場合は、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照して TP1-Converter が正常に開始しなかった原因を取り除いてください。

なお、TP1-Converter と TP1-Bridge は、どちらを先に開始してもかまいません。

4.8　データ変換定義の再作成（通常版かつ TP1-Bridge 限定）

TP1-Converter の起動中に OpenTP1 の SPP を修正し、データ変換定義ファイルの更新が必要になった場合は、データ変換定義ファイルを再作成してください。

再作成したデータ変換定義ファイルは、「4.4　TP1-Converter の資材の配置（通常版かつ TP1-Bridge 限定）」のデータ変換定義ファイルの配置先に配置してください。

データ変換定義ファイルは、SPP の修正内容に応じて更新してください。SSP の修正内容とデータ変換定義ファイルの更新方法を次に示します。

表 4-3　SSP の修正内容とデータ変換定義ファイルの更新方法

項番	SSP の修正内容	データ変換定義ファイルの更新方法
1	サービスの要求データ、または応答データの変更	サービスに対応したデータ変換定義ファイルを更新する。
2	サービスの追加	サービスに対応したデータ変換定義ファイルを追加する。
3	サービスグループの追加	サービスグループに対応したディレクトリを追加する。
4	サービスの削除	サービスに対応したデータ変換定義ファイルを削除する。
5	サービスグループの削除	サービスグループに対応したディレクトリを削除する。

データ変換定義ファイルの作成方法については、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編」の、データ変換の定義手順に関する説明を参照してください。

4.9 キャッシュのリフレッシュ（通常版かつ TP1-Bridge 限定）

TP1-Converter の起動中にデータ変換定義ファイルを更新した場合は、TP1-Converter のキャッシュのリフレッシュを実行してください。キャッシュのリフレッシュを実行することで、キャッシュ中のデータ変換定義ファイルが更新され、以降のデータ変換には更新後のデータ変換定義ファイルの定義が適用されます。

キャッシュのリフレッシュは、TP1-Converter に http リクエストを送信することで実行されます。キャッシュのリフレッシュを実行するための http リクエストの URL を、次に示します。

```
http://<NIO HTTPサーバのホスト名>:<NIO HTTPサーバのポート番号>/tp1converter/cache/refresh[?servicegroup=<サービスグループ名>]
```

キャッシュのリフレッシュは、サービスグループ単位のリフレッシュ、またはキャッシュ全体のリフレッシュを選択できます。1 つの SPP を更新した場合は、サービスグループ単位でのリフレッシュを推奨します。

次に示す条件の場合のキャッシュのリフレッシュの実行例を示します。

- http リクエスト送信に使用するクライアント：curl コマンド
- NIO HTTP サーバのホスト名：hostname
- NIO HTTP サーバのポート番号：8008

＜サービスグループ名 svg1 のリフレッシュの実行例＞

```
curl -X GET http://hostname:8008/tp1converter/cache/refresh?servicegroup=svg1
```

＜キャッシュ全体のリフレッシュの実行例＞

```
curl -X GET http://hostname:8008/tp1converter/cache/refresh
```

キャッシュのリフレッシュが成功した場合は、http ステータスコード 200 が返されます。

キャッシュのリフレッシュが失敗した場合は、http ステータスコード 200 以外が返されます。http ステータスコードが 400 番台の場合は、指定した URL に誤りがないか、または TP1-Converter が停止していないかを確認して、失敗の原因を取り除いてください。http ステータスコードが 500 番台の場合は、TP1-Converter のキャッシュリフレッシュ処理で障害が発生しています。TP1-Converter が出力するログを参照して、失敗の原因を取り除いてください。

重要

- データ変換時に参照するキャッシュの不整合を避けるために、更新する SPP を停止した状態で、キャッシュのリフレッシュリクエストを送信してください。キャッシュのリフレッシュが終了したあとに、SPP を再起動してください。

- キャッシュのリフレッシュ中はキャッシュに対して排他制御をするため、データ変換処理が待たされることがあります。そのため、キャッシュのリフレッシュは、動作しているトランザクションが少ない時間帯に実行することを推奨します。

4.10　TP1-Converter の終了（通常版かつ TP1-Bridge 限定）

uCosminexus Service Platform の cjstopapp（J2EE アプリケーションの停止）コマンドで、TP1-Converter を終了します。

次に示すコマンドを実行してください。

```
/opt/Cosminexus/CC/admin/bin/cjstopapp TP1Converter -name tp1converter
```

usrconf.properties の ejbserver.naming.host または ejbserver.naming.port の値をデフォルト値から変更している場合は、-nameserver オプションでプロバイダ URL を指定してください。

コマンドの戻り値が 0 の場合は、TP1-Converter は正常に終了しています。

コマンドの戻り値が 0 以外の場合は、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照して TP1-Converter が正常に終了しなかった原因を取り除いてください。

なお、TP1-Converter と TP1-Bridge は、どちらを先に終了してもかまいません。

4.11　J2EE サーバの終了（通常版かつ TP1-Bridge 限定）

uCosminexus Service Platform の cjstopsv（J2EE サーバの終了）コマンドで、J2EE サーバを終了します。

次に示すコマンドを実行してください。

```
/opt/Cosminexus/CC/server/bin/cjstopsv TP1Converter
```

コマンドの戻り値が 0 の場合は、J2EE サーバは正常に終了しています。

コマンドの戻り値が 0 以外の場合は、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照して J2EE サーバが正常に終了しなかった原因を取り除いてください。

4.12　キャッシュの参照（通常版かつ TP1-Bridge 限定）

TP1-Converter のキャッシュに登録されているデータ変換定義ファイルの内容を確認するために、キャッシュを参照できます。キャッシュの参照は、TP1-Converter に http リクエストを送信することで実行されます。キャッシュの参照を実行するための http リクエストの URL を、次に示します。

```
http://<NIO HTTPサーバのホスト名>:<NIO HTTPサーバのポート番号>/tp1converter/cache/refer
```

次に示す条件の場合のキャッシュの参照の実行例を示します。

- http リクエスト送信に使用するクライアント：curl コマンド
- NIO HTTP サーバのホスト名：hostname
- NIO HTTP サーバのポート番号：8008

＜キャッシュの参照の実行例＞

```
curl -X GET http://hostname:8008/tp1converter/cache/refer
```

キャッシュの参照に成功した場合は、レスポンスのボディ部に、次の形式でキャッシュに登録されているデータ変換定義ファイルの情報が返されます。

```
[
  {
    "servicegroup": "gg…gg",
      "service": ["vv…vv",
        :    （サービスの数だけ繰り返されます）
      ],
    "date": "yyyymmdd hh:mm:ss"
  },
    :        （サービスグループの数だけ繰り返されます）
]
```

参照結果の項目

- *gg…gg*：サービスグループ名
- *vv…vv*：サービス名
- *yyyymmdd hh:mm:ss*：
  サービスグループのデータ変換定義ファイルをキャッシュに登録した日時

キャッシュが空の場合は、レスポンスのボディ部に、次の形式で参照結果が返されます。

```
[
]
```

キャッシュの参照が成功した場合は、http ステータスコード 200 が返されます。

キャッシュの参照が失敗した場合は、http ステータスコード 200 以外が返されます。http ステータスコードが 400 番台の場合は、指定した URL に誤りがないか、または TP1-Converter が停止していないかを確認して、失敗の原因を取り除いてください。http ステータスコードが 500 番台の場合は、TP1-Converter のキャッシュ参照処理で障害が発生しています。TP1-Converter が出力するログを参照して、失敗の原因を取り除いてください。

4.13 TP1-Converter の削除と J2EE サーバのアンセットアップ（通常版かつ TP1-Bridge 限定）

TP1-Converter が不要になった場合は、次の作業を実施してください。

- TP1-Converter の削除
- J2EE サーバのアンセットアップ

4.13.1 TP1-Converter の削除（通常版かつ TP1-Bridge 限定）

uCosminexus Service Platform の cjdeleteapp（J2EE アプリケーションの削除）コマンドで、TP1-Converter を削除します。

次に示すコマンドを実行してください。

```
/opt/Cosminexus/CC/admin/bin/cjdeleteapp TP1Converter -name tp1converter
```

usrconf.properties の ejbserver.naming.host または ejbserver.naming.port の値をデフォルト値から変更している場合は、-nameserver オプションでプロバイダ URL を指定してください。

コマンドの戻り値が 0 の場合は、TP1-Converter は正常に削除されています。

コマンドの戻り値が 0 以外の場合は、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照して TP1-Converter が正常に削除されなかった原因を取り除いてください。

4.13.2 J2EE サーバのアンセットアップ（通常版かつ TP1-Bridge 限定）

TP1-Converter を削除したあとに、uCosminexus Service Platform の cjsetup（J2EE サーバのセットアップとアンセットアップ）コマンドで、J2EE サーバをアンセットアップします。

次に示すコマンドを実行してください。

```
/opt/Cosminexus/CC/server/bin/cjsetup -d TP1Converter
```

コマンドの戻り値が 0 の場合は、J2EE サーバのアンセットアップは正常に終了しています。

コマンドの戻り値が 0 以外の場合は、マニュアル「Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド」、およびマニュアル「Cosminexus アプリケーションサーバ リファレンス コマンド編」を参照して J2EE サーバのアンセットアップが正常に終了しなかった原因を取り除いてください。

5 コンテナイメージの作成要件

5.1 コンテナイメージの一覧

HMP-PCTO は、ユーザアプリケーションに取り込んで使用する製品です。HMP-PCTO では、アプリケーションをコンテナオーケストレーションツール上にデプロイして実行します。このため、アプリケーションはコンポーネントごとにコンテナイメージ形式で作成します。

HMP-PCTO の運用として、コンテナイメージは、ユーザが用意する資材で作成するものと、HMP-PCTO が提供する資材を使用してユーザが作成するものがあります。コンテナイメージの分類を次に示します。

- ユーザ責務のコンテナイメージ
- HMP-PCTO のコントロールプレーンのコンテナイメージ

また、HMP-PCTO のコントロールプレーンのコンテナイメージには、次のバリエーションがあります。

- 通常版
  Java アプリケーションを実行するコンテナイメージの場合、ユーザはどちらかを選択できます。
  - uCosminexus Application Runtime with Java for Spring Boot 導入あり
  - uCosminexus Application Runtime with Java for Spring Boot 導入なし

  ただし、次に示すコンテナイメージは例外とします。
  - ユーザのビジネスロジックを含まないコンテナイメージは、「uCosminexus Application Runtime with Java for Spring Boot 導入あり」だけとします（HMP-PCTO としてトラブルシュートができるようにするため）。
  - HMP-PCTO 以外の日立製品のコンテナイメージ（例：EADS）では、uCosminexus Application Runtime with Java for Spring Boot 導入有無は HMP-PCTO の仕様に従います。
- トライアル版

HMP-PCTO のコンテナイメージの一覧を次の表に示します。

表 5-1 HMP-PCTO のコンテナイメージの一覧

コンテナイメージ	コンテナイメージの分類		バリエーション			備考
	ユーザ責務のコンテナイメージ	HMP-PCTO のコントロールプレーンのコンテナイメージ	通常版 uCosminexus Application Runtime with Java for Spring Boot		トライアル版	
			導入あり	導入なし		
uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ	○	−	○	−	−	通常版のユーザ責務のコンテナイメージで uCosminexus Application Runtime with Java for Spring Boot を

コンテナイメージ	コンテナイメージの分類		バリエーション			備考
	ユーザ責務のコンテナイメージ	HMP-PCTO のコントロールプレーンのコンテナイメージ	通常版 uCosminexus Application Runtime with Java for Spring Boot		トライアル版	
			導入あり	導入なし		
						導入する場合だけ作成します。
JDK コンテナベースイメージ	○	−	−	○	○	トライアル版の場合、または通常版のユーザ責務のコンテナイメージで uCosminexus Application Runtime with Java for Spring Boot を導入しない場合だけ作成します。
Orchestrator コンテナイメージ	○	−	○	○	○	−
Entity-Service コンテナイメージ	○	−	○	○	○	サービス間を中継するサービスとしての Entity-Service がある場合だけ作成します。 詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のトランザクションのモデル（その 2）に関する説明を参照してください。
SQL-Participant コンテナイメージ	○ (JDBC Driver を組み込むだけ)	−	○	−	○	SQL-Participant を使用する場合だけ作成します。
Entity-Module/TCC-Participant コンテナイメージ	○	−	○	○	○	TCC-Participant を使用する場合だけ作成します。
Mediator コンテナイメージ	−	○	○	−	○	−

コンテナイメージ	コンテナイメージの分類		バリエーション			備考
	ユーザ責務のコンテナイメージ	HMP-PCTO のコントロールプレーンのコンテナイメージ	通常版 uCosminexus Application Runtime with Java for Spring Boot		トライアル版	
			導入あり	導入なし		
TP1-Bridge コンテナイメージ	−	○	○	−	−	TP1-Bridge を使用する場合だけ作成します。
Dependency-Checker コンテナイメージ	−	○	−	○	○	−
Ext-Cons コンテナイメージ	−	○	−	−	○	トライアル版の場合だけ作成します。
EADS サーバ用コンテナイメージ	−	○	−	○	−	通常版の場合だけ作成します。
EADS コマンド実行用コンテナイメージ	−	○	−	○	−	通常版の場合だけ作成します。

（凡例）

○：該当します。

−：該当しません。

5.2　コンテナイメージの提供方式

HMP-PCTO のコンテナイメージの提供方式の一覧を次の表に示します。

表 5-2　HMP-PCTO のコンテナイメージの提供方式の一覧

コンテナイメージの分類	通常版/トライアル版	提供方式	ユーザへの提供物	ユーザの対応作業
ユーザ責務のコンテナイメージ	通常版およびトライアル版	Containerfile 方式	• Containerfile の記述例（「2.2　Java アプリケーションの開発」参照） • コンテナベースイメージおよび前提プログラムの一覧 • 【SQL-Participant を使用する場合】HMP-PCTO のアプリケーション JAR ファイルビルド用 Java プロジェクト • HMP-PCTO の各ライブラリ JAR ファイル • HMP-PCTO の各シェルスクリプト	1. HMP-PCTO の各ライブラリ JAR ファイルを使用した、SQL-Participant 以外のユーザ責務の Kubernetes アプリケーションの JAR ファイルをビルドする。ユーザが作成するアプリケーションのため、ユーザ任意の方法でビルドする。 2. 【SQL-Participant を使用する場合】HMP-PCTO の SQL-Participant アプリケーションの JAR ファイルをビルドする 3. 一覧を基に、コンテナベースイメージおよび前提プログラムを準備する 4. Containerfile の記述例を基に Containerfile を作成する 5. Containerfile を使用しコンテナイメージをビルドする
HMP-PCTO のコントロールプレーンのコンテナイメージ	通常版	Containerfile 方式	• Containerfile のテンプレート • コンテナベースイメージおよび前提プログラムの一覧 • HMP-PCTO のアプリケーション JAR ファイルビルド用 Java プロジェクト • HMP-PCTO の各ライブラリ JAR ファイル	1. HMP-PCTO の各ライブラリ JAR ファイルを使用した HMP-PCTO のアプリケーションの JAR ファイルをビルドする 2. 一覧を基に、コンテナベースイメージおよび前提プログラムを準備する

コンテナイメージの分類	通常版/トライアル版	提供方式	ユーザへの提供物	ユーザの対応作業
			• HMP-PCTO の各シェルスクリプト	3. ユーザの開発環境に応じて、Containerfile のテンプレートの内容を編集する 4. Containerfile を使用しコンテナイメージをビルドする
	トライアル版	コンテナイメージ方式	• HMP-PCTO の各コンテナイメージ	なし

5.3　各コンテナイメージの作成要件

各コンテナイメージの詳細について説明します。

以降で説明するコンテナベースイメージおよび前提プログラムのバージョンは、「1.4.1　前提 OS および前提プログラム」を参照してください。

5.3.1　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（通常版限定）

このコンテナベースイメージは、ユーザ責務のコンテナイメージで uCosminexus Application Runtime with Java for Spring Boot を導入する場合にだけ作成します。

次の表に示す uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの要件を満たすように作成します。

表 5-3　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージの要件

項目	要件
コンテナベースイメージ	Red Hat Universal Base Image 8※
前提プログラム	• Java SE 8 の uCosminexus Application Runtime with Java for Spring Boot • curl • jq
配置ファイル	規定しません。
コンテナ実行ユーザ・グループ	規定しません。
コンテナ起動時カレントディレクトリ	規定しません。
コンテナ起動コマンドライン	指定なしとしてください（コマンドについてはこのコンテナベースイメージを使用する各コンテナイメージ側で指定します）。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します）。

注※

UBI 8 の任意のベースイメージ（UBI 8 Minimal など）

5.3.2 JDK コンテナベースイメージ

このコンテナベースイメージは、トライアル版の場合、または通常版のユーザ責務のコンテナイメージで uCosminexus Application Runtime with Java for Spring Boot を導入しない場合だけ作成します。

次の表に示す JDK コンテナベースイメージの要件を満たすように作成します。

表 5-4 JDK コンテナベースイメージの要件

項目	要件
コンテナベースイメージ	■通常版の場合 • Red Hat Universal Base Image 8※ ■トライアル版の場合 • Almalinux 8 Minimal 8.8
前提プログラム	■通常版の場合 • Java SE 8 準拠の JDK • curl • jq ■トライアル版の場合 • Java SE 8 準拠の JDK • curl • jq
配置ファイル	規定しません。
コンテナ実行ユーザ・グループ	規定しません。
コンテナ起動時カレントディレクトリ	規定しません。
コンテナ起動コマンドライン	指定なしとしてください（コマンドについてはこのコンテナベースイメージを使用する各コンテナイメージ側で指定します）。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します）。

注※

UBI 8 の任意のベースイメージ（UBI 8 Minimal など）

5.3.3 Orchestrator コンテナイメージ

次の表に示す Orchestrator コンテナイメージの要件を満たすように作成します。

表 5-5 Orchestrator コンテナイメージの要件

項目	要件
コンテナベースイメージ	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（「5.3.1 uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（通常版限定）」を参照してください） ■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入なし）、トライアル版の場合 • JDK コンテナベースイメージ（「5.3.2 JDK コンテナベースイメージ」を参照してください）
前提プログラム	規定しません。
配置ファイル	次に示すファイルを配置してください。 • ファイル：ユーザの Orchestrator アプリケーションの JAR ファイル（ユーザ任意のファイル名称） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。
コンテナ実行ユーザ・グループ	規定しません。
コンテナ起動時カレントディレクトリ	規定しません。
コンテナ起動コマンドライン	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト(starter.sh)」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Orchestrator アプリケーションの JAR ファイル」を指定してください。 ■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入なし）、トライアル版の場合 • コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Orchestrator アプリケーションの JAR ファイル」を指定してください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。

項目	要件
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します）。

5.3.4 Entity-Service コンテナイメージ

サービス間を中継するサービスとしての Entity-Service がある場合だけ作成します。

詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のトランザクションのモデル（その 2）に関する説明を参照してください。

次の表に示す Entity-Service コンテナイメージの要件を満たすように作成します。

表 5-6 Entity-Service コンテナイメージの要件

項目	要件
コンテナベースイメージ	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（「5.3.1 uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（通常版限定）」を参照してください） ■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入なし）、トライアル版の場合 • JDK コンテナベースイメージ（「5.3.2 JDK コンテナベースイメージ」を参照してください）
前提プログラム	規定しません。
配置ファイル	次に示すファイルを配置してください。 • ファイル：ユーザの Entity-Service アプリケーションの JAR ファイル（ユーザ任意のファイル名称） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。
コンテナ実行ユーザ・グループ	規定しません。
コンテナ起動時カレントディレクトリ	規定しません。

項目	要件
コンテナ起動コマンドライン	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト(starter.sh)」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Service アプリケーションの JAR ファイル」を指定してください。 ■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入なし）、トライアル版の場合 • コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Service アプリケーションの JAR ファイル」を指定してください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します）。

5.3.5 SQL-Participant コンテナイメージ

次の表に示す SQL-Participant コンテナイメージの要件を満たすように作成します。

表 5-7 SQL-Participant コンテナイメージの要件

項目	要件
コンテナベースイメージ	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（「5.3.1 uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（通常版限定）」を参照してください） ■トライアル版の場合 • JDK コンテナベースイメージ（「5.3.2 JDK コンテナベースイメージ」を参照してください）
前提プログラム	■通常版の場合 • Hitachi Elastic Application Data Store Client for Java • ncompress • ncurses

項目	要件
	• tar • cpio • gzip ■トライアル版の場合 • 規定しません。
配置ファイル	次に示すファイルを配置してください。 • ファイル：HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル（sql-participant-V.R.S.jar） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。 • ファイル：JDBC Driver の JAR ファイル（ユーザ任意のファイル名称） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。 • ファイル：Hitachi Elastic Application Data Store Client for Java の JAR ファイル（トライアル版の場合は不要） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。 • ファイル：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの実行権限を付与してください。 • ファイル：HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの実行権限を付与してください。
コンテナ実行ユーザ・グループ	規定しません。
コンテナ起動時カレントディレクトリ	規定しません。
コンテナ起動コマンドライン	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト(starter.sh)」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル(sql-participant-V.R.S.jar)」を指定してください。 ■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入なし）、トライアル版の場合

項目	要件
	• コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「HMP-PCTO の SQL-Participant アプリケーションの JAR ファイル(sql-participant-V.R.S.jar)」を指定してください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します)。
環境変数	次に示す環境変数を配置してください。 • 環境変数名：LOADER_PATH 設定値：Hitachi Elastic Application Data Store Client for Java の JAR ファイルを配置したパス
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します)。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します)。

注　V.R.S には、バージョンを設定します。

5.3.6　Entity-Module/TCC-Participant コンテナイメージ

次の表に示す Entity-Module/TCC-Participant コンテナイメージの要件を満たすように作成します。

表 5-8　Entity-Module/TCC-Participant コンテナイメージの要件

項目	要件
コンテナベースイメージ	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（「5.3.1　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（通常版限定)」を参照してください) ■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入なし)、トライアル版の場合 • JDK コンテナベースイメージ（「5.3.2　JDK コンテナベースイメージ」を参照してください)
前提プログラム	■通常版の場合 • Hitachi Elastic Application Data Store Client for Java • ncompress • ncurses • tar • cpio

項目	要件
	• gzip ■トライアル版の場合 • 規定しません。
配置ファイル	次に示すファイルを配置してください。 • ファイル：ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル（ユーザ任意のファイル名称） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。 • ファイル：Hitachi Elastic Application Data Store Client for Java の JAR ファイル（トライアル版の場合は不要） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。 • ファイル：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの実行権限を付与してください。 • ファイル：HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの実行権限を付与してください。
コンテナ実行ユーザ・グループ	規定しません。
コンテナ起動時カレントディレクトリ	規定しません。
コンテナ起動コマンドライン	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト(starter.sh)」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」を指定してください。 ■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入なし）、トライアル版の場合 • コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「ユーザの Entity-Module/TCC-Participant アプリケーションの JAR ファイル」を指定してください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。
環境変数	次に示す環境変数を配置してください。 • 環境変数名：LOADER_PATH

項目	要件
	設定値：Hitachi Elastic Application Data Store Client for Java の JAR ファイルを配置したパス
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します）。

5.3.7 Mediator コンテナイメージ

このコンテナイメージは、HMP-PCTO として提供します。通常版の場合は、次の表に示す要件を満たすように作成します。

表 5-9 Mediator コンテナイメージの要件

項目	要件
コンテナベースイメージ	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • Red Hat Universal Base Image 8 Minimal ■トライアル版の場合 • Almalinux 8 Minimal 8.8
前提プログラム	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • Java SE 8 の uCosminexus Application Runtime with Java for Spring Boot • curl • glibc • jq ■トライアル版の場合 • java-1.8.0-openjdk-headless • curl • glibc • jq
配置ファイル	次に示すファイルを配置してください。 • ファイル：HMP-PCTO の Mediator アプリケーションの JAR ファイル（mediator-V.R.S.jar） 配置パス：/deployments 下 オーナー・グループ：「1001:0」 パーミッション：「0440」

項目	要件
	• ファイル：HMP-PCTO の Mediator アプリケーションのダイレクト IO 用ライブラリファイル（libhmppcto-mediator-V.R.S.so） 配置パス：/deployments 下 オーナー・グループ：「1001:0」 パーミッション：「0440」 • ファイル：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 配置パス：/deployments 下 オーナー・グループ：「1001:0」 パーミッション：「0550」 • ファイル：HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh） 配置パス：/deployments 下 オーナー・グループ：「1001:0」 パーミッション：「0550」
コンテナ実行ユーザ・グループ	「1001:0」
コンテナ起動時カレントディレクトリ	「/deployments」
コンテナ起動コマンドライン	■通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト(starter.sh)」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「HMP-PCTO の Mediator アプリケーションの JAR ファイル(mediator-V.R.S.jar)」を指定してください。 ■トライアル版の場合 • コマンドを「java コマンド」とし、引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「HMP-PCTO の Mediator アプリケーションの JAR ファイル(mediator-V.R.S.jar)」を指定してください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します）。

注　V.R.S には、バージョンを設定します。

5.3.8　TP1-Bridge コンテナイメージ（通常版かつ TP1-Bridge 限定）

次の表に示す TP1-Bridge コンテナイメージの要件を満たすように作成します。

表 5-10　TP1-Bridge コンテナイメージの要件

項目	要件
コンテナベースイメージ	• uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（「5.3.1　uCosminexus Application Runtime with Java for Spring Boot コンテナベースイメージ（通常版限定）」を参照してください）
前提プログラム	• Hitachi Elastic Application Data Store Client for Java • ncompress • ncurses • tar • cpio • gzip
配置ファイル	次に示すファイルを配置してください。 • ファイル：HMP-PCTO の TP1-Bridge アプリケーションの JAR ファイル（tp1-bridge-V.R.S.jar） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。 • ファイル：Hitachi Elastic Application Data Store Client for Java の JAR ファイル 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの読み取り権限を付与してください。 • ファイル：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの実行権限を付与してください。 • ファイル：HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh） 配置パス：規定しません。 オーナー・グループ：規定しません。 パーミッション：コンテナ実行ユーザの実行権限を付与してください。
コンテナ実行ユーザ・グループ	規定しません。
コンテナ起動時カレントディレクトリ	規定しません。
コンテナ起動コマンドライン	• コマンドを「uCosminexus Application Runtime with Java for Spring Boot のプロセスモニタ起動スクリプト（starter.sh）」とし、引数に「java コマンド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「HMP-

項目	要件
	PCTO の TP1-Bridge アプリケーションの JAR ファイル（tp1-bridge-V.R.S.jar)」を指定してください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します)。
環境変数	次に示す環境変数を配置してください。 • 環境変数名：LOADER_PATH 設定値：Hitachi Elastic Application Data Store Client for Java の JAR ファイルを配置したパス
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します)。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します)。

注　V.R.S には、バージョンを設定します。

5.3.9　Dependency-Checker コンテナイメージ

このコンテナイメージは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 5-11　Dependency-Checker コンテナイメージの要件

項目	要件
コンテナベースイメージ	■通常版の場合 • Red Hat Universal Base Image 8 Minimal ■トライアル版の場合 • Almalinux 8 Minimal 8.8
前提プログラム	■通常版の場合 • curl • jq ■トライアル版の場合 • curl • jq
配置ファイル	次に示すファイルを配置してください。 • ファイル：HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh） 配置パス：/deployments 下 オーナー・グループ：「1001:0」 パーミッション：「0550」

項目	要件
コンテナ実行ユーザ・グループ	「1001:0」
コンテナ起動時カレントディレクトリ	「/deployments」
コンテナ起動コマンドライン	エントリポイントを「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」とします（引数については、このコンテナベースイメージを使用する各 Kubernetes マニフェスト側で指定します）。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します）。

5.3.10　Ext-Cons コンテナイメージ（トライアル版限定）

このコンテナイメージは、HMP-PCTO として提供します。トライアル版の場合は、次の表に示す要件を満たすように作成します。

表 5-12　Ext-Cons コンテナイメージの要件

項目	要件
コンテナベースイメージ	• Almalinux 8 Minimal 8.8
前提プログラム	• java-1.8.0-openjdk-headless
配置ファイル	次に示すファイルを配置してください。 • ファイル：HMP-PCTO の Ext-Cons アプリケーションの JAR ファイル（ext-cons-V.R.S.jar） 配置パス：/deployments 下 オーナー・グループ：「1001:0」 パーミッション：「0440」
コンテナ実行ユーザ・グループ	「1001:0」
コンテナ起動時カレントディレクトリ	「/deployments」
コンテナ起動コマンドライン	コマンドを「java コマンド」とし、引数に「HMP-PCTO の Ext-Cons アプリケーションの JAR ファイル（ext-cons-V.R.S.jar）」を指定してください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。

項目	要件
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェックは Kubernetes マニフェストで設定します）。

注　V.R.S には、バージョンを設定します。

5.3.11　EADS サーバ用コンテナイメージ（通常版限定）

このコンテナイメージは、HMP-PCTO として提供します。通常版の場合は、次の表に示す要件を満たすように作成します。

表 5-13　EADS コンテナイメージの要件

項目	要件
コンテナベースイメージ	• Red Hat Universal Base Image 8 Minimal
前提プログラム	• Hitachi Elastic Application Data Store • Hitachi Elastic Application Data Store のリリースノートの前提プログラムに従う
配置ファイル	次に示すファイルを配置してください。 • ファイル：/opt/hitachi/xeads/server/servers/template/ディレクトリと配下のファイルをコピーした node ディレクトリ※1 配置パス：/opt/hitachi/xeads/server/servers 下 オーナー・グループ：「0:0」 パーミッション："chmod -R g=u <node ディレクトリのパス>"コマンドを実行した状態とすること • ファイル：orchestration_eads_server_start.sh 配置パス：/opt/hitachi/xeads/server/servers/node/bin 下 オーナー・グループ：「0:0」 パーミッション：「0554」 • ファイル：「node<n※2>」という名称の/opt/hitachi/xeads/server/servers/node ディレクトリへのシンボリックリンク 配置パス：/opt/hitachi/xeads/server/servers 下 オーナー・グループ：「0:0」 パーミッション：「0777」
コンテナ実行ユーザ・グループ	「0:0」
コンテナ起動時カレントディレクトリ	/opt/hitachi/xeads/server/servers/node/

項目	要件
コンテナ起動コマンドライン	コマンドを「orchestration_eads_server_start.sh」としてください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェック機能がある場合は Kubernetes マニフェストで設定します）。

注※1

ディレクトリ配下のファイルのパーミッションは、EADS の仕様に従ってください。

注※2

n は 1 から始まる EADS サーバの ID を指定してください。

5.3.12 EADS コマンド実行用コンテナイメージ（通常版限定）

このコンテナイメージは、HMP-PCTO として提供します。通常版の場合は、次の表に示す要件を満たすように作成します。

表 5-14 EADS コンテナイメージの要件

項目	要件
コンテナベースイメージ	• Red Hat Universal Base Image 8 Minimal
前提プログラム	• kubectl
配置ファイル	次に示すファイルを配置してください。 • ファイル：orchestration_eads_command_start.sh 配置パス：/opt/hitachi/xeads/orchestration/container/command 下 オーナー・グループ：「0:0」 パーミッション：「0777」
コンテナ実行ユーザ・グループ	「0:0」
コンテナ起動時カレントディレクトリ	/opt/hitachi/xeads/orchestration/container/command
コンテナ起動コマンドライン	コマンドを「orchestration_eads_command_start.sh」としてください。
公開ポート	規定しません（HMP-PCTO で必要な公開ポートは Kubernetes マニフェストで設定します）。
環境変数	規定しません（HMP-PCTO で必要な環境変数は Kubernetes マニフェストで設定します）。

項目	要件
ボリューム	規定しません（HMP-PCTO で必要なボリュームは Kubernetes マニフェストで設定します）。
コンテナ終了シグナル	規定しません。
コンテナヘルスチェック	規定しません（HMP-PCTO で必要なヘルスチェック機能がある場合は Kubernetes マニフェストで設定します）。

6 Kubernetes クラスタの作成要件

6.1 Kubernetes アプリケーションのデプロイ方法

HMP-PCTO のコンテナイメージは、コンテナオーケストレーションツール上にデプロイして実行します。HMP-PCTO のコンテナイメージについては、「5. コンテナイメージの作成要件」を参照してください。

前提とする Kubernetes ディストリビューションおよびバージョンは、「1.4.1 前提 OS および前提プログラム」を参照してください。

HMP-PCTO の Kubernetes アプリケーションのデプロイ方法の一覧を次の表に示します。

表 6-1 HMP-PCTO の Kubernetes アプリケーションのデプロイ方法の一覧

Kubernetes アプリケーションの分類	デプロイ方法	ユーザへの提供物	ユーザの対応作業
ユーザ責務の Kubernetes アプリケーション	ユーザ任意の方法によるデプロイ	• Kubernetes マニフェストの記述例（「2.6 Kubernetes マニフェスト作成」参照）	1. Kubernetes マニフェストの記述例を基に Kubernetes マニフェストを作成する 2. Kubernetes マニフェストを使用し Kubernetes アプリケーションをデプロイする
HMP-PCTO のコントロールプレーンの Kubernetes アプリケーション	Helm によるデプロイ	• Helm チャート • Helm チャートのカスタマイズ例（「3.7 Helm チャートのカスタマイズ」参照）	1. Helm チャートのカスタマイズ例を基に Helm チャートをカスタマイズする 2. Helm チャートを使用し Kubernetes アプリケーションをデプロイする
RAS の Kubernetes アプリケーション			

6.2 Kubernetes クラスタ内外の通信暗号化方針

Kubernetes クラスタ内および Kubernetes クラスタ外の通信暗号化方針を次に示します。

- 通常版では Pod 間通信（HTTP、gRPC）を TLS（Transport Layer Security）プロトコルを使用して通信データを暗号化することで、通信データの漏洩や改ざんを防止する
- サーバ認証／クライアント認証（mTLS）によって、通信相手の妥当性を検証し、意図しない相手との通信を抑止する
- 暗号化有無はプロパティで切り替える
- 次の通信個所は、通信暗号化の対象外とする
  - Pod 内通信

    ・トランザクションヘルスチェックスクリプト/未決着トランザクション待機スクリプトと SQL-Participant/Mediator、TCC-Participant/Mediator、または TP1-Bridge/Mediator の間

    ・Entity-Service と SQL-Participant の間
  - SQL-Participant と DB の間
  - SQL-Participant、TCC-Participant、または TP1-Bridge と EADS の間
  - TP1-Bridge と TP1-Converter の間、および TP1-Bridge と OpenTP1 の間
  - TP1-Converter と REST API 発行元の間
  - Filebeat と Logstash の間
  - Prometheus と Metricbeat の間
  - Jaeger-collector と Orchestrator/Entity-service/SQL-Participant/TCC-Participant/TP1-Bridge/Mediator の間

6.3 Kubernetes アプリケーションのタイムゾーン設定方針

Kubernetes アプリケーションのタイムゾーン設定方針を次に示します。

- Kubernetes アプリケーションのタイムゾーンを設定可能とすることで、トラブルシュート時に参照するログの日時を確認しやすくする
- 次の個所は、タイムゾーン設定の対象とする
  - ユーザ責務の Kubernetes アプリケーション内の Java アプリケーションのコンテナ
  - HMP-PCTO のコントロールプレーンの Kubernetes アプリケーション内の Java アプリケーションのコンテナ
- 次の個所は、タイムゾーン設定の対象外とする
  - ユーザ責務の Kubernetes アプリケーション内の Java アプリケーション以外のコンテナ
  - HMP-PCTO のコントロールプレーンの Kubernetes アプリケーション内の Java アプリケーション以外のコンテナ
  - RAS の Kubernetes アプリケーション内のコンテナ
  - EADS のコンテナ
- タイムゾーンの設定は任意とする
- タイムゾーンの設定方法は、OS で用意されている環境変数 TZ とする

6.4　各 Kubernetes アプリケーションの作成要件

各 Kubernetes アプリケーションの作成要件を次に示します。

6.4.1　Orchestrator の要件

この Kubernetes アプリケーションは、次の表に示す Orchestrator の要件を満たすように作成します。

表 6-2　Orchestrator の要件

項目	要件
ワークロードリソース	規定しません。任意（Deployment など）とします。
ネームスペース	規定しません。
ラベル	規定しません。※1
アノテーション	規定しません。
初期化コンテナ構成	• Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	• Orchestrator コンテナ（「5.3.3　Orchestrator コンテナイメージ」を参照してください） • Filebeat コンテナ
レプリカ数	「1」
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	規定しません。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で、次に示すとおり対象リソースへのアクセス許可を定義してください。 ＜ Mediator 選択機能用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints • アクセス許可 get、list、watch 上記アクセス許可を定義するための Kubernetes リソースは ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用すること。 ＜デプロイ依存関係チェック機能用ロール＞ • API グループ

項目	要件
	空（コアグループ）、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list これらのアクセス許可を定義するための Kubernetes リソースは、次に示すとおり使い分けてください。 • Orchestrator と Orchestrator の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace がすべて同一の場合： Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 • Orchestrator と Orchestrator の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace が 1 つでも異なる場合： ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定）。
Pod の割り当てノードの制御	規定しません。
永続化ボリューム	• uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを割り当ててください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。
Pod ライフサイクル設定	規定しません。

注※1

Deployment リソースや StatefulSet リソースの場合は、Kubernetes のマニフェストの記述仕様に従って、.spec.selector と.spec.template.metadata.labels の紐づけのために最低 1 つ以上のラベルを定義する必要があります。Kubernetes マニフェストの記述仕様の詳細については、Kubernetes のマニュアルなどを参照してください。

注※2

Orchestrator の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「表 6-3 Orchestrator の Dependency-Checker コンテナの要件」を参照してください。

コンテナごとの要件を次に示します。

表 6-3　Orchestrator の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数）

項目	要件
	推奨値「10」 • 第３引数（接続タイムアウト秒数） 推奨値「10」 • 第４引数（応答タイムアウト秒数） 推奨値「10」 • 第５引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Participant 例を次に示します。 （例） SQL-Participant を使用する場合：my-namespace/deployments/sql-participant TCC-Participant を使用する場合：my-namespace/deployments/tcc-participant TP1-Bridge を使用する場合（TP1 連携機能を使用する場合）：my-namespace/deployments/tp1-bridge※ （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

注※

tp1-bridge の deployments には、「(15)　Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。

（例）

リリース名が tp1-bridge1 の場合：my-namespace/deployments/tp1-bridge1

表 6-4　Orchestrator コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 • ユーザ責務の Kubernetes アプリケーションで使用する各種ポート番号 • Prometheus メトリクス収集で使用する http ポート番号 プロトコル：TCP 公開ポート：Orchestrator の server.port パラメタに設定したポート番号 コンテナポート：Orchestrator の server.port パラメタに設定したポート番号

項目	要件
コンフィグ設定	次の HMP-PCTO のパラメタを、Spring Boot のコンフィグソース（application.properties、application.yml、環境変数など）に設定してください。 • 「8.2.1　共通」のパラメタ • 「8.2.2　Orchestrator」のパラメタ
	HMP-PCTO の gRPC 通信の暗号化を有効にする場合、取扱説明書「HMP-PCTO テクニカルガイド」の各プロセスの認証種別ごとに必要となるファイルの一覧に関する説明に示すとおり暗号化通信で必要となるファイルをコンテナ内にマウントしてください（Secret リソースなど）。
	タイムゾーンの設定は任意です。
	環境変数 HMPPCTO_JAVA_OPTIONS で Orchestrator の Java プロセス起動時に Java VM に渡すオプションを指定してください。
	uCosminexus Application Runtime with Java for Spring Boot の設定ファイル（config.properties）をコンテナ内にマウントしてください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。 次のとおりファイルを配置してください。 • ファイル：uCosminexus Application Runtime with Java for Spring Boot の設定ファイル キー名：config.properties マウント先：/opt/hitachi/ucars/conf/config.properties
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	規定しません。
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

表 6-5　Orchestrator の Filebeat コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数 「-e」 標準エラー出力にログを記録します。syslog/file へのログ出力を無効化します。 • 第 2 引数 「-E」 このパラメタ以降に記載された設定を反映した状態で起動します。 • 第 3 引数（http.enabled） 「true」 HTTP エンドポイントを有効にします。死活監視のために設定を有効化します。
ネットワーク設定	規定しません。

項目	要件
コンフィグ設定	filebeat.yml を「3.9.1　ログおよびトラブルシュート情報の取得の設定」のとおりに設定してください。
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」
CPU/メモリリソース制限	規定しません。

6.4.2　Entity-Service（Participant コンテナを含まない Pod とする場合）の要件

この Kubernetes マニフェストは、サービス間を中継するサービスとしての Entity-Service がある場合だけ作成します。詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のトランザクションのモデルに関する説明を参照してください。

この Kubernetes アプリケーションは、次の表に示す Entity-Service（Participant コンテナを含まない Pod とする場合）の要件を満たすように作成します。

表 6-6　Entity-Service（Participant コンテナを含まない Pod とする場合）の要件

項目	要件
ワークロードリソース	規定しません。任意（Deployment など）とします。
ネームスペース	規定しません。
ラベル	規定しません。※1

項目	要件
アノテーション	規定しません。
初期化コンテナ構成	• Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	• Entity-Service コンテナ（「5.3.4　Entity-Service コンテナイメージ」を参照してください）（SQL-Participant を使用する場合） • Filebeat コンテナ
レプリカ数	規定しません。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	規定しません。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で次のとおり対象リソースへのアクセス許可を定義してください。 ＜ Mediator 選択機能用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints • アクセス許可 get、list、watch 上記アクセス許可を定義するための Kubernetes リソースは ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用すること。 ＜デプロイ依存関係チェック機能用ロール＞ • API グループ 空（コアグループ）、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list これらのアクセス許可を定義するための Kubernetes リソースは、次に示すとおり使い分けてください。 • Entity-Service と Entity-Service の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace がすべて同一の場合： Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 • Entity-Service と Entity-Service の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace が 1 つでも異なる場合：

項目	要件
	ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定）。
Pod の割り当てノードの制御	規定しません。
永続化ボリューム	uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを割り当ててください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「330」

注※1

Deployment リソースや StatefulSet リソースの場合は、Kubernetes のマニフェストの記述仕様に従って、.spec.selector と.spec.template.metadata.labels の紐づけのために最低 1 つ以上のラベルを定義する必要があります。Kubernetes マニフェストの記述仕様の詳細については、Kubernetes のマニュアルなどを参照してください。

注※2

Entity-Service の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「表 6-7 Entity-Service（Participant コンテナを含まない Pod とする場合）の Dependency-Checker コンテナの要件」を参照してください。

コンテナごとの要件を次に示します。

表 6-7 Entity-Service（Participant コンテナを含まない Pod とする場合）の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Mediator、Participant 例を次に示します。 （例） SQL-Participant を使用する場合：

項目	要件
	my-namespace/statefulsets/mediator my-namespace/deployments/sql-participant TCC-Participant を使用する場合： my-namespace/statefulsets/mediator my-namespace/deployments/tcc-participant TP1-Bridge を使用する場合（TP1 連携機能を使用する場合）： my-namespace/statefulsets/mediator my-namespace/deployments/tp1-bridge※ （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

注※

tp1-bridge の deployments には、「(15)　Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。

（例）

リリース名が tp1-bridge1 の場合：my-namespace/deployments/tp1-bridge1

表 6-8　Entity-Service（Participant コンテナを含まない Pod とする場合）コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 • ユーザ責務の Kubernetes アプリケーションで使用する各種ポート番号 • Prometheus メトリクス収集で使用する http ポート番号 プロトコル：TCP 公開ポート：Entity-Service の server.port パラメタに設定したポート番号 コンテナポート：Entity-Service の server.port パラメタに設定したポート番号
コンフィグ設定	次の HMP-PCTO のパラメタを、Spring Boot のコンフィグソース（application.properties、application.yml、環境変数など）に設定してください。 •「8.2.1　共通」のパラメタ •「8.2.3　Entity-Service/Entity-Module」のパラメタ
	タイムゾーンの設定は任意です。

項目	要件
	環境変数 HMPPCTO_JAVA_OPTIONS で Entity-Service の Java プロセス起動時に Java VM に渡すオプションを指定してください。
	uCosminexus Application Runtime with Java for Spring Boot の設定ファイル（config.properties）をコンテナ内にマウントしてください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。 次のとおりファイルを配置してください。 • ファイル：uCosminexus Application Runtime with Java for Spring Boot の設定ファイル キー名：config.properties マウント先：/opt/hitachi/ucars/conf/config.properties
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	規定しません。
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

表 6–9　Entity-Service（Participant コンテナを含まない Pod とする場合）の Filebeat コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数 「-e」 標準エラー出力にログを記録します。syslog/file へのログ出力を無効化します。 • 第 2 引数 「-E」 このパラメタ以降に記載された設定を反映した状態で起動します。 • 第 3 引数（http.enabled） 「true」 HTTP エンドポイントを有効にします。死活監視のために設定を有効化します。
ネットワーク設定	規定しません。
コンフィグ設定	filebeat.yml を「3.9.1　ログおよびトラブルシュート情報の取得の設定」のとおりに設定してください。
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」

項目	要件
	• Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」
CPU/メモリリソース制限	規定しません。

6.4.3 SQL-Participant の要件

この Kubernetes アプリケーションは、次の表に示す SQL-Participant の要件を満たすように作成します。

表 6–10 SQL-Participant の要件

項目	要件
ワークロードリソース	規定しません。任意（Deployment など）とします。
ネームスペース	SQL-Participant を構成する Pod を定義した Kubernetes の Service の Service 名(.metadata.name フィールド)とネームスペース名(.metadata.namespace フィールド)を「.」でつないだ文字列が、41 バイト以内の文字列となるように、ネームスペース名を検討してください。
ラベル	規定しません。※1
アノテーション	規定しません。
初期化コンテナ構成	Dependency-Checker コンテナ（「5.3.9 Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	Entity-Service コンテナ（「5.3.4 Entity-Service コンテナイメージ」を参照してください） SQL-Participant コンテナ（「5.3.5 SQL-Participant コンテナイメージ」を参照してください） Filebeat コンテナ
レプリカ数	規定しません。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。

項目	要件
アップデートストラテジー	規定しません。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で次のとおり対象リソースへのアクセス許可を定義してください。 ＜ Mediator 選択機能用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints • アクセス許可 get、list、watch 上記アクセス許可を定義するための Kubernetes リソースは ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用すること。 ＜デプロイ依存関係チェック機能用ロール＞ • API グループ 空（コアグループ）、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list これらのアクセス許可を定義するための Kubernetes リソースは、次に示すとおり使い分けてください。 • Participant と Participant の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace がすべて同一の場合： Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 • Participant と Participant の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace が 1 つでも異なる場合： ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定）。
Pod の割り当てノードの制御	規定しません。
永続化ボリューム	• uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを割り当ててください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。

項目	要件
	• 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「330」

注※1

Deployment リソースや StatefulSet リソースの場合は、Kubernetes のマニフェストの記述仕様に従って、.spec.selector と.spec.template.metadata.labels の紐づけのために最低 1 つ以上のラベルを定義する必要があります。Kubernetes マニフェストの記述仕様の詳細については、Kubernetes のマニュアルなどを参照してください。

注※2

Participant の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「表 6–11 SQL-Participant の Dependency-Checker コンテナの要件」を参照してください。

コンテナごとの要件を次に示します。

表 6–11　SQL-Participant の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Mediator、EADS、Ext-Cons（トライアル版の場合だけ）、Participant が使用する DB/外部サービス 例を次に示します。 （例） my-namespace/statefulsets/mediator my-namespace/statefulsets/eads-server my-namespace/deployments/ext-cons（トライアル版の場合だけ） my-namespac /statefulsets/postgresql （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。

項目	要件
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-12　Entity-Service コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 • ユーザ責務の Kubernetes アプリケーションで使用する各種ポート番号 • Prometheus メトリクス収集で使用する http ポート番号 プロトコル：TCP 公開ポート：Entity-Service の server.port パラメタに設定したポート番号 コンテナポート：Entity-Service の server.port パラメタに設定したポート番号
コンフィグ設定	次の HMP-PCTO のパラメタを、Spring Boot のコンフィグソース（application.properties、application.yml、環境変数など）に設定してください。 •「8.2.1　共通」のパラメタ •「8.2.3　Entity-Service/Entity-Module」のパラメタ
	タイムゾーンの設定は任意です。
	環境変数 HMPPCTO_JAVA_OPTIONS で Entity-Service の Java プロセス起動時に Java VM に渡すオプションを指定してください。
	uCosminexus Application Runtime with Java for Spring Boot の設定ファイル（config.properties）をコンテナ内にマウントしてください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。 次のとおりファイルを配置してください。 • ファイル：uCosminexus Application Runtime with Java for Spring Boot の設定ファイル キー名：config.properties マウント先：/opt/hitachi/ucars/conf/config.properties
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	規定しません。
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

表 6-13　SQL-Participant コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	SQL-Participant:を構成する Pod を定義した Kubernetes の Service の Service 名(.metadata.name フィールド)とネームスペース名

項目	要件
	(.metadata.namespace フィールド)を「.」でつないだ文字列が、41 バイト以内の文字列となるように、Service の Service 名を検討してください。
	Kubernetes クラスタ内で次のポートを公開してください（Service リソース)。 • SQL-Participant の gRPC ポート番号（Mediator 向け） プロトコル：TCP 公開ポート：SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.mediator.port パラメタに設定したポート番号 コンテナポート：SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.mediator.port パラメタに設定したポート番号 • SQL-Participant の http ポート番号（ヘルスチェック、未決着トランザクション確認、Prometheus メトリクス収集用） プロトコル：TCP 公開ポート：SQL-Participant の server.port パラメタに設定したポート番号 コンテナポート：SQL-Participant の server.port パラメタに設定したポート番号
コンフィグ設定	次の HMP-PCTO のパラメタを、Spring Boot のコンフィグソース（application.properties、application.yml、環境変数など）に設定してください。 •「8.2.1　共通」のパラメタ •「8.2.4　SQL-Participant」のパラメタ
	HMP-PCTO の gRPC 通信の暗号化を有効にする場合、取扱説明書「HMP-PCTO テクニカルガイド」の各プロセスの認証種別ごとに必要となるファイルの一覧に関する説明に示すとおり暗号化通信で必要となるファイルをコンテナ内にマウントしてください（Secret リソースなど)。
	タイムゾーンの設定は任意です。
	環境変数 HMPPCTO_JAVA_OPTIONS で SQL-Participant の Java プロセス起動時に Java VM に渡すオプションを指定してください。
	uCosminexus Application Runtime with Java for Spring Boot の設定ファイル（config.properties）をコンテナ内にマウントしてください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ)。 次のとおりファイルを配置してください。 • ファイル：uCosminexus Application Runtime with Java for Spring Boot の設定ファイル キー名：config.properties マウント先：/opt/hitachi/ucars/conf/config.properties
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • コマンド実行

項目	要件
	パス：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 第 1 引数（リトライ回数）：「0」固定（Liveness Probe の設定でリトライするため） 第 2 引数（リトライ間隔秒数）：「0」固定（Liveness Probe の設定でリトライするため） 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：SQL-Participant の server.port パラメタに設定したポート番号 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「180」 • Probe 実行間隔秒数（periodSeconds） 推奨値「30」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「10」 postStart フックを次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 第 1 引数（リトライ回数）：推奨値「30」 第 2 引数（リトライ間隔秒数）：推奨値「10」 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：SQL-Participant の server.port パラメタに設定したポート番号 preStop フックを次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh） 第 1 引数（リトライ回数）：推奨値「30」 第 2 引数（リトライ間隔秒数）：推奨値「10」 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：SQL-Participant の server.port パラメタに設定したポート番号
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

表 6-14　SQL-Participant の Filebeat コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数 「-e」 標準エラー出力にログを記録します。syslog/file へのログ出力を無効化します。 • 第 2 引数 「-E」 このパラメタ以降に記載された設定を反映した状態で起動します。 • 第 3 引数（http.enabled） 「true」 HTTP エンドポイントを有効にします。死活監視のために設定を有効化します。
ネットワーク設定	規定しません。
コンフィグ設定	filebeat.yml を「3.9.1　ログおよびトラブルシュート情報の取得の設定」のとおりに設定してください。
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」
CPU/メモリリソース制限	規定しません。

6.4.4　TCC-Participant の要件

この Kubernetes アプリケーションは、次の表に示す TCC-Participant の要件を満たすように作成します。

表 6-15　TCC-Participant の要件

項目	要件
ワークロードリソース	規定しません。任意（Deployment など）とします。
ネームスペース	TCC-Participant を構成する Pod を定義した Kubernetes の Service の Service 名(.metadata.name フィールド)とネームスペース名(.metadata.namespace フィールド)を「.」でつないだ文字列が、41 バイト以内の文字列となるように、ネームスペース名を検討してください。
ラベル	規定しません。※1
アノテーション	規定しません。
初期化コンテナ構成	Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	Entity-Module/TCC-Participant コンテナ（「5.3.6　Entity-Module/TCC-Participant コンテナイメージ」を参照してください） Filebeat コンテナ
レプリカ数	規定しません。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してくださいと（Kubernetes のデフォルト設定）。
アップデートストラテジー	規定しません。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で次のとおり対象リソースへのアクセス許可を定義してください。 ＜ Mediator 選択機能用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints • アクセス許可 get、list、watch 上記アクセス許可を定義するための Kubernetes リソースは ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用すること。 ＜デプロイ依存関係チェック機能用ロール＞ • API グループ 空（コアグループ）、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可

項目	要件
	get、list これらのアクセス許可を定義するための Kubernetes リソースは、次に示すとおり使い分けてください。 • Participant と Participant の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace がすべて同一の場合： Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 • Participant と Participant の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace が 1 つでも異なる場合： ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定）。
Pod の割り当てノードの制御	規定しません。
永続化ボリューム	• uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを割り当ててください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「330」

注※1

Deployment リソースや StatefulSet リソースの場合は、Kubernetes のマニフェストの記述仕様に従って、.spec.selector と.spec.template.metadata.labels の紐づけのために最低 1 つ以上のラベルを定義する必要があります。Kubernetes マニフェストの記述仕様の詳細については、Kubernetes のマニュアルなどを参照してください。

注※2

Participant の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「表 6-16 TCC-Participant の Dependency-Checker コンテナの要件」を参照してください。

コンテナごとの要件を次に示します。

表 6-16　TCC-Participant の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数）

項目	要件
	推奨値「10」 • 第４引数（応答タイムアウト秒数） 推奨値「10」 • 第５引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Mediator、EADS、Ext-Cons（トライアル版の場合だけ） 例を次に示します。 （例） my-namespace/statefulsets/mediator my-namespace/statefulsets/eads-server my-namespace/deployments/ext-cons（トライアル版の場合だけ） （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-17　Entity-Module/TCC-Participant コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	TCC-Participant を構成する Pod を定義した Kubernetes の Service の Service 名(.metadata.name フィールド)とネームスペース名(.metadata.namespace フィールド)を「.」でつないだ文字列が、41 バイト以内の文字列となるように、Service の Service 名を検討してください。
	Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 • ユーザ責務の Kubernetes アプリケーションで使用する各種ポート番号 • Prometheus メトリクス収集で使用する http ポート番号 プロトコル：TCP 公開ポート：Entity-Module の server.port パラメタに設定したポート番号 コンテナポート：Entity-Module の server.port パラメタに設定したポート番号 • TCC-Participant の gRPC ポート番号（Mediator 向け） プロトコル：TCP 公開ポート：TCC-Participant の jp.co.Hitachi.soft.hmppcto.tcc.mediator.port パラメタに設定したポート番号

項目	要件
	コンテナポート：TCC-Participant の jp.co.Hitachi.soft.hmppcto.tcc.mediator.port パラメタに設定したポート番号
コンフィグ設定	次の HMP-PCTO のパラメタを、Spring Boot のコンフィグソース（application.properties、application.yml、環境変数など）に設定してください。 • 「8.2.1　共通」のパラメタ • 「8.2.3　Entity-Service/Entity-Module」のパラメタ • 「8.2.5　TCC-Participant」のパラメタ
	HMP-PCTO の gRPC 通信の暗号化を有効にする場合、取扱説明書「HMP-PCTO テクニカルガイド」の各プロセスの認証種別ごとに必要となるファイルの一覧に関する説明に示すとおり暗号化通信で必要となるファイルをコンテナ内にマウントしてください（Secret リソースなど）。
	タイムゾーンの設定は任意です。
	環境変数 HMPPCTO_JAVA_OPTIONS で Entity-Module/TCC-Participant の Java プロセス起動時に Java VM に渡すオプションを指定してください。
	uCosminexus Application Runtime with Java for Spring Boot の設定ファイル（config.properties）をコンテナ内にマウントしてください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。 次のとおりファイルを配置してください。 • ファイル：uCosminexus Application Runtime with Java for Spring Boot の設定ファイル キー名：config.properties マウント先：/opt/hitachi/ucars/conf/config.properties
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 第 1 引数（リトライ回数）：「0」固定（Liveness Probe の設定でリトライするため） 第 2 引数（リトライ間隔秒数）：「0」固定（Liveness Probe の設定でリトライするため） 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：TCC-Participant の server.port パラメタに設定したポート番号 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「180」 • Probe 実行間隔秒数（periodSeconds） 推奨値「30」

項目	要件
	• Probe 最大失敗回数（failureThreshold） 推奨値「3」 Probe タイムアウト秒数（timeoutSeconds） 推奨値「10」 postStart フックを次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 第 1 引数（リトライ回数）：推奨値「30」 第 2 引数（リトライ間隔秒数）：推奨値「10」 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：TCC-Participant の server.port パラメタに設定したポート番号 preStop フックを次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO の「未決着トランザクション待機スクリプト（wait_transactions.sh） 第 1 引数（リトライ回数）：推奨値「30」 第 2 引数（リトライ間隔秒数）：推奨値「10」 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：TCC-Participant の server.port パラメタに設定したポート番号
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

表 6–18　TCC-Participant の Filebeat コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数 「-e」 標準エラー出力にログを記録します。syslog/file へのログ出力を無効化します。 • 第 2 引数 「-E」 このパラメタ以降に記載された設定を反映した状態で起動します。 • 第 3 引数（http.enabled） 「true」 HTTP エンドポイントを有効にします。死活監視のために設定を有効化します。
ネットワーク設定	規定しません。
コンフィグ設定	filebeat.yml を「3.9.1　ログおよびトラブルシュート情報の取得の設定」のとおりに設定してください。

項目	要件
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」
CPU/メモリリソース制限	規定しません。

6.4.5 Mediator の要件

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6-19 Mediator の要件

項目	要件
ワークロードリソース	StatefulSet リソースとしてください。
ネームスペース	規定しません。
ラベル	metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：mediator 備考：チャート名 • キー名：app.kubernetes.io/version

項目	要件
	値：*V.R.S*※2 備考：バージョン • キー名：helm.sh/chart 値：mediator-*V.R.S*※2 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> 備考： • キー名：app.kubernetes.io/name 値：mediator 備考：チャート名
アノテーション	設定しません（デフォルト）。
初期化コンテナ構成	• Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	• Mediator コンテナ（「5.3.7　Mediator コンテナイメージ」を参照してください）
レプリカ数	• ThreeQuaters 合意方式の場合：5～64 • PrivilegedValue 合意方式の場合：2～64
サービス名 （StatefulSet リソース特有）	「mediator」
Pod の管理ポリシー （StatefulSet リソース特有）	レプリカ数の Pod の起動/停止を並行に実施してください。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	ローリングアップデートとしてください（Kubernetes のデフォルト設定）。
Pod セキュリティコンテキスト	設定しません（デフォルト）。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で次のとおり対象リソースへのアクセス許可を定義してください。 <デプロイ依存関係チェック機能用ロール> • API グループ 空（コアグループ）、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list これらのアクセス許可を定義するための Kubernetes リソースは、次に示すとおり使い分けてください。

項目	要件
	• Mediator と Mediator の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※1 の Namespace がすべて同一の場合： Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 • Mediator と Mediator の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※1 の Namespace が 1 つでも異なる場合： ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効（true)」にしてください（Kubernetes のデフォルト設定)。
Pod の割り当てノードの制御	• 1 つのノードに割り当てられる Pod 数は 1 つまでです（例として、ワーカーノード数「5」に対してレプリカ数「5」のとき、各ノードに Pod が 1 つ起動します)。 • 停止した Pod を次に起動する際は前回起動したノードと必ず同一のノードに割り当ててください。該当するノードが障害から復旧していない場合に割り当て不可のときは、ほかの空いているノードが存在していたとしても Pod を割り当てないでください（フェイルオーバーしないでください)。
永続化ボリューム	Mediator の ConsensusLog の永続ボリュームを割り当ててください。コンテナ内の「/volume」に、永続ボリュームをマウントします。 ＜割り当て先をローカル永続ボリューム（local）とする場合＞ • 1 つの Pod につき割り当てる永続化ボリュームは 1 つとしてください（共有しないでください)。 • 永続ボリュームの種類を「ローカル永続ボリューム（local)」（ノードにアタッチされたローカルストレージデバイス）としてください。 • ブロックデバイスとしてください。 ＜割り当て先を NFS ストレージとする場合とする場合＞ • 永続ボリュームは、Mediator が存在するすべての Pod からアクセスできるようにしてください（共有しないでください)。 • 永続ボリュームは、NFS 上に作成してください。 • NFS サーバ側の条件として、ダイレクト I/O（O_DIRECT）に対応している必要があります。ダイレクト I/O（O_DIRECT）に対応しているサービスは、Azure Blob Storag、Amazon EFS、Amazon EBS です。
	uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを割り当ててください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ)。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「330」

注※1

Orchestrator の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「表 6–20 Mediator の Dependency-Checker コンテナの要件」を参照してください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

＜制限事項＞

スケーリングは不可とします（手動/自動、スケールアウト/インに関係なく）。

コンテナごとの要件を次に示します。

表 6-20　Mediator の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Logstash、Jaeger-query、Jaeger-collector、Prometheus、Metricbeat、Filebeat 例を次に示します。 （例） my-namespace/deployments/logstash my-namespace/deployments/jaeger-query my-namespace/deployments/jaeger-collector my-namespace/deployments/prometheus my-namespace/deployments/metricbeat my-namespace/daemonsets/filebeat （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-21　Mediator コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	Kubernetes クラスタ内の Headless Service（ClusterIP: None）で次のポートを公開してください（Service リソース）。 • Mediator の gRPC ポート番号（Participant,Orchestrator 向け） プロトコル：TCP 公開ポート：Mediator の jp.co.Hitachi.soft.hmppcto.grpc.port パラメタに設定したポート番号 コンテナポート：Mediator の jp.co.Hitachi.soft.hmppcto.grpc.port パラメタに設定したポート番号 • Prometheus メトリクス収集で使用する http ポート番号 プロトコル：TCP 公開ポート：Mediator の server.port パラメタに設定したポート番号 コンテナポート：Mediator の server.port パラメタに設定したポート番号
コンフィグ設定	次の HMP-PCTO のパラメタを、Spring Boot のコンフィグソースに設定してください。 •「8.2.1　共通」のパラメタ •「8.3.1　Mediator」のパラメタ 次のとおりファイルを配置します。 • ファイル：Spring Boot のコンフィグソース キー名：application.properties マウント先：/deployments/config/application.properties
	HMP-PCTO の gRPC 通信の暗号化を有効にする場合（Helm チャートの grpc.useTls が true）、取扱説明書「HMP-PCTO テクニカルガイド」の各プロセスの認証種別ごとに必要となるファイルの一覧に関する説明に示すとおり gRPC 通信機能の暗号化通信で必要となるファイルをコンテナ内にマウントしてください。 次のとおりファイルを配置します。 • ファイル：接続先サーバのサーバ証明書を発行した CA の証明書（PEM 形式）ファイル キー名：ca.crt マウント先：/cert/ca.crt • ファイル：自サーバ証明書（PEM 形式）ファイル キー名：server.crt マウント先：/cert/server.crt • ファイル：自サーバ証明書を作成した秘密鍵ファイル キー名：server.key マウント先：/cert/server.key さらに、クライアント認証を行う場合（Helm チャートの grpc.tls.clientAuth.enabled が true）は、次のとおりファイルを配置します。 • ファイル：自クライアント証明書ファイル（PEM 形式） キー名：client.crt

項目	要件
	マウント先：/cert/client.crt • ファイル：自クライアント証明書を作成した秘密鍵ファイル キー名：client.key マウント先：/cert/client.key • ファイル：接続元クライアント（orchestrator,participant）の秘密鍵が格納されているキーストア（pkcs12 または JKS）ファイル キー名：keystore マウント先：/cert/keystore • ファイル：キーストアファイルのパスワード キー名：keystore_password マウント先：application.properties の jp.co.Hitachi.soft.hmppcto.grpc.tls.server.client-auth.password パラメタの値に設定 「Mediator の gRPC 通信機能の暗号化通信用シークレット」をユーザが作成します（「(3)　TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成」を参照してください）。 ユーザが作成した「Mediator の gRPC 通信機能の暗号化通信用シークレット」を StatefulSet から参照する際、Helm チャートの grpc.tls.secretName パラメタに指定されている Secret オブジェクト名で参照してください。
	環境変数 TZ でタイムゾーンを設定します。ユーザ責務の Kubernetes アプリケーションと同じタイムゾーンに設定することを推奨します。デフォルト値は日本標準時（Asia/Tokyo、UTC+09:00）です。
	Helm チャートの timezone パラメタで、Mediator コンテナのタイムゾーンを指定します。
	Helm チャートの java.options パラメタで Mediator の Java プロセス起動時に Java VM に渡すオプションを指定してください。
	uCosminexus Application Runtime with Java for Spring Boot の設定ファイル（config.properties）をコンテナ内にマウントしてください（通常版（uCosminexus Application Runtime with Java for Spring Boot 導入あり）の場合だけ）。 次のとおりファイルを配置してください。 • ファイル：uCosminexus Application Runtime with Java for Spring Boot の設定ファイル キー名：config.properties マウント先：/opt/hitachi/ucars/conf/config.properties
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 第 1 引数（リトライ回数）：「0」固定（Liveness Probe の設定でリトライするため）

項目	要件
	第 2 引数（リトライ間隔秒数）：「0」固定（Liveness Probe の設定でリトライするため） 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：Mediator の server.port パラメタに設定したポート番号 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「180」 • Probe 実行間隔秒数（periodSeconds） 推奨値「30」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「10」 postStart フックを次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 第 1 引数（リトライ回数）：推奨値「30」 第 2 引数（リトライ間隔秒数）：推奨値「10」 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：Mediator の server.port パラメタに設定したポート番号 preStop フックを次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh） 第 1 引数（リトライ回数）：推奨値「30」 第 2 引数（リトライ間隔秒数）：推奨値「10」 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：Mediator の server.port パラメタに設定したポート番号
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-22　Mediator の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル	説明
templates/	Helm チャートの template 配置用ディレクトリ

ディレクトリ/ファイル			説明
	includes/		Helm チャートの ConfigMap オブジェクトの設定内容外部化ファイル配置用ディレクトリ
		application.properties.tpl	Mediator の ConfigMap オブジェクトの設定内容外部化ファイル
		mediator-config-map-data.yaml	Mediator の ConfigMap オブジェクトの設定内容外部化ファイル
		config.properties.tpl	uCosminexus Application Runtime with Java for Spring Boot の ConfigMap オブジェクトの設定内容外部化ファイル
		ucars-config-map-data.yaml	uCosminexus Application Runtime with Java for Spring Boot の ConfigMap オブジェクトの設定内容外部化ファイル
	mediator-config-map.yaml		Mediator の ConfigMap オブジェクトのマニフェスト
	ucars-config-map.yaml		uCosminexus Application Runtime with Java for Spring Boot の ConfigMap オブジェクトのマニフェスト
	service.yaml		Service オブジェクトのマニフェスト
	serviceaccount.yaml		ServiceAccount オブジェクトのマニフェスト
	statefulset.yaml		StatefulSet オブジェクトのマニフェスト
Chart.yaml			Chart.yaml
values.yaml			values.yaml
values.schema.json			values.yaml の設定値のバリデーション用ファイル

6.4.6 TP1-Bridge の要件（通常版かつ TP1-Bridge 限定）

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6–23 TP1-Bridge の要件

項目	要件
ワークロードリソース	Deployment リソースとしてください。
ネームスペース	規定しません。
ラベル	metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：<リリース名> • キー名：app.kubernetes.io/version

項目	要件
	値：*V.R.S*※1 備考：バージョン • キー名：helm.sh/chart 値：tp1-bridge-*V.R.S*※1 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/name 値：<リリース名>
アノテーション	設定しません（デフォルト）。
初期化コンテナ構成	• Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	• TP1-Bridge コンテナ（「5.3.8　TP1-Bridge コンテナイメージ（通常版かつ TP1-Bridge 限定）」を参照してください） • Filebeat コンテナ
レプリカ数	「1」としてください。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	規定しません。
Pod セキュリティコンテキスト	設定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で次のとおり対象リソースへのアクセス許可を定義してください。 ＜ Mediator 選択機能用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints • アクセス許可 get、list、watch これらのアクセス許可を定義するための Kubernetes リソースは、ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。 ＜デプロイ依存関係チェック機能用ロール＞ • API グループ 空（コアグループ）、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可

項目	要件
	get、list これらのアクセス許可を定義するための Kubernetes リソースは、次に示すとおり使い分けてください。 • TP1-Bridge と TP1-Bridge の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace がすべて同一の場合： Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 • TP1-Bridge と TP1-Bridge の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群※2 の Namespace が 1 つでも異なる場合： ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効（true)」にしてください（Kubernetes のデフォルト設定)。
Pod の割り当てノードの制御	規定しません。
永続化ボリューム	uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームを割り当ててください。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「330」

注※1

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

注※2

TP1-Bridge の Dependency-Checker コンテナで指定するターゲット Kubernetes オブジェクト群については、「表 6–24 TP1-Bridge の Dependency-Checker コンテナの要件」を参照してください。

＜制限事項＞

スケーリングは不可とします（手動/自動、スケールアウト/インに関係なく)。

コンテナごとの要件を次に示します。

表 6–24　TP1-Bridge の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数）

項目	要件
	推奨値「10」 • 第４引数（応答タイムアウト秒数） 推奨値「10」 • 第５引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Mediator、EADS 例を次に示します。 （例） my-namespace/statefulsets/mediator my-namespace/statefulsets/eads-server （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-25　TP1-Bridge コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 • TP1-Bridge の gRPC ポート番号（Mediator 向け） プロトコル：TCP 公開ポート：TP1-Bridge の jp.co.Hitachi.soft.hmppcto.tp1.mediator.port パラメタに設定したポート番号 コンテナポート：TP1-Bridge の jp.co.Hitachi.soft.hmppcto.tp1.mediator.port パラメタに設定したポート番号 • TP1-Bridge の http ポート番号（ヘルスチェック、未決着トランザクション確認、Prometheus メトリクス収集、および AP からの OpenTP1 のサービス呼び出し要求の受信用） プロトコル：TCP 公開ポート：TP1-Bridge の server.port パラメタに設定したポート番号 コンテナポート：TP1-Bridge の server.port パラメタに設定したポート番号
コンフィグ設定	次の HMP-PCTO のパラメタを、Spring Boot のコンフィグソースに設定してください。 •「8.2.1　共通」のパラメタ •「8.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）」のパラメタ

項目	要件
	次のとおりファイルを配置します。 • ファイル：Spring Boot のコンフィグソース キー名：application.properties マウント先：/deployments/config/application.properties
	HMP-PCTO の gRPC 通信の暗号化を有効にする場合（Helm チャートの grpc.useTls が true）、取扱説明書「HMP-PCTO テクニカルガイド」の各プロセスの認証種別ごとに必要となるファイルの一覧に関する説明に示すとおり、gRPC 通信機能の暗号化通信で必要となるファイルをコンテナ内にマウントしてください。 次のとおりファイルを配置します。 • ファイル：接続先サーバのサーバ証明書を発行した CA の証明書（PEM 形式）ファイル キー名：ca.crt マウント先：/cert/ca.crt • ファイル：自サーバ証明書（PEM 形式）ファイル キー名：server.crt マウント先：/cert/server.crt • ファイル：自サーバ証明書を作成した秘密鍵ファイル キー名：server.key マウント先：/cert/server.key さらに、クライアント認証を行う場合（Helm チャートの grpc.tls.clientAuth.enabled が true）は、次のとおりファイルを配置します。 • ファイル：自クライアント証明書ファイル（PEM 形式） キー名：client.crt マウント先：/cert/client.crt • ファイル：自クライアント証明書を作成した秘密鍵ファイル キー名：client.key マウント先：/cert/client.key • ファイル：接続元クライアント（orchestrator,participant）の秘密鍵が格納されているキーストア（pkcs12 または JKS）ファイル キー名：keystore マウント先：/cert/keystore • ファイル：キーストアファイルのパスワード キー名：keystore_password マウント先：application.properties の jp.co.Hitachi.soft.hmppcto.grpc.tls.server.client-auth.password パラメタの値に設定 「TP1-Bridge の gRPC 通信機能の暗号化通信用シークレット」をユーザが作成します（「(3)　TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成」を参照してください）。 ユーザが作成した「TP1-Bridge の gRPC 通信機能の暗号化通信用シークレット」を Deployments から参照する際、Helm チャートの grpc.tls.secretName パラメタに指定されている Secret オブジェクト名で参照してください。

項目	要件
	環境変数 TZ でタイムゾーンを設定します。ユーザ責務の Kubernetes アプリケーションと同じタイムゾーンに設定することを推奨します。デフォルト値は日本標準時（Asia/Tokyo、UTC+09:00）です。
	Helm チャートの timezone パラメタで、TP1-Bridge コンテナのタイムゾーンを指定します。
	Helm チャートの java.options パラメタで、TP1-Bridge の Java プロセス起動時に Java VM に渡すオプションを指定してください。
	uCosminexus Application Runtime with Java for Spring Boot の設定ファイル（config.properties）をコンテナ内にマウントしてください。 次のとおりファイルを配置してください。 • ファイル：uCosminexus Application Runtime with Java for Spring Boot の設定ファイル キー名：config.properties マウント先：/opt/hitachi/ucars/conf/config.properties
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 第 1 引数（リトライ回数）：「0」固定（Liveness Probe の設定でリトライするため） 第 2 引数（リトライ間隔秒数）：「0」固定（Liveness Probe の設定でリトライするため） 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：TP1-Bridge の server.port パラメタに設定したポート番号 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「180」 • Probe 実行間隔秒数（periodSeconds） 推奨値「30」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「10」 postStart フックを次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO のトランザクションヘルスチェックスクリプト（health_check.sh） 第 1 引数（リトライ回数）：推奨値「30」 第 2 引数（リトライ間隔秒数）：推奨値「10」 第 3 引数（接続タイムアウト秒数）：推奨値「10」

項目	要件
	第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：TP1-Bridge の server.port パラメタに設定したポート番号 preStop フックを次のとおり設定してください。 • コマンド実行 パス：HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh） 第 1 引数（リトライ回数）：推奨値「30」 第 2 引数（リトライ間隔秒数）：推奨値「10」 第 3 引数（接続タイムアウト秒数）：推奨値「10」 第 4 引数（応答タイムアウト秒数）：推奨値「10」 第 5 引数（HTTP ポート番号）：TP1-Bridge の server.port パラメタに設定したポート番号
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

表 6-26　TP1-Bridge の Filebeat コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数 「-e」 標準エラー出力にログを記録します。syslog/file へのログ出力を無効化します。 • 第 2 引数 「-E」 このパラメタ以降に記載された設定を反映した状態で起動します。 • 第 3 引数（http.enabled） 「true」 HTTP エンドポイントを有効にします。死活監視のために設定を有効化します。
ネットワーク設定	規定しません。
コンフィグ設定	• filebeat.yml を「3.9.1　ログおよびトラブルシュート情報の取得の設定」のとおりに設定してください。 • 環境変数 HMPPCTO_POD_NAMESPACE、HMPPCTO_POD_NAME、および HMPPCTO_NODE_NAME で Kibana などの可視化ツールで検索する際の条件として使用するフィールドを付与してください。
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds）

項目	要件
	推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」
CPU/メモリリソース制限	規定しません。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6–27　TP1-Bridge の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル			説明
templates/			Helm チャートの template 配置用ディレクトリ
	includes/		Helm チャートの ConfigMap オブジェクトの設定内容外部化ファイル配置用ディレクトリ
		application.properties.tp1	TP1-Bridge の ConfigMap オブジェクトの設定内容外部化ファイル
		tp1-bridge-config-map-data.yaml	TP1-Bridge の ConfigMap オブジェクトの設定内容外部化ファイル
		config.properties.tpl	uCosminexus Application Runtime with Java for Spring Boot の ConfigMap オブジェクトの設定内容外部化ファイル
		ucars-config-map-data.yaml	uCosminexus Application Runtime with Java for Spring Boot の ConfigMap オブジェクトの設定内容外部化ファイル
		filebeat.yml.tpl	Filebeat の ConfigMap オブジェクトの設定内容外部化ファイル
		filebeat-config-map-data.yaml	Filebeat の ConfigMap オブジェクトの設定内容外部化ファイル
		eads-config-map-data.yaml	EADS クライアントの ConfigMap オブジェクトの設定内容外部化ファイル
		hmp-pcto.eads-client.properties.tpl	EADS クライアントの ConfigMap オブジェクトの設定内容外部化ファイル
	tp1-bridge-config-map.yaml		TP1-Bridge の ConfigMap オブジェクトのマニフェスト

ディレクトリ/ファイル		説明
	ucars-config-map.yaml	uCosminexus Application Runtime with Java for Spring Boot の ConfigMap オブジェクトのマニフェスト
	filebeat-config-map.yaml	Filebeat の ConfigMap オブジェクトのマニフェスト
	service.yaml	Service オブジェクトのマニフェスト
	serviceaccount.yaml	ServiceAccount オブジェクトのマニフェスト
	deployment.yaml	Deployment オブジェクトのマニフェスト
	eads-config-map.yaml	EADS クライアントの ConfigMap オブジェクトのマニフェスト
Chart.yaml		Chart.yaml
values.yaml		values.yaml
values.schema.json		values.yaml の設定値のバリデーション用ファイル

6.4.7 EADS の要件（通常版限定）

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6–28 EADS の要件

項目	要件
ワークロードリソース	StatefulSet リソースとしてください。
ネームスペース	規定しません。
ラベル	StatefulSet リソースの metadata.label には次を設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：eads-server 備考：チャート名 • キー名：app.kubernetes.io/version 値：*VV-RR*※1 備考：EADS サーババージョン • キー名：helm.sh/chart 値：eads-server-*V.R.S*※2 備考：<チャート名>-<バージョン> StatefulSet リソースの spec.selector.matchLabels には次を設定します。 • キー名：app.kubernetes.io/instance

項目	要件
	値：<リリース名> • キー名：app.kubernetes.io/name 値：eads-server 備考：チャート名
アノテーション	設定しません（デフォルト）。
初期化コンテナ構成	設定しません（デフォルト）。
コンテナ構成	• EADS サーバ用コンテナ(「5.3.11　EADS サーバ用コンテナイメージ（通常版限定）」を参照してください) • EADS コマンド実行用コンテナ(「5.3.12　EADS コマンド実行用コンテナイメージ（通常版限定）」を参照してください)
レプリカ数	「3」以上の数値としてください。
サービス名 (StatefulSet リソース特有)	「eads-server-eads-cluster」としてください。
Pod の管理ポリシー (StatefulSet リソース特有)	podManagementPolicy: Parallel としてください。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	type: OnDelete としてください。
Pod セキュリティコンテキスト	設定しません（デフォルト）。
アクセス制御	EADS サーバ・EADS の運用コマンドの実行に必要な EADS の helm チャート固有のアクセス制御だけとします。
Pod の割り当てノードの制御	設定しません（デフォルト）。 注　「ローカル永続ボリューム(local)」の永続化ボリュームをマウントすることで割り当てを制御します。
永続化ボリューム	• 1 つの Pod につき割り当てる永続化ボリュームは 3 つ（ログファイル用、ストアデータファイル用、キャッシュデータファイルおよびキャッシュインデックスファイル用）としてください（共有しないでください）。 • 永続ボリュームの種類を「ローカル永続ボリューム（local)」（ノードにアタッチされたローカルストレージデバイス）としてください。 • ファイルシステム（volumeModes:Filesystem）としてください。
Pod ライフサイクル設定	設定しません（デフォルト）。

注※1

VV-RR は使用する EADS のバージョンに合わせて読み替えてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

＜制限事項＞

スケーリングは不可とします（手動/自動、スケールアウト/インに関係なく）。

コンテナごとの要件を次に示します。

表 6-29　EADS サーバ用コンテナの要件

項目	要件
コマンド	./bin/orchestration_eads_server_start.sh
ネットワーク設定	Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 EADS のリクエスト受付用ポート番号（Mediator 向け） • プロトコル TCP • 公開ポート EADS の eads.server.port パラメタに設定したポート番号 • コンテナポート EADS の eads.server.port パラメタに設定したポート番号
コンフィグ設定	次の HMP-PCTO のパラメタを、EADS サーバの helm チャートが参照する values.yaml に設定してください。 •「8.3.3　EADS（通常版限定）」のパラメタ
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-30　EADS コマンド実行用コンテナの要件

項目	要件
コマンド	./orchestration_eads_command_start.sh
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	次の HMP-PCTO のパラメタを、EADS サーバの helm チャートが参照する values.yaml に設定してください。 •「8.3.3　EADS（通常版限定）」のパラメタ
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

eads-server-chart.tgz を解凍し、中にあるファイル（values.yaml など）を使用できます。

表 6-31　EADS の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル	説明
/opt/hitachi/xeads/orchestration/chart/	Helm チャートの template 配置用ディレクトリ

ディレクトリ/ファイル		説明
	eads-service-chart.tgz	EADS サーバ専用 Service チャート
	eads-server-chart.tgz	EADS サーバチャート
	eads-command-job-chart.tgz	EADS コマンド用ファイル
	eads-command-cronjob-chart.tgz	EADS コマンド用ファイル

6.4.8 Ext-Cons の要件（トライアル版限定）

この Kubernetes アプリケーションは、HMP-PCTO として提供します。トライアル版の場合は、次の表に示す要件を満たすように作成します。

表 6-32 Ext-Cons の要件

項目	要件
ワークロードリソース	Deployment リソースとしてください。
ネームスペース	規定しません。
ラベル	metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：ext-cons 備考：チャート名 • キー名：app.kubernetes.io/version 値：*V.R.S*※ 備考：バージョン • キー名：helm.sh/chart 値：ext-cons-*V.R.S*※ 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/name 値：mediator 備考：チャート名
アノテーション	設定しません（デフォルト）。
初期化コンテナ構成	• Dependency-Checker コンテナ（「5.3.9 Dependency-Checker コンテナイメージ」を参照してください）

項目	要件
コンテナ構成	• Ext-Cons コンテナ（「5.3.10　Ext-Cons コンテナイメージ（トライアル版限定）」を参照してください）
レプリカ数	「1」
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	デフォルト（ローリングアップデート）から変更ありません。
Pod セキュリティコンテキスト	設定しません（デフォルト）。
アクセス制御	ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用して、Pod に対するロールベースアクセス制御（RBAC）で次のとおり対象リソースへのアクセス許可を定義してください。 • API グループ 空（コアグループ）、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定）。
Pod の割り当てノードの制御	設定しません（デフォルト）。
永続化ボリューム	設定しません（デフォルト）。
Pod ライフサイクル設定	設定しません（デフォルト）。

注※

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コンテナごとの要件を次に示します。

表 6-33　Ext-Cons の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群）

項目	要件
	＜対象 Kubernetes アプリケーション＞ Logstash、Jaeger-query、Jaeger-collector、Prometheus、Metricbeat、Filebeat 例を次に示します。 （例） my-namespace/deployments/logstash my-namespace/deployments/jaeger-query my-namespace/deployments/jaeger-collector my-namespace/deployments/prometheus my-namespace/deployments/metricbeat my-namespace/daemonsets/filebeat （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-34　Ext-Cons コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 Ext-Cons の HTTP ポート番号 • プロトコル TCP • 公開ポート Ext-Cons の server.port パラメタに設定したポート番号 • コンテナポート Ext-Cons の server.port パラメタに設定したポート番号
コンフィグ設定	次の HMP-PCTO のパラメタを、Spring Boot のコンフィグソース（application.properties、application.yml、環境変数など）に設定してください。 •「8.2.1　共通」のパラメタ •「8.3.4　Ext-Cons（トライアル版限定）」のパラメタ 次のとおりファイルを配置します。 • ファイル：Spring Boot のコンフィグソース キー名：application.properties

項目	要件
	マウント先：/deployments/config/application.properties
	環境変数 TZ でタイムゾーンを設定します。ユーザ責務の Kubernetes アプリケーションと同じタイムゾーンに設定することを推奨します。デフォルト値は日本標準時（Asia/Tokyo、UTC+09:00）です。 Helm チャートの timezone パラメタで、Ext-Cons コンテナのタイムゾーンを指定します。
	Helm チャートの java.options パラメタで Ext-Cons の Java プロセス起動時に Java VM に渡すオプションを指定してください。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-35　Ext-Cons の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル			説明
templates/			Helm チャートの template 配置用ディレクトリ
	includes/		Helm チャートの ConfigMap オブジェクトの設定内容外部化ファイル配置用ディレクトリ
		application.properties.tpl	ConfigMap オブジェクトの設定内容外部化ファイル
		config-map-data.yaml	ConfigMap オブジェクトの設定内容外部化ファイル
	config-map.yaml		ConfigMap オブジェクトのマニフェスト
	deployment.yaml		Deployment オブジェクトのマニフェスト
	service.yaml		Service オブジェクトのマニフェスト
	serviceaccount.yaml		ServiceAccount オブジェクトのマニフェスト
Chart.yaml			Chart.yaml
values.yaml			values.yaml
values.schema.json			values.yaml の設定値のバリデーション用ファイル

6.4.9　Filebeat の要件

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6-36　Filebeat の要件

項目	要件
ワークロードリソース	DaemonSet リソースとしてください。
ネームスペース	任意の名称を指定できます。ただし、「6.4.9　Filebeat の要件」～「6.4.15 Logstash の要件」では同一のネームスペースを指定してください。
ラベル	.metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：filebeat 備考：チャート名 • キー名：app.kubernetes.io/version 値：*V.R.S*※1 備考：バージョン • キー名：helm.sh/chart 値：filebeat-*V.R.S*※2 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> 備考： • キー名：app.kubernetes.io/name 値：filebeat 備考：チャート名
アノテーション	規定しません。
初期化コンテナ構成	Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	• Filebeat コンテナ
レプリカ数	規定しません（DaemonSet のため）。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	ローリングアップデートとしてください（Kubernetes のデフォルト設定）。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で、次に示すとおり対象リソースへのアクセス許可を定義してください。 <デプロイ依存関係チェック機能用ロール> • API グループ

項目	要件
	空(コアグループ)、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list 上記アクセス許可を定義するための Kubernetes リソースは Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 ＜コンテナログ収集用ロール＞ • API グループ 空(コアグループ) • リソース pods、nodes、namespaces • アクセス許可 get、watch、list 上記アクセス許可を定義するための Kubernetes リソースは ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定）。
Pod の割り当てノードの制御	HMP-PCTO の Pod が割り当てられる Kubernetes ノードすべてに必ず割り当ててください。
永続化ボリューム	規定しません。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「30」

注※1

V.R.S は使用する Filebeat のバージョンに合わせて読み替えてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コンテナごとの要件を次に示します。

表 6-37　Filebeat の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数）

項目	要件
	推奨値「10」 • 第４引数（応答タイムアウト秒数） 推奨値「10」 • 第５引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Elasticsearch 例を次に示します。 （例） my-namespace/statefulsets/elasticsearch-master （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-38　Filebeat コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数 「-e」 標準エラー出力にログを記録します。syslog/file へのログ出力を無効化します。 • 第 2 引数 「-E」 このパラメタ以降に記載された設定を反映した状態で起動します。 • 第 3 引数（http.enabled） 「true」 HTTP エンドポイントを有効にします。死活監視のために設定を有効化します。
ネットワーク設定	規定しません。
コンフィグ設定	規定しません。
コンテナセキュリティコンテキスト	次のとおり設定します。 • 実行ユーザ：0
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds）

項目	要件
	推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-39　Filebeat の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル		説明
templates/		Helm チャートの template 配置用ディレクトリ
	configmap-config.yaml	ConfigMap オブジェクトのマニフェスト
	daemonset.yaml	Daemonset オブジェクトのマニフェスト
	serviceaccount.yaml	ServiceAccount オブジェクトのマニフェスト
Chart.yaml		Chart.yaml
values.yaml		values.yaml
values.schema.json		values.yaml の設定値のバリデーション用ファイル

6.4.10　Jaeger-query の要件

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6-40　Jaeger-query の要件

項目	要件
ワークロードリソース	Deployment リソースとしてください。
ネームスペース	任意の名称を指定できます。ただし、「6.4.9　Filebeat の要件」～「6.4.15 Logstash の要件」では同一のネームスペースを指定してください。
ラベル	metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：jaeger-query 備考：チャート名 • キー名：app.kubernetes.io/version 値：*V.R.S*※1 備考：バージョン • キー名：helm.sh/chart 値：jaeger-query-*V.R.S*※2 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> 備考： • キー名：app.kubernetes.io/name 値：jaeger-query 備考：チャート名
アノテーション	規定しません。
初期化コンテナ構成	Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	Jaeger-query コンテナ
レプリカ数	「1」
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	ローリングアップデートとしてください（Kubernetes のデフォルト設定）。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で、次に示すとおり対象リソースへのアクセス許可を定義してください。 <デプロイ依存関係チェック機能用ロール> • API グループ

項目	要件
	空(コアグループ)、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list 上記アクセス許可を定義するための Kubernetes リソースは Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定)。
Pod の割り当てノードの制御	規定しません。
永続化ボリューム	規定しません。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「30」

注※1

V.R.S は使用する Jaeger-query のバージョンに合わせて読み替えてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コンテナごとの要件を次に示します。

表 6-41　Jaeger-query の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Elasticsearch 例を次に示します。 （例） my-namespace/statefulsets/elasticsearch-master

項目	要件
	（エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-42　Jaeger-query コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • --es.max-span-age Jaeger-query の Helm チャートパラメタの jaegerQuery.maxSpanAge の設定値 パラメタの詳細については、「(1)　Helm チャートのパラメタ」を参照してください。
ネットワーク設定	デプロイしたノードの IP で、次のポートを公開してください（Service リソース）。 Jaeger の Web UI アクセス用ポート番号 • プロトコル TCP • 公開ポート デプロイしたノード上のほかと被らないポート番号 • コンテナポート 16686
コンフィグ設定	規定しません。
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「30」 • Probe 実行間隔秒数（periodSeconds） 推奨値「15」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「10」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」

項目	要件
	Readiness Probe を次のとおり設定すること。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「30」 • Probe 実行間隔秒数（periodSeconds） 推奨値「5」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「4」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-43　Jaeger-query の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル		説明
templates/		Helm チャートの template 配置用ディレクトリ
	deployment.yaml	Deployment オブジェクトのマニフェスト
	service.yaml	Service オブジェクトのマニフェスト
	serviceaccount.yaml	ServiceAccount オブジェクトのマニフェスト
Chart.yaml		Chart.yaml
values.yaml		values.yaml
values.schema.json		values.yaml の設定値のバリデーション用ファイル

6.4.11　Jaeger-collector の要件

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6-44　Jaeger-collector の要件

項目	要件
ワークロードリソース	Deployment リソースとしてください。
ネームスペース	任意の名称を指定できます。ただし、「6.4.9　Filebeat の要件」～「6.4.15 Logstash の要件」では同一のネームスペースを指定してください。

項目	要件
ラベル	.metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：jaeger-collector 備考：チャート名 • キー名：app.kubernetes.io/version 値：*V.R.S*※1 備考：バージョン • キー名：helm.sh/chart 値：jaeger-collector-*V.R.S*※2 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> 備考： • キー名：app.kubernetes.io/name 値：jaeger-collector 備考：チャート名
アノテーション	規定しません。
初期化コンテナ構成	Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	Jaeger-collector コンテナ
レプリカ数	規定しません。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	ローリングアップデートとしてください（Kubernetes のデフォルト設定）。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で、次に示すとおり対象リソースへのアクセス許可を定義してください。 <デプロイ依存関係チェック機能用ロール> • API グループ 空(コアグループ)、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list

項目	要件
	上記アクセス許可を定義するための Kubernetes リソースは Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定）。
Pod の割り当てノードの制御	• 1 つのノードに割り当てられる Pod 数は 1 つまでです（例として、ワーカーノード数「5」に対してレプリカ数「5」のとき、各ノードに Pod が 1 つ起動します）。
永続化ボリューム	規定しません。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「30」

注※1

V.R.S は使用する Jaeger-collector のバージョンに合わせて読み替えてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コンテナごとの要件を次に示します。

表 6-45　Jaeger-collector の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Elasticsearch 例を次に示します。 （例） my-namespace/statefulsets/elasticsearch-master （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。

項目	要件
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-46　Jaeger-collector コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • --collector.queue-size Jaeger-collector の Helm チャートパラメタの collector.queueSize の設定値 • --collector.num-workers Jaeger-collector の Helm チャートパラメタの collector.numWorkers の設定値 • --es.num-replicas Jaeger-collector の Helm チャートパラメタの replicas の設定値 • --es.num-shards Jaeger-collector の Helm チャートパラメタの collector.numShards の設定値 パラメタの詳細については、「(1)　Helm チャートのパラメタ」を参照してください。
ネットワーク設定	デプロイしたノードの IP によるポートの公開は不要です。 Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 分散トレースの Span 受付用ポート番号（Participant,Orchestrator,Mediator 向け） • プロトコル TCP • コンテナポート 14268
コンフィグ設定	規定しません。
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「300」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold）

項目	要件
	推奨値「1」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「60」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「3」
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-47　Jaeger-collector の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル		説明
templates/		Helm チャートの template 配置用ディレクトリ
	deployment.yaml	Deployment オブジェクトのマニフェスト
	service.yaml	Service オブジェクトのマニフェスト
	serviceaccount.yaml	ServiceAccount オブジェクトのマニフェスト
Chart.yaml		Chart.yaml
values.yaml		values.yaml
values.schema.json		values.yaml の設定値のバリデーション用ファイル

6.4.12　Prometheus の要件

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6-48　Prometheus の要件

項目	要件
ワークロードリソース	Deployment リソースとしてください。
ネームスペース	任意の名称を指定できます。ただし、「6.4.9　Filebeat の要件」～「6.4.15 Logstash の要件」では同一のネームスペースを指定してください。

項目	要件
ラベル	.metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：prometheus 備考：チャート名 • キー名：app.kubernetes.io/version 値：*V.R.S*※1 備考：バージョン • キー名：helm.sh/chart 値：prometheus-*V.R.S*※2 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> 備考： • キー名：app.kubernetes.io/name 値：prometheus 備考：チャート名
アノテーション	規定しません。
初期化コンテナ構成	Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	• Prometheus コンテナ
レプリカ数	「1」
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	ローリングアップデートとしてください（Kubernetes のデフォルト設定）。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で、次に示すとおり対象リソースへのアクセス許可を定義してください。 <デプロイ依存関係チェック機能用ロール> • API グループ 空(コアグループ)、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list

項目	要件
	上記アクセス許可を定義するための Kubernetes リソースは Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 ＜メトリクス収集用ロール＞ • API グループ 空(コアグループ) • リソース pods、nodes、nodes/proxy、services、endpoints • アクセス許可 get、watch、list 上記アクセス許可を定義するための Kubernetes リソースは ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効（true)」にしてください（Kubernetes のデフォルト設定)。
Pod の割り当てノードの制御	規定しません。
永続化ボリューム	規定しません。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「300」

注※1

V.R.S は使用する Prometheus のバージョンに合わせて読み替えてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コンテナごとの要件を次に示します。

表 6-49　Prometheus の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Elasticsearch

項目	要件
	例を次に示します。 （例） my-namespace/statefulsets/elasticsearch-master （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-50　Prometheus コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • --config.file 「/etc/prometheus/prometheus.yml」 Prometheus の設定ファイルを配置したパス名を設定します。ConfigMap で定義した prometheus.yml は、/etc/prometheus/に配置されます。 • --storage.tsdb.retention.time Prometheus の Helm チャートパラメタの prometheus.storage.retentionTime の設定値 パラメタの詳細については、「(1)　Helm チャートのパラメタ」を参照してください。
ネットワーク設定	デプロイしたノードの IP で、次のポートを公開してください（Service リソース）。 Prometheus の Web UI アクセス用ポート番号 • プロトコル TCP • 公開ポート デプロイしたノード上のほかと被らないポート番号 • コンテナポート 9090
コンフィグ設定	規定しません。
コンテナセキュリティコンテキスト	規定しません。
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「30」 • Probe 実行間隔秒数（periodSeconds）

項目	要件
	推奨値「15」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「10」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「30」 • Probe 実行間隔秒数（periodSeconds） 推奨値「5」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「4」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-51　Prometheus の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル			説明
templates/			Helm チャートの template 配置用ディレクトリ
	includes/		Helm チャートの ConfigMap オブジェクトの設定内容外部化ファイル配置用ディレクトリ
		config-map-data.yaml	ConfigMap オブジェクトの設定内容外部化ファイル
		prometheus.yml.tpl	ConfigMap オブジェクトの設定内容外部化ファイル
	config-map.yaml		ConfigMap オブジェクトのマニフェスト
	deployment.yaml		Deployment オブジェクトのマニフェスト
	service.yaml		Service オブジェクトのマニフェスト
	serviceaccount.yaml		ServiceAccount オブジェクトのマニフェスト
Chart.yaml			Chart.yaml
values.yaml			values.yaml
values.schema.json			values.yaml の設定値のバリデーション用ファイル

6.4.13　Metricbeat の要件

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6-52　Metricbeat の要件

項目	要件
ワークロードリソース	Deployment リソースとしてください。
ネームスペース	任意の名称を指定できます。ただし、「6.4.9　Filebeat の要件」～「6.4.15 Logstash の要件」では同一のネームスペースを指定してください。
ラベル	.metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：metricbeat 備考：チャート名 • キー名：app.kubernetes.io/version 値：*V.R.S*※1 備考：バージョン • キー名：helm.sh/chart 値：metricbeat-*V.R.S*※2 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> 備考： • キー名：app.kubernetes.io/name 値：metricbeat 備考：チャート名
アノテーション	規定しません。
初期化コンテナ構成	Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	Metricbeat コンテナ
レプリカ数	「1」
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	ローリングアップデートとしてください（Kubernetes のデフォルト設定）。

項目	要件
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で、次に示すとおり対象リソースへのアクセス許可を定義してください。 ＜デプロイ依存関係チェック機能用ロール＞ • API グループ 空(コアグループ)、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list 上記アクセス許可を定義するための Kubernetes リソースは Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。 ＜メトリクス向け Kubernetes メタデータ付与用ロール＞ • API グループ 空(コアグループ) • リソース pods、nodes、namespaces、events • アクセス許可 get、watch、list 上記アクセス許可を定義するための Kubernetes リソースは ClusterRole リソース、ClusterRoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定）。
Pod の割り当てノードの制御	規定しません。
永続化ボリューム	規定しません。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「30」

注※1

V.R.S は使用する Metricbeat のバージョンに合わせて読み替えてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コンテナごとの要件を次に示します。

表 6-53　Metricbeat の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。

項目	要件
	• 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Elasticsearch 例を次に示します。 （例） my-namespace/statefulsets/elasticsearch-master （エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-54　Metricbeat コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数 「-e」 標準エラー出力にログを記録します。syslog/file へのログ出力を無効化します。 • 第 2 引数 「-E」 このパラメタ以降に記載された設定を反映した状態で起動します。 • 第 3 引数（http.enabled） 「true」 HTTP エンドポイントを有効にします。死活監視のために設定を有効化します。
ネットワーク設定	デプロイしたノードの IP によるポートの公開は不要です。 Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。

項目	要件
	メトリクス情報受付用ポート番号（Prometheus 向け） • プロトコル TCP • コンテナポート 9201
コンフィグ設定	規定しません。
コンテナセキュリティコンテキスト	次のとおり設定します。 • 実行ユーザ：0
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「10」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-55　Metricbeat の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル		説明
templates/		Helm チャートの template 配置用ディレクトリ
	config-map.yaml	ConfigMap オブジェクトのマニフェスト
	deployment.yaml	Deployment オブジェクトのマニフェスト

ディレクトリ/ファイル		説明
	service.yaml	Service オブジェクトのマニフェスト
	serviceaccount.yaml	ServiceAccount オブジェクトのマニフェスト
Chart.yaml		Chart.yaml
values.yaml		values.yaml
values.schema.json		values.yaml の設定値のバリデーション用ファイル

6.4.14　Elasticsearch の要件

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6-56　Elasticsearch の要件

項目	要件
ワークロードリソース	StatefulSet リソースとしてください。
ネームスペース	任意の名称を指定できます。ただし、「6.4.9　Filebeat の要件」～「6.4.15 Logstash の要件」では同一のネームスペースを指定してください。
ラベル	metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：次のどちらかの値を設定します。 ・elasticsearch-master(シングル構成、クラスタ構成のマスターノードの場合) ・elasticsearch-data (クラスタ構成のデータノードの場合) • キー名：app.kubernetes.io/version 値：*V.R.S*※1 備考：バージョン • キー名：helm.sh/chart 値：elasticsearch-*V.R.S*※2 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> 備考： • キー名：app.kubernetes.io/name

項目	要件
	値：elasticsearch 備考：チャート名
アノテーション	規定しません。
初期化コンテナ構成	sysctl コマンド実行用コンテナ
コンテナ構成	Elasticsearch コンテナ
レプリカ数	1 以上。 クラスタ構成の場合、master ノードのレプリカ数は 3 以上。
サービス名 （StatefulSet リソース特有）	「elasticsearch-master-headless」
Pod の管理ポリシー （StatefulSet リソース特有）	規定しません。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	ローリングアップデートとしてください（Kubernetes のデフォルト設定）。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	規定しません。
Pod の割り当てノードの制御	• 1 つのノードに割り当てられる Pod 数は 1 つまでです（例として、ワーカーノード数「5」に対してレプリカ数「5」のとき、各ノードに Pod が 1 つ起動します）。 • 停止した Pod を次に起動する際は前回起動したノードと必ず同一のノードに割り当ててください。該当するノードが障害から復旧していない場合に割り当て不可のときは、ほかの空いているノードが存在していたとしても Pod を割り当てないでください（フェイルオーバーしないでください）。
永続化ボリューム	• データディレクトリに割り当てる永続化ボリュームは 1 つの Pod につき 1 つとしてください（共有しないでください）。 • リポジトリディレクトリに割り当てる永続化ボリュームは、全 Pod からアクセス可能としてください（共有してください）。 • データディレクトリに割り当てる永続ボリュームの種類を「ローカル永続ボリューム（local）」（ノードにアタッチされたローカルストレージデバイス）としてください。 • リポジトリディレクトリに割り当てる永続化ボリュームは、共有できる NFS 上に作成してください。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「120」

注※1

V.R.S は使用する Elasticsearch のバージョンに合わせて読み替えてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コンテナごとの要件を次に示します。

表 6-57　Elasticsearch の sysctl コマンド実行用コンテナの要件

項目	要件
コマンド	カーネルパラメタ「vm.max_map_count」を変更するため、次のとおり sysctl コマンドを実行します。 sysctl -w vm.max_map_count=262144（値はパラメタで設定します）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	次の HMP-PCTO のパラメタを、Elasticsearch の helm チャートが参照する values.yaml に設定してください。 •「8.3.10　Elasticsearch」のパラメタ「sysctlVmMaxMapCount」
コンテナセキュリティコンテキスト	次のとおり設定します。 • 実行ユーザ：root • 特権コンテナの許可
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-58　Elasticsearch コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	デプロイしたノードの IP で、次のポートを公開してください（Service リソース）。 Elasticsearch REST API 用ポート番号 • プロトコル TCP • 公開ポート デプロイしたノード上のほかと被らないポート番号 • コンテナポート 9200
コンフィグ設定	規定しません。
コンテナセキュリティコンテキスト	次のとおり設定します。 • 実行ユーザ：1000（Elasticsearch 実行ユーザ）
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「300」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」

項目	要件
	• Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」 Readiness Probe を次のとおり設定すること。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「60」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「3」
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-59　Elasticsearch の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル		説明
templates/		Helm チャートの template 配置用ディレクトリ
	_helpers.tpl	Helm チャート内で使用する変数を定義したファイル
	service-endpoint.yaml	API 通信を行うための Service オブジェクトのマニフェスト
	service-headless.yaml	クラスタ構成時、Elasticsearch Pod 間通信を行うための Service オブジェクトのマニフェスト
	statefulset.yaml	Statefulset オブジェクトのマニフェスト
Chart.yaml		Chart.yaml
values.yaml		values.yaml
values.schema.json		values.yaml の設定値のバリデーション用ファイル

6.4.15　Logstash の要件

この Kubernetes アプリケーションは、HMP-PCTO として提供します。次の表に示す要件を満たすように作成します。

表 6-60　Logstash の要件

項目	要件
ワークロードリソース	Deployment リソースとしてください。
ネームスペース	任意の名称を指定できます。ただし、「6.4.9　Filebeat の要件」～「6.4.15 Logstash の要件」では同一のネームスペースを指定してください。
ラベル	metadata.label には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> • キー名：app.kubernetes.io/managed-by 値：Helm • キー名：app.kubernetes.io/name 値：logstash 備考：チャート名 • キー名：app.kubernetes.io/version 値：*V.R.S*※1 備考：バージョン • キー名：helm.sh/chart 値：logstash-*V.R.S*※2 備考：<チャート名>-<バージョン> .spec.selector.matchLabels、および.spec.template.metadata.labels には次に示すキーを設定します。 • キー名：app.kubernetes.io/instance 値：<リリース名> 備考： • キー名：app.kubernetes.io/name 値：logstash 備考：チャート名
アノテーション	規定しません。
初期化コンテナ構成	Dependency-Checker コンテナ（「5.3.9　Dependency-Checker コンテナイメージ」を参照してください）
コンテナ構成	Logstash コンテナ
レプリカ数	規定しません。
再起動ポリシー	コンテナ障害時は必ず再起動（オートヒーリング）してください（Kubernetes のデフォルト設定）。
アップデートストラテジー	ローリングアップデートとしてください（Kubernetes のデフォルト設定）。
Pod セキュリティコンテキスト	規定しません。
アクセス制御	Pod に対するロールベースアクセス制御（RBAC）で、次に示すとおり対象リソースへのアクセス許可を定義してください。 <デプロイ依存関係チェック機能用ロール> • API グループ

項目	要件
	空(コアグループ)、apps • リソース pods、replicasets、deployments、statefulsets、daemonsets • アクセス許可 get、list 上記アクセス許可を定義するための Kubernetes リソースは Role リソース、RoleBinding リソース、ServiceAccount リソースを使用してください。
	ServiceAccount リソースのトークンの自動マウント（automountServiceAccountToken フィールド）を「有効(true)」にしてください（Kubernetes のデフォルト設定)。
Pod の割り当てノードの制御	1 つのノードに割り当てられる Pod 数は 1 つまでです（例として、ワーカーノード数「5」に対してレプリカ数「5」のとき、各ノードに Pod が 1 つ起動します)。
永続化ボリューム	規定しません。
Pod ライフサイクル設定	terminationGracePeriodSeconds を次のとおり設定してください。 • 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「120」

注※1

V.R.S は使用する Logstash のバージョンに合わせて読み替えてください。

注※2

V.R.S は使用する HMP-PCTO のバージョンに合わせて読み替えてください。

コンテナごとの要件を次に示します。

表 6-61　Logstash の Dependency-Checker コンテナの要件

項目	要件
コマンド	次に示すコマンド引数（args）を設定します。 • 第 1 引数（リトライ回数） 推奨値「30」 • 第 2 引数（リトライ間隔秒数） 推奨値「10」 • 第 3 引数（接続タイムアウト秒数） 推奨値「10」 • 第 4 引数（応答タイムアウト秒数） 推奨値「10」 • 第 5 引数以降（ターゲット Kubernetes オブジェクト群） ＜対象 Kubernetes アプリケーション＞ Elasticsearch 例を次に示します。 （例） my-namespace/statefulsets/elasticsearch-master

項目	要件
	（エントリポイントは Dependency-Checker コンテナイメージで「HMP-PCTO のデプロイ依存関係チェックスクリプト（dependency_check.sh）」を設定済みです）
ネットワーク設定	設定しません（デフォルト）。
コンフィグ設定	設定しません（デフォルト）。
コンテナセキュリティコンテキスト	設定しません（デフォルト）。
コンテナライフサイクル設定	設定しません（デフォルト）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 6-62　Logstash コンテナの要件

項目	要件
コマンド	設定しません（デフォルト）。
ネットワーク設定	デプロイしたノードの IP によるポートの公開は不要です。 Kubernetes クラスタ内で次のポートを公開してください（Service リソース）。 ログメッセージ受付用ポート番号（Filebeat 向け） • プロトコル TCP • コンテナポート 5044
コンフィグ設定	規定しません。
コンテナセキュリティコンテキスト	次のとおり設定します。 • 実行ユーザ：1000（Logstash 実行ユーザ）
コンテナライフサイクル設定	Liveness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「300」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」 • Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「1」 Readiness Probe を次のとおり設定してください。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「60」 • Probe 実行間隔秒数（periodSeconds） 推奨値「10」

項目	要件
	• Probe 最大失敗回数（failureThreshold） 推奨値「3」 • Probe タイムアウト秒数（timeoutSeconds） 推奨値「5」 • Probe 失敗後の成功判定最小回数（successThreshold） 推奨値「3」
CPU/メモリリソース制限	実稼働を通して、環境に合わせた設定を実施してください。

Helm チャートのディレクトリ/ファイル構成を次の表に示します。

表 6-63　Logstash の Helm チャートのディレクトリ/ファイル構成

ディレクトリ/ファイル		説明
templates/		Helm チャートの template 配置用ディレクトリ
	configmap-config.yaml	Logstash の起動と制御設定を定義した ConfigMap オブジェクトのマニフェスト
	configmap-pipeline.yaml	Logstash で受け付けるデータの入力・出力先に関する設定を定義した ConfigMap オブジェクトのマニフェスト
	deployment.yaml	Deployment オブジェクトのマニフェスト
	service.yaml	Service オブジェクトのマニフェスト
	serviceaccount.yaml	ServiceAccount オブジェクトのマニフェスト
Chart.yaml		Chart.yaml
values.yaml		values.yaml
values.schema.json		values.yaml の設定値のバリデーション用ファイル

6.5 Kubernetes アプリケーションで使用するスクリプト

Kubernetes アプリケーションで使用するシェルスクリプトについて説明します。

6.5.1 デプロイ依存関係チェックスクリプト（dependency_check.sh）

コマンド引数で指定されたターゲット Kubernetes オブジェクト群が起動状態になるまで待機するスクリプトです。各 Kubernetes アプリケーションの初期化コンテナとして起動する Dependency-Checker コンテナで使用します。

Kubernetes API サーバ（(kube-apiserver）に対して Kubernetes オブジェクトの情報を取得する GET リクエストを実行し、JSON レスポンスの内容が起動済みを示している場合は、ステータスコード「0」で正常終了します。初期化コンテナが正常終了した場合、Kubernetes クラスタは該当する Pod 内の各コンテナの起動を開始します。

一方、JSON レスポンスの内容が未起動または起動中を示している場合は、コマンド引数で指定されたリトライ回数・リトライ間隔秒数でリトライします。最大リトライ回数を超えた場合は、ステータスコード「1」で異常終了します。リトライオーバーし、初期化コンテナが異常終了した場合、Pod の状態は CrashLoopBackOff になります。

＜コマンド構文＞

```
dependency_check.sh△<リトライ回数>△<リトライ間隔秒数>△<接続タイムアウト秒数>△<応答タイムアウト秒数>△{<Namespace>/<Kubernetesリソース種別>/[<Kubernetesオブジェクト名>|<ラベルセレクタ>]}△[...]
```

（凡例）

△：半角スペース 1 文字

ターゲット Kubernetes オブジェクトの指定方法の詳細を次に説明します。

- ターゲット Kubernetes オブジェクトは、次の形式で指定します。

  ```
  {<Namespace>/<Kubernetesリソース種別>/[<Kubernetesオブジェクト名>|<ラベルセレクタ>]}
  ```

- <Namespace>は、ターゲット Kubernetes オブジェクトの Namespace を指定します。なお、存在しない Namespace を指定した場合は、KFSG62103-W のメッセージを出力し、リトライします。
- <Kubernetes リソース種別>は、ターゲット Kubernetes オブジェクトの Kubernetes リソース種別を指定します。指定可能な Kubernetes リソース種別は次のとおりです。なお、これら以外の文字列を指定した場合は、KFSG62103-W のメッセージを出力し、リトライします。
  - pods
  - replicasets
  - deployments

  - statefulsets
  - daemonsets
- <Kubernetes オブジェクト名>は、ターゲット Kubernetes オブジェクトを Kubernetes オブジェクト名で指定する場合に指定します。なお、存在しない Kubernetes オブジェクト名を指定した場合は、KFSG62103-W のメッセージを出力し、リトライします。
- <ラベルセレクタ>は、ターゲット Kubernetes オブジェクトをラベルセレクタで指定する場合に指定します。ラベルセレクタは「<キー>=<値>」の形式で指定します（(例)「app=mediator」)。ラベルセレクタで指定すると、該当するすべての Kubernetes オブジェクトが起動済みとなるまで待機します。なお、存在しないラベルを指定した場合は、KFSG62110-W のメッセージを出力し、リトライします。
- ターゲット Kubernetes オブジェクトは、複数指定できます。ターゲット Kubernetes オブジェクトを複数指定すると、指定したすべての Kubernetes オブジェクトが起動済みとなるまで待機します。

＜実行例＞

ターゲット Kubernetes オブジェクトを Kubernetes オブジェクト名で指定する例：

```
dependency_check.sh 30 10 10 10 my-namespace/deployments/sql-participant
```

ターゲット Kubernetes オブジェクトをラベルセレクタで指定する例：

```
dependency_check.sh 30 10 10 10 my-namespace/pods/app=sql-participant
```

ターゲット Kubernetes オブジェクトを複数指定する例：

```
dependency_check.sh 30 10 10 10 ¥
  my-namespace/statefulsets/mediator ¥
  my-namespace/statefulsets/eads-server ¥
  my-namespace/statefulsets/postgresql
```

＜コマンド引数チェック仕様＞

次の条件をどちらか 1 つでも満たしていない場合は、KFSG62109-E のメッセージを出力し、ステータスコード「1」で異常終了します。

- コマンド引数の数が「5 以上」である
- コマンド引数で指定されたリトライ回数・リトライ間隔秒数・接続タイムアウト秒数・応答タイムアウト秒数の値が「整数値」である

＜バージョン表示オプション＞

次のようにコマンドに「--version」オプションを指定して実行すると、該当するスクリプト名およびバージョン番号を表示して終了します。

```
dependency_check.sh△--version
```

（凡例）

△：半角スペース 1 文字

＜コマンドの実行結果＞

```
dependency_check.sh△V.R.S
```

（凡例）

△：半角スペース 1 文字

V.R.S：バージョン

＜コマンドの実行結果の例＞

```
dependency_check.sh 1.0.0
```

6.5.2 トランザクションヘルスチェックスクリプト（health_check.sh）

Participant や Mediator が新規トランザクションを受け付けられる状態になるまで待機するスクリプトです。Participant および Mediator の postStart フックで使用します。

localhost に対してコマンド引数で指定された HTTP ポートの「トランザクションヘルスチェック API (/hmppcto/health)」へ GET リクエストを実行し、JSON レスポンスに含まれる「Mediator/Participant の起動状態（"status"キー)」の値が「起動状態（"SERVING")」の場合は、ステータスコード「0」で正常終了します。postStart フックが正常終了した場合、Pod のステータスは Running になります。

一方、値が「停止状態（非"SERVING")」の場合は、コマンド引数で指定されたリトライ回数・リトライ間隔秒数でリトライします。最大リトライ回数を超えた場合は、ステータスコード「1」で異常終了します。リトライオーバーし、postStart フックが異常終了した場合、Pod の状態は PostStartHookError になります。

＜コマンド構文＞

```
health_check.sh△<リトライ回数>△<リトライ間隔秒数>△<接続タイムアウト秒数>△<応答タイムアウト秒数> △<HTTPポート番号>
```

（凡例）

△：半角スペース 1 文字

＜実行例＞

```
health_check.sh 30 10 10 10 20555
```

＜コマンド引数チェック仕様＞

次の条件をどちらか 1 つでも満たしていない場合は、KFSG62100-E のメッセージを出力し、ステータスコード「1」で異常終了します。

- コマンド引数の数が「5 以上」である
- 各コマンド引数の値が「整数値」である

＜バージョン表示オプション＞

次のようにコマンドに「--version」オプションを指定して実行すると、該当するスクリプト名およびバージョン番号を表示して終了します。

```
health_check.sh△--version
```

（凡例）

△：半角スペース 1 文字

＜コマンドの実行結果＞

```
health_check.sh△V.R.S
```

（凡例）

△：半角スペース 1 文字

V.R.S：バージョン

＜コマンドの実行結果の例＞

```
health_check.sh 1.0.0
```

6.5.3 未決着トランザクション待機スクリプト（wait_transactions.sh）

Participant や Mediator の未決着トランザクションがなくなるまで待機するスクリプトです。Participant および Mediator の preStop フックで使用します。

localhost に対してコマンド引数で指定された HTTP ポートの「トランザクションサマリ API（/hmppcto/transactions/summary）」へ GET リクエストを実行し、JSON レスポンスに含まれる「未決着トランザクション数（"active"キー）」の値が「0」の場合は、ステータスコード「0」で正常終了します。

一方、値が「非 0」の場合は、コマンド引数で指定されたリトライ回数・リトライ間隔秒数でリトライします。最大リトライ回数を超えた場合は、ステータスコード「1」で異常終了します。

preStop フックが正常終了した場合および異常終了した場合のどちらの場合も、Pod は削除されます。

＜コマンド構文＞

```
wait_transactions.sh△<リトライ回数>△<リトライ間隔秒数>△<接続タイムアウト秒数>△<応答タイムアウト秒数> △<HTTPポート番号>
```

（凡例）

△：半角スペース 1 文字

＜実行例＞

wait_transactions.sh 30 10 10 10 20555

＜コマンド引数チェック仕様＞

次の記条件をどちらか 1 つでも満たしていない場合は、KFSG62101-E のメッセージを出力し、ステータスコード「1」で異常終了します。

- コマンド引数の数が「5 以上」である
- 各コマンド引数の値が「整数値」である

＜バージョン表示オプション＞

次のようにコマンドに「--version」オプションを指定して実行すると、該当するスクリプト名およびバージョン番号を表示して終了します。

```
wait_transactions.sh△--version
```

（凡例）

△：半角スペース 1 文字

＜コマンドの実行結果＞

```
wait_transactions.sh△V.R.S
```

（凡例）

△：半角スペース 1 文字

V.R.S：バージョン

＜コマンドの実行結果の例＞

```
wait_transactions.sh 1.0.0
```

7 API

HMP-PCTO がサポートするクラス・インタフェース、アノテーションおよび例外の仕様について説明します。

7.1 API の記述形式

HMP-PCTO がサポートするクラス・インタフェース、アノテーションおよび例外の記述形式について説明します。

各項で説明する項目は次のとおりです。

クラス・インタフェース

説明

クラス・インタフェースの機能について説明します。

メソッド

クラス・インタフェースのメソッドの一覧について説明します。

なお、親クラスから継承したメソッドは記載していません。

パッケージ名称

クラス・インタフェースのパッケージ名称について説明します。

アノテーション

説明

アノテーションの機能について説明します。

適用可能要素

アノテーションの適用可能要素について説明します。

属性

アノテーションの属性の一覧について説明します。

パッケージ名称

クラス・インタフェースのパッケージ名称について説明します。

例外

説明

例外の機能について説明します。

メソッド

例外のメソッドの一覧について説明します。

なお、親クラスから継承したメソッドは記載していません。

パッケージ名称

例外のパッケージ名称について説明します。

継承関係

例外の継承関係について説明します。

メソッド

説明

メソッドの機能について説明します。

形式

メソッドの形式について説明します。

引数

メソッドの引数について説明します。

例外

メソッド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

メソッドの戻り値について説明します。

7.2　トランザクション制御の API 仕様

HMP-PCTO のトランザクション制御で使用する API の仕様を説明します。

7.2.1　クラス・インタフェースの一覧

HMP-PCTO のトランザクション制御がサポートするクラス・インタフェースの一覧を示します。

表 7-1　クラス・インタフェースの一覧

項番	パッケージ名	クラス・インタフェース名	説明
1	jp.co.Hitachi.soft.hmppcto.common.transaction	XidHeaders インタフェース	HMP-PCTO がトランザクション制御に使用する HTTP ヘッダ名を定義したインタフェースです。
2	jp.co.Hitachi.soft.hmppcto.filter.propagation	TxPropagationIscImperativePlugin クラス	HMP-ADIF の（Imperative な）ISC コマンドによる REST 通信を使用したトランザクションを伝搬する ISC コマンドプラグインクラスです。
3	jp.co.Hitachi.soft.hmppcto.filter.propagation	TxPropagationRestTemplateInterceptor クラス	Spring Framework の RestTemplate を使用したトランザクションを伝搬するインタセプタクラスです。

7.2.2　アノテーションの一覧

HMP-PCTO のトランザクション制御がサポートするアノテーションの一覧を示します。

項番	パッケージ名	アノテーション名	説明
1	jp.co.Hitachi.soft.hmppcto.branch.api	@PctoBranchRollbackFor アノテーション	トランザクションブランチの属性を設定するアノテーションです。
2	jp.co.Hitachi.soft.hmppcto.transaction.api	@PctoTransactional アノテーション	トランザクション境界（トランザクションの開始/終了）と属性を設定するアノテーションです。

7.2.3　例外の一覧

HMP-PCTO のトランザクション制御がサポートする例外の一覧を示します。

表 7-2　例外の一覧

項番	パッケージ名	例外名	説明	ユーザによる例外送出可否
1	jp.co.Hitachi.soft.hmppcto.interceptor	RollbackTerminationException	トランザクションがロールバック決着したときに送出する例外です。	否
2	jp.co.Hitachi.soft.hmppcto.interceptor	TerminationFailureException	トランザクションの決着結果が不明の場合に送出する例外です。	否
3	jp.co.Hitachi.soft.hmppcto.TransactionSystemException	TransactionSystemException	サービス情報の参照時に障害が発生した場合に送出する例外です。 主に Orchestrator の初期化時、トランザクションの開始時に送出します。	否

7.2.4　XidHeaders インタフェース（SQL-Participant 限定）

説明

HMP-PCTO がトランザクション制御に使用する HTTP ヘッダ名を定義したインタフェースです。
先行プリペアする場合は、AP が XidHeaders.IS_LAST_CALL を使用して HTTP ヘッダに追加します。
ほかの定数は、HMP-PCTO が使用する HTTP ヘッダです。AP で使用する必要はありません。

定数

XidHeaders インタフェースの定数の一覧を次の表に示します。

表 7-3　XidHeaders インタフェースの定数の一覧

定数	備考
public static final String BRANCH_ID	HMP-PCTO がトランザクション制御に使用する HTTP ヘッダ名の定数です。
public static final String IS_LAST_CALL	先行プリペアをする場合に、AP が HTTP ヘッダに追加する HTTP ヘッダ名の定数です。 AP が同一のブランチで最後に SQL を実行する場合にこの HTTP ヘッダに XidHeaders.TRUE を設定してサービスを呼び出すと、HMP-PCTO は先行プリペアを実行します。

定数	備考
public static final String FORMAT_ID	HMP-PCTO がトランザクション制御に使用する HTTP ヘッダ名の定数です。
public static final String PARTICIPANT_GRPC_URI	HMP-PCTO がトランザクション制御に使用する HTTP ヘッダ名の定数です。
public static final String TRANSACTION_CONTEXT	HMP-PCTO がトランザクション制御に使用する HTTP ヘッダ名の定数です。
public static final String TRANSACTION_ID	HMP-PCTO がトランザクション制御に使用する HTTP ヘッダ名の定数です。
public static final String TRANSACTION_KEY	HMP-PCTO がトランザクション制御に使用する HTTP ヘッダ名の定数です。
public static final String TRUE	HTTP ヘッダに XidHeaders.IS_LAST_CALL を追加する際に、HTTP ヘッダの値に設定する定数です。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.common.transaction
```

7.2.5 TxPropagationIscImperativePlugin クラス

説明

HMP-ADIF の（Imperative な）ISC コマンドによる REST 通信時に、トランザクションを伝搬する ISC コマンドプラグインクラスです。AP は HMP-PCTO のライブラリを組み込んで、HMP-ADIF の ISC コマンドを使用します。TxPropagationIscImperativePlugin クラスの各メソッドは、HMP-ADIF が呼び出します。AP が TxPropagationIscImperativePlugin クラスを直接使用する必要はありません。

TxPropagationIscImperativePlugin の preExecute()メソッドでは、ISC コマンドのエラー情報を設定しません。そのため、ほかの ISC コマンドプラグインや、ISC コマンドの実行には影響しません。

メソッド

TxPropagationIscImperativePlugin クラスのメソッドの一覧を次の表に示します。

表 7-4 TxPropagationIscImperativePlugin クラスのメソッドの一覧

メソッド	説明
applicable(java.lang.String communicationType)	ISC コマンドプラグインの適用可否を判定するメソッドです。
getOrder()	ISC コマンドプラグインの実行順序を判定するメソッドです。
postExecute(org.globallogic.microservice.modules.isc.imperative.plugins. IscImperativeCommandExecutionPluginContext pluginContext,	ISC コマンド実行後に任意の処理を実行するメソッドです。

メソッド	説明
org.globallogic.microservice.modules.IscImperativeCommand command)	
preExecute(org.globallogic.microservice.modules.isc.imperative.plugins. IscImperativeCommandExecutionPluginContext pluginContext, org.globallogic.microservice.modules.IscImperativeCommand command)	ISC コマンド実行前に任意の処理を実行するメソッドです。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.filter.propagation
```

(1) applicable(java.lang.String communicationType)

説明

ISC コマンドプラグインの適用可否を返します。

形式

```
public boolean applicable(java.lang.String communicationType)
```

引数

java.lang.String communicationType：

次のとおり、通信方式が渡されます。

REST の場合："REST"

メッセージングの場合："MESSAGING"

戻り値

HMP-PCTO は、REST にだけ対応します。

引数が"REST"の場合は true、"MESSAGING"の場合は、false を返します。

例外

なし。

(2) getOrder()

説明

ISC コマンドプラグインの実行順番を返します。

形式

```
public int getOrder()
```

引数

なし。

戻り値

パラメタ jp.co.Hitachi.soft.hmppcto.branch.qualifier.propagation-order で設定した値を返します。デフォルト値は 1 です。

TxPropagationIscImperativePlugin 以外の ISC コマンドプラグインクラスを使用する場合は、特に理由がないかぎり TxPropagationIscImperativePlugin クラスの preExecute()が最初、postExecute()が最後に呼ばれるようにデフォルト値で使用してください。

例外

なし。

(3) postExecute(IscImperativeCommandExecutionPluginContext pluginContext, IscImperativeCommand command)

説明

ISC コマンド実行後の処理を実行するメソッドです。レスポンスから HMP-PCTO がトランザクション制御に使用する HTTP ヘッダを入力してトランザクション制御を実行します。

形式

```
public void postExecute(org.globallogic.microservice.modules.isc.imperative.
plugins.IscImperativeCommandExecutionPluginContext pluginContext,
org.globallogic.microservice.modules.IscImperativeCommand command)
```

引数

org.globallogic.microservice.modules.isc.imperative.plugins.IscImperativeCommandExecutionPluginContext pluginContext：

IscImperativeCommandExecutionPluginContext インスタンスを指定します。

org.globallogic.microservice.modules.IscImperativeCommand.IscImperativeCommand command：

IscImperativeCommand インスタンスを指定します。

戻り値

なし。

例外

なし。

(4) preExecute(IscImperativeCommandExecutionPluginContext pluginContext, IscImperativeCommand command)

説明

ISC コマンド実行前の処理を実行するメソッドです。HMP-PCTO がトランザクション制御に使用する HTTP ヘッダをリクエストに追加してトランザクションを伝搬します。

形式

```
public void preExecute(IscImperativeCommandExecutionPluginContext pluginContext, IscImperativeCommand command)
```

引数

org.globallogic.microservice.modules.isc.imperative.plugins.IscImperativeCommandExecutionPluginContext pluginContext：

IscImperativeCommandExecutionPluginContext インスタンスを指定します。

org.globallogic.microservice.modules.IscImperativeCommand.IscImperativeCommand command：

IscImperativeCommand インスタンスを指定します。

戻り値

なし。

例外

なし。

7.2.6 TxPropagationRestTemplateInterceptor クラス

説明

Spring Framework の RestTemplate による REST 通信時に、トランザクションを伝搬するインタセプタクラスです。AP が RestTemplate のインスタンスに登録して使用します。
TxPropagationRestTemplateInterceptor の各メソッドは、Spring Framework が呼び出します。AP が直接 TxPropagationRestTemplateInterceptor の各メソッドを使用する必要はありません。AP がほかのインタセプタを登録する場合は、特に理由がないかぎりは、
TxPropagationRestTemplateInterceptor クラスが最初に呼び出されるように登録してください。

メソッド

TxPropagationRestTemplateInterceptor クラスのメソッドの一覧を次の表に示します。

表 7-5 TxPropagationRestTemplateInterceptor クラスのメソッドの一覧

メソッド	説明
intercept(org.springframework.http.HttpRequest request, byte[] body, org.springframework.http.client.ClientHttpRequestExecution execution)	指定されたリクエストをインタセプトし、レスポンスを返すメソッドです。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.filter.propagation
```

(1) intercept(org.springframework.http.HttpRequest request, byte[] body, org.springframework.http.client.ClientHttpRequestExecution execution)

説明

指定されたリクエストをインタセプトし、レスポンスを返します。次の順序で処理を実行します。

1. HMP-PCTO がトランザクション制御に使用する HTTP ヘッダをリクエストに追加してトランザクションを伝搬します。
2. 引数 execution の excute()メソッドを実行して後続のインタセプタを実行、およびリクエストを送信し、レスポンスを受け取ります。
3. レスポンスから HMP-PCTO がトランザクション制御に使用する HTTP ヘッダを入力してトランザクション制御を実行します。

形式

```
org.springframework.http.client.ClientHttpResponse intercept(
org.springframework.http.HttpRequest request, byte[] body,
org.springframework.http.client.ClientHttpRequestExecution execution) throws java.io.IOEx
ception
```

引数

org.springframework.http.HttpRequest request：

メソッド、URI、ヘッダを含むリクエストを指定します。

byte[] body：

リクエストの本文を指定します。

org.springframework.http.client.ClientHttpRequestExecution execution：

リクエストの実行に使用するインスタンスを指定します。

戻り値

org.springframework.http.client.ClientHttpResponse：

レスポンスを返します。

例外

java.io.IOException：

I/O エラーです。リクエストの実行時（引数 execution の excute()メソッド実行時）に送出された例外をそのまま送出します。このメソッドで新たな例外の送出はしません。

7.2.7　@PctoBranchRollbackFor アノテーション

説明

トランザクションブランチの属性を設定するアノテーションです。Entity-Service または Entity-Module で実行される任意のクラス、メソッドに指定できます。複数のメソッドに指定した場合は、最後に実行されたメソッドで設定した属性が有効になります。クラスレベルで指定することもできますが、メソッドレベルでの指定を推奨します。

rollbackOn 属性、rollbackOnFamily 属性でこのトランザクションをロールバックにマークするかどうかを決める HTTP ステータス、HTTP ステータスファミリーを設定します。

トランザクションブランチのサービス終了後に動作する HMP-PCTO のフィルタが、HTTP ステータスを判定してロールバックにマークしたかどうかを Orchestrator、および Participant に連絡します。ロールバックにマークしたトランザクションブランチが 1 つでもあると、グローバルトランザクションはロールバックします。

rollbackOn 属性、rollbackOnFamily 属性を両方指定した場合は、両方の設定値が有効です。

デフォルトの動作は、4xx、5xx をロールバックにマークします。

ただし、次の場合には、rollbackOn 属性、rollbackOnFamily 属性の設定に関係なくロールバックにマークしません。

- HTTP ステータス 404 の場合
- HTTP ステータス 405 の場合

適用可能要素

クラス、メソッド

属性

@PctoBranchRollbackFor アノテーションの属性の一覧を次の表に示します。

表 7-6　@PctoBranchRollbackFor アノテーションの属性の一覧

属性名	説明	必須/任意	型	デフォルト値
rollbackOn	トランザクションにロールバックのマークを付ける必要がある HTTP ステータスを設定します。	任意	javax.ws.rs.core.Response.Status[]	なし
rollbackOnFamily	トランザクションにロールバックのマークを付ける必要がある HTTP ステータスファミリーを設定します。	任意	javax.ws.rs.core.Response.Status.Family[]	{javax.ws.rs.core.Response.Status.Family.CLIENT_ERROR, javax.ws.rs.core.Response.Status.Family.SERVER_ERROR}

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.interceptor
```

7.2.8 @PctoTransactional アノテーション

説明

トランザクション境界を設定し、トランザクションの属性を設定するアノテーションです。このアノテーションを指定したメソッドの呼び出し前にトランザクションを開始し、メソッドの終了後にトランザクションを終了します。このアノテーションを指定したメソッドから呼び出され、同じスレッドで動作するメソッドはトランザクションの範囲内です。トランザクションの範囲内でこのアノテーションを指定したメソッドがネストしている場合は、ネストしている指定を無視します（元のトランザクションの範囲内です）。クラスレベルで指定することもできますが、メソッドレベルでの指定を推奨します。

このアノテーションを設定したメソッドの実行後に動作する HMP-PCTO のインタセプタが、アプリケーションが送出した例外をキャッチして、トランザクションにロールバックのマークを付けるか判定し、Mediator にトランザクションの決着をリクエストします。

トランザクションがロールバックに決着した場合、および通信障害によって Mediator からトランザクションの決着結果を受け取れなかった場合は、このアノテーションを指定したメソッドの呼び出し元に例外を送出します。サービスのルートとなるメソッドにこのアノテーションを指定した場合、例外がどのようなエラー処理にマッピングされるかは、使用しているフレームワーク、AP サーバの仕様に依存します。そのため、このアノテーションはルートとなるメソッド以外に設定してアプリケーションでエラーハンドリングすることを推奨します。

rollbackOn 属性を設定して、インタセプタがトランザクションにロールバックのマークを付ける必要がある例外を示すことができます。

逆に、dontRollbackOn 属性を設定して、インタセプタがトランザクションにロールバックのマークを付けてはならない例外を示すことができます。設定しなかった例外の場合にロールバックのマークを付けます。

これらの属性に設定した例外クラスのサブクラスにも同じ属性が適用されます。

両方の属性が指定されている場合、dontRollbackOn が優先されます。

両方の属性を省略した場合、java.io.IOException とそのサブクラスをロールバックにマークします。

なお、java.lang.RuntimeException とそのサブクラスは、属性の設定に関係なくロールバックのマークを付けます。

適用可能要素

クラス、メソッド

属性

@PctoTransactional アノテーションの属性の一覧を次の表に示します。

表 7-7 @PctoTransactional アノテーションの属性の一覧

属性名	説明	必須/任意	型	デフォルト値
dontRollbackOn	トランザクションにロールバックのマークを付けてはならない例外を指定します。	任意	Class<?>[]	なし

属性名	説明	必須/任意	型	デフォルト値
rollbackOn	トランザクションにロールバックのマークを付ける必要がある例外を指定します。	任意	Class<?>[]	{java.io.IOException}

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.transaction.api
```

7.2.9 RollbackTerminationException 例外

説明

トランザクションがロールバック決着した場合に送出する例外です。@PctoTransactional を設定したメソッドの呼び出し元に送出します。@PctoTransactional を設定したメソッドが送出した例外は、getSuppressed()メソッドで受け取ることができます。

トランザクションがロールバックした理由は、ログおよび getMessage()メソッドで受け取るメッセージで確認できます。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.interceptor
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      jp.co.Hitachi.soft.hmppcto.interceptor.RollbackTerminationException
```

7.2.10 TerminationFailureException 例外

説明

トランザクションの決着結果が、Mediator との通信障害のために不明の場合に送出する例外です。@PctoTransactional を設定したメソッドの呼び出し元に送出します。@PctoTransactional を設定したメソッドが送出した例外は、getSuppressed()メソッドで受け取ることができます。

トランザクションの決着結果は、ログおよび getMessage()メソッドで受け取るメッセージ中の XID をキーに、Mediator のログを確認してください。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.interceptor
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      jp.co.Hitachi.soft.hmppcto.interceptor.TerminationFailureException
```

7.2.11 TransactionSystemException 例外

説明

サービス情報の参照時に障害が発生した場合に送出する例外です。

主に Orchestrator の初期化時、トランザクションの開始時に送出します。トランザクションの開始時に送出する場合は、@PctoTransactional を設定したメソッドを呼び出さないで、@PctoTransactional を設定したメソッドの呼び出し元に TransactionSystemException を送出します。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.TransactionSystemException
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      jp.co.Hitachi.soft.hmppcto.TransactionSystemException.TransactionSystemException
```

7.3　JDBC Proxy Driver の API 仕様（SQL-Participant 限定）

HMP-PCTO の JDBC Proxy Driver で使用する API の仕様を説明します。

制限事項

- JDBC Driver の仕様で定められているメソッドのうち、HMP-PCTO がサポートしていないメソッドを呼び出した場合は PctoSqlFeatureNotSupportedException 例外、または PctoSQLClientInfoException 例外を送出します。
- 親クラスから継承されたメソッドについては、親クラスから継承されたメソッドの一覧に記載されているメソッドだけをサポートします。親クラスでサポートしているメソッドについては、親クラスから継承されたメソッドの一覧に記載がない場合、使用時に例外を送出しませんが、動作の保証はしません。
- JDBC Proxy Driver は、DB の列に配列型が使われている場合、1 次元配列にだけ対応しています。2 次元以上の配列には対応していません。2 次元以上の配列を使用した場合の動作は保証しません 。

7.3.1　クラス・インタフェースの一覧（SQL-Participant 限定）

HMP-PCTO の JDBC Proxy Driver がサポートするクラス・インタフェースの一覧を次に示します。

表 7-8　クラス・インタフェースの一覧

項番	パッケージ名	クラス・インタフェース名	説明
1	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoDataSource クラス	データベースに接続するためのファクトリクラスです。
2	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoConnection クラス	データベースとの接続（セッション）を表現するクラスです。
3	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoStatement クラス	静的 SQL 文を実行し、作成された結果を返すために使用するクラスです。
4	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoPreparedStatement クラス	プリコンパイルされた SQL 文を実行するクラスです。
5	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoCallableStatement クラス	SQL ストアド・プロシージャを実行するときに使用するクラスです。
6	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoResultSet クラス	データベースの結果を表すデータの表のクラスです。

項番	パッケージ名	クラス・インタフェース名	説明
7	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoDatabaseMetaData クラス	データベースに関する包括的な情報を提供するクラスです。
8	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoResultSetMetadata クラス	ResultSet オブジェクトの列の型とプロパティに関する情報を取得するのに使用するクラスです。
9	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoSQLXML クラス	Java プログラミング言語で SQL XML 型を表現するクラスです。
10	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoBlob クラス	Java プログラミング言語で SQL BLOB 値を表現するクラスです。
11	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoClob クラス	Java プログラミング言語で SQL CLOB 型を表現するクラスです。
12	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoInputStream クラス	バイト入力ストリームを表現するクラスです。
13	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoReader クラス	文字ストリームを読み込むためのクラスです。
14	jp.co.Hitachi.soft.hmppcto.jdbc.client	PctoArray クラス	SQL 型 ARRAY の Java プログラミング言語でのマッピングをするクラスです。

7.3.2 例外の一覧（SQL-Participant 限定）

HMP-PCTO の JDBC Proxy Driver がサポートする例外の一覧を次に示します。

表 7-9 例外の一覧

項番	パッケージ名	例外名	説明	ユーザによる例外送出可否
1	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoSqlFeatureNotSupportedException 例外	JDBC Proxy Driver がサポートしていないメソッドを呼び出した場合に送出する例外です。	否
2	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoSQLClientInfoException 例外	JDBC Proxy Driver がサポートしていないメソッドを呼び出した場合に送出する例外です。	否

項番	パッケージ名	例外名	説明	ユーザによる例外送出可否
3	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoCommunicationFailureException 例外	Entity-Service と Participant 間の通信に失敗した場合に送出する例外です。	否
4	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoInternalContradictionException 例外	Participant の内部で障害が発生した場合に送出する例外です。	否
5	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoTransactionStartException 例外	Participant のトランザクションブランチを開始できない場合に送出する例外です。	否
6	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoPrePreparedException 例外	先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出する例外です。	否
7	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoConnectionLackedException 例外	パラメタ「db.pool.max-pool-size」の指定値よりも DB コネクション数を多く使用しようとした場合に送出する例外です。	否
8	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoIOException 例外	JDBC Proxy Driver のメソッド実行時に入出力エラーが発生した場合に送出する例外です。	否
9	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoIOCommunicationFailureException 例外	Entity-Service と Participant 間の通信に失敗した場合に送出する例外です。	否
10	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoIOInternalContradictionException 例外	Participant の内部で障害が発生した場合に送出する例外です。	否
11	jp.co.Hitachi.soft.hmppcto.jdbc.client.exception	PctoIOTransactionStartException 例外	Participant のトランザクションブランチを開始できない場合に送出する例外です。	否

7.3.3　PctoDataSource クラス（SQL-Participant 限定）

説明

データベースに接続するためのファクトリクラスです。この DataSource オブジェクトが表す物理データ・ソースへの接続で使用します。DriverManager 機能の代わりに、DataSource オブジェクトが接続を得るための手段となります。

DataSource インタフェースを実装するオブジェクトは、Java Naming and Directory (JNDI) API を基にしたネーム・サービスに登録されます。

DataSource オブジェクトがルックアップ操作によって取得されて、Connection オブジェクトを作成するために使用されます。基本実装では、DataSource オブジェクトによって取得された接続は、DriverManager 機能によって取得した接続と同じものになります。

メソッド

PctoDataSource クラスのメソッドの一覧を次の表に示します。

表 7-10　PctoDataSource クラスのメソッドの一覧

メソッド	説明
getConnection()	この DataSource オブジェクトが表すデータ・ソースへの接続の確立を試みます。
isWrapperFor(Class<?> iface)	指定されたインタフェースを実装しているラッパーである場合かを返します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1)　getConnection()

説明

この DataSource オブジェクトが表すデータ・ソースへの接続の確立を試みます。

形式

```
public Connection getConnection() throws SQLException
```

引数

なし。

戻り値

java.sql.Connection：

Spring Boot の設定ファイル(application.properties、application.yml など)に設定している次の項目の URL の接続を表すオブジェクトを返します。

- spring.datasource.url

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

次の場合に例外を送出します。

- 設定ファイルで指定された接続先 Participant の URL の形式が不正の場合
- Entity-Service と Participant 間の通信に失敗した場合

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

次の場合に例外を送出します。

- Participant の内部で障害が発生しトランザクションを開始できない場合
- トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoConnectionLackedException：

パラメタ「db.pool.max-pool-size」の指定値よりも DB コネクション数を多く使用しようとした場合に送出します。

(2) isWrapperFor(Class<?> iface)

説明

指定されたインタフェースを実装しているラッパーである場合かを返します。

形式

```
public boolean isWrapperFor(Class<?> iface) throws SQLException
```

引数

iface：

インタフェースを定義するクラス。

戻り値

false を返します。

7.3.4 PctoConnection クラス（SQL-Participant 限定）

説明

データベースとの接続（セッション）を表現するクラスです。接続のコンテキスト内で SQL 文が実行され結果が返されます。Connection オブジェクト中のデータベースは、表、サポートしている SQL 文法、ストアド・プロシージャ、およびこの接続の能力などについての情報を提供します。

メソッド

PctoConnection クラスのメソッドの一覧を次の表に示します。

表 7-11　PctoConnection クラスのメソッドの一覧

メソッド	説明
close()	自動的な解除を待たないで、すぐにこの Connection オブジェクトのデータベースと JDBC リソースを解除します。
createArrayOf(java.lang.String typeName, java.lang.Object[] elements)	java.sql.Array オブジェクトを生成するファクトリ・メソッドです。
createSQLXML()	SQLXML インタフェースを実装しているオブジェクトを構築します。
createStatement()	SQL 文をデータベースに送るための Statement オブジェクトを生成します。
createStatement(int resultSetType, int resultSetConcurrency)	指定された型と並行処理で ResultSet オブジェクトを生成する Statement オブジェクトを生成します。
getMetaData()	この Connection オブジェクトが接続を表すデータベースに関するメタデータを格納する DatabaseMetaData オブジェクトを取得します。
prepareCall(java.lang.String sql)	データベースのストアド・プロシージャを呼び出すための CallableStatement オブジェクトを生成します。
prepareCall(java.lang.String sql, int resultSetType, int resultSetConcurrency)	指定された型と並行処理で ResultSet オブジェクトを生成する CallableStatement オブジェクトを生成します。
prepareStatement(java.lang.String sql)	パラメタ付き SQL 文をデータベースに送るための PreparedStatement オブジェクトを生成します。
prepareStatement(java.lang.String sql, int autoGeneratedKeys)	自動生成キーを取得する機能を持つデフォルトの PreparedStatement オブジェクトを生成します。
prepareStatement(java.lang.String sql, int resultSetType, int resultSetConcurrency)	指定された型と並行処理で ResultSet オブジェクトを生成する PreparedStatement オブジェクトを生成します。
prepareStatement(java.lang.String sql, java.lang.String[] columnNames)	指定された配列によって、指定された自動生成キーを返す機能を持つデフォルトの PreparedStatement オブジェクトを生成します。
setTransactionIsolation(int level)	この Connection オブジェクトのトランザクション遮断レベルを変更します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1) close()

説明

自動的な解除を待たないで、すぐにこの Connection オブジェクトのデータベースと JDBC リソースを解除します。

形式

```
public void close()
```

引数

なし。

戻り値

なし。

例外

なし。

(2) createArrayOf(java.lang.String typeName, java.lang.Object[] elements)

説明

java.sql.Array オブジェクトを生成するファクトリ・メソッドです。Participant の JDBC Driver は、Object 配列の要素を、Object を指定したクラスについて java.sql.Types で定義されたデフォルトの JDBC SQL 型にマッピングします。

形式

```
public java.sql.Array createArrayOf(java.lang.String typeName, java.lang.Object[] element
s) throws java.sql.SQLException
```

引数

java.lang.String typeName：

配列の要素がマッピングされる型の SQL 名を指定します。typeName は、データベース固有の名前であり、組み込み型、ユーザ定義型、またはこのデータベースでサポートされる標準 SQL 型の名前のことです。

java.lang.Object[] elements：

返されるオブジェクトを生成する要素を指定します。Object に指定できるクラスは、プリミティブデータ型に対応する参照型、プリミティブデータ型のラッパークラスのインスタンス、または HMP-PCTO が提供するクラスのインスタンスです。それ以外が指定された場合は、例外を送出します。

戻り値

指定された SQL 型に、要素がマッピングされる java.sql.Array オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- JDBC 型が typeName に適していなくて、かつ変換がサポートされていない場合
- 引数 typeName が null の場合
- メソッドがクローズされたあとに接続された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのデータ型をサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) createSQLXML()

説明

SQLXML インタフェースを実装しているオブジェクトを構築します。返されるオブジェクトには、初期状態のときはデータが格納されていません。データを SQLXML オブジェクトに追加する場合は、SQLXML インタフェースの setString メソッドを使用します。

形式

```
public java.sql.SQLXML createSQLXML() throws java.sql.SQLException
```

引数

なし。

戻り値

SQLXML インタフェースを実装しているオブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- SQLXML インタフェースを実装するオブジェクトを構築できなかった場合
- このメソッドがクローズされた接続に対して呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのデータ型をサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(4) createStatement()

説明

SQL 文をデータベースに送るための Statement オブジェクトを生成します。パラメタなしの SQL 文は、通常 Statement オブジェクトを使用して実行されます。同じ SQL 文が多数回実行される場合は、PreparedStatement オブジェクトを使用する方が効率的なことがあります。

形式

```
public java.sql.Statement createStatement() throws java.sql.SQLException
```

引数

なし。

戻り値

java.sql.Statement：

新しい Statement オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(5) createStatement(int resultSetType, int resultSetConcurrency)

説明

指定された型と並行処理で ResultSet オブジェクトを生成する Statement オブジェクトを生成します。このメソッドは、デフォルトの結果セットの型および並行処理をオーバーライドできます。

形式

```
public java.sql.Statement createStatement(int resultSetType, int resultSetConcurrency) th
rows java.sql.SQLException
```

引数

int resultSetType：

結果セットの型を指定します。指定できる結果セットの型は、ResultSet.TYPE_FORWARD_ONLY、ResultSet.TYPE_SCROLL_INSENSITIVE、または ResultSet.TYPE_SCROLL_SENSITIVE のうちの 1 つです。

int resultSetConcurrency：

並行処理の種類を指定します。指定できる並行処理の種類は、ResultSet.CONCUR_READ_ONLY または ResultSet.CONCUR_UPDATABLE のどちらかです。

戻り値

指定された型および並行処理で ResultSet オブジェクトを生成する新しい Statement オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- 指定されたパラメタが、型および並行処理を示す ResultSet 定数ではない場合

java.sql.SQLFeatureNotSupportedException：

次の場合に例外を送出します。

- Participant の JDBC Driver がこのメソッドをサポートしない場合
- このメソッドが、指定された結果セットの型と結果セットの並行処理でサポートされない場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(6) getMetaData()

説明

この Connection オブジェクトが接続を表すデータベースに関するメタデータを格納する DatabaseMetaData オブジェクトを取得します。メタデータには、データベースの表、サポートしている SQL 文の文法、ストアド・プロシージャ、および接続設定などの情報が含まれています。

形式

```
public java.sql.DatabaseMetaData getMetaData() throws java.sql.SQLException
```

引数

なし。

戻り値

この Connection オブジェクトの DatabaseMetaData オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(7) prepareCall(java.lang.String sql)

説明

データベースのストアド・プロシージャを呼び出すための CallableStatement オブジェクトを生成します。CallableStatement オブジェクトは、その IN と OUT パラメタを設定するメソッドと、ストアド・プロシージャの呼び出しを実行するメソッドを提供します。このメソッドは、ストアド・プロシージャ呼び出し文を処理するために最適化されています。

形式

```
public java.sql.CallableStatement prepareCall(java.lang.String sql) throws java.sql.SQLException
```

引数

java.lang.String sql：

1 つ以上の'?'パラメタ・プレースホルダーを含めることができる SQL 文を指定します。

戻り値

プリコンパイルされた SQL 文を含む新しいデフォルトの CallableStatement オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(8) prepareCall(java.lang.String sql, int resultSetType, int resultSetConcurrency)

説明

指定された型と並行処理で ResultSet オブジェクトを生成する CallableStatement オブジェクトを生成します。このメソッドは、デフォルトの結果セットの型と並行処理をオーバーライドできます。

形式

```
public java.sql.CallableStatement prepareCall(java.lang.String sql, int resultSetType, in
t resultSetConcurrency) throws java.sql.SQLException
```

引数

java.lang.String sql：

データベースに送る SQL 文が記述された java.lang.String オブジェクトを指定します。1 つ以上の'?'パラメタを含めることができます。

Int resultSetType：

結果セットの型を指定します。指定できる結果セットの型は、ResultSet.TYPE_FORWARD_ONLY、ResultSet.TYPE_SCROLL_INSENSITIVE、または ResultSet.TYPE_SCROLL_SENSITIVE のうちの 1 つです。

int resultSetConcurrency：

並行処理の種類を指定します。指定できる並行処理の種類は、ResultSet.CONCUR_READ_ONLY または ResultSet.CONCUR_UPDATABLE のどちらかです。

戻り値

指定された型および並行処理で ResultSet オブジェクトを生成する、プリコンパイルされた SQL 文を含む新しい CallableStatement オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- 指定されたパラメタが型および並行処理を示す ResultSet 定数ではない場合

java.sql.SQLFeatureNotSupportedException：

次の場合に例外を送出します。

- Participant の JDBC Driver がこのメソッドをサポートしない場合
- このメソッドが指定された結果セットの型と結果セットの並行処理でサポートされない場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(9) prepareStatement(java.lang.String sql)

説明

パラメタ付き SQL 文をデータベースに送るための PreparedStatement オブジェクトを生成します。IN パラメタ付きまたは IN パラメタなしの SQL 文は、プリコンパイルして、PreparedStatement オブジェクトに格納できます。その後、このオブジェクトは、この文を複数回効率的に実行するために使用できます。このメソッドは、プリコンパイルが効果のある、パラメタ付き SQL 文を処理するために最適化されています。Participant の JDBC Driver が、プリコンパイルをサポートしている場合、prepareStatement メソッドは、その文をプリコンパイル用としてデータベースに送ります。

形式

```
public java.sql.PreparedStatement prepareStatement(java.lang.String sql) throws java.sql.
SQLException
```

引数

java.lang.String sql：

1 つ以上の'?'IN パラメタ・プレースホルダーを含めることができる SQL 文を指定します。

戻り値

java.sql.PreparedStatement：

プリコンパイルされた SQL 文を含む新しい PreparedStatement オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(10)　prepareStatement(java.lang.String sql, int autoGeneratedKeys)

説明

自動生成キーを取得する機能を持つデフォルトの PreparedStatement オブジェクトを生成します。指定された定数によって、JDBC Driver が自動生成キーを取得可能にするかどうかが指定されます。SQL 文が INSERT 文以外の場合、または自動生成キーを返すことができる SQL 文以外の場合、このパラメタは無視されます。このメソッドは、プリコンパイルが効果的となるパラメタ付き SQL 文を処理するために最適化されています。Participant の JDBC Driver がプリコンパイルをサポートしている場合、prepareStatement メソッドは、その SQL 文をプリコンパイル用としてデータベースに送ります。

形式

```
public java.sql.PreparedStatement prepareStatement(java.lang.String sql, int autoGenerate
dKeys) throws java.sql.SQLException
```

引数

java.lang.String sql：

1 つ以上の'?'IN パラメタ・プレースホルダーを含めることができる SQL 文を指定します。

int autoGeneratedKeys：

自動生成キーを返すかどうかを示すフラグを指定します。
Statement.RETURN_GENERATED_KEYS または Statement.NO_GENERATED_KEYS のどちらかを指定します。

戻り値

自動生成キーを返す機能を持つ、プリコンパイルされた SQL 文を含む新しい PreparedStatement オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- 指定されたパラメタが、自動生成キーを返すかどうかを示す Statement 定数ではない場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver が、定数 Statement.RETURN_GENERATED_KEYS を指定したこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(11) prepareStatement(java.lang.String sql, int resultSetType, int resultSetConcurrency)

説明

指定された型と並行処理で ResultSet オブジェクトを生成する PreparedStatement オブジェクトを生成します。このメソッドは、デフォルトの結果セットの型と並行処理をオーバーライドできます。

形式

```
public java.sql.PreparedStatement prepareStatement(java.lang.String sql, int resultSetTyp
e, int resultSetConcurrency) throws java.sql.SQLException
```

引数

java.lang.String sql：

データベースに送る SQL 文が記述された java.lang.String オブジェクトを指定します。1 つ以上の'?'IN パラメタを含めることができます。

int resultSetType：

結果セットの型を指定します。指定できる結果セットの型は、ResultSet.TYPE_FORWARD_ONLY、ResultSet.TYPE_SCROLL_INSENSITIVE、または ResultSet.TYPE_SCROLL_SENSITIVE のうちの 1 つです。

int resultSetConcurrency：

並行処理の種類を指定します。指定できる並行処理の種類は、ResultSet.CONCUR_READ_ONLY または ResultSet.CONCUR_UPDATABLE のどちらかです。

戻り値

指定された型および並行処理で ResultSet オブジェクトを生成する、プリコンパイルされた SQL 文を含む新しい PreparedStatement オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- 指定されたパラメタが、型および並行処理を示す ResultSet 定数ではない場合

java.sql.SQLFeatureNotSupportedException：

次の場合に例外を送出します。

- Participant の JDBC Driver がこのメソッドをサポートしない場合
- このメソッドが、指定された結果セットの型と結果セットの並行処理でサポートされない場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(12) prepareStatement(java.lang.String sql, java.lang.String[] columnNames)

説明

指定された配列によって、指定された自動生成キーを返す機能を持つデフォルトの PreparedStatement オブジェクトを生成します。この配列には、返される自動生成キーを含むターゲット表の列の名前を含みます。SQL 文が INSERT 文以外の場合、または自動生成キーを返すことができる SQL 文以外の場合、JDBC Driver はこの配列を無視します。

IN パラメタ付きまたは IN パラメタなしの SQL 文をプリコンパイルして、PreparedStatement オブジェクトに格納できます。そのあとに、この SQL 文を複数回効率的に実行するためにこのオブジェクトを使用します。このメソッドは、プリコンパイルが効果的となるパラメタ付き SQL 文を処理するために最適化されています。Participant の JDBC Driver が、プリコンパイルをサポートしている場合、prepareStatement メソッドは、その SQL 文をプリコンパイル用としてデータベースに送ります。

形式

```
public java.sql.PreparedStatement prepareStatement(java.lang.String sql, java.lang.Strin
g[] columnNames) throws java.sql.SQLException
```

引数

java.lang.String sql：

1 つ以上の'?'IN パラメタ・プレースホルダーを含めることができる SQL 文を指定します。

java.lang.String[] columnNames：

挿入された行から返される列を示す列名の配列を指定します。

戻り値

指定された列名の配列によって指定される自動生成キーを返す機能を持つ、プリコンパイルされた文を含む新しい PreparedStatement オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(13) setTransactionIsolation(int level)

説明

この Connection オブジェクトのトランザクション遮断レベルを変更します。指定できるトランザクション遮断レベルの定数は、Connection インタフェースで定義されています。

形式

```
public void setTransactionIsolation(int level) throws java.sql.SQLException
```

引数

int level：

変更後のトランザクション遮断レベルを指定します。次のどれかの Connection 定数を指定します。

- Connection.TRANSACTION_READ_UNCOMMITTED
- Connection.TRANSACTION_READ_COMMITTED
- Connection.TRANSACTION_REPEATABLE_READ
- Connection.TRANSACTION_SERIALIZABLE

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- 引数に指定されたパラメタが Connection 定数以外の場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoPrePreparedException：

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.5 PctoStatement クラス（SQL-Participant 限定）

説明

静的 SQL 文を実行し、作成された結果を返すために使用するクラスです。デフォルトでは、Statement オブジェクトごとに、同時に開くことができる ResultSet オブジェクトは 1 つだけです。したがって、1 つの ResultSet オブジェクトの読み込みが、別の読み込みによって割り込まれると、それぞれが異なった Statement オブジェクトによって生成されたことになります。

メソッド

PctoStatement クラスのメソッドの一覧を次の表に示します。

表 7-12 PctoStatement クラスのメソッドの一覧

メソッド	説明
addBatch(java.lang.String sql)	この Statement オブジェクトの現在のコマンドのリストに、指定された SQL コマンドを追加します。
close()	自動的にクローズされるときに Statement オブジェクトのデータベースと JDBC リソースが解放されるのを待つのではなく、すぐにそれらを解放します。
closeOnCompletion()	この Statement オブジェクトに依存するすべての結果セットがクローズされたときに、この Statement オブジェクトをクローズします。
execute(java.lang.String sql)	複数の結果を返す可能性がある SQL 文を実行します。

メソッド	説明
execute(java.lang.String sql, int autoGeneratedKeys)	複数の結果を返す可能性がある SQL 文を実行し、すべての自動生成キーを検索可能にするかどうかについて JDBC Driver に通知します。
executeBatch()	コマンドのバッチをデータベースに送信して実行し、すべてのコマンドが正常に実行された場合、更新カウントの配列を返します。
executeQuery(java.lang.String sql)	単一の ResultSet オブジェクトを返す、指定された SQL 文を実行します。
executeUpdate(java.lang.String sql)	指定された SQL 文を実行します。
getConnection()	この Statement オブジェクトを生成した Connection オブジェクトを取得します。
getGeneratedKeys()	この Statement オブジェクトを実行した結果として作成された自動生成キーを取得します。
getMoreResults()	Statement オブジェクトの次の結果に移動します。この結果が ResultSet オブジェクトの場合は true を返し、getResultSet メソッドで得られた現在の ResultSet オブジェクトをすべて暗黙にクローズします。
getQueryTimeout()	JDBC Driver が Statement オブジェクトの実行を待つ秒数を取得します。
getResultSet()	ResultSet オブジェクトとして現在の結果を取得します。
getUpdateCount()	更新カウントとして現在の結果を取得します。結果が ResultSet オブジェクトの場合、または結果がない場合は-1 を返します。
getWarnings()	この Statement オブジェクトに関する呼び出しによって報告される最初の警告を取得します。
setEscapeProcessing(boolean enable)	エスケープの処理をオンまたはオフに設定します。
setFetchSize(int rows)	この Statement オブジェクトで作成された ResultSet オブジェクトでより多くの行が必要なときに、データベースから取り出す必要がある行数についてのヒントを Participant の JDBC Driver に提供します。
setQueryTimeout(int seconds)	JDBC Driver が Statement オブジェクトの実行を待つ秒数を設定します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1) addBatch(java.lang.String sql)

説明

この Statement オブジェクトの現在のコマンドのリストに、指定された SQL コマンドを追加します。このリストのコマンドは、executeBatch メソッドを呼び出せばバッチとして実行できます。

このメソッドは、PreparedStatement または CallableStatement で呼び出すことはできません。

形式

```
public void addBatch(java.lang.String sql) throws java.sql.SQLException
```

引数

java.lang.String sql：

通常は、SQL INSERT 文または UPDATE 文を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- JDBC Driver がバッチ更新をサポートしない場合
- メソッドが PreparedStatement または CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) close()

説明

自動的にクローズされるときに Statement オブジェクトのデータベースと JDBC リソースが解放されるのを待つのではなく、すぐにそれらを解放します。

形式

```
public void close()
```

引数

なし。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に例外を送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) closeOnCompletion()

説明

この Statement オブジェクトに依存するすべての結果セットがクローズされたときに、この Statement オブジェクトをクローズします。Statement オブジェクトの実行によって結果セットが生成されない場合、このメソッドは無効になります。

形式

```
public void closeOnCompletion() throws java.sql.SQLException
```

引数

なし。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

メソッドがクローズされたあとに接続された場合に例外を送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(4) execute(java.lang.String sql)

説明

複数の結果を返す可能性がある SQL 文を実行します。幾つかの（一般的でない）状況では、単一の SQL 文が複数の結果セットまたは更新カウントを返すことがあります。複数の結果を返すことが分かっているストアド・プロシージャを実行中、または未知の SQL 文字列を動的に実行中ではないかぎり、通常はこれを無視できます。

execute メソッドは、SQL 文を実行し最初の結果の形式を示します。次に、getResultSet メソッドまたは getUpdateCount メソッドを使用して結果を取得します。そして、getMoreResults メソッドによって引き続き任意の結果の取得（複数可）に移動します。

このメソッドは、PreparedStatement または CallableStatement で呼び出すことはできません。

形式

```
public boolean execute(java.lang.String sql) throws java.sql.SQLException
```

引数

java.lang.String sql：

任意の SQL 文を指定します。

戻り値

最初の結果が ResultSet オブジェクトの場合は true を返します。更新カウントの場合、または結果がない場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- メソッドが PreparedStatement または CallableStatement で呼び出された場合

java.sql.SQLTimeoutException：

Participant の JDBC Driver が、現在実行中の Statement の取り消しを試みたときに送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(5) execute(java.lang.String sql, int autoGeneratedKeys)

説明

複数の結果を返す可能性がある SQL 文を実行し、すべての自動生成キーを検索可能にするかどうかについて JDBC Driver に通知します。SQL 文が INSERT 文以外の場合、または自動生成キーを返すことができる SQL 文以外の場合（そのような文のリストはベンダ固有）、JDBC Driver はこの信号を無視します。

幾つかの（一般的でない）状況では、単一の SQL 文が複数の結果セットまたは更新カウントを返すことがあります。複数の結果を返すことが分かっているストアド・プロシージャを実行中、または未知の SQL 文字列を動的に実行中でないかぎり、通常はこれを無視できます。

execute メソッドは、SQL 文を実行して最初の結果の形式を示します。次に、getResultSet メソッドまたは getUpdateCount メソッドを使用して結果を取得します。そして、getMoreResults メソッドによって引き続き任意の結果の取得（複数可）に移動します。

このメソッドは、PreparedStatement または CallableStatement で呼び出すことはできません。

形式

```
public boolean execute(java.lang.String sql, int autoGeneratedKeys) throws java.sql.SQLException
```

引数

java.lang.String sql：

任意の SQL 文を指定します。

int autoGeneratedKeys：

getGeneratedKeys メソッドを使用して、自動生成キーを検索可能にするかどうかを示す定数（Statement.RETURN_GENERATED_KEYS または Statement.NO_GENERATED_KEYS）を指定します。

戻り値

最初の結果が ResultSet オブジェクトの場合は true を返します。更新カウントであるか、または結果がない場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- メソッドに指定された 2 番目のパラメタが Statement.RETURN_GENERATED_KEYS または Statement.NO_GENERATED_KEYS 以外の場合
- メソッドが PreparedStatement または CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver が定数 Statement.RETURN_GENERATED_KEYS を指定したこのメソッドをサポートしない場合に送出します。

java.sql.SQLTimeoutException：

Participant の JDBC Driver が、現在実行中の Statement の取り消しを試みたときに送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(6) executeBatch()

説明

コマンドのバッチをデータベースに送信して実行し、すべてのコマンドが正常に実行された場合、更新カウントの配列を返します。返される配列の int 要素は、バッチ内のコマンドに対応して順序づけられます（バッチに追加された順序に従って順序づけられます）。executeBatch メソッドによって返される配列内の要素は、次のうちのどれか 1 つになります

- ゼロ以上の数値

  コマンドが正常に処理されたことを示し、コマンドの実行によって影響を受けたデータベース内の行数を指定する更新カウントです。

- SUCCESS_NO_INFO の値

  コマンドは正常に処理されたが、影響を受けた行数が不明なことを示します。

  バッチ更新内のコマンドの 1 つが正常に実行できなかった場合、このメソッドは BatchUpdateException をスローし、Participant の JDBC Driver はバッチ内の残りのコマンドの処理を継続することも、中止することもできます。ただし、JDBC Driver の動作は、特定の DBMS と一貫性を保つ必要があり、常にコマンドの処理を継続するか、または継続しないかのどちらかになります。JDBC Driver が処理を継続する場合は、BatchUpdateException.getUpdateCounts メソッドで返される配列には、バッチ内のコマンドと同じ数の要素が格納され、それらの要素の 1 つ以上が次に該当します。

- EXECUTE_FAILED の値

  正常に実行できなかったコマンドが正常に実行されたことを示します。コマンドが失敗したあとに、JDBC Driver がコマンドの処理を継続した場合にかぎり発生します。

形式

```
public int[] executeBatch() throws java.sql.SQLException
```

引数

なし。

戻り値

バッチ内のコマンドごとに 1 つの要素が格納されている更新カウントの配列を返します。配列の要素は、コマンドがバッチに追加された順序で並べられます。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- JDBC Driver がバッチ文をサポートしない場合

java.sql.BatchUpdateException：

次の場合に例外を送出します。

- データベースに送信されたコマンドの 1 つが正常に実行されなかった場合
- 結果セットを返そうとした場合

java.sql.SQLTimeoutException：

Participant の JDBC Driver が、現在実行中の Statement の取り消しを試みたときに送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(7) executeQuery(java.lang.String sql)

説明

単一の ResultSet オブジェクトを返す、指定された SQL 文を実行します。このメソッドを PreparedStatement で呼び出すことはできません。

形式

```
public ResultSet executeQuery(java.lang.String sql) throws SQLException
```

引数

java.lang.String sql：

データベースに送られる SQL 文です。

戻り値

指定されたクエリによって作成されたデータを含む ResultSet オブジェクトです。null にはなりません。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- 指定された SQL 文が、単一の ResultSet オブジェクトだけ生成する場合
- メソッドが PreparedStatement で呼び出された場合

java.sql.SQLTimeoutException：

Participant の JDBC Driver が、現在実行中の Statement の取り消しを試みたときに送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(8) executeUpdate(java.lang.String sql)

説明

指定された SQL 文を実行します。SQL 文は、INSERT 文、UPDATE 文、DELETE 文、または SQL DDL 文のような何も返さない SQL 文の場合があります。このメソッドを PreparedStatement で呼び出すことはできません。

形式

```
public int executeUpdate(java.lang.String sql) throws SQLException
```

引数

java.lang.String sql：

SQL INSERT 文、UPDATE 文、または DELETE 文のような SQL データ操作言語（DML）文、あるいは DDL 文のような何も返さない SQL 文を指定します。

戻り値

SQL データ操作言語（DML）文の場合は行数を、何も返さない SQL 文の場合は 0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- 指定された SQL 文が、単一の ResultSet オブジェクトだけ生成する場合
- メソッドが PreparedStatement で呼び出された場合

java.sql.SQLTimeoutException：

Participant の JDBC Driver が、現在実行中の Statement の取り消しを試みたときに送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(9) getConnection()

説明

この Statement オブジェクトを生成した Connection オブジェクトを取得します。

形式

```
public java.sql.Connection getConnection() throws java.sql.SQLException
```

引数

なし。

戻り値

この文を生成した接続を返します。

例外

なし。

(10) getGeneratedKeys()

説明

この Statement オブジェクトを実行した結果として作成された自動生成キーを取得します。この Statement オブジェクトがキーを生成しなかった場合は、空の ResultSet オブジェクトが返されます。

形式

```
public java.sql.ResultSet getGeneratedKeys() throws java.sql.SQLException
```

引数

なし。

戻り値

この Statement オブジェクトの実行で生成された自動生成キーを含む ResultSet オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(11) getMoreResults()

説明

Statement オブジェクトの次の結果に移動します。この結果が ResultSet オブジェクトの場合は true を返し、getResultSet メソッドで得られた現在の ResultSet オブジェクトをすべて暗黙にクローズします。

形式

```
public boolean getMoreResults() throws java.sql.SQLException
```

引数

なし。

戻り値

次の結果が ResultSet オブジェクトの場合は true を返します。更新カウントの場合、または結果がない場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(12) getQueryTimeout()

説明

JDBC Driver が Statement オブジェクトの実行を待つ秒数を取得します。この制限を超えると、SQLException がスローされます。

形式

```
public int getQueryTimeout() throws java.sql.SQLException
```

引数

なし。

戻り値

現在のクエリ・タイムアウトの制限値の秒数を返します。0 は無制限を意味します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(13) getResultSet()

説明

ResultSet オブジェクトとして現在の結果を取得します。このメソッドは、1 つの結果に対して 1 回だけ呼び出す必要があります。

形式

```
public java.sql.ResultSet getResultSet() throws java.sql.SQLException
```

引数

なし。

戻り値

ResultSet オブジェクトとしての現在の結果を返します。結果が更新カウントの場合、または結果がない場合は null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(14) getUpdateCount()

説明

更新カウントとして現在の結果を取得します。結果が ResultSet オブジェクトの場合、または結果がない場合は-1 を返します。このメソッドは、1 つの結果に対して 1 回だけ呼び出す必要があります。

形式

```
public int getUpdateCount() throws java.sql.SQLException
```

引数

なし。

戻り値

更新カウントとして現在の結果を返します。現在の結果が ResultSet オブジェクトの場合、または結果がない場合は-1 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合

- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(15) getWarnings()

説明

この Statement オブジェクトに関する呼び出しによって報告される最初の警告を取得します。後続の Statement オブジェクトの警告は、この SQLWarning オブジェクトにチェーンされます。

警告チェーンは、SQL 文が（再）実行されるたびに自動的にクリアされます。このメソッドは、クローズされた Statement オブジェクトの呼び出しには使用しません。使用された場合、SQLException がスローされます。

ResultSet オブジェクトが処理中の場合、ResultSet オブジェクトの読み込みに関連する警告はそのオブジェクトを生成した Statement オブジェクトではなく、すべて ResultSet オブジェクトにチェーンされます。

形式

```
public java.sql.SQLWarning getWarnings() throws java.sql.SQLException
```

引数

なし。

戻り値

最初の SQLWarning オブジェクトを返します。警告がない場合は null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(16) setEscapeProcessing(boolean enable)

説明

エスケープの処理をオンまたはオフに設定します。エスケープ・スキャニングがオンの場合（デフォルト）、JDBC Driver は SQL 文をデータベースに送る前にエスケープ置き換えを実行します。

Connection および DataSource のプロパティの escapeProcessing を使用して、デフォルトのエスケープ処理の動作を変更できます。true（デフォルト）を指定した場合、すべての Statement オブジェクトのエスケープ処理が有効になります。false を指定した場合、すべての Statement オブジェクトのエスケープ処理が無効になります。setEscapeProcessing メソッドを使用して、個々の Statement オブジェクトに対するエスケープ処理の動作を指定できます。

用意された文は、通常は呼び出す前に構文解析されているため、PreparedStatements オブジェクトに対するエスケープ処理を無効にしても効果はありません。

形式

```
public void setEscapeProcessing(boolean enable) throws java.sql.SQLException
```

引数

boolean enable：

エスケープ処理を有効にする場合は true を、無効にする場合は false を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(17) setFetchSize(int rows)

説明

この Statement オブジェクトで作成された ResultSet オブジェクトでより多くの行が必要なときに、データベースから取り出す必要がある行数についてのヒントを Participant の JDBC Driver に提供します。指定された値が 0 の場合、ヒントは無視されます。デフォルト値は 0 です。

形式

```
public void setFetchSize(int rows) throws java.sql.SQLException
```

引数

int rows：

フェッチする行数を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- rows≧0 の条件が満たされていない場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(18) setQueryTimeout(int seconds)

説明

JDBC Driver が Statement オブジェクトの実行を待つ秒数を設定します。デフォルトでは、実行中の SQL 文が完了するまでの時間に制限はありません。制限時間を経過した場合、SQLTimeoutException がスローされます。

形式

```
public void setQueryTimeout(int seconds) throws java.sql.SQLException
```

引数

int seconds：

新しいクエリ・タイムアウトの制限時間の秒数を指定します。0 は無制限を意味します

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドがクローズされたあとに接続された場合
- seconds≧0 の条件が満たされていない場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.6 PctoPreparedStatement クラス（SQL-Participant 限定）

説明

プリコンパイルされた SQL 文を表すクラスです。SQL 文は、プリコンパイルされ、PreparedStatement オブジェクトに格納されます。その後、このオブジェクトは、この文を複数回効率的に実行するために使用できます。IN パラメタ値を設定する設定機能メソッド（setInt、setString など）は、入力パラメ

タの定義された SQL 型と互換のある型を指定する必要があります。例えば、IN パラメタが INTEGER という SQL 型の場合、setInt メソッドを使用するようにしてください。

メソッド

PctoPreparedStatement クラスのメソッドの一覧を次の表に示します。

表 7-13　PctoPreparedStatement クラスのメソッドの一覧

メソッド	説明
addBatch()	この PreparedStatement オブジェクトのコマンドのバッチに、パラメタのセットを追加します。
execute()	この PreparedStatement オブジェクトの、すべての種類の SQL 文を実行します。
executeQuery()	PreparedStatement オブジェクトの SQL クエリを実行し、そのクエリによって生成された ResultSet オブジェクトを返します。
executeUpdate()	PreparedStatement オブジェクトの SQL 文を実行します。
setArray(int parameterIndex, java.sql.Array x)	指定されたパラメタを、指定された java.sql.Array オブジェクトに設定します。
setBigDecimal(int parameterIndex, java.math.BigDecimal x)	指定されたパラメタを、指定された java.math.BigDecimal 値に設定します。
setBinaryStream(int parameterIndex, java.io.InputStream x, int length)	指定されたパラメタを、指定されたバイト数を持つ、指定された入力ストリームに設定します。
setBlob(int parameterIndex, java.io.InputStream inputStream)	指定されたパラメタを InputStream オブジェクトに設定します。
setBoolean(int parameterIndex, boolean x)	指定されたパラメタを、指定された Java の boolean 値に設定します。
setByte(int parameterIndex, byte x)	指定されたパラメタを、指定された Java の byte 値に設定します。
setBytes(int parameterIndex, byte[] x)	指定されたパラメタを、指定された Java の byte 配列に設定します。
setCharacterStream(int parameterIndex, java.io.Reader reader, int length)	指定されたパラメタを、指定された文字数である指定された Reader オブジェクトに設定します。
setClob(int parameterIndex, java.io.Reader reader)	指定されたパラメタを、Reader オブジェクトに設定します。
setDate(int parameterIndex, java.sql.Date x)	アプリケーションを実行している仮想マシンのデフォルトのタイムゾーンを使用して、指定されたパラメタを、指定された java.sql.Date 値に設定します。
setDouble(int parameterIndex, double x)	指定されたパラメタを、指定された Java の double 値に設定します。
setFloat(int parameterIndex, float x)	指定されたパラメタを、指定された Java の float 値に設定します。
setInt(int parameterIndex, int x)	指定されたパラメタを、指定された Java の int 値に設定します。
setLong(int parameterIndex, long x)	指定されたパラメタを、指定された Java の long 値に設定します。

メソッド	説明
setNString(int parameterIndex, java.lang.String value)	指定されたパラメタを、指定された String オブジェクトに設定します。
setNull(int parameterIndex, int sqlType)	指定されたパラメタを SQL NULL に設定します。
setObject(int parameterIndex, java.lang.Object x)	指定されたパラメタの値を、指定されたオブジェクトで設定します。
setObject(int parameterIndex, java.lang.Object x, int targetSqlType)	指定されたパラメタの値を、指定されたオブジェクトで設定します。
setShort(int parameterIndex, short x)	指定されたパラメタを、指定された Java の short 値に設定します。
setString(int parameterIndex, java.lang.String x)	指定されたパラメタを、指定された Java の String 値に設定します。
setTime(int parameterIndex, java.sql.Time x)	指定されたパラメタを、指定された java.sql.Time 値に設定します。
setTimestamp(int parameterIndex, java.sql.Timestamp x)	指定されたパラメタを、指定された java.sql.Timestamp 値に設定します。

PctoStatement クラスから継承された PctoPreparedStatement クラスのメソッドの一覧を次の表に示します。

表 7-14　PctoStatement クラスから継承された PctoPreparedStatement クラスのメソッドの一覧

メソッド	説明
close()	詳細については、「(2)　close()」を参照してください。
getConnection()	詳細については、「(9)　getConnection()」を参照してください。
getGeneratedKeys()	詳細については、「(10)　getGeneratedKeys()」を参照してください。
getUpdateCount()	詳細については、「(14)　getUpdateCount()」を参照してください。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1)　addBatch()

説明

この PreparedStatement オブジェクトのコマンドのバッチに、パラメタのセットを追加します。

形式

```
public void addBatch() throws java.sql.SQLException
```

引数

なし。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PctoPreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) execute()

説明

この PreparedStatement オブジェクトの、すべての種類の SQL 文を実行します。PreparedStatement オブジェクトで用意された SQL 文には、複数の結果を返すものがあります。execute メソッドは、executeQuery メソッドまたは executeUpdate メソッドによって処理される、より簡単な形式の文と同様に、複雑な文も処理します。

execute メソッドは、boolean を返して最初の結果の形式を示します。getResultSet メソッドまたは getUpdateCount メソッドを呼び出して結果を取得します。後続の結果がある場合は、getMoreResults メソッドを呼び出してその結果に移動します。

形式

```
public boolean execute() throws java.sql.SQLException
```

引数

なし。

戻り値

最初の結果が ResultSet オブジェクトの場合は true を返します。更新カウントの場合、または結果がない場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PctoPreparedStatement で呼び出された場合
- 引数がこのメソッドに指定された場合

java.sql.SQLTimeoutException：

Participant の JDBC Driver が、現在実行中の Statement の取り消しを試みたときに送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) executeQuery()

説明

PreparedStatement オブジェクトの SQL クエリを実行し、そのクエリによって生成された ResultSet オブジェクトを返します。

形式

```
public java.sql.ResultSet executeQuery() throws java.sql.SQLException
```

引数

なし。

戻り値

クエリによって作成されたデータを含む ResultSet オブジェクトです。null にはなりません。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合

- このメソッドがクローズされた PctoPreparedStatement で呼び出された場合
- SQL 文が ResultSet オブジェクトを返さない場合

java.sql.SQLTimeoutException：

Participant の JDBC Driver が、現在実行中の Statement の取り消しを試みたときに送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(4) executeUpdate()

説明

PreparedStatement オブジェクトの SQL 文を実行します。実行する SQL 文は、SQL データ操作言語（DML）文（INSERT 文、UPDATE 文、DELETE 文など）であるか、または DDL 文のような何も返さない SQL 文である必要があります。

形式

```
public int executeUpdate() throws java.sql.SQLException
```

引数

なし。

戻り値

SQL データ操作言語（DML）文の場合は行数を返します。何も返さない SQL 文の場合は 0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- このメソッドがクローズされた PctoPreparedStatement で呼び出された場合
- SQL 文が ResultSet オブジェクトを返した場合

java.sql.SQLTimeoutException：

Participant の JDBC Driver が、現在実行中の Statement の取り消しを試みたときに送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(5) setArray(int parameterIndex,java.sql.Array x)

説明

指定されたパラメタを、指定された java.sql.Array オブジェクトに設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL ARRAY 値に変換します。

形式

```
public void setArray(int parameterIndex, java.sql.Array x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.sql.Array x：

SQL ARRAY 値をマッピングする Array オブジェクトを指定します。指定できるオブジェクトは、PctoArray クラスのオブジェクトです。それ以外が指定された場合は、例外を送出します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(6) setBigDecimal(int parameterIndex, java.math.BigDecimal x)

説明

指定されたパラメタを、指定された java.math.BigDecimal 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL NUMERIC 値に変換します。

形式

```
public void setBigDecimal(int parameterIndex, java.math.BigDecimal x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.math.BigDecimal x：

設定したい java.math.BigDecimal の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドがクローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(7) setBinaryStream(int parameterIndex, java.io.InputStream x, int length)

説明

指定されたパラメタを、指定されたバイト数を持つ、指定された入力ストリームに設定します。

なお、LONGVARBINARY パラメタに非常に大きなバイナリ値が入力される場合は、java.io.InputStream オブジェクトを使用すると便利です。ファイルの終わりに達するまで必要に応じてストリームからデータが読み込まれます。

形式

```
public void setBinaryStream(int parameterIndex, java.io.InputStream x, int length) throws
 java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.io.InputStream x：

バイナリ・パラメタ値を含む Java 入力ストリームを指定します。

int length：

ストリームのバイト数を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外、PctoInternalContradictionException 例外、および PctoIOException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOException：

引数で指定した Java 入力ストリームの読み込みに失敗した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(8) setBlob(int parameterIndex, java.io.InputStream inputStream)

説明

指定されたパラメタを InputStream オブジェクトに設定します。

このメソッドは、setBinaryStream メソッドとは異なり、パラメタ値を BLOB でサーバに送信することを Participant の JDBC Driver に通知します。setBinaryStream メソッドを使用すると、Participant の JDBC Driver はパラメタ値を LONGVARBINARY と BLOB のどちらでサーバに送信するかを判断するための余計な処理が発生するおそれがあります。

形式

```
public void setBlob(int parameterIndex, java.io.InputStream inputStream) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.io.InputStream inputStream：

パラメタ値に設定されるデータを含むオブジェクトを指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOException：

引数で指定した Java 入力ストリームの読み込みに失敗した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(9) setBoolean(int parameterIndex, boolean x)

説明

指定されたパラメタを、指定された Java の boolean 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL BIT または BOOLEAN 値に変換します。

形式

```
public void setBoolean(int parameterIndex, boolean x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

boolean x：

設定したい boolean の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(10) setByte(int parameterIndex, byte x)

説明

指定されたパラメタを、指定された Java の byte 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL TINYINT 値に変換します。

形式

```
public void setByte(int parameterIndex, byte x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

byte x：

設定したい byte の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(11) setBytes(int parameterIndex, byte[] x)

説明

指定されたパラメタを、指定された Java の byte 配列に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL VARBINARY または LONGVARBINARY（Participant の JDBC Driver の VARBINARY 値の制限に対する引数のサイズに依存）に変換します。

形式

```
public void setBytes(int parameterIndex, byte[] x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

byte[] x：

設定したい byte 配列の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(12) setCharacterStream(int parameterIndex, java.io.Reader reader, int length)

説明

指定されたパラメタを、指定された文字数である指定された Reader オブジェクトに設定します。

なお、LONGVARCHAR パラメタに非常に大きな UNICODE 値が入力される場合は、java.io.Reader オブジェクトを使用すると便利です。ファイルの終わりに達するまで必要に応じてストリームからデータが読み込まれます。Participant の JDBC Driver は、データを UNICODE からデータベースの char 形式に変換します。

形式

```
public void setCharacterStream(int parameterIndex, java.io.Reader reader, int length) thr
ows java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.io.Reader reader：

Unicode データを格納する java.io.Reader オブジェクトを指定します。

int length：

ストリーム内の文字数を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(13) setClob(int parameterIndex, java.io.Reader reader)

説明

指定されたパラメタを、Reader オブジェクトに設定します。

このメソッドは、setCharacterStream メソッドとは異なり、パラメタ値を CLOB でサーバに送信することを Participant の JDBC Driver に通知します。setCharacterStream メソッドを使用すると、Participant の JDBC Driver はパラメタ値を LONGVARCHAR と CLOB のどちらでサーバに送信するかを判断するための余計な処理が発生するおそれがあります。

形式

```
public void setClob(int parameterIndex, java.io.Reader reader) throws java.sql.SQLExcepti
on
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

java.io.Reader reader：

パラメタ値に設定されるデータを含むオブジェクトを指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドがクローズされた PreparedStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOException：

引数で指定した Java 入力ストリームの読み込みに失敗した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(14) setDate(int parameterIndex, java.sql.Date x)

説明

アプリケーションを実行している仮想マシンのデフォルトのタイムゾーンを使用して、指定されたパラメタを、指定された java.sql.Date 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL DATE 値に変換します。

形式

```
public void setDate(int parameterIndex, java.sql.Date x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

java.sql.Date x：

設定したい java.sql.Date の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(15) setDouble(int parameterIndex, double x)

説明

指定されたパラメタを、指定された Java の double 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL DOUBLE 値に変換します。

形式

```
public void setDouble(int parameterIndex, double x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

double x：

設定したい double の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(16) setFloat(int parameterIndex, float x)

説明

指定されたパラメタを、指定された Java の float 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL REAL 値に変換します。

形式

public void setFloat(int parameterIndex, float x) throws java.sql.SQLException

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

float x：

設定したい float の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(17) setInt(int parameterIndex, int x)

説明

指定されたパラメタを、指定された Java の int 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL INTEGER 値に変換します。

形式

```
public void setInt(int parameterIndex, int x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

int x：

設定したい int の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(18) setLong(int parameterIndex, long x)

説明

指定されたパラメタを、指定された Java の long 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL BIGINT 値に変換します。

形式

```
public void setLong(int parameterIndex, long x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

long x：

設定したい long の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(19) setNString(int parameterIndex, java.lang.String value)

説明

指定されたパラメタを、指定された String オブジェクトに設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL NCHAR、NVARCHAR、または LONGNVARCHAR 値（Participant の JDBC Driver の NVARCHAR 値の制限に対する引数のサイズに依存）に変換します。

形式

```
public void setNString(int parameterIndex, java.lang.String value) throws java.sql.SQLExc
eption
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

java.lang.String value：

設定したい文字列の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- JDBC Driver が各国の文字セットをサポートしない場合
- JDBC Driver がデータ変換エラーが発生したことを検出できる場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(20) setNull(int parameterIndex, int sqlType)

説明

指定されたパラメタを SQL NULL に設定します。パラメタの SQL の型を指定する必要があります。

形式

```
public void setNull(int parameterIndex, int sqlType) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

int sqlType：

java.sql.Types で定義される SQL 型コードを指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

引数 sqlType のデータ型が、ARRAY、BLOB、CLOB、DATALINK、JAVA_OBJECT、NCHAR、NCLOB、NVARCHAR、LONGNVARCHAR、REF、ROWID、SQLXML、または STRUCT で、Participant の JDBC Driver がこのデータ型をサポートしていない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(21) setObject(int parameterIndex, java.lang.Object x)

説明

指定されたパラメタの値を、指定されたオブジェクトで設定します。

JDBC の仕様で、Java の Object 型から SQL 型への標準マッピングが規定されています。指定された引数は、データベースに送られる前に対応する SQL 型に変換されます。

なお、すべてのデータベースで、バックエンドに対する型のない Null の送信が許可されるわけではありません。移植性を最大限にするには、setObject(int parameterIndex, Object x)メソッドの代わり

に、setNull メソッドまたは setObject(int parameterIndex, Object x, int sqlType)メソッドを使用してください。

オブジェクトがこれらのインタフェースを複数実装するクラスであるなどあいまいさがある場合、このメソッドは例外をスローします。

形式

```
public void setObject(int parameterIndex, java.lang.Object x) throws java.sql.SQLExceptio
n
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.lang.Object x：

入力パラメタ値を含むオブジェクトを指定します。Object に指定できる値は、プリミティブデータ型、プリミティブデータ型のラッパークラスのインスタンス、または HMP-PCTO が提供するクラスのインスタンスです。それ以外が指定された場合は、例外を送出します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(22) setObject(int parameterIndex, java.lang.Object x, int targetSqlType)

説明

指定されたパラメタの値を、指定されたオブジェクトで設定します。

形式

```
public void setObject(int parameterIndex, java.lang.Object x, int targetSqlType) throws j
ava.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.lang.Object x：

入力パラメタ値を含むオブジェクトを指定します。Object に指定できる値は、プリミティブデータ型、プリミティブデータ型のラッパークラスのインスタンス、または HMP-PCTO が提供するクラスのインスタンスです。それ以外が指定された場合は、例外を送出します。

int targetSqlType：

java.sql.Types で定義される SQL 型コードを指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(23) setShort(int parameterIndex, short x)

説明

指定されたパラメタを、指定された Java の short 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL SMALLINT 値に変換します。

形式

```
public void setShort(int parameterIndex, short x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

short x：

設定したい short の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(24) setString(int parameterIndex, java.lang.String x)

説明

指定されたパラメタを、指定された Java の String 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQLVARCHAR または LONGVARCHAR 値（Participant の JDBC Driver の VARCHAR 値の制限に対する引数のサイズに依存）に変換します。

形式

```
public void setString(int parameterIndex, java.lang.String x) throws java.sql.SQLExceptio
n
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.lang.String x：

設定したい文字列の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(25) setTime(int parameterIndex, java.sql.Time x)

説明

指定されたパラメタを、指定された java.sql.Time 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL TIME 値に変換します。

形式

```
public void setTime(int parameterIndex, java.sql.Time x) throws java.sql.SQLException
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

java.sql.Time x：

設定したい java.sql.Time の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(26) setTimestamp(int parameterIndex, java.sql.Timestamp x)

説明

指定されたパラメタを、指定された java.sql.Timestamp 値に設定します。データベースに送るときに、Participant の JDBC Driver はこれを SQL TIMESTAMP 値に変換します。

形式

```
public void setTimestamp(int parameterIndex, java.sql.Timestamp x) throws java.sql.SQLExc
eption
```

引数

int parameterIndex：

設定したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

java.sql.Timestamp x：

設定したい java.sql.Timestamp の値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数のインデックスが SQL 文のパラメタ・マーカーに対応しない場合
- データベースアクセスエラーが発生した場合
- このメソッドが、クローズされた PreparedStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.7 PctoCallableStatement クラス（SQL-Participant 限定）

説明

SQL ストアド・プロシージャを実行するときに使用するクラスです。

JDBC API は、ストアド・プロシージャの SQL エスケープ構文を提供します。これによって、すべての RDBMS に対して標準の方法でストアド・プロシージャを呼び出せます。SQL エスケープ構文には、結果パラメタを含める場合と、含めない場合があります。結果パラメタを含める場合は、結果パラメタを OUT パラメタとして登録しておく必要があります。そのほかのパラメタは、入力または出力、ある

いはその両方に対して使用できます。パラメタは、番号によって順番に参照されます。最初のパラメタは 1 です。

IN パラメタ値は、PreparedStatement から継承した set メソッドを使用して設定されます。すべての OUT パラメタの型は、ストアド・プロシージャの実行前に登録する必要があります。実行後に、それらの値はここで提供される get メソッド経由で取り出されます。

CallableStatement は、1 つの ResultSet オブジェクトまたは複数の ResultSet オブジェクトを返すことができます。複数の ResultSet オブジェクトは、Statement から継承した操作を使用して処理されます。

移植性を高めるには出力パラメタを取得する前に、呼び出しの ResultSet オブジェクトと更新カウントを処理するようにします。

メソッド

PctoCallableStatement クラスのメソッドの一覧を次の表に示します。

表 7-15　PctoCallableStatement クラスのメソッドの一覧

メソッド	説明
getArray(int parameterIndex)	指定された JDBC ARRAY パラメタの値を、Java プログラミング言語の java.sql.Array オブジェクトとして取り出します。
getBigDecimal(int parameterIndex, int scale)	指定された JDBC NUMERIC パラメタの値を、引数 scale で指定された小数点以下の桁数を持つ java.math.BigDecimal オブジェクトとして取り出します。
getBlob(int parameterIndex)	指定された JDBC BLOB パラメタの値を、Java プログラミング言語の java.sql.Blob オブジェクトとして取り出します。
getBytes(int parameterIndex)	指定された JDBC BINARY または VARBINARY パラメタの値を、Java プログラミング言語の byte 値の配列として取り出します。
getClob(int parameterIndex)	指定された JDBC CLOB パラメタの値を、Java プログラミング言語の java.sql.Clob オブジェクトとして取り出します。
getDate(int parameterIndex)	指定された JDBC DATE パラメタの値を、java.sql.Date オブジェクトとして取り出します。
getDouble(int parameterIndex)	指定された JDBC DOUBLE パラメタの値を、Java プログラミング言語の double として取り出します。
getFloat(int parameterIndex)	指定された JDBC FLOAT パラメタの値を、Java プログラミング言語の float として取り出します。
getInt(int parameterIndex)	指定された JDBC INTEGER パラメタの値を、Java プログラミング言語の int として取り出します。
getLong(int parameterIndex)	指定された JDBC BIGINT パラメタの値を、Java プログラミング言語の long として取り出します。
getNString(int parameterIndex)	指定された NCHAR、NVARCHAR、または LONGNVARCHAR パラメタの値を、Java プログラミング言語の String として取り出します。
getObject(int parameterIndex)	指定されたパラメタの値を、Java プログラミング言語の Object として取り出します。

メソッド	説明
getObject(int parameterIndex, Class<T> type)	OUT パラメタ parameterIndex の値を表すオブジェクトを返します。変換がサポートされている場合は、そのパラメタの SQL 型を要求された Java データ型に変換します。
getShort(int parameterIndex)	指定された JDBC SMALLINT パラメタの値を、Java プログラミング言語の short として取り出します。
getSQLXML(int parameterIndex)	指定された SQL XML パラメタの値を、Java プログラミング言語の java.sql.SQLXML オブジェクトとして取り出します。
getString(int parameterIndex)	指定された JDBC CHAR、VARCHAR、または LONGVARCHAR パラメタの値を、Java プログラミング言語の String として取り出します。
getTime(int parameterIndex)	指定された JDBC TIME パラメタの値を、java.sql.Time オブジェクトとして取り出します。
getTimestamp(int parameterIndex)	指定された JDBC TIMESTAMP パラメタの値を、java.sql.Timestamp オブジェクトとして取り出します。
registerOutParameter(int parameterIndex, int sqlType)	引数 parameterIndex に指定した OUT パラメタを、JDBC 型 sqlType として登録します。
registerOutParameter(int parameterIndex, int sqlType, int scale)	引数 parameterIndex に指定した OUT パラメタを、JDBC 型 sqlType として登録します。
registerOutParameter(int parameterIndex, int sqlType, java.lang.String typeName)	指定された出力パラメタを登録します。
wasNull()	最後に読み込まれた OUT パラメタの値が SQL NULL かどうかを取得します。

PctoPreparedStatement クラスから継承された PctoCallableStatement クラスのメソッドの一覧を次の表に示します。

表 7-16　PctoPreparedStatement クラスから継承された PctoCallableStatement クラスのメソッドの一覧

メソッド	説明
addBatch()	詳細については、「(1)　addBatch()」を参照してください。
execute()	詳細については、「(2)　execute()」を参照してください。

PctoStatement クラスから継承された PctoCallableStatement クラスのメソッドの一覧を次の表に示します。

表 7-17　PctoStatement クラスから継承された PctoCallableStatement クラスのメソッドの一覧

メソッド	説明
close()	詳細は「(2)　close()」を参照してください。
getUpdateCount()	詳細については、「(14)　getUpdateCount()」を参照してください。

パッケージ名称

jp.co.Hitachi.soft.hmppcto.jdbc.client

(1) getArray(int parameterIndex)

説明

指定された JDBC ARRAY パラメタの値を、Java プログラミング言語の java.sql.Array オブジェクトとして取り出します。

形式

```
public java.sql.Array getArray(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

戻り値

Java プログラミング言語の java.sql.Array オブジェクトとして表されたパラメタ値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) getBigDecimal(int parameterIndex, int scale)

説明

指定された JDBC NUMERIC パラメタの値を、引数 scale で指定された小数点以下の桁数を持つ java.math.BigDecimal オブジェクトとして取り出します。

形式

```
public java.math.BigDecimal getBigDecimal(int parameterIndex, int scale) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

int scale：

小数点以下の桁数を指定します。

戻り値

パラメタ値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) getBlob(int parameterIndex)

説明

指定された JDBC BLOB パラメタの値を、Java プログラミング言語の java.sql.Blob オブジェクトとして取り出します。

形式

```
public java.sql.Blob getBlob(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

戻り値

Java プログラミング言語の java.sql.Blob オブジェクトとして表されたパラメタ値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(4) getBytes(int parameterIndex)

説明

指定された JDBC BINARY または VARBINARY パラメタの値を、Java プログラミング言語の byte 値の配列として取り出します。

形式

```
public byte[] getBytes(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

パラメタ値を返します。値が SQL NULL の場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(5) getClob(int parameterIndex)

説明

指定された JDBC CLOB パラメタの値を、Java プログラミング言語の java.sql.Clob オブジェクトとして取り出します。

形式

```
public java.sql.Clob getClob(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

戻り値

Java プログラミング言語の java.sql.Clob オブジェクトとして表されたパラメタ値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(6) getDate(int parameterIndex)

説明

指定された JDBC DATE パラメタの値を、java.sql.Date オブジェクトとして取り出します。

形式

```
public java.sql.Date getDate(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

戻り値

パラメタ値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(7) getDouble(int parameterIndex)

説明

指定された JDBC DOUBLE パラメタの値を、Java プログラミング言語の double として取り出します。

形式

```
public double getDouble(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

戻り値

パラメタ値を返します。値が存在しない場合は、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(8) getFloat(int parameterIndex)

説明

指定された JDBC FLOAT パラメタの値を、Java プログラミング言語の float として取り出します。

形式

```
public float getFloat(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

戻り値

パラメタ値を返します。値が存在しない場合は、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合

- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(9) getInt(int parameterIndex)

説明

指定された JDBC INTEGER パラメタの値を、Java プログラミング言語の int として取り出します。

形式

```
public int getInt(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

パラメタ値を返します。値が存在しない場合は、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(10) getLong(int parameterIndex)

説明

指定された JDBC BIGINT パラメタの値を、Java プログラミング言語の long として取り出します。

形式

```
public long getLong(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

パラメタ値を返します。値が存在しない場合は、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(11) getNString(int parameterIndex)

説明

指定された NCHAR、NVARCHAR、または LONGNVARCHAR パラメタの値を、Java プログラミング言語の String として取り出します。

固定長の型の JDBC NCHAR の場合、返される String オブジェクトは、データベースで SQL NCHAR 値が持っていた値と正確に同じ値（データベースが付加するパディングを含む）を持ちます。

形式

```
public java.lang.String getNString(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

NCHAR、NVARCHAR、または LONGNVARCHAR 値をマッピングする String オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(12) getObject(int parameterIndex)

説明

指定されたパラメタの値を、Java プログラミング言語の Object として取り出します。値が SQL NULL の場合、JDBC Driver は Java null を返します。

このメソッドは、Java オブジェクトを返します。そのオブジェクトの型は、registerOutParameter メソッドを使用してこのパラメタに登録された JDBC の型に対応します。

HMP-PCTO は、ターゲットの JDBC の型を java.sql.Types.OTHER として登録するデータベース固有の抽象データ型には対応していません。

形式

```
public java.lang.Object getObject(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

OUT パラメタ値を保持する java.lang.Object オブジェクトを返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(13) getObject(int parameterIndex, Class<T> type)

説明

OUT パラメタ parameterIndex の値を表すオブジェクトを返します。変換がサポートされている場合は、そのパラメタの SQL 型を要求された Java データ型に変換します。変換がサポートされていない場合、または type に null が指定された場合、SQLException がスローされます。

形式

```
public <T> T getObject(int parameterIndex, Class<T> type) throws java.sql.SQLException
```

型

<T> T：

この Class オブジェクトでモデル化されるクラスの型です。

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

Class<T> type：

指定されたパラメタの変換後の Java データ型を表すクラスを指定します。

戻り値

OUT パラメタ値を保持する、引数 type で指定したクラスのインスタンスを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 変換がサポートされていない場合
- 引数 type が null の場合
- 上記以外のエラーが発生した場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(14) getShort(int parameterIndex)

説明

指定された JDBC SMALLINT パラメタの値を、Java プログラミング言語の short として取り出します。

形式

```
public short getShort(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

パラメタ値を返します。値が存在しない場合は、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(15) getSQLXML(int parameterIndex)

説明

指定された SQL XML パラメタの値を、Java プログラミング言語の java.sql.SQLXML オブジェクトとして取り出します。

形式

```
public java.sql.SQLXML getSQLXML(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、…となります。

戻り値

SQL XML 値をマッピングする SQLXML オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(16) getString(int parameterIndex)

説明

指定された JDBC CHAR、VARCHAR、または LONGVARCHAR パラメタの値を、Java プログラミング言語の String として取り出します。

固定長の型の JDBC CHAR の場合、返される String オブジェクトは、データベースで SQL CHAR 値が持っていた値と正確に同じ値（データベースが付加するパディングを含む）を持ちます。

形式

```
public java.lang.String getString(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

パラメタ値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(17) getTime(int parameterIndex)

説明

指定された JDBC TIME パラメタの値を、java.sql.Time オブジェクトとして取り出します。

形式

```
public java.sql.Time getTime(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

パラメタ値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(18) getTimestamp(int parameterIndex)

説明

指定された JDBC TIMESTAMP パラメタの値を、java.sql.Timestamp オブジェクトとして取り出します。

形式

```
public java.sql.Timestamp getTimestamp(int parameterIndex) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

戻り値

パラメタ値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(19) registerOutParameter(int parameterIndex, int sqlType)

説明

引数 parameterIndex に指定した OUT パラメタを、JDBC 型 sqlType として登録します。ストアド・プロシージャが実行される前に、すべての OUT パラメタを登録しておく必要があります。

OUT パラメタ用に引数 sqlType で指定した JDBC の型は、そのパラメタの値を読み込むために get メソッドで使用する Java の型を判定します。HMP-PCTO は、データベース固有の型には対応していません。sqlType に java.sql.Types.OTHER を指定した場合は、例外を送出します。

パラメタが JDBC 型 NUMERIC または DECIMAL の場合は、スケール値を受け入れるバージョンの registerOutParameter を使用します。

形式

```
public void registerOutParameter(int parameterIndex, int sqlType) throws java.sql.SQLExce
ption
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

int sqlType：

java.sql.Types によって定義される JDBC 型のコードを指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

引数 sqlType のデータ型が、ARRAY、BLOB、CLOB、DATALINK、JAVA_OBJECT、NCHAR、NCLOB、NVARCHAR、LONGNVARCHAR、REF、ROWID、SQLXML、または STRUCT で、Participant の JDBC Driver がこのデータ型をサポートしていない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(20) registerOutParameter(int parameterIndex, int sqlType, int scale)

説明

引数 parameterIndex に指定した OUT パラメタを、JDBC 型 sqlType として登録します。ストアド・プロシージャが実行される前に、すべての OUT パラメタを登録しておく必要があります。

OUT パラメタ用に引数 sqlType で指定した JDBC の型は、そのパラメタの値を読み込むために get メソッドで使用する Java の型を判定します。

パラメタが JDBC の型 NUMERIC または DECIMAL の場合は、この registerOutParameter メソッドを使用します。

形式

```
public void registerOutParameter(int parameterIndex, int sqlType, int scale) throws java.
sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

int sqlType：

java.sql.Types によって定義される SQL 型のコードを指定します。

int scale：

設定する小数点以下の桁数を指定します。0 以上の数値を指定してください。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

引数 sqlType のデータ型が、ARRAY、BLOB、CLOB、DATALINK、JAVA_OBJECT、NCHAR、NCLOB、NVARCHAR、LONGNVARCHAR、REF、ROWID、SQLXML、または STRUCT で、Participant の JDBC Driver がこのデータ型をサポートしていない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(21) registerOutParameter(int parameterIndex, int sqlType, java.lang.String typeName)

説明

指定された出力パラメタを登録します。ユーザ定義出力パラメタまたは REF 出力パラメタには、このバージョンの registerOutParameter メソッドを使用してください。ユーザ定義型の例には、STRUCT、DISTINCT、JAVA_OBJECT、および名前付き配列があります。

ストアド・プロシージャが実行される前に、すべての OUT パラメタを登録しておいてください。

ユーザ定義パラメタの場合は、そのパラメタの完全指定 SQL 型名も指定してください。REF パラメタの場合は、参照する型の完全指定型名を指定してください。型コードや型名の情報を必要としない Participant の JDBC Driver はそれを無視します。ただし、移植性を高めるためには、アプリケーションはユーザ定義パラメタおよび REF パラメタに対して、これらの値を提供する必要があります。このメソッドは、ユーザ定義パラメタと REF パラメタを対象としていますが、任意の JDBC 型のパラメタを登録するために使用することもできます。パラメタがユーザ定義の型または REF 型を持たない場合、typeName パラメタは無視されます。

出力パラメタの値を読み込むときには、そのパラメタの登録済み SQL 型に対応する Java 型用の取得メソッドを使用してください。

形式

```
public void registerOutParameter(int parameterIndex, int sqlType, java.lang.String typeName) throws java.sql.SQLException
```

引数

int parameterIndex：

取得したいパラメタのインデックスを指定します。最初のパラメタは 1、2 番目のパラメタは 2、… となります。

int sqlType：

java.sql.Types の値を指定します。

java.lang.String typeName：

SQL 構造化型の完全指定された名前を指定してください。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合

- メソッドが、クローズされた CallableStatement で呼び出された場合

java.sql.SQLFeatureNotSupportedException：

引数 sqlType のデータ型が、ARRAY、BLOB、CLOB、DATALINK、JAVA_OBJECT、NCHAR、NCLOB、NVARCHAR、LONGNVARCHAR、REF、ROWID、SQLXML、または STRUCT で、Participant の JDBC Driver がこのデータ型をサポートしていない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(22) wasNull()

説明

最後に読み込まれた OUT パラメタの値が SQL NULL かどうかを取得します。読み込み用メソッドの呼び出しのあとに、このメソッドを呼び出してください。そうでないと、null かどうかを判別するために使用する値がありません。

形式

```
public boolean wasNull() throws java.sql.SQLException
```

引数

なし。

戻り値

最後に読み込まれたパラメタが SQL NULL の場合は true を、それ以外の場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた CallableStatement で呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.8 PctoResultSet クラス（SQL-Participant 限定）

説明

データベースの結果を表すデータの表のクラスです。データベースの結果セットを表すデータの表は、通常、データベースに照会する文を実行すると生成されます。ResultSet オブジェクトは、カーソルがデータの現在の行を指し示すよう維持します。初期状態では、カーソルは最初の行の先頭に配置されます。next メソッドによって、カーソルは次の行に移動します。ResultSet は、現在の行から列値を取得する getter メソッド（getBoolean、getLong など）を提供します。値は、列のインデックス番号か列名のどちらかを使って取得できます。getter メソッドへの入力として使用される列名では、大文字と小文字は区別されません。列名で getter メソッドが呼び出され、複数の列が同じ名前を持つ場合は、最初に一致する列の値が返されます。ResultSet オブジェクトは、このオブジェクトを生成した Statement オブジェクトが閉じられるとき、再実行されるとき、または一連の複数の結果から次の結果を取り出すために使われるときに、自動的に閉じられます。

メソッド

PctoResultSet クラスのメソッドの一覧を次の表に示します。

表 7-18 PctoResultSet クラスのメソッドの一覧

メソッド	説明
absolute(int row)	この ResultSet オブジェクト内の指定された行番号にカーソルを移動します。
close()	自動的にクローズされるときに ResultSet オブジェクトのデータベースと JDBC リソースが解放されるのを待つのではなく、すぐにそれらを解放します。
getArray(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Array オブジェクトとして取り出します。
getArray(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Array オブジェクトとして取り出します。
getBigDecimal(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、完全な精度の java.math.BigDecimal として取り出します。
getBigDecimal(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、完全な精度の java.math.BigDecimal として取り出します。
getBlob(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Blob オブジェクトとして取り出します。
getBlob(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Blob オブジェクトとして取り出します。

メソッド	説明
getBoolean(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の boolean として取り出します。
getBoolean(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の boolean として取り出します。
getByte(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の byte として取り出します。
getByte(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の byte として取り出します。
getBytes(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の byte 配列として取り出します。
getBytes(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の byte 配列として取り出します。
getClob(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Clob オブジェクトとして取り出します。
getClob(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Clob オブジェクトとして取り出します。
getDate(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Date オブジェクトとして取り出します。
getDate(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Date オブジェクトとして取り出します。
getDouble(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の double として取り出します。
getDouble(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の double として取り出します。
getFloat(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の float として取り出します。
getFloat(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の float として取り出します。
getInt(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の int として取り出します。
getInt(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の int として取り出します。
getLong(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の long として取り出します。
getLong(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の long として取り出します。
getMetaData()	この ResultSet オブジェクトの列の数、型、およびプロパティを取得します。

メソッド	説明
getNString(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の String として取り出します。
getNString(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の String として取り出します。
getObject(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Object として取り出します。
getObject(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Object として取り出します。
getObject(int columnIndex, Class<T> type)	この ResultSet オブジェクトの現在の行の指定された列の値を取得します。変換がサポートされている場合は、その列の SQL 型を要求された Java データ型に変換します。
getObject(java.lang.String columnLabel, Class<T> type)	この ResultSet オブジェクトの現在行にある指定された列の値を取得します。変換がサポートされている場合は、その列の SQL 型を要求された Java データ型に変換します。
getShort(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の short として取り出します。
getShort(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の short として取り出します。
getSQLXML(int columnIndex)	この ResultSet の現在行にある指定された列の値を、Java プログラミング言語の java.sql.SQLXML オブジェクトとして取り出します。
getSQLXML(java.lang.String columnLabel)	この ResultSet の現在行にある指定された列の値を、Java プログラミング言語の java.sql.SQLXML オブジェクトとして取り出します。
getString(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の String として取り出します。
getString(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の String として取り出します。
getTime(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Time オブジェクトとして取り出します。
getTime(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Time オブジェクトとして取り出します。
getTimestamp(int columnIndex)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Timestamp オブジェクトとして取り出します。
getTimestamp(java.lang.String columnLabel)	この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Timestamp オブジェクトとして取り出します。
getType()	この ResultSet オブジェクトの型を返します。

メソッド	説明
isClosed()	この ResultSet オブジェクトがクローズされているかどうかを取得します。
next()	カーソルを現在の位置から順方向に 1 行移動します。
wasNull()	最後に読み込まれた列の値が SQL NULL であるかどうかを通知します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1) absolute(int row)

説明

この ResultSet オブジェクト内の指定された行番号にカーソルを移動します。

指定された行番号が正の値の場合、カーソルは結果セットの先頭を基準にして指定された行番号に移動します。最初の行が行 1、2 番目が行 2 となります。

指定された行番号が負の値の場合、カーソルは結果セットの終端に対する絶対行位置に移動します。例えば、absolute(-1)メソッドを呼び出すと、カーソルは最終行に移動します。absolute(-2)メソッドを呼び出すと、カーソルは最終行の前の行に移動します。

指定された行番号が 0 の場合、カーソルは最初の行の前に移動します。

結果セット内の先頭行または最終行を越える位置にカーソルを移動しようとすると、カーソルは先頭行の前または最終行の後ろに置かれます。

形式

```
public boolean absolute(int row) throws java.sql.SQLException
```

引数

int row：

カーソルの移動先の行番号を指定します。0 を指定した場合、カーソルは最初の行の前に置かれます。正の番号を指定した場合、行番号は結果セットの先頭からカウントされます。負の番号を指定した場合、行番号は結果セットの終端からカウントされます。

戻り値

カーソルがこの ResultSet オブジェクト内の位置に移動する場合は true を、最初の行の前または最終行の後ろにカーソルが位置する場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合

- メソッドが、クローズされた結果セットで呼び出された場合
- 結果セットのタイプが TYPE_FORWARD_ONLY の場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) close()

説明

自動的にクローズされるときに ResultSet オブジェクトのデータベースと JDBC リソースが解放されるのを待つのではなく、すぐにそれらを解放します。

形式

```
public void close()
```

引数

なし。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に例外を送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) getArray(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Array オブジェクトとして取り出します。

形式

```
public java.sql.Array getArray(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

指定された列の SQL ARRAY 値を表す java.sql.Array オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(4) getArray(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Array オブジェクトとして取り出します。

形式

```
public java.sql.Array getArray(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

指定された列の SQL ARRAY 値を表す java.sql.Array オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(5) getBigDecimal(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、完全な精度の java.math.BigDecimal として取り出します。

形式

```
public java.math.BigDecimal getBigDecimal(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

全精度の列の値を返します。値が存在しない場合、返される値は Java プログラミング言語の null です。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(6) getBigDecimal(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、完全な精度の java.math.BigDecimal として取り出します。

形式

```
public java.math.BigDecimal getBigDecimal(java.lang.String columnLabel) throws java.sql.S
QLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

全精度の列の値を返します。値が存在しない場合、返される値は Java プログラミング言語の null です。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(7) getBlob(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Blob オブジェクトとして取り出します。

形式

```
public java.sql.Blob getBlob(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

指定された列の SQL BLOB 値を表す Blob オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合

- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(8) getBlob(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Blob オブジェクトとして取り出します。

形式

```
public java.sql.Blob getBlob(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

指定された列の SQL BLOB 値を表す Blob オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(9) getBoolean(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の boolean として取り出します。

なお、次のどちらかに該当する場合は、false が返されます。

- 指定された列のデータ型が CHAR または VARCHAR で、"0"を含む場合
- データ型が BIT、TINYINT、SMALLINT、INTEGER、または BIGINT で、0 を含む場合

次のどちらかに該当する場合は、true が返されます。

- 指定された列のデータ型が CHAR または VARCHAR で、"1"を含む場合
- データ型が BIT、TINYINT、SMALLINT、INTEGER、または BIGINT で、1 を含む場合

形式

```
public boolean getBoolean(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合、false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(10) getBoolean(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の boolean として取り出します。

なお、次のどちらかに該当する場合は、false が返されます。

- 指定された列のデータ型が CHAR または VARCHAR で、"0"を含む場合
- データ型が BIT、TINYINT、SMALLINT、INTEGER、または BIGINT で、0 を含む場合

次のどちらかに該当する場合は、true が返されます。

- 指定された列のデータ型が CHAR または VARCHAR で、"1"を含む場合
- データ型が BIT、TINYINT、SMALLINT、INTEGER、または BIGINT で、1 を含む場合

形式

```
public boolean getBoolean(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合、false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(11) getByte(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の byte として取り出します。

形式

```
public byte getByte(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(12) getByte(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の byte として取り出します。

形式

```
public byte getByte(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(13) getBytes(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の byte 配列として取り出します。バイトは JDBC Driver によって返された raw 値を表します。

形式

```
public byte[] getBytes(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(14) getBytes(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の byte 配列として取り出します。byte 配列には、JDBC Driver によって返された raw 値が格納されます。

形式

```
public byte[] getBytes(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(15) getClob(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Clob オブジェクトとして取り出します。

形式

```
public java.sql.Clob getClob(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

指定された列の SQL CLOB 値を表す Clob オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(16) getClob(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Clob オブジェクトとして取り出します。

形式

```
public java.sql.Clob getClob(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

指定された列の SQL CLOB 値を表す Clob オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(17) getDate(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Date オブジェクトとして取り出します。

形式

```
public java.sql.Date getDate(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(18) getDate(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Date オブジェクトとして取り出します。

形式

```
public java.sql.Date getDate(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(19) getDouble(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の double として取り出します。

形式

```
public double getDouble(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(20) getDouble(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の double として取り出します。

形式

```
public double getDouble(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(21) getFloat(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の float として取り出します。

形式

```
public float getFloat(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合は、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(22) getFloat(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の float として取り出します。

形式

```
public float getFloat(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合

- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(23) getInt(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の int として取り出します。

形式

```
public int getInt(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(24) getInt(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の int として取り出します。

形式

```
public int getInt(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(25) getLong(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の long として取り出します。

形式

```
public long getLong(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(26) getLong(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の long として取り出します。

形式

```
public long getLong(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(27) getMetaData()

説明

この ResultSet オブジェクトの列の数、型、およびプロパティを取得します。

形式

```
public java.sql.ResultSetMetaData getMetaData() throws java.sql.SQLException
```

引数

なし。

戻り値

この ResultSet オブジェクトの列の記述を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(28) getNString(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の String として取り出します。NCHAR、NVARCHAR、および LONGNVARCHAR 列にアクセスするときに使用します。

形式

```
public java.lang.String getNString(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(29) getNString(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の String として取り出します。NCHAR、NVARCHAR、および LONGNVARCHAR 列にアクセスするときに使用します。

形式

```
public java.lang.String getNString(java.lang.String columnLabel) throws java.sql.SQLExcep
tion
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(30) getObject(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Object として取り出します。

このメソッドは、指定された列の値を Java オブジェクトとして返します。Java オブジェクトの型は、JDBC の仕様で指定されている組み込み型のマッピングに従って、列の SQL 型に対応するデフォルトの Java オブジェクト型になります。値が SQL NULL の場合、JDBC Driver は Java null を返します。

HMP-PCTO は、データベース固有の抽象データ型には対応しません。

形式

```
public java.lang.Object getObject(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を保持している java.lang.Object を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(31) getObject(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の Object として取り出します。

このメソッドは、指定された列の値を Java オブジェクトとして返します。Java オブジェクトの型は、JDBC の仕様で指定されている組み込み型のマッピングに従って、列の SQL 型に対応するデフォルトの Java オブジェクト型になります。値が SQL NULL の場合、JDBC Driver は Java null を返します。

HMP-PCTO は、データベース固有の抽象データ型には対応しません。

形式

```
public java.lang.Object getObject(java.lang.String columnLabel) throws java.sql.SQLExcept
ion
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を保持している java.lang.Object を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(32) getObject(int columnIndex, Class<T> type)

説明

この ResultSet オブジェクトの現在の行の指定された列の値を取得します。変換がサポートされている場合は、その列の SQL 型を要求された Java データ型に変換します。変換がサポートされていない場合、または type に null が指定された場合は、SQLException がスローされます。

形式

```
public <T>T getObject(int columnIndex, Class<T> type) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

Class<T> type：

指定された列の変換後の Java データ型を表すクラスを指定します。

戻り値

列値を保持する type のインスタンスを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

Java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合
- 変換がサポートされていない場合
- type が null の場合
- 変換でエラーが発生した場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(33) getObject(java.lang.String columnLabel, Class<T> type)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を取得します。変換がサポートされている場合は、その列の SQL 型を要求された Java データ型に変換します。変換がサポートされていない場合、または type に null が指定された場合は、SQLException がスローされます。

形式

```
public <T>T getObject(java.lang.String columnLabel, Class<T> type) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

Class<T> type：

指定された列の変換後の Java データ型を表すクラスを指定します。

戻り値

列値を保持する type のインスタンスを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合
- 変換がサポートされていない場合
- type が null の場合
- 変換でエラーが発生した場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(34) getShort(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の short として取り出します。

形式

```
public short getShort(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(35) getShort(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の short として取り出します。

形式

```
public short getShort(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合、0 を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(36) getSQLXML(int columnIndex)

説明

この ResultSet の現在行にある指定された列の値を、Java プログラミング言語の java.sql.SQLXML オブジェクトとして取り出します。

形式

```
public java.sql.SQLXML getSQLXML(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

SQL XML 値をマッピングする SQLXML オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(37) getSQLXML(java.lang.String columnLabel)

説明

この ResultSet の現在行にある指定された列の値を、Java プログラミング言語の java.sql.SQLXML オブジェクトとして取り出します。

形式

```
public java.sql.SQLXML getSQLXML(java.lang.String columnLabel) throws java.sql.SQLExcepti
on
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

SQL XML 値をマッピングする SQLXML オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(38) getString(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の String として取り出します。

形式

```
public java.lang.String getString(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(39) getString(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の String として取り出します。

形式

```
public java.lang.String getString(java.lang.String columnLabel) throws java.sql.SQLExcept
ion
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(40) getTime(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Time オブジェクトとして取り出します。

形式

```
public java.sql.Time getTime(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(41) getTime(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Time オブジェクトとして取り出します。

形式

```
public java.sql.Time getTime(java.lang.String columnLabel) throws java.sql.SQLException
```

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(42) getTimestamp(int columnIndex)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Timestamp オブジェクトとして取り出します。

形式

```
public java.sql.Timestamp getTimestamp(int columnIndex) throws java.sql.SQLException
```

引数

int columnIndex：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(43) getTimestamp(java.lang.String columnLabel)

説明

この ResultSet オブジェクトの現在行にある指定された列の値を、Java プログラミング言語の java.sql.Timestamp オブジェクトとして取り出します。

形式

public java.sql.Timestamp getTimestamp(java.lang.String columnLabel) throws java.sql.SQLException

引数

java.lang.String columnLabel：

SQL AS 節として指定された列のラベルを指定します。SQL AS 節が指定されていない場合、このラベルには列の名前を指定します。

戻り値

列値を返します。値が存在しない場合は、null を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- 引数が無効な場合
- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(44) getType()

説明

この ResultSet オブジェクトの型を返します。型は、結果セットを作成した Statement オブジェクトによって決まります。

形式

```
public int getType() throws java.sql.SQLException
```

引数

なし。

戻り値

ResultSet.TYPE_FORWARD_ONLY（1003）、ResultSet.TYPE_SCROLL_INSENSITIVE（1004）、または ResultSet.TYPE_SCROLL_SENSITIVE（1005）を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(45) isClosed()

説明

この ResultSet オブジェクトがクローズされているかどうかを取得します。ResultSet オブジェクトで close メソッドが呼び出されるか、または自動的にクローズされる場合に、ResultSet オブジェクトがクローズされます。

形式

```
public boolean isClosed() throws java.sql.SQLException
```

引数

なし。

戻り値

この ResultSet オブジェクトがクローズされている場合は true を、オープン状態の場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(46) next()

説明

カーソルを現在の位置から順方向に 1 行移動します。ResultSet のカーソルは、初期状態では最初の行の前に位置づけられています。next メソッドの最初の呼び出しによって、最初の行が現在行になります。2 番目の呼び出しによって 2 行目が現在行になり、以下同様に続きます。next メソッドの呼び出しで false が返された場合、カーソルは最終行の後ろに位置します。現在行で入力ストリームがオープンしている場合、next メソッドへの呼び出しは暗黙的にそのストリームをクローズさせます。

形式

```
public boolean next() throws java.sql.SQLException
```

引数

なし。

戻り値

新しい現在の行が有効の場合は true を、行がそれ以上存在しない場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(47) wasNull()

説明

最後に読み込まれた列の値が SQL NULL であるかどうかを通知します。最初に getter メソッドの 1 つを呼び出して、列の値を読み込みます。次に wasNull メソッドを呼び出して、読み込まれた値が SQL NULL かどうかを判定します。

形式

```
public boolean wasNull() throws java.sql.SQLException
```

引数

なし。

戻り値

最後に読み込まれた列の値が SQL NULL の場合は true を、それ以外の場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- データベースアクセスエラーが発生した場合
- メソッドが、クローズされた結果セットで呼び出された場合

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.9 PctoDatabaseMetaData クラス（SQL-Participant 限定）

説明

データベースに関する包括的な情報を提供するクラスです。

このクラスは、基本となる DBMS の操作方法を見つける必要があるツールで使用します。複数の DBMS と一緒に使用されるアプリケーションで使います。

メソッド

PctoDatabaseMetaData クラスのメソッドの一覧を次の表に示します。

表 7-19　PctoDatabaseMetaData クラスのメソッドの一覧

メソッド	説明
getDatabaseProductName()	Connection オブジェクトが接続を表すデータベース製品の名前を取得します。
supportsMultipleResultSets()	Connection オブジェクトが接続を表すデータベースで、execute メソッドの単一の呼び出しからの複数の ResultSet オブジェクトの取得がサポートされているかどうかを取得します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1)　getDatabaseProductName()

説明

Connection オブジェクトが接続を表すデータベース製品の名前を取得します。

形式

```
public java.lang.String getDatabaseProductName() throws java.sql.SQLException
```

引数

なし。

戻り値

データベース製品の名前を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) supportsMultipleResultSets()

説明

Connection オブジェクトが接続を表すデータベースで、execute メソッドの単一の呼び出しからの複数の ResultSet オブジェクトの取得がサポートされているかどうかを取得します。

形式

```
public boolean supportsMultipleResultSets() throws java.sql.SQLException
```

引数

なし。

戻り値

ResultSet オブジェクトの取得がサポートされている場合は true を、そうでない場合は false を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.10 PctoResultSetMetadata クラス（SQL-Participant 限定）

説明

ResultSet オブジェクトの列の型とプロパティに関する情報を取得するのに使用するクラスです。

メソッド

PctoResultSetMetadata クラスのメソッドの一覧を次の表に示します。

表 7-20 PctoResultSetMetadata クラスのメソッドの一覧

メソッド	説明
getColumnClassName(int column)	Java クラスの完全指定された名前を返します。

メソッド	説明
getColumnCount()	この ResultSet オブジェクトの列数を返します。
getColumnLabel(int column)	印刷や表示に使用する、指定された列の推奨タイトルを取得します。
getColumnName(int column)	指定された列の名前を取得します。
getColumnType(int column)	指定された列の SQL 型を取得します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1) getColumnClassName(int column)

説明

Java クラスの完全指定された名前を返します。

形式

```
public java.lang.String getColumnClassName(int column) throws java.sql.SQLException
```

引数

int column：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

Java プログラミング言語のクラスの完全指定された名前を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) getColumnCount()

説明

この ResultSet オブジェクトの列数を返します。

形式

```
public int getColumnCount() throws java.sql.SQLException
```

引数

なし。

戻り値

列数を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) getColumnLabel(int column)

説明

印刷や表示に使用する、指定された列の推奨タイトルを取得します。通常、推奨タイトルは、SQL AS 節として指定されます。SQL AS 節が指定されていない場合、getColumnLabel から返される値は、getColumnName メソッドによって返される値と同じになります。

形式

```
public java.lang.String getColumnLabel(int column) throws java.sql.SQLException
```

引数

int column：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列の推奨タイトルを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(4) getColumnName(int column)

説明

指定された列の名前を取得します。

形式

```
public java.lang.String getColumnName(int column) throws java.sql.SQLException
```

引数

int column：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

列名を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(5) getColumnType(int column)

説明

指定された列の SQL 型を取得します。

形式

```
public int getColumnType(int column) throws java.sql.SQLException
```

引数

int column：

列インデックスを指定します。最初の列は 1、2 番目の列は 2、…となります。

戻り値

java.sql.Types からの SQL 型を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

データベースアクセスエラーが発生した場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.11 PctoSQLXML クラス（SQL-Participant 限定）

説明

Java プログラミング言語で SQL XML 型を表現するクラスです。XML は、XML 値をデータベース表の行内に列の値として格納する組み込みの型です。デフォルトでは、JDBC Driver は SQLXML オブジェクトをデータそのものではなく、XML データへの論理ポインタとして格納します。SQLXML オブジェクトは、それが生成されたトランザクションの期間中に有効です。

PctoResultSet または PctoCallableStatement の各クラスのメソッド（getSQLXML など）を使用すると、XML 値にアクセスできます。

メソッド

PctoSQLXML クラスのメソッドの一覧を次の表に示します。

表 7-21　PctoSQLXML クラスのメソッドの一覧

メソッド	説明
free()	このメソッドは、このオブジェクトをクローズし、このオブジェクトが保持していたリソースを解放します。
getString()	この SQLXML インスタンスによって指定された XML 値の文字列表現を返します。
setString(java.lang.String value)	この SQLXML インスタンスによって指定された XML 値を、指定された String 表現に設定します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1)　free()

説明

このメソッドは、このオブジェクトをクローズし、このオブジェクトが保持していたリソースを解放します。このメソッドが呼び出されると、SQL XML オブジェクトは無効になり、読み込み可能にも書き込み可能にもなりません。

なお、free メソッドを呼び出した場合に、free 以外のメソッドを呼び出すと、SQLException がスローされます。

また、free メソッドを複数回呼び出した場合、2 回目以降の free メソッドの呼び出しでは何も行われません。

形式

```
public void free() throws java.sql.SQLException
```

引数

なし。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

XML 値を解放するときにエラーが発生した場合に送出します。

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) getString()

説明

この SQLXML インスタンスによって指定された XML 値の文字列表現を返します。String のエンコーディングは Unicode です。ResultSet の指定された列が java.sql.Types 型の SQLXML である場合、このメソッドの動作は ResultSet.getString()と同じになります。

このメソッドが呼び出されると、SQL XML オブジェクトは読み込みができなくなります。実装によっては、書き込みができなくなることもあります。

形式

```
public java.lang.String getString() throws java.sql.SQLException
```

引数

なし。

戻り値

この SQLXML インスタンスによって指定された XML 値の文字列表現を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- XML 値を処理するときにエラーが発生した場合
- 状態が読み込み不可能である場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) setString(java.lang.String value)

説明

この SQLXML インスタンスによって指定された XML 値を、指定された String 表現に設定します。String のエンコーディングは Unicode です。このメソッドが呼び出されると、SQL XML オブジェクトは書き込みができなくなります。実装によっては、読み込みができなくなることもあります。

形式

```
public void setString(java.lang.String value) throws java.sql.SQLException
```

引数

java.lang.String value：

XML 値を指定します。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- XML 値を処理するときにエラーが発生した場合
- 状態が書き込み不可能である場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.12　PctoBlob クラス（SQL-Participant 限定）

説明

Java プログラミング言語で SQL BLOB 値を表現するクラスです。SQL の BLOB は、バイナリ・ラージ・オブジェクトをデータベースの表の行内に列の値として格納する組み込みの型です。

ResultSet、CallableStatement、および PreparedStatement の各インタフェースのメソッド（getBlob、setBlob など）を使用すると、SQL の BLOB 値にアクセスできます。

メソッド

PctoBlob クラスのメソッドの一覧を次の表に示します。

表 7-22　PctoBlob クラスのメソッドの一覧

メソッド	説明
getBinaryStream()	この Blob インスタンスによって指定された BLOB 値をストリームとして取り出します。
getBytes(long pos, int length)	この Blob オブジェクトが表す BLOB 値の全部または一部を byte 配列として取り出します。
length()	この Blob オブジェクトによって指定される BLOB 値内にあるバイトの数を返します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1)　getBinaryStream()

説明

この Blob インスタンスによって指定された BLOB 値をストリームとして取り出します。

形式

```
public java.io.InputStream getBinaryStream() throws java.sql.SQLException
```

引数

なし。

戻り値

BLOB データを格納するストリームを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

BLOB 値にアクセスするときにエラーが発生した場合に送出します。

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) getBytes(long pos, int length)

説明

この Blob オブジェクトが表す BLOB 値の全部または一部を byte 配列として取り出します。byte 配列には、引数 pos に指定された位置から、引数 length に指定された連続するバイト数が格納されます。

形式

```
public byte[] getBytes(long pos, int length) throws java.sql.SQLException
```

引数

long pos：

BLOB 値内の抽出される最初のバイトの位置（序数）を指定します。最初のバイトの位置は 1 です。

int length：

コピーされる連続したバイト数を指定します。長さの値は、0 以上である必要があります。

戻り値

この Blob オブジェクトによって指定された BLOB 値内の、引数 pos に指定された位置から、引数 length に指定された連続するバイト数が格納されている配列を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- BLOB 値にアクセスするときにエラーが発生した場合
- 引数 pos が 1 より小さいか、または引数 length が 0 より小さい場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) length()

説明

この Blob オブジェクトによって指定される BLOB 値内にあるバイトの数を返します。

形式

```
public long length() throws java.sql.SQLException
```

引数

なし。

戻り値

BLOB の長さ（バイト単位）を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

BLOB の長さにアクセスするときにエラーが発生した場合に送出します。

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.13　PctoClob クラス（SQL-Participant 限定）

説明

Java プログラミング言語で SQL CLOB 型を表現するクラスです。SQL の CLOB は、キャラクタ・ラージ・オブジェクトをデータベースの表の行内の列値として格納する組み込みの型です。

ResultSet、CallableStatement、および PreparedStatement の各インタフェースのメソッド（getClob、setClob など）を使用すると、SQL の CLOB 値にアクセスできます。

メソッド

PctoClob クラスのメソッドの一覧を次の表に示します。

表 7-23　PctoClob クラスのメソッドの一覧

メソッド	説明
getCharacterStream()	この Clob オブジェクトによって指定された CLOB 値を、java.io.Reader オブジェクト（または文字のストリーム）として取り出します。
getSubString(long pos, int length)	この Clob オブジェクトによって指定された CLOB 値内の指定された部分文字列のコピーを取得します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1)　getCharacterStream()

説明

この Clob オブジェクトによって指定された CLOB 値を、java.io.Reader オブジェクト（または文字のストリーム）として取り出します。

形式

```
public java.io.Reader getCharacterStream() throws java.sql.SQLException
```

引数

なし。

戻り値

CLOB データを格納する java.io.Reader オブジェクトを返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

CLOB 値にアクセスするときにエラーが発生した場合に送出します。

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) getSubString(long pos, int length)

説明

この Clob オブジェクトによって指定された CLOB 値内の指定された部分文字列のコピーを取得します。部分文字列は、引数 pos に指定された位置から、引数 length に指定された数の連続する文字です。

形式

```
public java.lang.String getSubString(long pos, int length) throws java.sql.SQLException
```

引数

long pos：

抽出される部分文字列の最初の文字を指定します。最初の文字の位置は 1 です。

int length：

コピーされる連続した文字数を指定します。0 以上を指定してください。

戻り値

この Clob オブジェクトによって指定された CLOB 値内の指定された部分文字列である String を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

次の場合に例外を送出します。

- CLOB 値にアクセスするときにエラーが発生した場合
- 引数 pos が 1 より小さいか、または引数 length が 0 より小さい場合

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.14 PctoInputStream クラス（SQL-Participant 限定）

説明

バイト入力ストリームを表現するクラスです。

メソッド

PctoInputStream クラスのメソッドの一覧を次の表に示します。

表 7-24 PctoInputStream クラスのメソッドの一覧

メソッド	説明
available()	この入力ストリームのメソッドの次の呼び出しによって、ブロックをしないでこの入力ストリームから読み込むことができる（またはスキップできる）データの推定バイト数を返します。
close()	この入力ストリームを閉じて、そのストリームに関連するすべてのシステム・リソースを解放します。
mark(int readlimit)	この入力ストリームの現在位置にマークを設定します。
markSupported()	この入力ストリームが、mark および reset メソッドをサポートしているかどうかを判定します。
read()	入力ストリームから最初の 1 バイトのデータを読み込み、0 - 255 の範囲の int でバイト値を返します。
read(byte[] b)	入力ストリームから数バイトのデータを読み込み、それを引数 b に指定したバッファ配列に読み込みます。また、実際に読み込んだバイト数を整数で返します。
read(byte[] b, int off, int len)	引数 len に指定したバイト数のデータを、入力ストリームから byte 配列に読み込みます。
reset()	このストリームを、この入力ストリームで最後に mark メソッドが呼び出されたときの位置に再配置します。
skip(long n)	この入力ストリームから、引数 n に指定したバイト数のデータをスキップして破棄します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1) available()

説明

この入力ストリームのメソッドの次の呼び出しによって、ブロックをしないでこの入力ストリームから読み込むことができる（またはスキップできる）データの推定バイト数を返します。次の呼び出しは、同じスレッドのときもあれば、別のスレッドのときもあります。大きなデータを 1 回で読み込んだり、スキップしたりすることでブロックされることはありませんが、読み込むデータまたはスキップするデータのバイト数が少なくなることはあります。

形式

```
public int available() throws java.io.IOException
```

引数

なし。

戻り値

ブロックをしないでこの入力ストリームから読み込むことができる（またはスキップできる）データの推定バイト数を返します。入力ストリームの終わりに達した場合は、0 を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) close()

説明

この入力ストリームを閉じて、ストリームに関連するすべてのシステム・リソースを解放します。

形式

```
public void close() throws java.io.IOException
```

引数

なし。

戻り値

なし。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) mark(int readlimit)

説明

この入力ストリームの現在位置にマークを設定します。これに続いて reset メソッドを呼び出すと、最後にマークされた位置にこのストリームが再配置されます。以降の読み込みでは、同じバイト数のデータが再度読み込まれます。

引数 readlimit では、この入力ストリームに対してマーク位置が無効になる前にこの引数が指定するバイト数を読み込むように指示します。

形式

```
public void mark(int readlimit)
```

引数

int readlimit：

マーク位置が無効になる前に読み込み可能なバイト数の上限を指定します。

戻り値

なし。

例外

java.lang.RuntimeException：

次の例外を cause に設定した java.lang.RuntimeException を送出します。

- PctoIOCommunicationFailureException
  Entity-Service と Participant 間の通信に失敗した場合
- PctoIOInternalContradictionException
  Participant の内部で障害が発生した場合
- PctoIOTransactionStartException
  トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合

(4) markSupported()

説明

この入力ストリームが、mark メソッドおよび reset メソッドをサポートしているかどうかを判定します。mark メソッドおよび reset がサポートされているかどうかは、入力ストリーム・インスタンスごとに異なります。

形式

```
public boolean markSupported()
```

引数

なし。

戻り値

このストリーム・インスタンスが mark メソッドおよび reset メソッドをサポートしている場合は true を、それ以外の場合は false を返します。

例外

java.lang.RuntimeException：

次の例外を cause に設定した java.lang.RuntimeException を送出します。

- PctoIOCommunicationFailureException
  Entity-Service と Participant 間の通信に失敗した場合
- PctoIOInternalContradictionException
  Participant の内部で障害が発生した場合
- PctoIOTransactionStartException
  トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合

(5) read()

説明

入力ストリームから最初の 1 バイトのデータを読み込み、0 - 255 の範囲の int でバイト値を返します。入力ストリームの終わりに達したために読み込むデータがない場合は、-1 が返されます。

このメソッドは、入力データが読み込めるようになるか、入力ストリームの終わりが検出されるか、または例外がスローされるまでブロックされます。

形式

```
public int read() throws java.io.IOException
```

引数

なし。

戻り値

入力ストリームの最初の 1 バイトのデータを、0 - 255 の範囲の int で返します。ストリームの終わりに達した場合は、-1 を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(6) read(byte[] b)

説明

入力ストリームから数バイトのデータを読み込み、それを引数 b に指定したバッファ配列に読み込みます。また、実際に読み込んだバイト数を整数で返します。

このメソッドは、入力データが読み込めるようになるか、入力ストリームの終わりが検出されるか、または例外がスローされるまでブロックされます。

なお、引数 b の値が 0 の場合、データは読み込まれません。このとき、戻り値には 0 が返されます。

形式

```
public int read(byte[] b) throws java.io.IOException
```

引数

byte[] b：

データの読み込み先のバッファを指定します。

戻り値

バッファに格納されたデータのバイト数の合計値を返します。ストリームの終わりに達してデータがない場合は、-1 を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

次の場合に例外を送出します。

- ファイルの終わりに達していること以外の理由で最初のバイトを読み込めない場合
- 入力ストリームが閉じられた場合
- そのほかの入出力エラーが発生した場合

java.lang.NullPointerException：

引数 b が null である場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(7) read(byte[] b, int off, int len)

説明

引数 len に指定したバイト数のデータを、入力ストリームから byte 配列に読み込みます。

なお、引数 len に指定したバイト数は、読み込みをする際の最大バイト数のため、実際に読み込まれるバイト数は引数 len の値より少ないこともあります。戻り値には、実際にバッファに読み込まれたデータのバイト数の合計値を整数で返します。

このメソッドは、入力データが読み込めるようになるか、入力ストリームの終わりが検出されるか、または例外がスローされるまでブロックされます。

なお、引数 len の値が 0 の場合、データは読み込まれません。このとき、戻り値には 0 が返されます。入力ストリームの終わりに達したために読み込むデータがない場合は、-1 が返されます。

形式

```
public int read(byte[] b, int off, int len) throws java.io.IOException
```

引数

byte[] b：

データの読み込み先のバッファを指定します。

int off：

データが書き込まれる配列 b の開始オフセットを指定します。

int len：

読み込む最大バイト数を指定します。

戻り値

バッファに読み込まれたデータのバイト数の合計値を返します。ストリームの終わりに達してデータがない場合は、-1 を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

次の場合に例外を送出します。

- ファイルの終わりに達していること以外の理由で最初のバイトを読み込めない場合
- 入力ストリームが閉じられた場合
- そのほかの入出力エラーが発生した場合

java.lang.NullPointerException：

引数 b が null である場合に送出します。

java.lang.IndexOutOfBoundsException：

次の場合に例外を送出します。

- 引数 off が負の値の場合
- 引数 len が負の値の場合
- 引数 len が b.length - off より大きい場合

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(8) reset()

説明

このストリームを、この入力ストリームで最後に mark メソッドが呼び出されたときの位置に再配置します。

形式

```
public void reset() throws java.io.IOException
```

引数

なし。

戻り値

なし。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

次の場合に例外を送出します。

- ストリームにマークが設定されていない場合
- マークが無効の場合

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(9) skip(long n)

説明

この入力ストリームから、引数 n に指定したバイト数のデータをスキップして破棄します。

なお、引数 n に指定したバイト数のデータをスキップする際に入力ストリームの終わりに達した場合、スキップされるデータは引数 n に指定したバイト数よりも少なくなります。戻り値には、実際にスキップされたデータのバイト数が返されます。

形式

```
public long skip(long n) throws java.io.IOException
```

引数

long n：

スキップするバイト数を指定します。

戻り値

実際にスキップされたデータのバイト数を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

次の場合に例外を送出します。

- ストリームがシークをサポートしない場合
- そのほかの入出力エラーが発生した場合

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.15　PctoReader クラス（SQL-Participant 限定）

説明

文字ストリームを読み込むためのクラスです。

メソッド

PctoReader クラスのメソッドの一覧を次の表に示します。

表 7-25　PctoReader クラスのメソッドの一覧

メソッド	説明
close()	ストリームを閉じて、それに関連するすべてのシステム・リソースを解放します。
mark(int readAheadLimit)	ストリームの現在位置にマークを設定します。

メソッド	説明
markSupported()	このストリームが mark()オペレーションをサポートするかどうかを判定します。
read()	単一の文字を読み込みます。
read(char[] cbuf)	配列に文字を読み込みます。
read(char[] cbuf, int off, int len)	配列の一部に文字を読み込みます。
read(CharBuffer target)	指定された文字バッファに文字列を読み込みます。
ready()	このストリームが読み込めるかどうかを判定します。
reset()	ストリームをリセットします。
skip(long n)	文字をスキップします。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1) close()

説明

ストリームを閉じて、それに関連するすべてのシステム・リソースを解放します。ストリームが閉じられたあとに read()、ready()、mark()、reset()、または skip()を呼び出すと、IOException がスローされます。すでに閉じられているストリームを閉じても、何の影響もありません。

形式

```
public void close() throws java.io.IOException
```

引数

なし。

戻り値

なし。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) mark(int readAheadLimit)

説明

ストリームの現在位置にマークを設定します。以降の reset()の呼び出しでは、この位置へのストリームの再配置が試みられます。

形式

```
public void mark(int readAheadLimit) throws java.io.IOException
```

引数

int readAheadLimit：

マークを保持しながら読み込むことができる文字数の上限を指定します。この数の文字を読み込んだあとにストリームをリセットしようとすると失敗することがあります。

戻り値

なし。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

次の場合に例外を送出します。

- ストリームが mark()をサポートしない場合
- そのほかの入出力エラーが発生した場合

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(3) markSupported()

説明

このストリームが mark()オペレーションをサポートするかどうかを判定します。

形式

```
public boolean markSupported()
```

引数

なし。

戻り値

このストリームが mark()オペレーションをサポートする場合は、true を返します。

例外

java.lang.RuntimeException：

次の例外を cause に設定した java.lang.RuntimeException を送出します。

- PctoIOCommunicationFailureException
  Entity-Service と Participant 間の通信に失敗した場合
- PctoIOInternalContradictionException
  Participant の内部で障害が発生した場合
- PctoIOTransactionStartException
  トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合

(4) read()

説明

単一の文字を読み込みます。このメソッドは、1 文字が読み込まれるか、入出力エラーが発生するか、またはストリームの終わりに達するまでブロックされます。

形式

```
public int read() throws java.io.IOException
```

引数

なし。

戻り値

読み込まれた文字を 0 - 65535（0x00 - 0xffff）の範囲の整数で返します。ストリームの終わりに達した場合は、-1 を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(5) read(char[] cbuf)

説明

配列に文字を読み込みます。このメソッドは、入力の一部が有効になるか、入出力エラーが発生するか、またはストリームの終わりに達するまでブロックされます。

形式

```
public int read(char[] cbuf) throws java.io.IOException
```

引数

char[] cbuf：

転送先バッファを指定します。

戻り値

読み込まれた文字数を返します。ストリームの終わりに達した場合は、-1 を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(6) read(char[] cbuf, int off, int len)

説明

配列の一部に文字を読み込みます。このメソッドは、入力の一部が有効になるか、入出力エラーが発生するか、またはストリームの終わりに達するまでブロックされます。

形式

```
public int read(char[] cbuf, int off, int len) throws java.io.IOException
```

引数

char[] cbuf：

転送先バッファを指定します。

int off：

文字の格納開始オフセットを指定します。

int len：

読み込む文字の最大数を指定します。

戻り値

読み込まれた文字数を返します。ストリームの終わりに達した場合は、-1 を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(7) read(CharBuffer target)

説明

指定された文字バッファに文字列を読み込みます。バッファは、put 操作の結果によって変更される以外は、文字列そのままのリポジトリとして使用されます。バッファのフリッピング（反転）やリワインド（巻き戻し）は行われません。

形式

```
public int read(CharBuffer target) throws java.io.IOException
```

引数

CharBuffer target：

文字を読み込むバッファを指定します。

戻り値

バッファに追加された文字の数を返します。文字のソースが最後の位置にある場合は-1 を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

java.lang.NullPointerException：

引数 target が null である場合に送出します。

java.nio.ReadOnlyBufferException：

target が読み取り専用バッファの場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(8) ready()

説明

このストリームが読み込めるかどうかを判定します。

形式

```
public boolean ready() throws java.io.IOException
```

引数

なし。

戻り値

次の read()が入力をブロックしないことが確実な場合は true、そうでない場合は false を返します。false が返されても、次の読み込みが確実にブロックするというわけではありません。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

次の場合に例外を送出します。

- 入出力エラーが発生した場合

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(9) reset()

説明

ストリームをリセットします。ストリームにマークが設定されている場合は、マークに再配置しようとします。マークが設定されていない場合は、開始位置への再配置など、個々のストリームに適した方法でリセットを試みます。

形式

```
public void reset() throws java.io.IOException
```

引数

なし。

戻り値

なし。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

次の場合に例外を送出します。

- ストリームにマークが設定されていない場合
- マークの設定が無効な場合
- ストリームが reset()をサポートしない場合
- ほかの入出力エラーが発生した場合

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(10) skip(long n)

説明

文字をスキップします。このメソッドは、文字が読み込まれるか、入出力エラーが発生するか、またはストリームの終わりに達するまでブロックされます。

形式

```
public long skip(long n) throws java.io.IOException
```

引数

long n：

スキップする文字数を指定します。

戻り値

実際にスキップした文字数を返します。

例外

PctoIOCommunicationFailureException 例外と PctoIOInternalContradictionException 例外以外は、Participant が送出した例外を送出します。

java.io.IOException：

入出力エラーが発生した場合に送出します。

java.lang.IllegalArgumentException：

引数 n が負の値の場合に送出します。

PctoIOCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoIOInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoIOTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.16　PctoArray クラス（SQL-Participant 限定）

説明

SQL 型 ARRAY の Java プログラミング言語でのマッピングをするクラスです。デフォルトでは、ARRAY 値は SQL ARRAY 値へのトランザクション期間中の参照です。デフォルトでは、Array オブジェクトは内部的に SQL LOCATOR(array)を使って実装されます。したがって、Array オブジェクトは、ARRAY 値のデータを含むのではなく、SQL ARRAY 値のデータへの論理ポインタを含みます。

Array インタフェースは、SQL ARRAY 値のデータを配列または ResultSet オブジェクトとしてクライアントに渡すメソッドを提供します。

基底型のエントリを持つ型マップが、getArray メソッドおよび getResultSet メソッドに提供されると、その型マップに含まれるマッピングは、ARRAY 値の要素をマッピングするのに使用されます。通常、型マップは提供されません。その場合、接続の型マップがデフォルトで使用されます。接続の型マップまたはメソッドに提供された型マップが基底型のエントリを持っていない場合、要素は標準マップに従ってマッピングされます。

メソッド

PctoArray クラスのメソッドの一覧を次の表に示します。

表 7-26　PctoArray クラスのメソッドの一覧

メソッド	説明
free()	このメソッドは、Array オブジェクトを解放し、保持されているリソースを解放します。
getArray()	この Array オブジェクトによって指定された SQL ARRAY 値の内容を、Java プログラミング言語の配列の形式で取り出します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client
```

(1)　free()

説明

このメソッドは、Array オブジェクトを解放し、保持されているリソースを解放します。free メソッドが一度呼び出されたあとは、オブジェクトは無効になります。

free メソッドが呼び出されると、free メソッド以外のメソッドを呼び出そうとしても、SQLException がスローされます。free メソッドを複数回呼び出した場合、2 回目以降の free メソッドの呼び出しでは何も行われません。

形式

```
public void free() throws java.sql.SQLException
```

引数

なし。

戻り値

なし。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

Array のリソースの解放時にエラーが発生した場合に送出します。

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

(2) getArray()

説明

この Array オブジェクトによって指定された SQL ARRAY 値の内容を、Java プログラミング言語の配列の形式で取り出します。この getArray メソッドは、型マップのカスタマイズのための接続に関連した型マップを使います。

なお、getArray メソッドを使用してプリミティブデータ型にマッピングする基底型を生成する場合、返される配列がそのプリミティブデータ型の配列と java.lang.Object の配列のどちらであるかは、実装で定義されます。

形式

```
public java.lang.Object getArray() throws java.sql.SQLException
```

引数

なし。

戻り値

この Array オブジェクトによって指定された SQL ARRAY 値の順序付き要素が格納されている、Java プログラミング言語の配列を返します。

例外

PctoCommunicationFailureException 例外と PctoInternalContradictionException 例外以外は、Participant で JDBC Driver が送出した例外を送出します。

java.sql.SQLException：

配列へのアクセス中にエラーが発生した場合に送出します。

java.sql.SQLFeatureNotSupportedException：

Participant の JDBC Driver がこのメソッドをサポートしない場合に送出します。

PctoCommunicationFailureException：

Entity-Service と Participant 間の通信に失敗した場合に送出します。

PctoInternalContradictionException：

Participant の内部で障害が発生した場合に送出します。

PctoTransactionStartException：

トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗した場合に送出します。

7.3.17 PctoSqlFeatureNotSupportedException 例外（SQL-Participant 限定）

説明

JDBC Proxy Driver がサポートしていないメソッドを呼び出した場合に送出する例外です。サポートしていないメソッドの呼び出し元に送出します。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.sql.SQLException
        java.sql.SQLNonTransientException
          java.sql.SQLFeatureNotSupportedException
            jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoSqlFeatureNotSupportedEx
ception
```

7.3.18 PctoSQLClientInfoException 例外（SQL-Participant 限定）

説明

JDBC Proxy Driver がサポートしていないメソッドを呼び出した場合に送出する例外です。サポートしていないメソッドの呼び出し元に送出します。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.sql.SQLException
        java.sql.SQLClientInfoException
          jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoSQLClientInfoException
```

7.3.19 PctoCommunicationFailureException 例外（SQL-Participant 限定）

説明

Entity-Service と Participant 間の通信に失敗した場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.sql.SQLException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoCommunicationFailureExceptio
n
```

7.3.20 PctoInternalContradictionException 例外（SQL-Participant 限定）

説明

Participant の内部で障害が発生した場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.sql.SQLException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoInternalContradictionExcepti
on
```

7.3.21　PctoTransactionStartException 例外（SQL-Participant 限定）

説明

Participant のトランザクションブランチを開始できない場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.sql.SQLException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoTransactionStartException
```

7.3.22　PctoPrePreparedException 例外（SQL-Participant 限定）

説明

先行プリペア実行後に JDBC Proxy Driver のメソッドを実行した場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
```

```
    java.lang.Exception
      java.sql.SQLException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoPrePreparedException
```

7.3.23　PctoConnectionLackedException 例外（SQL-Participant 限定）

説明

パラメタ「db.pool.max-pool-size」の指定値よりも DB コネクション数を多く使用しようとした場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.sql.SQLException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoConnectionLackedException
```

7.3.24　PctoIOException 例外（SQL-Participant 限定）

説明

JDBC Proxy Driver のメソッド実行時に入出力エラーが発生した場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.sql.SQLException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoIOException
```

7.3.25 PctoIOCommunicationFailureException 例外（SQL-Participant 限定）

説明

Entity-Service と Participant 間の通信に失敗した場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.io.IOException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoIOCommunicationFailureExcept
ion
```

7.3.26 PctoIOInternalContradictionException 例外（SQL-Participant 限定）

説明

Participant の内部で障害が発生した場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.io.IOException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoIOInternalContradictionExcep
tion
```

7.3.27　PctoIOTransactionStartException 例外（SQL-Participant 限定）

説明

Participant のトランザクションブランチを開始できない場合に送出する例外です。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.jdbc.client.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.io.IOException
        jp.co.Hitachi.soft.hmppcto.jdbc.client.exception.PctoIOTransactionStartException
```

7.3.28　SQLState・ErrorCode 一覧（SQL-Participant 限定）

HMP-PCTO の JDBC Proxy Driver の例外が出力する SQLState・ErrorCode の一覧を次に示します。

表 7-27　SQLState・ErrorCode の一覧

項番	SQLState	ErrorCode	例外	意味
1	08100	1013	PctoConnectionLackedException 例外	コネクションプールの Connection インスタンスが不足しました。
2	08101	1016	PctoInternalContradictionException 例外	Connection インスタンスが失われていました。
3	08102	5	PctoInternalContradictionException 例外	JDBC Driver のインスタンスが失われていました。
4	08103	1030	PctoInternalContradictionException 例外	Connection インスタンスの取得時に DB で障害が発生しました。
5	0A000	0	PctoSqlFeatureNotSupportedException 例外 PctoSQLClientInfoException 例外	JDBC Proxy Driver がサポートしていないメソッドを呼び出しました。
6	0A001	7	PctoSqlFeatureNotSupportedException 例外	HMP-PCTO がサポートしていない Object クラスのデータが指定されました。

項番	SQLState	ErrorCode	例外	意味
7	0A002	8	PctoSqlFeatureNotSupportedException 例外	HMP-PCTO がサポートしていない Java データ型を表すクラスが指定されました。
8	0A003	9	PctoSqlFeatureNotSupportedException 例外	HMP-PCTO がサポートしていない JDBC 型を示す値が sqlType に指定されました。
9	A0000	1	PctoCommunicationFailureException 例外	設定ファイルに記載している URL の形式が不正です。
10	A0001	2	PctoCommunicationFailureException 例外	Participant との通信に失敗しました。
11	A0002	1010	PctoInternalContradictionException 例外 または PctoTransactionStartException 例外	SQL-Participant が閉塞しています。
12	A0003	1011	PctoTransactionStartException 例外	トランザクションブランチの開始に失敗しました。
13	A0004	1012	PctoPrePreparedException 例外	プリペア済みのトランザクションブランチで SQL 実行を要求されました。
14	A0005	3	PctoTransactionStartException 例外	トランザクションの範囲外で実行されたためトランザクションブランチの開始に失敗しました。
15	A0006	6	PctoIOException 例外	ストリームからのデータの読み込みに失敗しました。
16	A0007	1029	PctoInternalContradictionException 例外	トランザクション決着監視タイマによる rollback 処理後に SQL 実行を要求されました。

7.4　TCC-Participant の API 仕様（TCC-Participant 限定）

HMP-PCTO の TCC-Participant で使用する API の仕様を説明します。

7.4.1　クラス・インタフェースの一覧（TCC-Participant 限定）

HMP-PCTO の TCC-Participant がサポートするクラス・インタフェースの一覧を次に示します。

表 7-28　クラス・インタフェースの一覧

項番	パッケージ名	クラス・インタフェース名	説明
1	jp.co.Hitachi.soft.hmppcto.participant.tcc.api	TccResponse<T extends java.io.Serializable>クラス	トランザクションを一意に表す文字列に紐づいたデータを管理するためのクラスです。
2	jp.co.Hitachi.soft.hmppcto.participant.tcc.api	TccService<I, O extends java.io.Serializable>インタフェース	TCC パターンの Try、Confirm、Cancel、Recover の定義を持つインタフェースです。

7.4.2　アノテーションの一覧（TCC-Participant 限定）

HMP-PCTO の TCC-Participant がサポートするアノテーション一覧を次に示します。

表 7-29　アノテーションの一覧

項番	パッケージ名	アノテーション名	説明
1	jp.co.Hitachi.soft.hmppcto.participant.tcc.api	@Tcc アノテーション	executeTry メソッドと executeTry メソッドの前後処理を紐づけるアノテーションです。

7.4.3　例外の一覧（TCC-Participant 限定）

なし。

7.4.4 TccResponse<T extends java.io.Serializable>クラス（TCC-Participant 限定）

説明

transactionKey に紐づいた data を管理するためのクラスです。TccResponse は transactionKey と data のフィールドを持ちます。transactionKey はトランザクションを一意に表す文字列を格納しています。data は任意の型のデータを格納できます。data は任意の型のため、業務ロジック内で利用している一意に識別できる ID のようなデータを持つことを想定しています。トランザクションを一意に表す文字列とデータを紐づけていることで、data をトランザクション管理下に置くことができます。TccService インタフェースに定義されているメソッドの戻り値や引数として利用し、TCC-Participant とユーザアプリケーション間でデータのやり取りを行います。

このクラスは、「(b)　トランザクションを一意に表す文字列とデータの紐づけの永続化」でこのクラスのインスタンスそのものを外部 API などを使用して永続化することも可能とするために java.io.Serializable インタフェースを継承しています。

ジェネリクス T には data の型を指定します。data の型についても java.io.Serializable インタフェースを継承するように実装してください。

transactionKey の長さは最大 256 バイトです。

メソッド

TccResponse クラスのメソッドの一覧を次の表に示します。

表 7-30　TccResponse クラスのメソッドの一覧

メソッド	説明
TccResponse(final T data)	コンストラクタです。
getTransactionKey()	フィールド transactionKey の getter メソッドです。
getData()	フィールド data の getter メソッドです。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.participant.tcc.api
```

(1)　TccResponse(final T data)

説明

コンストラクタです。

引数の値をフィールドの data にセットします。

コンストラクタを呼び出すタイミングはトランザクション中である必要があります。

コンストラクタを呼び出すタイミングがトランザクション外である場合、HMP-PCTO は KFSG62016-E メッセージを出力し、java.lang.IllegalStateException を送出します。

形式

```
public TccResponse(final T data)
```

引数

T data：

任意の型のデータオブジェクトです。

戻り値

なし。

例外

なし。

(2) getTransactionKey()

説明

フィールド transactionKey の getter メソッドです。

形式

```
public String getTransactionKey()
```

引数

なし。

戻り値

フィールドの transactionKey を返します。

例外

なし。

(3) getData()

説明

フィールドの data の getter メソッドです。

形式

```
public T getData()
```

引数

なし。

戻り値

任意の型のデータオブジェクトです。

例外

なし。

7.4.5　TccService<I, O extends java.io.Serializable>インタフェース（TCC-Participant 限定）

説明

TCC パターンの Try、Confirm、Cancel、Recover の定義を持つインタフェースです。このインタフェースをユーザが実装することで、TCC パターンを利用したトランザクション制御を実現します。ジェネリクス I は executeTry の第一引数の型、O は executeTry の戻り値、executeConfirm、executeCancel の引数で利用される TccResponse のジェネリクスで利用される型を指定します。

メソッド

TccService インタフェースのメソッドの一覧を次の表に示します。

表 7-31　TccService インタフェースのメソッドの一覧

メソッド	説明
executeTry(I input)	データの仮登録を行う処理を実装します。
executeConfirm(TccResponse<O> response)	トランザクションが正常に決着したときの処理を実装します。
executeCancel(TccResponse<O> response)	トランザクションが異常に決着したときの処理を実装します。
executeRecover()	TCC-Participant が再起動した際に、未決着かつ自 Participant が登録した TccResponse の一覧を返す処理を実装します。
getOperationId()	TCC-Participant 内のユーザアプリケーション内に実装された TccService インタフェースの複数の実装クラスを一意に識別する文字列を返す処理を実装します。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.participant.tcc.api
```

(1)　executeTry(I input)

説明

TCC パターンにおける Try の処理を実装してください。例えば Try はデータの仮登録を行います。詳細な実装方法は「2.2　Java アプリケーションの開発」の executeTry()メソッドの実装の説明を参照してください。

戻り値の TccResponse<O>の O はジェネリクスで指定した任意の型です。

この処理中に例外が発生した場合、HMP-PCTO でキャッチしてメッセージを出すなどの処理はしません。メッセージやスタックトレースの出力処理はユーザが必要に応じて実装してください。

executeTry の呼び出し元で@PctoBranchRollbackFor アノテーションを利用した上で、例外の種類に応じて HTTP ステータスを変更することで、トランザクションをコミット決着させるかロールバック決着させるかを指定できます。@PctoBranchRollbackFor アノテーションに関しては「7.2.7 @PctoBranchRollbackFor アノテーション」や「2.2　Java アプリケーションの開発」のトランザクションの伝搬の説明を参照してください。

形式

```
public TccResponse<O> executeTry(I input) throws java.lang.Exception
```

引数

I input：

任意の型のデータオブジェクトを指定します。

戻り値

TccResponse< O>クラスのインスタンスを返します。

例外

java.lang.Exception:

executeTry の処理中に異常が発生した場合に送出してください。

(2)　executeConfirm(TccResponse<O> response)

説明

TCC パターンにおける Confirm の処理を実装してください。例えば Confirm は Try で仮登録したデータを確定させる処理を行います。詳細な実装方法は「2.2　Java アプリケーションの開発」の executeConfirm()メソッドの実装の説明を参照してください。

HMP-PCTO から分散合意結果が Commit 決着であった場合に実行されます。

なお、同一トランザクションに参加している TccService インタフェースの複数の実装クラスの間で、HMP-PCTO からの各 executeConfirm()メソッドの実行順序は順不同です。

引数の TccResponse<O>の O はジェネリクスで指定した任意の型で、executeTry の戻り値を利用します。

この処理で送出された例外は HMP-PCTO でキャッチし、KFSG62007-E (Y)M のメッセージおよびスタックトレースを出力し、トランザクションを終了します。

形式

```
public void executeConfirm(TccResponse<O> response)
```

引数

TccResponse<O> response：

executeTry メソッドで戻り値として受け取った TccResponse クラスの値を指定します。

戻り値

なし。

例外

なし。

(3) executeCancel(TccResponse<O> response)

説明

TCC パターンにおける Cancel の処理を実装してください。例えば Cancel は Try で仮登録したデータを取り消す処理を行います。詳細な実装方法は「2.2 Java アプリケーションの開発」の executeCancel()メソッドの実装の説明を参照してください。

HMP-PCTO から分散合意結果が Rollback であった場合に実行されます。

引数の TccResponse<O>の O はジェネリクスで指定した任意の型で、executeTry の戻り値を利用します。

この処理で送出された例外は HMP-PCTO でキャッチし、KFSG62008-E (Y)M のメッセージおよびスタックトレースを出力し、トランザクションを終了します。

形式

```
public void executeCancel(TccResponse<O> response)
```

引数

TccResponse<O> response：

executeTry メソッドで戻り値として受け取った TccResponse クラスの値を指定します。

戻り値

なし。

例外

なし。

(4) executeRecover()

説明

TCC-Participant が再起動した際に実行されます。

このメソッドが呼び出された時点で未決着かつ自 Participant が登録した TccResponse の一覧を返す処理を実装してください。

詳細な実装方法は「2.2 Java アプリケーションの開発」の executeRecover()メソッドの実装の説明を参照してください。

この処理で送出された例外は HMP-PCTO でキャッチし、KFSG62004-E (Y)M のメッセージおよびスタックトレースを出力し、Pod の起動を終了します。

形式

```
public java.util.Set<TccResponse<O>> executeRecover()
```

引数

なし。

戻り値

このメソッドが呼び出された時点で未決着かつ自 Participant が登録した TccResponse の一覧を返します。

未決着かつ自 Participant が登録した TccResponse がない場合は、空の java.util.set を返します。

例外

なし。

(5) getOperationId()

説明

TCC-Participant 内のユーザアプリケーション内に実装された TccService インタフェースの複数の実装クラスを一意に識別する ID を返す処理を実装してください。

詳細な実装方法は「2.2 Java アプリケーションの開発」の getOperationId()メソッドの実装の説明を参照してください。

例外が発生したとき、executeRecover 中の場合は HMP-PCTO でキャッチし、KFSG62012-E (Y)M のメッセージおよびスタックトレースを出力し、Pod の起動を終了します。

一方、トランザクション開始時の場合は HMP-PCTO でキャッチし、KFSG62012-E (Y)M のメッセージおよびスタックトレースを出力し、トランザクションの要求を無視します。

形式

```
public java.lang.String getOperationId()
```

引数

なし。

戻り値

TccService を一意に識別する文字列を返します。

文字列の長さは 1～256 バイトの範囲内である必要があります。

例外

なし。

7.4.6 @Tcc アノテーション（TCC-Participant 限定）

説明

executeTry メソッドと、executeTry メソッドの前後処理を紐づけるアノテーションです。TccService インタフェースに付与することで HMP-PCTO が前後処理を行います。

詳細な実装方法は「2.2 Java アプリケーションの開発」の@Tcc アノテーションの付与を参照してください。

適用可能要素

クラス

属性

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.participant.tcc.api
```

8 パラメタ

8.1 HMP-PCTO の各コンポーネントおよび連携先製品に設定するパラメタ

HMP-PCTO の各コンポーネントおよび連携先製品に設定するパラメタを説明します。

HMP-PCTO のパラメタは、次のとおりに設定してください。

表 8-1 HMP-PCTO のパラメタ設定方法

コンポーネントの分類	パラメタ設定方法	コンポーネント	参照先
ユーザのコンポーネント	Spring Boot のコンフィグソース（application.properties、application.yml、環境変数など）に設定	共通	8.2.1 共通
		Orchestrator	8.2.2 Orchestrator
		Entity-Service/Entity-Module	8.2.3 Entity-Service/Entity-Module
		SQL-Participant	8.2.4 SQL-Participant
		TCC-Participant	8.2.5 TCC-Participant
HMP-PCTO のコンポーネント	Helm チャートの values.yaml に設定	Mediator	8.3.1 Mediator
		TP1-Bridge	8.3.2 TP1-Bridge（通常版かつ TP1-Bridge 限定）
		EADS（通常版限定）	8.3.3 EADS（通常版限定）
		Ext-Cons（トライアル版の場合）	8.3.4 Ext-Cons（トライアル版限定）
		Filebeat	8.3.5 Filebeat
		Jaeger-query	8.3.6 Jaeger-query
HMP-PCTO と RAS のコンポーネント		Jaeger-collector	8.3.7 Jaeger-collector
		Prometheus	8.3.8 Prometheus
		Metricbeat	8.3.9 Metricbeat
		Elasticsearch	8.3.10 Elasticsearch
		Logstash	8.3.11 Logstash

連携先製品のパラメタは、次のとおりに設定してください。

表 8-2 連携先製品のパラメタ設定方法

連携先製品	パラメタ設定方法	参照先
PostgreSQL（SQL-Participant 限定）	PostgreSQL のサーバ側のパラメタに設定	8.4.1 PostgreSQL（SQL-Participant 限定）
SQL Server（SQL-Participant 限定）	SQL Server のサーバ側のオプションに設定	8.4.2 SQL Server（SQL-Participant 限定）

連携先製品	パラメタ設定方法	参照先
HiRDB（SQL-Participant 限定）	HiRDB の次の定義に設定 • サーバ定義 • クライアント環境定義	8.4.3　HiRDB（SQL-Participant 限定）
OpenTP1（通常版かつ TP1-Bridge 限定）	連携先の OpenTP1 の次の定義に設定 • rap リスナーサービス定義 • システム共通定義 • トランザクションサービス定義 • ユーザサービス定義	8.5　TP1 連携時の連携先 OpenTP1 に設定が必要なパラメタ（通常版かつ TP1-Bridge 限定）
uCosminexus Application Server（通常版かつ TP1-Bridge 限定）	uCosminexus Application Server の次のファイルに設定 • usrconf.properties ファイル • usrconf.cfg ファイル	8.6　uCosminexus Application Server に設定が必要な TP1-Converter のパラメタ（通常版かつ TP1-Bridge 限定）

重要

パラメタ設定の注意事項

パラメタの型が String（文字列）の場合に設定できる文字は ASCII 文字です。マルチバイト文字は設定できません。

8.2　ユーザのコンポーネントに設定するパラメタ

ユーザのコンポーネントに設定するパラメタについて説明します。

8.2.1　共通

Orchestrator、Entity-Service、Mediator、SQL-Participant、TCC-Participant および TP1-Bridge で共通のパラメタを説明します。

(1)　jp.co.Hitachi.soft.hmppcto のパラメタ

各パラメタの接頭辞は、jp.co.Hitachi.soft.hmppcto です。

表 8-3　共通の jp.co.Hitachi.soft.hmppcto のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値

tracing.instance-id	Orchestrator/Entity-Service/Participant/TP1-Bridge/Mediator を一意に識別するための ID です。この指定値は、分散トレースのサービス名として表示します。
	32 文字以内の任意の文字列を設定してください。
	省略した場合、デフォルト値としてユニークな ID が設定されます。
	ID がシステムで一意になるように、Orchestrator/Entity-Service/Participant/TP1-Bridge/Mediator ごとに異なる値を設定してください。
	一意ではない場合、分散トレース情報がどのプロセスの情報かの判別ができません。
	ただし、レプリカ設定を実施する場合は、このパラメタを省略してください。
	設定してしまうと、レプリカ設定を実施している場合に複数台で同じ ID となり、分散トレース情報がどのサービスかを判別できません。
		String	32 文字	• Mediator 以外の場合
				省略した場合および指定しなかった場合は、次の形式でサービス名が設定されます。
				`<コンポーネント名>-<12桁のランダムな数字>`
				• Mediator の場合
				省略した場合および指定しなかった場合は、次の形式でサービス名が設定されます。
				`<ホスト名>@Mediator`

tracing.url	HMP-PCTO が分散トレースの送信先に使用する URL です。 Jaeger-collector の URL を設定してください。	String	−	なし

パラメタ	説明	型	設定範囲	デフォルト値
	パラメタ tracing.trace-use の指定値が true の場合は省略不可です。 次の形式で指定してください。 • Jaeger-collector が同一 namespace の場合 http://jaeger-collector:<ポート番号>/api/traces • Jaeger-collector が異なる namespace の場合 http://jaeger-collector.<ネームスペース名>.svc.cluster.local:<ポート番号>/api/traces 指定例 • Jaeger-collector が同一 namespace の場合 http://jaeger-collector:14268/api/traces • Jaeger-collector が異なる namespace の場合 http://jaeger-collector.my-namespace.svc.cluster.local:14268/api/traces			
tracing.trace-use	分散トレースの取得有無を指定します。 true 分散トレースを取得します。 false 分散トレースを取得しません。テスト環境などで Jaeger を使用できない場合に指定してください。本番環境では指定しないでください。	Boolean	true|false	true
tracing.queue-size	メモリ上のキューに保持できる分散トレース数を指定します。tracing.trace-use パラメタの指定値が true の場合は省略できません。 この設定値を上回る数の分散トレースが発生した場合、その分散トレースは破棄されます。 キューに存在する分散トレースは、次のどちらかの条件を満たすごとに分散トレース基盤に送信されます。	Integer	10000-100000	20000

パラメタ	説明	型	設定範囲	デフォルト値
	• キュー内の分散トレース数が jp.co.Hitachi.soft.hmppcto.tracing.batch-size パラメタで設定した値を上回った場合 • 前回の送信から jp.co.Hitachi.soft.hmppcto.tracing.delay-time パラメタで設定した値以上の時間が経過した場合			
tracing.delay-time	分散トレースを分散トレース基盤に送信する間隔をミリ秒単位で指定します。tracing.trace-use パラメタの指定値が true の場合は省略できません。 前回の送信からこのパラメタに設定した値以上の時間が経過すると、キューに存在する分散トレースが分散トレース基盤に送信されます。	Integer	1000-5000	1000
tracing.batch-size	分散トレースを分散トレース基盤に送信する際に、一度に送信する分散トレース数を指定します。tracing.trace-use パラメタの指定値が true の場合は省略できません。 キューに存在する分散トレース数が、このパラメタに設定した値を上回ると、キューに存在する分散トレースが分散トレース基盤に送信されます。	Integer	1000-10000	2000

（凡例）－：該当しません。

8.2.2 Orchestrator

Orchestrator のパラメタを説明します。

(1) jp.co.Hitachi.soft.hmppcto のパラメタ

各パラメタの接頭辞は、jp.co.Hitachi.soft.hmppcto です。

表 8-4　Orchestrator の jp.co.Hitachi.soft.hmppcto のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
instance-id	Orchestrator を一意に識別するための ID です。 32 文字以内の任意の文字列を設定してください。 同じ Mediator に複数の Orchestrator から接続する場合は、Orchestrator ごとに異なる値を設定してください。	String	32 文字	なし 必ず設定してください。
learner.consensus-type	トランザクションの決着に使用する分散合意のアルゴリズムを設定します。	String	Three-Quarters | Privileged-Value	なし 必ず設定してください。
mediator.name	Mediator の pod の DNS 名中の StatefulSet 名を設定してください。 Mediator の pod は StatefulSet のため、DNS 名は次の形式です。この中の StatefulSet 名を設定します。 `<StatefulSet名>-<連番>.<Service名>.<ネームスペース名>.svc.cluster.local` 例えば DNS 名が「mediator-0.mediator.space1.svc.cluster.local」の場合は、「mediator」を設定してください。 HMP-PCTO は、Kubernetes と連携して mediator.name と mediator.name-suffix をキーに Orchestrator が接続する Mediator を決定します。	String	−	なし 必ず設定してください。
mediator.name-suffix	Mediator の pod の DNS 名中のサービス名以降の文字を指定してください。 Mediator の pod は StatefulSet のため、DNS 名は次の形式です。この中の Service 名以降の文字列を設定します。 `<StatefulSet名>-<連番>.<Service名>.<ネームスペース名>.svc.cluster.local` 例えば、DNS 名が「mediator-0.mediator.space1.svc.cluster.local」の場合は、「.mediator.space1.svc.cluster.local」を設定してください。	String	−	なし 必ず設定してください。
mediator.port	Mediator の HMP-PCTO が gRPC を受信するポート番号を指定します。 Mediator のパラメタ grpc.port に設定した値と同じ値を設定してください。	Integer	1024-65535	20551

パラメタ	説明	型	設定範囲	デフォルト値
mediator.size	トランザクションの合意に参加する Mediator 数を設定してください。 learner.consensus-type に Three-Quarters を設定した場合は、5 以上の値を設定してください。	Integer	2-16	なし 必ず設定してください。
mediator.minimum-start-size	Orchestrator でトランザクション開始時に、mediator.size で設定した数の Mediator の起動を確認できなかった場合に、トランザクションを開始する最小限の Mediator 数を設定してください。 mediator.size で設定した数以下の値を設定してください。mediator.size で設定した数より大きい値を設定した場合は Orchestrator の初期化時に異常終了します。 トランザクション開始時の Mediator の起動数がこのパラメタに設定した数未満の場合は、トランザクションの開始に失敗します。 learner.consensus-type に Three-Quarters を設定した場合は、5 以上の値を設定してください。	Integer	2-16	mediator.size 設定値
mediator.termination-thread-pool-size	Orchestrator から Mediator への送信に使用するスレッド数を指定してください。 1 トランザクションの終了時に mediator.size 指定値分のスレッドを使用して送信します。スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから送信します。 スレッドの空き待ちを回避するための理論上の必要なスレッド数は、サービスの同時実行数×mediator.size です。 設計指針とチューニング方法については、「3.10.1　スレッドプール」を参照してください。	Integer	1-1000	20
mediator.discovery-retry-count	Orchestrator 起動時に、トランザクションの合意に必要な Mediator が起動しているかのチェックをリトライする回数を設定します。 Mediator の起動が確認できるまで最大で、mediator.discovery-retry-count × mediator.discovery-interval-in-millis（ミリ秒）待ち合わせます。	Integer	0-100	18
mediator.discovery-interval-in-millis	Orchestrator 起動時に、トランザクションの合意に必要な Mediator が起動しているかのチェックをリトライする際のリトライ間隔をミリ秒単位で設定します。	Integer	1-60000	10000

パラメタ	説明	型	設定範囲	デフォルト値

branch.qualifier.list	AP が Entity-Service を呼び出す際に、HMP-PCTO がトランザクションブランチを識別するために付与する識別子を設定してください。
	次の形式で指定します。
	`<Entity-Serviceのホスト名>:<Entity-Serviceのポート番号><Entity-Serviceのサービスのエンドポイント>@識別子`
	識別子は、32 文字以内で設定してください。
	複数の Entity-Service を設定する場合は「,」で区切って設定してください。
	設定例：
	`host1:80/service1@qualifier1,host2:80/service2@qualifier2`
		String	−	なし
				省略した場合、および指定しなかった URL の識別子は HMP-PCTO が UUID を生成して設定します。

branch.qualifier.propagation-order	HMP-PCTO と HMP-ADIF を連携して使用する場合の TxPropagationIscImperativePlugin クラスの ISC コマンドプラグイン実行順番を設定します。通常はデフォルト値を使用してください。ほかの ISC コマンドプラグインクラスとの実行順番を変更したい場合に設定してください。	Integer	1-2147483647	1
grpc.use-tls	暗号化通信をするかどうかを指定する。 true：暗号化通信をします。 false：暗号化通信をしません。	Boolean	true | false	false
grpc.tls.client.server-auth.certificate	接続先サーバのサーバ証明書を発行した CA の証明書(X.509 の PEM 形式)ファイルの絶対パス名を指定する。	String	−	なし 暗号化通信をする場合は、必ず指定してください。
grpc.tls.client.client-auth.certificate	クライアント認証をする場合に、自プロセスのクライアント証明書ファイル(X.509 の PEM 形式)の絶対パス名を指定する。	String	−	なし 暗号化通信でクライアント認証をする場合は、必ず指定してください。
grpc.tls.client.client-auth.key	クライアント認証をする場合に、自プロセスのクライアント証明書を作成した秘密鍵ファイル(pkcs8 の PEM 形式)の絶対パス名を指定する。	String	−	なし 暗号化通信でクライアント認証をする場合は、必ず指定してください。

パラメタ	説明	型	設定範囲	デフォルト値
grpc.tls.session.cache-size	SSL セッション情報のキャッシュ数を指定します。 0 を指定すると、gRPC のデフォルト値が使用されます。	Integer	0-2147483647	0
grpc.tls.session.timeout	キャッシュした SSL セッション情報の保持時間（秒）を指定します。 0 を指定すると、gRPC のデフォルト値が使用されます。	Integer	0-65535	0
grpc.keepalive.interval	PING 要求電文の送信間隔を秒単位で指定します。 0 を指定した場合は、PING 要求電文を送信しません。1～9 を指定した場合は 10 秒となります。 ネットワーク機器の無通信監視タイマなどによる TCP コネクション切断回避が目的の場合は、無通信監視タイマ値より小さい値を指定してください。	Integer	0-100000	0
grpc.keepalive.timeout	PING 応答電文の応答待ち時間を秒単位で指定します。 PING 要求電文の送信後、このタイマの指定値以内に PING 応答電文を受信できない場合は TCP コネクションを切断します。	Integer	1-65535	60
grpc.watchtime	Mediator への要求電文に対する応答電文の待ち時間を秒単位で指定します。 0 を指定した場合は、タイマ監視をせずに無限に応答を待ちます。 Orchestrator からトランザクションに参加した Mediator に commit/rollback の要求電文を送信して、すべての通信でこのタイマがタイムアウトした場合、Orchestrator は応答の待ち合わせを終了し、全 Mediator が障害と判断してトランザクションの決着結果は不明として AP に応答します。そのため、応答電文の待ち時間はトランザクション制御には依存せず、AP の業務処理として許容できる待ち時間となります。	Integer	0-65535	100
lb.list-timeout-in-seconds	Kubernetes からのサービス一覧取得の待ち時間を秒単位で指定します。タイマ指定値以内に応答を受信できない場合は、Orchestrator の起動処理を異常終了します。	Integer	1-10800	60

（凡例）－：該当しません。

8.2.3　Entity-Service/Entity-Module

Entity-Service/Entity-Module のパラメタについて説明します。

(1)　jp.co.Hitachi.soft.hmppcto のパラメタ

各パラメタの接頭辞は、jp.co.Hitachi.soft.hmppcto です。

表 8–5　Entity-Service/Entity-Module の jp.co.Hitachi.soft.hmppcto のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
client.proposer.type	SQL-Participant または TCC-Participant の使用を設定します。 SQL：SQL-Participant を使用する TCC：TCC-Participant を使用する 注 TP1-Bridge を使用する場合も、TP1-Bridge の内部で Entity-Service のライブラリを使用しますが、このパラメタの指定は不要です。指定しても無視されます。 Entity-Service のライブラリは、Java アプリケーションに TP1-Bridge のライブラリが組み込まれていることで、TP1-Bridge を使用すると自動的に判断します。	String	SQL|TCC	なし TP1-Bridge を使用しない場合は、必ず設定してください。
branch.qualifier.list	Orchestrator の branch.qualifier.list と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			−
branch.qualifier.propagation-order	Orchestrator の branch.qualifier.propagation-order と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			−
grpc.watchtime	SQL-Participant への要求電文に対する応答電文の待ち時間を秒単位で指定します。 0 を指定した場合は、タイマ監視をしないで無限に応答を待ちます。SQL-Participant の場合だけ設定できます。 Entity-Service の JDBC Proxy Driver から Participant に SQL 実行のための要求電文を送信して、このタイマがタイムアウトした場合、Entity-Service は SQL-Participant が障害（DB アクセスに時間が掛かることなども含む）と判断してトランザクションブランチをロールバックします。 そのため、応答待ち時間はトランザクション制御には依存しないで、AP のサービスとして許容できる待ち時間となります。	Integer	0-65535	60

（凡例）−：該当しません。

(2)　spring.datasource のパラメタ

各パラメタの接頭辞は、spring.datasource です。

spring.datasource のパラメタは、SQL-Participant で JDBC Proxy Driver を使用する場合に設定してください。SQL-Participant、JDBC Proxy Driver を使用しない場合は設定不要です。spring.datasource に JDBC Proxy Driver の設定をすることで、AP は JDBC Proxy Driver の DataSource を参照できます。

表 8–6　Entity-Service の spring.datasource のパラメタ

パラメタ	説明	型
driver-class-name	Spring Framework が JDBC Driver クラスを特定するパラメタです。 次の HMP-PCTO の JDBC Proxy Driver のクラス名を設定してください。 jp.co.Hitachi.soft.hmppcto.client.PctoDriver	String
type	Spring Framework が DB のコネクション制御に使用する DataSource を指定するパラメタです。 次の HMP-PCTO の JDBC Proxy Driver のクラス名を設定してください。 jp.co.Hitachi.soft.hmppcto.jdbc.client.PctoDataSource	String
url	Spring Framework が DB に接続するために使用する URL です。 Participant の HMP-PCTO が gRPC を受信する URL を設定してください。 ポート番号は、SQL-Participant のパラメタ sql.app.port に設定したポート番号を設定してください。	String
username	Spring Framework が DB に接続するために使用するユーザ名を設定します。 HMP-PCTO はこの設定値を使用しないため任意の文字列を設定してください。 DB に接続するためのユーザ名は SQL-Participant で設定します。	String
password	Spring Framework が DB に接続するために使用するパスワードを設定します。 HMP-PCTO はこの設定値を使用しないため、任意の文字列を設定してください。 DB に接続するためのパスワードは SQL-Participant で設定します。	String

なお、設定範囲とデフォルト値は、Spring Framework の仕様に依存します。

(3)　spring.sql のパラメタ

各パラメタの接頭辞は、spring.sql です。

spring.sql のパラメタは、SQL-Participant で JDBC Proxy Driver を使用する場合に設定してください。SQL-Participant、JDBC Proxy Driver を使用しない場合は設定不要です。

表 8–7　Entity-Service の spring.sql のパラメタ

パラメタ	説明	型
init.mode	Spring Framework が SQL データベースの初期化を行うかどうかを指定するパラメタです。 HMP-PCTO では SQL データベースの初期化を行う必要がないため、次の値を設定してください。	String

パラメタ	説明	型
	never	

なお、設定範囲とデフォルト値は、Spring Framework の仕様に依存します。

8.2.4 SQL-Participant

SQL-Participant のパラメタについて説明します。

(1) server のパラメタ

各パラメタの接頭辞は、server です。

表 8–8 SQL-Participant の server のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
port	SQL-Participant 用ポート番号を設定してください。 動的ポートなど、ほかのポートと重ならないように設定してください。	Integer	1024-65535	なし。 必ず指定してください。

(2) jp.co.Hitachi.soft.hmppcto のパラメタ

各パラメタの接頭辞は、jp.co.Hitachi.soft.hmppcto です。

表 8–9 SQL-Participant の jp.co.Hitachi.soft.hmppcto のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
external-consensus.interval-in-millis	Mediator などに障害が発生し、補助合意によってトランザクションの分散合意を行う場合の、補助合意を試行する間隔をミリ秒単位で指定してください。 このタイマが短ければ、Mediator などに障害が発生した場合のトランザクションの遅延を小さくできます。 ただし、このタイマが短過ぎると通信遅延を障害と誤認し、不要な補助合意が発生するため、通信遅延を考慮した値を設定してください。 この設定値に 1000 ミリ秒以上を指定した場合は 1000 の単位に切り上げたタイマ値とします。 なお、このタイマ値は初回だけ、最大で設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は最大 1000 ミリ秒です。	Integer	100-10000	1000

パラメタ	説明	型	設定範囲	デフォルト値
external-consensus.eads.properties	補助合意に使用する EADS のクライアントプロパティファイルの絶対パス名を設定してください。	String	ファイルパス	/eads-properties/hmp-pcto.eads-client.properties
external-consensus.eads.cache	補助合意に使用する EADS クラスタに対して、eztool createcache コマンドで作成したキャッシュ名を設定してください。	String	32 バイト以内の半角英数字（0～9、A～Z、および a～z）	hmppctoCache
sql.xa-server-name	SQL-Participant 起動時に未決着トランザクションが DB に残っているか確認する際に、DB から取得した未決着トランザクションが SQL-Participant が実行したトランザクションか判断するために使用する識別子です。 同じ DB に接続する SQL-Participant が複数ある場合ユニークな値を指定してください。	String	32 バイト以内の文字列	なし。 必ず指定してください。
sql.pool.allocation-wait-time-in-millis	SQL-Participant が DB アクセスのためのコネクションをコネクションプールから取得する際に、コネクションプールに空きが無かった場合の空き待ち時間をミリ秒単位で設定します。 0 を指定した場合は、コネクションプールの空き待ちを行いません。 空きがなかった場合、このパラメタに指定した時間だけコネクションの空きを待ちます。空き待ちを行わない、または待ち時間内にコネクションの空きができなかった場合はエラーとなり、JDBC Proxy Driver を使用してコネクションの取得を実行した AP に例外を送出します。	Long	0-65535000	61000
sql.pod.service-name	SQL-Participant を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名を「.」でつないだ形式で設定します。 Service 名とネームスペース名には、英小文字（a～z）、数字（0～9）、およびダッシュ（-）を使用できます。ただし、Service 名およびネームスペース名の先頭と終端に使用できるのは英小文字（a～z）、または数字（0～9）です。 このパラメタには、Mediator からホスト名として認識できる値を設定する必要があります。 設定例： `name.namespace`	String	3 バイト～41 バイトの文字列	なし。 必ず指定してください。
sql.app.port	SQL-Participant が Entity-Service から gRPC 通信を受信するためのポート番号を設定します。	Integer	1024-65535	20553

パラメタ	説明	型	設定範囲	デフォルト値
sql.mediator.port	SQL-Participant が Mediator から gRPC 通信を受信するためのポート番号を設定します。	Integer	1024-65535	20552
sql.jdbc.driver.path	SQL-Participant が DB に接続するために使用する JDBC Driver について、「2.4.7　SQL-Participant コンテナイメージの作成（SQL-Participant 限定）」でコンテナ内に配置したパスを設定します。絶対パスで指定してください。	String	−	なし。 必ず指定してください。
sql.datasource.class-name	SQL-Participant が JDBC Driver からロードするデータソースのクラス名を設定します。 接続する RDBMS ごとのデータソースのクラス名を次に示します。 • PostgreSQL の場合 org.postgresql.xa.PGXADataSource • SQL Server の場合 com.microsoft.sqlserver.jdbc.SQLServerXADataSource • HiRDB の場合 JP.co.Hitachi.soft.HiRDB.JDBC.PrdbXADataSource	String	−	なし。 必ず指定してください。
sql.datasource.property.values	JDBC Driver からロードしたデータソースに設定するプロパティの名前と値を「=」でつないだ形式で指定します。複数のプロパティを指定する場合は sql.datasource.property.delimiter で設定した値で区切って設定してください。 データソースに設定するプロパティには、次の値を指定しないでください。 • = • sql.datasource.property.delimiter で設定した値 設定例： ＜ PostgreSQL の場合＞ `serverName=sample-bank1;portNumber=5431;databaseName=appuser1;user=appuser1;password=appuser1` PostgreSQL の JDBC Driver の場合、次のプロパティを指定できます。 • serverName • databaseName • portNumber • user • password 詳細は、PostgreSQL JDBC Driver のサイトの DataSource のプロパティを参照してください。	String	−	なし。 必ず指定してください。

パラメタ	説明	型	設定範囲	デフォルト値

	PostgreSQL の JDBC Driver は、databaseName を省略すると、user に指定した値を databaseName の値として使用します。
	＜ SQL Server の場合＞
	`encrypt=false;serverName=sql-server-pod1;portNumber=5431;user=appuser1;password=appuser1`
	SQL Server の JDBC Driver の場合、次のプロパティを指定できます。
	• encrypt
	• serverName
	• portNumber
	• user
	• password
	詳細は、SQL Server のサイトの JDBC の接続プロパティを参照してください。
	＜ HiRDB の場合＞
	`XAOpenString=HDB1+/deployments/HiRDB.ini;XACloseString=HDB1+/deployments/HiRDB.ini`
	HiRDB の JDBC Driver の場合、次に示す形式で HiRDB のクライアント環境定義ファイル（HiRDB.ini）を指定してください。
	`XAOpenString=<HiRDBの環境変数グループ識別子（任意の4文字の英数字列）> + <HiRDB.iniファイルのパス>;XACloseString= <HiRDBの環境変数グループ識別子（任意の4文字の英数字列）> + <HiRDB.iniファイルのパス>`
	HiRDB.ini に指定する接続情報については、「8.4.3 HiRDB（SQL-Participant 限定）」を参照してください。			

sql.datasource.property.delimiter	sql.datasource.property.values で複数のプロパティを指定した場合の区切りとなる文字を設定します。	String	「=」以外の 1 バイトの文字	; (半角セミコロン)
sql.pool.max-pool-size	SQL-Participant がコネクションプールにプールする DB コネクションの数を設定します。 SQL-Participant 起動時にこのパラメタに指定した数の DB コネクションがプールされます。 データベースの同時接続数以下、かつ同時実行するトランザクション数以上の値を指定してください。 データベースの同時接続数が指定されているパラメタを次に示します。 • PostgreSQL の場合 max_connections • SQL Server の場合 user connections	Integer	1-100000	なし。 必ず指定してください。

パラメタ	説明	型	設定範囲	デフォルト値
	• HiRDB の場合 pd_max_users SQL-Participant 起動時に、SQL-Participant が使用する DB コネクションの数が sql.pool.max-pool-size に指定した値を超えた場合、コネクションの確立が失敗して、起動に失敗します。 また、SQL-Participant の起動後に、SQL-Participant が使用する DB コネクションの数が sql.pool.max-pool-size に指定した値を超えた場合、コネクションプールの空き待ちになります。			
sql.pool.transaction-timeout-in-sec	DB 側のトランザクションタイムアウト時間を秒単位で設定します。 0 を指定した場合、DB 側のトランザクションタイムアウト時間の設定が有効になります。 このトランザクションタイムアウト時間は javax.transaction.xa.XAResource.setTransactionTimeout()メソッドを使用して設定を行います。 そのため、使用する JDBC Driver が setTransactionTimeout()メソッドをサポートしているかご確認ください。サポートしていない場合は、DB 側のトランザクションタイムアウト時間設定が有効になります。 定義を省略した場合は 0 を指定した場合と同様に、DB 側のトランザクションタイムアウト時間の設定が有効になります。	Integer	0-65535 トランザクションタイムアウト時間に設定できる範囲は使用する JDBC Driver の仕様をご確認ください。	0
sql.transaction-termination.timeout-in-millis	トランザクション決着監視タイマのタイマ監視時間をミリ秒単位で設定します。 DB の排他ロックが解けない状態となることを許容できる時間をタイマ監視し、該当する時間を超えてもトランザクションが決着しなかった場合に、DB の排他ロック解除のためにトランザクションを決着に向かわせるタイマです。 AP がユーザにレスポンスを返すまでの許容時間、またはトランザクションブランチの処理時間の業務要件に応じた値を指定してください。 なお、複数の業務サービスがある場合は、すべてのサービスの業務要件に応じた値の合計時間で監視時間を設定する必要があります。 業務サービスは Orchestrator から呼び出され、DB などのリソースの更新を行う複数のサービスである Entity-Service を指します。 このタイマ値は初回だけ、最大で設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は最大 1000 ミリ秒です。また、	Long	100-6000000	60000

パラメタ	説明	型	設定範囲	デフォルト値
	この設定値に 1000 ミリ秒以上を指定した場合は、1000 の単位に切り上げたタイマ値とします。			
sql.transaction-termination.consensus.replay-interval-in-millis	トランザクション分散合意インターバルタイマのタイマ監視時間をミリ秒単位で設定します。 このパラメタの値は次の処理のために設定します。この値に設定した時間を超えてもトランザクションが決着しなかった場合、トランザクション決着処理を再実行（リプレイ）し、DB の排他ロック解除のためにトランザクションを決着に向かわせます。 • トランザクションをコミットまたはロールバックするためにトランザクション決着処理をする • トランザクション決着監視タイマのタイムアウトによるトランザクション決着処理をする • SQL-Participant ダウンからの再起動時にトランザクション決着処理をする このタイマを短く設定することで、SQL-Participant 内で合意が滞ったトランザクションを速やかに決着できますが、障害が長引いた場合にリトライの回数が多くなり、トランザクション決着のための通信に負荷が掛かるため、通信負荷を考慮した値を設定してください。 このタイマ値は初回だけ、最大で設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は最大 1000 ミリ秒です。また、この設定値に 1000 ミリ秒以上を指定した場合は、1000 の単位に切り上げたタイマ値とします。	Long	100-6000000	1000
sql.participant-processing.thread-pool-size	Entity-Service からのトランザクションブランチの Prepare 指示／キャンセル指示の実行に使用するスレッド数、および Mediator からの投票要求／トランザクション決着結果通知の実行に使用するスレッド数を指定してください。 スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから処理を実行します。 スレッドの空き待ちを防ぐ理論上の必要なスレッド数は、サービスの同時実行数×トランザクションに参加する Mediator 数です。 ただし、Mediator からの通知が重ならないかぎりは、理論上の必要なスレッド数は必要ありません。通常はデフォルト値を使用してください。	Integer	1-1000	12
sql.participant-processing.queue-size	SQL-Participant がトランザクション処理(Entity-Service からの SQL 実行、Prepare 指示／トランザクションのキャンセル指示、投票処理、分散合意処理およびトランザクション完了通知処理)に使用するスレッドの割り当てを待つキューのサイズを指定します。	Integer	1-2147483647	25

パラメタ	説明	型	設定範囲	デフォルト値
	理論上の必要な設定値は次のとおりです。 {m ＋（p−1）×m×m} ×m×t m：トランザクションに参加する Mediator 数 p：トランザクションに参加する Participant 数 t：同時実行するトランザクション数			
mediator-process-management.thread-pool-size	Mediator へ送信を行うスレッドを管理するスレッド数を指定します。 理論上必要なスレッド数は、トランザクションに参加する Mediator 数×2×同時実行するトランザクション数です。 ただし、処理の実行が重ならないかぎりは、理論上必要なスレッド数は必要ありません。通常はデフォルト値を使用してください。	Integer	1-32767	30
mediator-communication.thread-pool-size	Participant が Mediator と送受信を行うスレッド数を指定します。 スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから処理を実行します。スレッドの空き待ちを発生しないようにするための理論上必要なスレッド数は、トランザクションに参加する Mediator 数×2×同時実行するトランザクション数です。 ただし、送受信が重ならないかぎりは、理論上の必要なスレッド数は必要ありません。通常はデフォルト値を使用してください。	Integer	1-1000	30
mediator-communication.queue-size	Participant が Mediator との送受信に使用するスレッドの割り当てを待つキューのサイズを指定します。 理論上必要な設定値は次のとおりです。 {m ＋（p−1）×m×m} ×m×2×t m：トランザクションに参加する Mediator 数 p：トランザクションに参加する Participant 数 t：同時実行するトランザクション数	Integer	1-2147483647	25
sql.db-type	SQL-Participant が接続する RDBMS を指定します。	String	PostgreSQL ｜ SQLServer ｜ HiRDB	なし。 必ず指定してください。
external-consensus.type	トライアル版の補助合意サーバを使用する場合に指定します。 トライアル版の補助合意サーバは、補助合意サーバ障害時の補助合意を必要とするトランザクションの分散合意の一貫性を保証しません。	String	free	なし トライアル版を使用する場合必ず指定してください。
grpc.use-tls	暗号化通信をするかどうかを指定します。 true：暗号化通信をします。	Boolean	true ｜ false	false

パラメタ	説明	型	設定範囲	デフォルト値
	false：暗号化通信をしません。			
grpc.tls.client.server-auth.certificate	接続先サーバのサーバ証明書を発行した CA の証明書(X.509 の PEM 形式)ファイルの絶対パス名を指定する。	String	−	なし 暗号化通信をする場合は、必ず指定してください。
grpc.tls.server.server-auth.certificate	自プロセスのサーバ証明書(X.509 の PEM 形式)ファイルの絶対パス名を指定する。	String	−	なし 暗号化通信をする場合は、必ず指定してください。
grpc.tls.server.server-auth.key	自プロセスのサーバ証明書を作成した秘密鍵ファイル(pkcs8 の PEM 形式)の絶対パス名を指定する。	String	−	なし 暗号化通信をする場合は、必ず指定してください。
grpc.tls.client.client-auth.certificate	クライアント認証をする場合に、自プロセスのクライアント証明書ファイル(X.509 の PEM 形式)の絶対パス名を指定する。	String	−	なし 暗号化通信でクライアント認証をする場合は、必ず指定してください。
grpc.tls.client.client-auth.key	クライアント認証をする場合に、自プロセスのクライアント証明書を作成した秘密鍵ファイル(pkcs8 の PEM 形式)の絶対パス名を指定する。	String	−	なし 暗号化通信でクライアント認証をする場合は、必ず指定してください。
grpc.tls.server.client-auth.store	クライアント認証をする場合に、接続元クライアントの秘密鍵が格納されているキーストア（pkcs12 または JKS）ファイルの絶対パス名を指定します。	String	−	なし 暗号化通信でクライアント認証をする場合は、必ず指定してください。
grpc.tls.server.client-auth.password	クライアント認証をする場合に、grpc.tls.server.clientAuth.store で指定したキーストアファイルのパスワードを指定します。	String	−	なし 暗号化通信でクライア

パラメタ	説明	型	設定範囲	デフォルト値
				ント認証をする場合は、必ず指定してください。
grpc.tls.session.cache-size	SSL セッション情報のキャッシュ数を指定します。 0 を指定すると、gRPC のデフォルト値が使用されます。	Integer	0-2147483647	0
grpc.tls.session.timeout	キャッシュした SSL セッション情報の保持時間（秒）を指定します。 0 を指定すると、gRPC のデフォルト値が使用されます。	Integer	0-65535	0
grpc.keepalive.interval	PING 要求電文の送信間隔を秒単位で指定します。 0 を指定した場合は、PING 要求電文を送信しません。1～9 を指定した場合は 10 秒となります。 ネットワーク機器の無通信監視タイマなどによる TCP コネクション切断回避が目的の場合は、無通信監視タイマ値より小さい値を指定してください。	Integer	0-100000	0
grpc.keepalive.timeout	PING 応答電文の応答待ち時間を秒単位で指定します。 PING 要求電文の送信後、このタイマの指定値以内に PING 応答電文を受信できない場合は TCP コネクションを切断します。	Integer	1-65535	60
grpc.watchtime	Mediator への要求電文に対する応答電文の待ち時間を秒単位で指定します。 0 を指定した場合は、タイマ監視をしないで無限に応答を待ちます。 Participant から Mediator に要求電文を送信すると、Mediator からはすぐに要求を受け付けた旨の応答電文を送信するため、応答電文の待ち時間は gRPC 通信の送受信時間として許容できる待ち時間となります。	Integer	0-65535	5
mediator.name	Orchestrator の mediator.name と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			なし 必ず設定してください。
mediator.name-suffix	Orchestrator の mediator.name-suffix と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			なし 必ず設定してください。
mediator.port	Orchestrator の mediator.port と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			20551

（凡例）－：該当しません。

8.2.5　TCC-Participant

TCC-Participant のパラメタについて説明します。

(1)　server のパラメタ

各パラメタの接頭辞は、server です。

表 8–10　TCC-Participant の server のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
port	TCC-Participant 用ポート番号を設定してください。 動的ポートなど、ほかのポートと重ならないように設定してください。	Integer	1024-65535	なし。 必ず指定してください。

(2)　jp.co.Hitachi.soft.hmppcto のパラメタ

各パラメタの接頭辞は、jp.co.Hitachi.soft.hmppcto です。

表 8–11　TCC-Participant の jp.co.Hitachi.soft.hmppcto のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
tcc.pod.service-name	TCC-Participant を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名を「.」でつないだ形式で設定します。 Service 名とネームスペース名には、英小文字（a～z）、数字（0～9）、およびダッシュ（-）を使用できます。ただし、Service 名およびネームスペース名の先頭と終端に使用できるのは英小文字（a～z）、または数字（0～9）です。 このパラメタには、Mediator からホスト名として認識できる値を設定する必要があります。 設定例： `name.namespace`	String	3 バイト～41 バイトの文字列	なし。 必ず指定してください。
tcc.mediator.port	TCC-Participant が Mediator から gRPC 通信を受信するためのポート番号を設定します。	Integer	1024-65535	20552
tcc.recover.enabled	TCC-Participant 起動時に未決着トランザクションの Recovery 機能を実施するかどうかを設定します。true を設定した時 Recovery 機能が実行され、false を設定した時は実行されません。	Boolean	true | false	true
tcc.recover.order	未決着トランザクションの Recovery 機能について、Spring Framework の ApplicationListener 機能の「ApplicationReadyEvent」イベントハンドラでの実行優先度のオーダー値を指定します。	Integer	-2147483648-2147483647	2147483647

パラメタ	説明	型	設定範囲	デフォルト値
	AP でも「ApplicationReadyEvent」イベントハンドラを使用するケースを考慮し、HMP-PCTO の ApplicationReadyEvent イベントハンドラ（未決着トランザクションの Recovery 機能）と、AP の ApplicationReadyEvent イベントハンドラの実行順序を設定できるようにするためのパラメタです。 Spring Framework はオーダー値が小さいイベントハンドラから順に実行します。 もしオーダー値が一致しているイベントハンドラが複数存在した場合の実行順序は不定です。			
tcc.trn-termination.timeout-in-millis	トランザクション決着監視タイマのタイマ監視時間をミリ秒単位で設定します。 障害によってトランザクション決着に至らない状態となることを許容できる時間をタイマ監視し、設定時間を超えてもトランザクションが決着しなかった場合に、ユーザが許容できる時間（業務要件）を超過したと判断しトランザクションを決着に向かわせるタイマです。 トランザクションブランチの処理時間の業務要件に応じた値を指定してください。 なお、複数の業務サービスがある場合は、すべてのサービスの業務要件に応じた値の合計時間で監視時間を設定する必要があります。 このタイマ値が短過ぎるとトランザクションが Cancel 決着します。 このタイマ値は初回だけ、最大で設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は最大 1000 ミリ秒です。 また、この設定値に 1000 ミリ秒以上を指定した場合は 1000 の単位に切り上げたタイマ値とします。	Long	100-6000000	60000
tcc.consensus.interval-in-millis	トランザクション分散合意処理をリトライする間隔をミリ秒単位で設定します。 障害などによってトランザクションの分散合意ができないとき、設定した間隔でトランザクションの分散合意が決着するまで分散合意処理をリトライし、分散合意によるトランザクションの決着が必ず行われるようにするためのリトライ間隔です。 また、TCC-Participant が再起動したときに未決着となっているトランザクションがある場合にも分散合意によってトランザクションを決着させます。この場合も設定した間隔によってトランザクションの分散合意が決着するまで分散合意処理をリトライします。	Long	100-6000000	60000

パラメタ	説明	型	設定範囲	デフォルト値
	この間隔が短過ぎるとトランザクションの分散合意のリトライが無駄に実施され、トランザクションのレイテンシやシステム負荷に影響が出ます。 このタイマ値は初回だけ、最大で設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は最大 1000 ミリ秒です。また、この設定値に 1000 ミリ秒以上を指定した場合は 1000 の単位に切り上げたタイマ値とします。			
tcc.participant-processing.thread-pool-size	TCC-Participant がトランザクション処理で実行するスレッド数(投票処理、分散合意処理およびトランザクション完了通知処理)を指定してください。 スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから処理を実行します。スレッドの空き待ちを発生しないようにするための理論上必要なスレッド数は、トランザクションに参加する Mediator 数×同時実行するトランザクション数です。 ただし、処理の実行が重ならないかぎりは、理論上必要なスレッド数は必要ありません。通常はデフォルト値を使用してください。	Integer	1-1000	12
tcc.participant-processing.queue-size	TCC-Participant がトランザクション処理（投票処理、分散合意処理およびトランザクション完了通知処理）に使用するスレッドの割り当てを待つキューのサイズを指定します。 理論上必要な設定値は次のとおりです。 {m ＋（p−1）×m×m} ×m×t m：トランザクションに参加する Mediator 数 p：トランザクションに参加する Participant 数 t：同時実行するトランザクション数	Integer	1-2147483647	25
mediator-process-management.thread-pool-size	SQL-Participant の mediator-process-management.thread-pool-size と同じです。 「(2)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。	Integer	1-32767	30
mediator-communication.thread-pool-size	SQL-Participant の mediator-communication.thread-pool-size と同じです。 「(2)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。	Integer	1-1000	30
mediator-communication.queue-size	SQL-Participant の mediator-communication.queue-size と同じです。 「(2)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。	Integer	1-2147483647	25

パラメタ	説明	型	設定範囲	デフォルト値
external-consensus.interval-in-millis	Mediator などに障害が発生し、補助合意によってトランザクションの分散合意を行う場合の、補助合意を試行する間隔をミリ秒単位で指定してください。 このタイマが短ければ、Mediator などに障害が発生した場合のトランザクションの遅延を小さくできます。 ただし、このタイマが短過ぎると通信遅延を障害と誤認し、不要な補助合意が発生するため、通信遅延を考慮した値を設定してください。 この設定値に 1000 ミリ秒以上を指定した場合は 1000 の単位に切り上げたタイマ値とします。 なお、このタイマ値は初回だけ、最大で設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は最大 1000 ミリ秒です。	Integer	100-10000	1000
external-consensus.eads.properties	補助合意に使用する EADS のクライアントプロパティファイルの絶対パス名を設定してください。	String	ファイルパス	/eads-properties/hmp-pcto.eads-client.properties
external-consensus.eads.cache	補助合意に使用する EADS クラスタに対して、eztool createcache コマンドで作成したキャッシュ名を設定してください。	String	32 バイト以内の半角英数字（0～9、A～Z、および a～z）	hmppctoCache
external-consensus.type	トライアル版の補助合意サーバを使用する場合に指定します。 トライアル版の補助合意サーバは、補助合意サーバ障害時の補助合意を必要とするトランザクションの分散合意の一貫性を保証しません。	String	free	なし トライアル版を使用する場合必ず指定してください。
mediator.name	Orchestrator の mediator.name と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			なし 必ず設定してください。
mediator.name-suffix	Orchestrator の mediator.name-suffix と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			なし 必ず設定してください。
mediator.port	Orchestrator の mediator.port と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			20551
grpc.use-tls	暗号化通信をするかどうかを指定します。 true：暗号化通信をします。 false：暗号化通信をしません。	Boolean	true | false	false

パラメタ	説明	型	設定範囲	デフォルト値
grpc.tls.client.server-auth.certificate	接続先サーバのサーバ証明書を発行した CA の証明書(X.509 の PEM 形式)ファイルの絶対パス名を指定する。	String	−	なし 暗号化通信をする場合は、必ず指定してください。
grpc.tls.server.server-auth.certificate	自プロセスのサーバ証明書(X.509 の PEM 形式)ファイルの絶対パス名を指定する。	String	−	なし 暗号化通信をする場合は、必ず指定してください。
grpc.tls.server.server-auth.key	自プロセスのサーバ証明書を作成した秘密鍵ファイル(pkcs8 の PEM 形式)の絶対パス名を指定する。	String	−	なし 暗号化通信をする場合は、必ず指定してください。
grpc.tls.client.client-auth.certificate	クライアント認証をする場合に、自プロセスのクライアント証明書ファイル(X.509 の PEM 形式)の絶対パス名を指定する。	String	−	なし 暗号化通信でクライアント認証をする場合は、必ず指定してください。
grpc.tls.client.client-auth.key	クライアント認証をする場合に、自プロセスのクライアント証明書を作成した秘密鍵ファイル(pkcs8 の PEM 形式)の絶対パス名を指定する。	String	−	なし 暗号化通信でクライアント認証をする場合は、必ず指定してください。
grpc.tls.server.client-auth.store	クライアント認証をする場合に、接続元クライアントの秘密鍵が格納されているキーストア（pkcs12 または JKS）ファイルの絶対パス名を指定します。	String	−	なし 暗号化通信でクライアント認証をする場合は、必ず指定してください。
grpc.tls.server.client-auth.password	クライアント認証をする場合に、grpc.tls.server.clientAuth.store で指定したキーストアファイルのパスワードを指定します。	String	−	なし 暗号化通信でクライアント認証をする場合は、

パラメタ	説明	型	設定範囲	デフォルト値
				必ず指定してください。
grpc.tls.session.cache-size	SSL セッション情報のキャッシュ数を指定します。 0 を指定すると、gRPC のデフォルト値が使用されます。	Integer	0-2147483647	0
grpc.tls.session.timeout	キャッシュした SSL セッション情報の保持時間（秒）を指定します。 0 を指定すると、gRPC のデフォルト値が使用されます。	Integer	0-65535	0
grpc.keepalive.interval	PING 要求電文の送信間隔を秒単位で指定します。 0 を指定した場合は PING 要求電文を送信しません。1～9 を指定した場合は 10 秒となります。 ネットワーク機器の無通信監視タイマなどによる TCP コネクション切断回避が目的の場合は、無通信監視タイマ値より小さい値を指定してください。	Integer	0-100000	0
grpc.keepalive.timeout	PING 応答電文の応答待ち時間を秒単位で指定します。 PING 要求電文の送信後、このタイマの指定値以内に PING 応答電文を受信できない場合は TCP コネクションを切断します。	Integer	1-65535	60
grpc.watchtime	Mediator への要求電文に対する応答電文の待ち時間を秒単位で指定します。 0 を指定した場合は、タイマ監視をしないで無限に応答を待ちます。 Participant から Mediator に要求電文を送信すると、Mediator からはすぐに要求を受け付けた旨の応答電文を送信するため、応答電文の待ち時間は gRPC 通信の送受信時間として許容できる待ち時間となります。	Integer	0-65535	5

（凡例）－：該当しません。

8.3 HMP-PCTO と RAS のコンポーネントに設定するパラメタ

HMP-PCTO のコンポーネントと RAS のコンポーネントに設定するパラメタについて説明します。

8.3.1 Mediator

Mediator のパラメタを説明します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

次の表に記載されていないパラメタは設定を変更しないでください。

表 8-12 Mediator の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Mediator の Namespace 名を設定してください。	String	−	my-namespace
replicas	起動する Mediator 数を設定してください。 Orchestrator の jp.co.Hitachi.soft.hmppcto.learner.consensus-type に Three-Quarters を設定した場合は、5 以上の値を設定してください。	Integer	2-64	なし 必ず設定してください。
image	「3.4.3 Mediator コンテナイメージの作成（通常版限定）」で作成した Mediator のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。

imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。
	次のように指定します。
	`imagePullSecrets:`
	`- name: regcred`
	資格情報が不要な場合は、設定不要です。
		List<Object>	−	[]
				（半角左角括弧と半角右角括弧です）

imagePullPolicy	Mediator のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
resources	Mediator のコンテナのリソースを設定します。	Object	−	{} （波括弧です）

パラメタ	説明	型	設定範囲	デフォルト値

	コンテナに割り当てるリソースを明示的に指定した場合に設定してください。
	次の形式で requests や limits を設定します。
	`resources:`
	`  limits:`
	`  cpu: 2`
	`  memory: 1Gi`
	`  requests:`
	`  cpu: 1`
	`  memory: 512Mi`			

dependencyCheck.image	「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.targets	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのターゲット Kubernetes オブジェクト群を設定してください。 指定方法の詳細は「6.5.1　デプロイ依存関係チェックスクリプト（dependency_check.sh）」を参照してください。	List<String>	−	• - my-namespace/deployments/logstash • - my-namespace/deployments/jaeger-query • - my-namespace/deployments/jaeger-collector • - my-namespace/deployments/prometheus

パラメタ	説明	型	設定範囲	デフォルト値
				• - my-namespace/deployments/metricbeat • - my-namespace/daemonsets/filebeat
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※	180
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※	30
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※	10
livenessProbe.healthCheck.connectTimeoutSeconds	Liveness Probe で実行するトランザクションヘルスチェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
livenessProbe.healthCheck.readTimeoutSeconds	Liveness Probe で実行するトランザクションヘルスチェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
postStart.healthCheck.retryCount	postStart フックで実行するトランザクションヘルスチェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
postStart.healthCheck.retryIntervalSeconds	postStart フックで実行するトランザクションヘルスチェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
postStart.healthCheck.connectTimeoutSeconds	postStart フックで実行するトランザクションヘルスチェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
postStart.healthCheck.readTimeoutSeconds	postStart フックで実行するトランザクションヘルスチェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
preStop.waitTransactions.retryCount	preStop フックで実行する未決着トランザクション待機スクリプトのリトライ回数を設定してください。	Integer	0-1080	30

パラメタ	説明	型	設定範囲	デフォルト値
preStop.waitTransactions.retryIntervalSeconds	preStop フックで実行する未決着トランザクション待機スクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
preStop.waitTransactions.connectTimeoutSeconds	preStop フックで実行する未決着トランザクション待機スクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
preStop.waitTransactions.readTimeoutSeconds	preStop フックで実行する未決着トランザクション待機スクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	330
participant.sender.threadPoolSize	Mediator から Participant への送信に使用するスレッド数を設定してください。 1 トランザクションの分散合意時に、トランザクションに参加した Participant 数分のスレッドを使用して送信します。スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから送信します。 スレッドの空き待ちを回避するための理論上の必要なスレッド数は、同時実行トランザクション数× Participant 数です。 ただし、トランザクションの分散合意処理が重ならないかぎりは、理論上の必要なスレッド数は必要ありません。通常はデフォルト値を使用してください。	Integer	1-1000	30
participant.sender.queueSize	Mediator から Participant への送信に使用するスレッドの割り当て待ちキューのサイズを指定してください。 理論上の必要な設定値は次のとおりです。 {m ＋（p−1）×m×m} ×p×p ×t m：トランザクションに参加する Mediator 数 p：トランザクションに参加する Participant 数 t：同時実行するトランザクション数	Integer	1-2147483647	25

パラメタ	説明	型	設定範囲	デフォルト値
mediator.judge.threadPoolSize	Mediator の分散合意処理に使用するスレッド数を設定してください。 スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから処理を実行します。スレッドの空き待ちを発生しないようにするための理論上必要なスレッド数は、トランザクションに参加する Participant 数×同時実行するトランザクション数です。 ただし、処理の実行が重ならないかぎりは、理論上必要なスレッド数は必要ありません。通常はデフォルト値を使用してください。	Integer	1-1000	30
mediator.judge.queueSize	Mediator の分散合意処理に使用するスレッドの割り当て待ちキューのサイズを指定してください。 理論上の必要なスレッド数は次のとおりです。 {m ＋（p－1）×m×m} ×p×t m：トランザクションに参加する Mediator 数 p：トランザクションに参加する Participant 数 t：同時実行するトランザクション数	Integer	1-2147483647	25
grpc.server.threadPoolSize	Mediator が受信に使用するスレッド数を設定してください。 スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから処理を実行します。スレッドの空き待ちを発生しないようにするための理論上必要なスレッド数は、トランザクションに参加する Participant 数×同時実行するトランザクション数です。 ただし、処理の実行が重ならないかぎりは、理論上必要なスレッド数は必要ありません。通常はデフォルト値を使用してください。	Integer	1-1000	30
grpc.server.queueSize	Mediator が受信に使用するスレッドの割り当て待ちキューのサイズを指定してください。 理論上の必要なスレッド数は次のとおりです。 [{m ＋（p－1）×m×m} ×p ＋ 1] ×t	Integer	1-2147483647	25

パラメタ	説明	型	設定範囲	デフォルト値
	m：トランザクションに参加する Mediator 数 p：トランザクションに参加する Participant 数 t：同時実行するトランザクション数			
consensusLog.size	ConsensusLog ファイルに保持する ConsensusLog の数を設定してください。 TPS(1 秒当たりのトランザクション処理件数)×8 以上を設定してください。 この設定値を超える ConsensusLog の保持が発生した場合、該当する Mediator は分散合意に参加しません。 なお、この設定値に 1024（1 つの ConsensusLog のサイズ）を乗算し、1,048,576（ボリュームの書き込み容量不足にならないための予備サイズ（1MB））を加算したサイズを、Mediator の Pod に割り当てる ConsensusLog の永続ボリュームクレームのサイズに使用します。	Long	256-100000	256
consensusLog.io.threadPoolSize	ConsensusLog ファイルへの書き込みに使用するスレッド数を設定してください。 スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから処理を実行します。スレッドの空き待ちを発生しないようにするための理論上必要なスレッド数は、同時実行するトランザクション数です。 ただし、処理の実行が重ならないかぎりは、理論上必要なスレッド数は必要ありません。通常はデフォルト値を使用してください。	Integer	1-1000	12
consensusLog.io.queueSize	ConsensusLog ファイルへの書き込みに使用するスレッドの割り当てを待つキューのサイズを指定してください。 理論上必要な設定値は、同時実行するトランザクション数です。 ただし、トランザクションの分散合意処理が重ならないかぎりは、理論上必要な設定値は必要ありません。	Integer	1-2147483647	25

パラメタ	説明	型	設定範囲	デフォルト値
	通常はデフォルト値を使用してください。			
consensusLog.io.watchInSeconds	ConsensusLog ファイルへの書き込みおよび読み出し時の IO 監視時間を秒で設定してください。 0 を指定した場合、IO 監視を行いません。 この設定値を短く設定すると、IO ハングアップした場合に早期に検知できます。ただし、ハードウェアやドライバが障害検知するよりも短い時間を設定するとハードウェアやドライバによる障害理由が失われるおそれがあります。 IO 監視時間を超過した場合は障害メッセージを出力します。	Integer	0-3600	240
consensusLog.io.blockSize	ConsensusLog ファイルのブロックサイズをバイトで設定してください。 1024 の倍数を設定してください。 ブロックサイズは 1 回のファイル IO のサイズです。 ConsensusLog ファイルの永続ボリュームを NFS ストレージとする場合は、1024 を指定してください。 ブロックサイズの単位で NFS サーバと通信します。ブロックサイズを小さくすると、1 回のファイル IO のサイズが小さくなるため、NFS サーバとの通信量が減り、性能が向上します。	Integer	1024-16384	4096
consensusLog.storageClassName	「3.6.2　Mediator の ConsensusLog の永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で作成する Mediator の Pod に割り当てる ConsensusLog の永続ボリュームのストレージクラス名を設定してください。	String	−	なし 必ず設定してください。
http.port	Mediator の http ポート番号を設定してください。 動的ポートなど、ほかのポートと重ならないよう設定してください。 なお、この設定値はトランザクションヘルスチェックスクリプトおよび未決着トランザクション待機スクリ	Integer	1024-65535	20554

パラメタ	説明	型	設定範囲	デフォルト値
	プトのコマンド引数の http ポート番号にも使用します。			
grpc.port	Mediator の gRPC を受信するポート番号を設定してください。 動的ポートなど、ほかのポートと重ならないよう設定してください。	Integer	1024-65535	20551
grpc.useTls	暗号化通信をするかどうかを指定します。 true：暗号化通信をします。 false：暗号化通信をしません。	Boolean	true|false	false
grpc.tls.secretName	「(3)　TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成」でユーザが作成した「Mediator の gRPC 通信機能の暗号化通信用シークレット」の Kubernetes オブジェクト名称を指定します。	String	−	なし grpc.useTls に true を設定した場合は、必ず設定してください。
grpc.tls.clientAuth.enabled	クライアント認証をするかどうかを指定します。 true：クライアント認証をします。 false：クライアント認証をしません。 true を設定する場合は、必ず grpc.useTls に true を設定してください。	Boolean	true|false	false
grpc.tls.session.cacheSize	SSL セッション情報のキャッシュ数を指定します。 0 を指定すると gRPC のデフォルト値が使用されます。	Integer	0-2147483647	0
grpc.tls.session.timeout	キャッシュした SSL セッション情報の保持時間（秒）を指定します。 0 を指定すると gRPC のデフォルト値が使用されます。	Integer	0-65535	0
grpc.keepalive.interval	PING 要求電文の送信間隔を秒単位で指定します。 0 を指定した場合は PING 要求電文を送信しません。1～9 を指定した場合は 10 秒となります。 ネットワーク機器の無通信監視タイマなどによる TCP コネクション切断回避が目的の場合は、無通信監視タイマ値より小さい値を指定してください。	Integer	0-100000	0

パラメタ	説明	型	設定範囲	デフォルト値
grpc.keepalive.timeout	PING 応答電文の応答待ち時間を秒単位で指定します。 PING 要求電文の送信後、このタイマの指定値以内に PING 応答電文を受信できない場合は TCP コネクションを切断します。	Integer	1-65535	60
grpc.watchtime	Participant への要求電文に対する応答電文の待ち時間を秒単位で指定します。 0 を指定した場合は、タイマ監視をせずに無限に応答を待ちます。 Mediator から Participant に要求電文を送信すると、Participant からはすぐに要求を受け付けた旨の応答電文を送信するため、応答電文の待ち時間は gRPC 通信の送受信時間として許容できる待ち時間となります。	Integer	0-65535	5
mediator.judge.proposalLogLimitNum	Mediator が管理する Proposal Log が、Participant からトランザクション完了通知を受信できないで、滞留してしまった場合に Proposal Log として許容できる数（Proposal Log が保存を許容できるグローバルトランザクション数）を指定します。 滞留した Proposal Log がこのパラメタの件数を超えた場合、トランザクションの完了を確認するため Mediator から Participant へトランザクション完了確認通知を送信します。 このパラメタの件数が小さ過ぎる場合、トランザクション完了通知の送信頻度が多くなり、通信量が増加します。 このパラメタの件数が大き過ぎる場合、トランザクション完了通知の送信頻度が少なくなり、メモリを圧迫します。 Proposal Log に使用するメモリ量は、グローバルトランザクションに含まれる Participant の構成によって可変となります。 Proposal Log 1 件当たりのメモリ使用量は次のとおりです。 1400 ＋(800×トランザクションに参加した Participant 数)	Integer	1-17000	512

パラメタ	説明	型	設定範囲	デフォルト値
tracing.instanceId	共通の tracing.instance-id と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.url	共通の tracing.url と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.traceUse	共通の tracing.trace-use と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.queueSize	共通の tracing.queue-size と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.delayTime	共通の tracing.delay-time と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.batchSize	共通の tracing.batch-size と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
timezone	Mediator コンテナのタイムゾーンを指定します。ユーザ責務の Kubernetes アプリケーションと同じタイムゾーンに設定することを推奨します。 このパラメタに指定した値は環境変数 TZ ユーザに設定します。 このパラメタに指定した値がタイムゾーンとして不正な文字列である場合、Mediator コンテナのタイムゾーンは協定世界時(UTC)となります。	String	－	Asia/Tokyo
java.options	Mediator の Java プロセス起動時に Java VM に渡すオプションを指定します。 半角スペース区切りで複数のオプションを指定できます。 このパラメタには「-XX:+UseG1GC」オプションを必ず含めてください。「-XX:+UseG1GC」オプションを含めなかった場合、「-XX:+UseG1GC」オプションを含めた場合と比較して、Mediator の実行性能およびメモリ性能が低下することがあります。	String	－	-XX:+UseG1GC
ucars.enabled	Mediator コンテナで uCosminexus Application Runtime with Java for Spring Boot を有効にするかどうかを設定します。	Boolean	true|false	true

パラメタ	説明	型	設定範囲	デフォルト値
	true：有効にします（通常版の場合） false：無効にします（トライアル版の場合）			
ucars.snapshots.claimName	「3.6.10　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」で作成する uCosminexus Application Runtime with Java for Spring Boot スナップショットログの PersistentVolumeClaim リソースの名称(.metadata.name フィールドに指定した値)を設定してください。	String	－	なし ucars.enabled に true を設定した場合は、必ず設定してください。

（凡例）

－：該当しません。

注※

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

8.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）

TP1-Bridge のパラメタを説明します。

tp1.cltj から始まるパラメタは TP1/Client/J 環境定義に対応しています。パラメタの詳細については、マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」を参照してください。また、tp1.cltj から始まるパラメタと TP1/Client/J 環境定義の対応については、取扱説明書「HMP-PCTO テクニカルガイド」の TP1/Client/J の設定に関する説明を参照してください。

(1)　Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

パラメタの値に、Integer 型、Long 型で 7 桁以上の数値を指定すると、Helm の仕様によって、指数表示に変換されます。このため、7 桁以上の値を指定する場合は、ダブルクォーテーションで囲んでください。

表 8-13　TP1-Bridge の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	TP1-Bridge の Namespace 名を設定してください。	String	－	my-namespace

パラメタ	説明	型	設定範囲	デフォルト値
image	「3.4.4　TP1-Bridge コンテナイメージの作成（通常版かつ TP1-Bridge 限定）」で作成した TP1-Bridge のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。

imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。
	次のように設定します。
	`imagePullSecrets:`
	`- name: regcred`
	資格情報が不要な場合は、設定不要です。
		List<Object>	−	[]
				（半角左角括弧と半角右角括弧です）

imagePullPolicy	TP1-Bridge のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent

resources	TP1-Bridge のコンテナのリソースを設定します。
	コンテナに割り当てるリソースを明示的に指定した場合に設定してください。
	次の形式で requests や limits を設定します。
	`resources:`
	`  limits:`
	`  cpu: 2`
	`  memory: 1Gi`
	`  requests:`
	`  cpu: 1`
	`  memory: 512Mi`
		Object	−	{}
				（波括弧です）

dependencyCheck.image	「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10

パラメタ	説明	型	設定範囲	デフォルト値
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.targets	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのターゲット Kubernetes オブジェクト群を設定してください。 設定方法の詳細は、「6.5.1　デプロイ依存関係チェックスクリプト(dependency_check.sh)」を参照してください。	List<String>	−	• - my-namespace/statefulsets/mediator • - my-namespace/statefulsets/eads-server
filebeat.image	Filebeat のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。
filebeat.imagePullPolicy	Filebeat のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
filebeat.output.logstash.namespace	Logstash アプリケーションをデプロイする Namespace を設定してください。	String	−	なし 必ず設定してください。
filebeat.livenessProbe.initialDelaySeconds	Filebeat コンテナの Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	10
filebeat.livenessProbe.periodSeconds	Filebeat コンテナの Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
filebeat.livenessProbe.failureThreshold	Filebeat コンテナの Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
filebeat.livenessProbe.timeoutSeconds	Filebeat コンテナの Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
filebeat.readinessProbe.initialDelaySeconds	Filebeat コンテナの Readiness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	10
filebeat.readinessProbe.periodSeconds	Filebeat コンテナの Readiness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10

パラメタ	説明	型	設定範囲	デフォルト値
filebeat.readinessProbe.failureThreshold	Filebeat コンテナの Readiness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
filebeat.readinessProbe.timeoutSeconds	Filebeat コンテナの Readiness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	180
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	30
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	10
livenessProbe.healthCheck.connectTimeoutSeconds	Liveness Probe で実行するトランザクションヘルスチェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
livenessProbe.healthCheck.readTimeoutSeconds	Liveness Probe で実行するトランザクションヘルスチェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
postStart.healthCheck.retryCount	postStart フックで実行するトランザクションヘルスチェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
postStart.healthCheck.retryIntervalSeconds	postStart フックで実行するトランザクションヘルスチェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
postStart.healthCheck.connectTimeoutSeconds	postStart フックで実行するトランザクションヘルスチェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
postStart.healthCheck.readTimeoutSeconds	postStart フックで実行するトランザクションヘルスチェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
preStop.waitTransactions.retryCount	preStop フックで実行する未決着トランザクション待機スクリプトのリトライ回数を設定してください。	Integer	0-1080	30

パラメタ	説明	型	設定範囲	デフォルト値
preStop.waitTransactions.retryIntervalSeconds	preStop フックで実行する未決着トランザクション待機スクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
preStop.waitTransactions.connectTimeoutSeconds	preStop フックで実行する未決着トランザクション待機スクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
preStop.waitTransactions.readTimeoutSeconds	preStop フックで実行する未決着トランザクション待機スクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	330
server.tomcat.threads.max	Spring Boot のワーカースレッドの最大数を設定します。ワーカースレッドは、AP からの OpenTP1 のサービス呼び出し要求、メトリクスの収集などで使用します。 トランザクションの同時実行数＋ 3 以上の値を設定してください。 設定値を超えるリクエストを受信した場合は、ワーカースレッドの空きを待つため、AP から TP1-Bridge へのリクエストのレスポンスに時間が掛かることがあります。また、コネクションの確立に失敗して、送信時に例外が発生することがあります。	Integer	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。
server.tomcat.maxSwallowSize	Spring Boot が受信するリクエストボディの最大サイズを設定します。AP が TP1-Bridge に送信するリクエストボディの最大サイズ以上の値をバイト単位で設定してください。	String※2	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。
converter.host	TP1-Converter が動作する uCosminexus Application Server の NIO HTTP サーバのホスト名、または IP アドレスを設定してください。	String	－	なし 必ず設定してください。
converter.port	TP1-Converter が動作する uCosminexus Application Server の NIO HTTP サーバのポート番号を設定します。 uCosminexus Application Server の usrconf.properties の webserver.connector.nio_http.po	Integer	1-65535	8008

パラメタ	説明	型	設定範囲	デフォルト値
	rt の設定値と同じ値を設定してください。			
converter.connectTime	TP1-Converter が動作する uCosminexus Application Server の NIO HTTP サーバとのコネクション確立待ち時間を秒単位で設定してください。	Integer	1-60	10
converter.watchTime	TP1-Converter が動作する uCosminexus Application Server の NIO HTTP サーバからの応答待ちタイムアウト時間を秒単位で設定してください。 0 を設定した場合は、タイムアウトしないで無限に応答を待ちます。	Integer	0-65535	10
tp1.cltj.dchost	呼び出し先の OpenTP1 のホスト名を設定してください。	String	−	なし 必ず設定してください。
tp1.cltj.dcrapport	呼び出し先の OpenTP1 の rap リスナーのポート番号を設定してください。	Integer	5001-65535	なし 必ず設定してください。
tp1.cltj.dcwatchtim	呼び出し先の OpenTP1 に対しての要求を開始してから応答を受信するまでの待ち時間の最大値を秒単位で設定してください。 0 を設定した場合は、OpenTP1 からの応答の受信を無限に待ちます。	Integer	0-65535	180
tp1.cltj.dcwatchtiminherit	最大応答待ち時間を呼び出し先の OpenTP1 の rap サーバ側に引き継ぐかどうかを設定してください。 Y：最大応答待ち時間を引き継ぎます。 N：最大応答待ち時間を引き継ぎません。	String	Y|N	N
tp1.cltj.dcwatchtimrpcinherit	最大応答待ち時間を呼び出し先の OpenTP1 に引き継ぐかどうかを設定してください。 Y：最大応答待ち時間を引き継ぎます。 N：最大応答待ち時間を引き継ぎません。	String	Y|N	N
tp1.cltj.dccltdelay	呼び出し先の OpenTP1 の rap サーバ側の監視を早く終了させる場合に、その時間を秒単位で設定してください。	Integer	0-65535	0

パラメタ	説明	型	設定範囲	デフォルト値
tp1.cltj.dcuaptrace	UAP トレースを取得するかどうかを設定してください。 Y：UAP トレースを取得します。 N：UAP トレースを取得しません。	String	Y|N	N
tp1.cltj.dcuaptracesize	UAP トレースファイルのサイズをバイト単位で設定してください。	Integer	4096-1048576	4096
tp1.cltj.dcdatatrace	データトレースを取得するかどうかを設定してください。 Y：データトレースを取得します。 N：データトレースを取得しません。	String	Y|N	N
tp1.cltj.dcdatatracesize	データトレースファイルのサイズをバイト単位で設定してください。	Integer	4096-1048576	4096
tp1.cltj.dcdatatracemaxsize	データトレース 1 件のデータ長の最大値をバイト単位で設定してください。	Integer	16-1048576	128
tp1.cltj.dcerrtrace	エラートレースを取得するかどうかを設定してください。 Y：エラートレースを取得します。 N：エラートレースを取得しません。	String	Y|N	N
tp1.cltj.dcerrtracesize	エラートレースファイルのサイズをバイト単位で設定してください。	Integer	4096-1048576	4096
tp1.cltj.dcmethodtrace	メソッドトレースを取得するかどうかを設定してください。 Y：メソッドトレースを取得します。 N：メソッドトレースを取得しません。	String	Y|N	N
tp1.cltj.dcmethodtracesize	メソッドトレースファイルのサイズをバイト単位で設定してください。	Integer	4096-1048576	4096
tp1.cltj.dccltrpcmaxmsgsize	RPC で受け渡しできる電文の最大長をメガバイト単位で設定してください。	Integer	1-8	1
tp1.cltj.dccltsendbuffsize	コネクションごとに確保される TCP/IP 送信バッファのサイズをバイト単位で設定してください。 省略した場合は、JavaVM の送信バッファサイズが適用されます。	Integer	8192-2147483647	省略した場合は、JavaVM の送信バッファサイズを適用します。
tp1.cltj.dccltrecvbuffsize	コネクションごとに確保される TCP/IP 受信バッファのサイズをバイト単位で設定してください。 省略した場合は、JavaVM の受信バッファサイズが適用されます。	Integer	8192-2147483647	省略した場合は、JavaVM の受信バッファサイズを適用します。

パラメタ	説明	型	設定範囲	デフォルト値
externalConsensus.intervalInMillis	Mediator などに障害が発生し、補助合意によってトランザクションの分散合意を行う場合の、補助合意を試行する間隔をミリ秒単位で設定してください。 この間隔が短ければ、Mediator などに障害が発生した場合のトランザクションの遅延を小さくできます。ただし、この間隔を短くし過ぎると、通信遅延を障害と誤認し不要な補助合意が発生します。そのため、通信遅延を考慮した値を設定してください。 この設定値に 1000 ミリ秒以上を設定した場合は 1000 の単位に切り上げた値とします。 なお、この値は初回だけ、最大でこの設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は、最大 1000 ミリ秒です。	Integer	100-10000	1000
externalConsensus.eads.cache	補助合意に使用する EADS クラスタに対して、eztool createcache コマンドで作成したキャッシュ名を設定してください。	String	32 バイト以内の半角英数字（0-9，A-Z，および a-z）	hmppctoCache
tp1.xaServerName	TP1-Bridge 起動時に未決着トランザクションが OpenTP1 に残っているか確認する際、OpenTP1 から取得した未決着トランザクションが TP1-Bridge 自身の実行したトランザクションか判断するために使用する識別子です。 同じ OpenTP1 に接続する TP1-Bridge が複数ある場合、ユニークな値を設定してください。	String	32 バイト以内の文字列	なし 必ず設定してください。
tp1.pool.allocationWaitTimeInMillis	TP1-Bridge が OpenTP1 にアクセスするためのコネクションをコネクションプールから取得する際に、コネクションプールに空きがなかった場合の空き待ち時間をミリ秒単位で設定します。 0 を設定した場合は、コネクションプールの空き待ちをしません。 コネクションプールの空きがなかった場合、このパラメタに設定した時間だけ、コネクションプールの空き待ちをします。空き待ちをしない場	Long	0-65535000	61000

パラメタ	説明	型	設定範囲	デフォルト値
	合、または空き待ち時間内にコネクションプールの空きができなかった場合はエラーとなり、TP1-Bridge は、呼び出し元に http ステータス 502 を応答します。			

tp1.pod.serviceName	TP1-Bridge を構成する Pod を定義した Kubernetes の Service の Service 名とネームスペース名を「.」でつないだ形式で設定します。
	Service 名とネームスペース名には、英小文字（a～z）、数字（0～9）、およびダッシュ（-）を使用できます。ただし、Service 名およびネームスペース名の先頭と終端に使用できるのは英小文字（a～z）、または数字（0～9）です。
	なお、Service 名には、「(15) Kubernetes アプリケーションのデプロイ」で helm コマンドに指定するリリース名を指定してください。
	リリース名が tp1-bridge1 の場合の Service 名の設定例：
	`tp1-bridge1`
	このパラメタには、Mediator からホスト名として認識できる値を設定する必要があります。
	設定例：
	`name.namespace`
		String	3 バイト～41 バイトの文字列	なし
				必ず設定してください。

tp1.mediator.port	TP1-Bridge が Mediator から gRPC 通信を受信するためのポート番号を設定します。	Integer	1024-65535	20552
tp1.pool.maxPoolSize	TP1-Bridge がコネクションプールにプールする OpenTP1 とのコネクションの数を設定します。 TP1-Bridge 起動時にこのパラメタに設定した数のコネクションがプールされます。 TP1-Bridge が使用するコネクションの数がこのパラメタに設定した値を超えた場合、コネクションプールの空き待ちになります。 OpenTP1 の rap リスナーの同時接続する最大クライアント数以下、および rap サーバ数以下で、かつ同時	Integer	1-1024	なし 必ず設定してください。

パラメタ	説明	型	設定範囲	デフォルト値
	実行するトランザクション数以上の値を設定してください。			
tp1.transactionTermination.timeoutInMillis	トランザクション決着監視タイマのタイマ監視時間をミリ秒単位で設定します。 OpenTP1 の SPP によるリソースの排他ロックが解除できない状態となることを許容できる時間をタイマ監視します。このパラメタの値に設定した時間を超えてもトランザクションが決着しなかった場合、OpenTP1 の SPP によるリソースの排他ロック解除のためにトランザクションを決着に向かわせます。 AP がユーザにレスポンスを返すまでの許容時間、またはトランザクションブランチの処理時間の業務要件に応じた値を設定してください。 なお、同一トランザクションのすべてのサービスの業務要件に応じた値の合計時間で監視時間を設定する必要があります。 このタイマ値は初回だけ、最大でこの設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は、最大 1000 ミリ秒です。また、この設定値に 1000 ミリ秒以上を設定した場合は、1000 の単位に切り上げたタイマ値とします。	Long	100-6000000	60000
tp1.transactionTermination.consensus.replayIntervalInMillis	トランザクション分散合意リプレイインターバルタイマのタイマ監視時間をミリ秒単位で設定します。 このパラメタの値は次の処理のために設定します。この値に設定した時間を超えてもトランザクションが決着しなかった場合、トランザクション決着処理を再実行（リプレイ）し、OpenTP1 の SPP によるリソースの排他ロック解除のためにトランザクションを決着に向かわせます。 • トランザクションをコミットまたはロールバックするためにトランザクション決着処理をする • トランザクション決着監視タイマのタイムアウトによるトランザクション決着処理をする	Long	100-6000000	1000

パラメタ	説明	型	設定範囲	デフォルト値
	• TP1-Bridge ダウンからの再起動時にトランザクション決着処理をする このタイマを短く設定することで、TP1-Bridge 内で合意が滞ったトランザクションを速やかに決着できます。ただし、障害が長引いた場合にリトライの回数が多くなり、トランザクション決着のための通信に負荷が掛かります。そのため、通信負荷を考慮した値を設定してください。 このタイマ値は初回だけ、最大でこの設定値分の正の誤差を持ちます。ただし、この設定値が 1000 ミリ秒以上の場合の誤差は、最大 1000 ミリ秒です。また、この設定値に 1000 ミリ秒以上を設定した場合は 1000 の単位に切り上げたタイマ値とします。			
tp1.participantProcessing.threadPoolSize	TP1-Bridge がトランザクション処理（投票処理、分散合意処理、およびトランザクション完了通知処理）で使用するスレッド数を設定してください。 スレッドが不足した場合は、キューイングしてスレッドが空くのを待ってから処理を実行します。 スレッドの空き待ちを発生しないようにするための理論上必要なスレッド数は、「トランザクションに参加する Mediator 数×同時実行するトランザクション数」です。 ただし、処理の実行が重ならないかぎりは、理論上必要なスレッド数は必要ありません。通常は、デフォルト値を使用してください。	Integer	1-1000	12
tp1.participantProcessing.queueSize	TP1-Bridge がトランザクション処理（投票処理、分散合意処理、およびトランザクション完了通知処理）に使用するスレッドの割り当て待ちをするキューのサイズを設定します。 理論上必要な設定値は、次のとおりです。 {m ＋（p－1）×m×m} ×m×t m：トランザクションに参加する Mediator 数	Integer	1-2147483647	25

パラメタ	説明	型	設定範囲	デフォルト値
	p：トランザクションに参加する Participant および TP1-Bridge の数 t：同時実行するトランザクション数			
mediatorProcessManagement.threadPoolSize	SQL-Participant の mediator-process-management.thread-pool-size と同じです。 「(2)　jp.co.Hitachi.soft.hmppctoのパラメタ」を参照してください。	Integer	1-32767	30
mediatorCommunication.threadPoolSize	SQL-Participant の mediator-communication.thread-pool-size と同じです。 「(2)　jp.co.Hitachi.soft.hmppctoのパラメタ」を参照してください。	Integer	1-1000	30
mediatorCommunication.queueSize	SQL-Participant の mediator-communication.queue-size と同じです。 「(2)　jp.co.Hitachi.soft.hmppctoのパラメタ」を参照してください。	Integer	1-2147483647	25
http.port	TP1-Bridge の http ポート番号を設定してください。 動的ポートなどのほかのポートと重ならない値を設定してください。 なお、この設定値は、トランザクションヘルスチェックスクリプトおよび未決着トランザクション待機スクリプトのコマンド引数の http ポート番号、AP からの OpenTP1 のサービス呼び出し要求の受信にも使用します。	Integer	1024-65535	20554
grpc.port	TP1-Bridge の gRPC を受信するポート番号を設定してください。 tp1.mediator.port と同じ値を設定してください。なお、動的ポートなどのほかのポートと重ならない値を設定してください。	Integer	1024-65535	20552
grpc.useTls	暗号化通信をするかどうかを設定してください。 true：暗号化通信をします。 false：暗号化通信をします。	Boolean	true|false	false
grpc.tls.secretName	「(3)　TP1-Bridge、Mediator の gRPC 通信機能の暗号化通信用シークレットの作成」で作成した、TP1-Bridge の gRPC 通信機能の暗号化	String	—	なし grpc.useTls に true を設定した場合は、必ず設定してください。

パラメタ	説明	型	設定範囲	デフォルト値
	通信用シークレットの Kubernetes オブジェクト名を設定してください。			
grpc.tls.clientAuth.enabled	クライアント認証をするかどうかを設定してください。 true：クライアント認証をします。 false：クライアント認証をしません。 このパラメタに true を設定する場合は、grpc.useTls にも true を必ず設定してください。	Boolean	true|false	false
grpc.tls.session.cacheSize	SSL セッション情報のキャッシュ数を設定してください。 0 を設定すると gRPC のデフォルト値が使用されます。	Integer	0-2147483647	0
grpc.tls.session.timeout	キャッシュした SSL セッション情報の保持時間を、秒単位で設定してください。 0 を設定すると gRPC のデフォルト値が使用されます。	Integer	0-65535	0
grpc.keepalive.interval	PING 要求電文の送信間隔を、秒単位で設定してください。 0 を設定した場合は、PING 要求電文を送信しません。また、1～9 を設定した場合は、10 秒となります。 なお、ネットワーク機器の無通信監視タイマなどによる、TCP コネクションの切断回避が目的の場合は、無通信監視タイマ値より小さい値を設定してください。	Integer	0-100000	0
grpc.keepalive.timeout	PING 応答電文の応答待ち時間を、秒単位で設定してください。 PING 要求電文の送信後、このタイマの設定値の時間以内に PING 応答電文を受信できなかった場合は、TCP コネクションが切断されます。	Integer	1-65535	60
grpc.watchtime	Mediator への要求電文に対する応答電文の待ち時間を、秒単位で設定してください。 0 を設定した場合は、タイマ監視をしないで無限に応答を待ちます。 Participant から Mediator に要求電文を送信すると、Mediator は、すぐに要求を受け付けた旨の応答電文を送信します。そのため、応答電	Integer	0-65535	5

パラメタ	説明	型	設定範囲	デフォルト値
	文の待ち時間は gRPC 通信の送受信時間として許容できる待ち時間となります。			
mediator.name	Orchestrator の mediator.name と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			なし 必ず設定してください。
mediator.nameSuffix	Orchestrator の mediator.name-suffix と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			なし 必ず設定してください。
mediator.port	Orchestrator の mediator.port と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			20551
tracing.instanceId	共通の tracing.instance-id と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.url	共通の tracing.url と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.traceUse	共通の tracing.trace-use と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.queueSize	共通の tracing.queue-size と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.delayTime	共通の tracing.delay-time と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.batchSize	共通の tracing.batch-size と同じです。 「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
timezone	TP1-Bridge コンテナのタイムゾーンを設定してください。ユーザ責務の Kubernetes アプリケーションと同じタイムゾーンに設定することを推奨します。 このパラメタに設定した値は環境変数 TZ ユーザに設定されます。 なお、このパラメタに設定した値がタイムゾーンとして不正な文字列である場合は、TP1-Bridge コンテナのタイムゾーンは、協定世界時（UTC）となります。	String	−	Asia/Tokyo
java.options	TP1-Bridge の Java プロセス起動時に Java VM に渡すオプションを設定してください。複数のオプションを設定する場合は、半角スペースで区切ってください。 このパラメタには、「-XX:+UseG1GC」オプションを必ず含めてください。「-	String	−	-XX:+UseG1GC

パラメタ	説明	型	設定範囲	デフォルト値
	XX:+UseG1GC」オプションを含めなかった場合、「-XX:+UseG1GC」オプションを含めた場合と比較して、TP1-Bridgeの実行性能およびメモリ性能が低下することがあります。			
ucars.snapshots.claimName	「3.6.10　uCosminexus Application Runtime with Java for Spring Boot スナップショットログの永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの作成（通常版限定）」で作成する uCosminexus Application Runtime with Java for Spring Boot スナップショットログの PersistentVolumeClaim リソースの名称（.metadata.name フィールドに指定した値）を設定してください。	String	－	必ず設定してください。
eads.client.properties	EADS クライアントのプロパティを設定します。パラメタには、EADS クライアントのパラメタを設定してください。※3 詳細については、「(2)　EADS クライアントのパラメタ」を参照してください。	String	－	• eads.client.connect.servers=sv1,sv2,sv3 • eads.client.sv1.address=eads-service-eads-service-0.my-namespace • eads.client.sv1.port=24600 • eads.client.sv2.address=eads-service-eads-service-1.my-namespace • eads.client.sv2.port=24600 • eads.client.sv3.address=eads-service-eads-service-2.my-namespace • eads.client.sv3.port=24600 • eads.client.logger.dir=eads-log

パラメタ	説明	型	設定範囲	デフォルト値
				• eads.client.connectionPool.poolsize=1

（凡例）

－：該当しません。

注※1

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数、および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

注※2

Helm の仕様によって、7 桁以上の Integer、Long 型の入力は指数形式（例：「1e+06」）に自動で変換されるため、String 型の仕様とします。設定する際は設定値をダブルクォーテーションで囲んでください。

注※3

次に示すように「properties」の行に「|」を記述して、複数行の文字列を記述してください。

```
eads:
  client:
    properties: |
      eads.client.connect.servers=sv1,sv2,sv3
      eads.client.sv1.address=eads-service-eads-service-0.my-namespace
      eads.client.sv1.port=24600
      eads.client.sv2.address=eads-service-eads-service-1.my-namespace
      eads.client.sv2.port=24600
      eads.client.sv3.address=eads-service-eads-service-2.my-namespace
      eads.client.sv3.port=24600
      eads.client.logger.dir=eads-log
      eads.client.connectionPool.poolsize=1
```

8.3.3 EADS（通常版限定）

EADS を HMP-PCTO で動作させる際に、指定するパラメタを説明します。

EADS のパラメタの設定方法は、次の 2 とおりに分けられます。

- EADS サーバ

  EADS サーバチャート用 values.yaml ファイルを編集します。
- EADS クライアント

  Participant の Kubernetes マニフェストを編集します。

なお、EADS のデフォルト値のままとする設定は記載されていません。

プロパティの仕様や指定方法については、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照してください。

マニュアル「Elastic Application Data Store ユーザーズガイド」に記載されている見積もり式で使用するパラメタのうち、EADS パラメタに依存しないパラメタは次のとおりです。

表 8-14　EADS パラメタに依存しないパラメタ

パラメタ	値	説明
キャッシュ数	1	ディスクキャッシュを 1 個だけ使用します。 さらにそのキャッシュではデータ自動削除機能（FIFO モード）を有効にします。 詳細は「(f)　eads_cache.<キャッシュ名>.properties（キャッシュ定義ファイル)」を参照してください。
EADS サーバに格納される key の最大サイズ（単位：バイト）	164	最大 160 文字（ASCII コード）で構成された文字列です。 マニュアル「Elastic Application Data Store ユーザーズガイド」から、160+4 で見積もっています。
EADS サーバに格納される value の最大サイズ（単位：バイト）	17	最大 8 文字（ASCII コード）で構成された Java の java.lang.String 型インスタンスであり、java.io.ObjectOutputStream クラスで直列化したあとのバイト数は最大 15 バイトになります。 マニュアル「Elastic Application Data Store ユーザーズガイド」から、15+2 で見積もっています。
最大同時更新データ数	1	EADS クライアントの API である putAll および getAll は使用していません。
ノンブロッキング通信制御	不採用	ノンブロッキング通信制御は使用していません。
グループ名の数	0	グループは使用していません。
補助合意結果保持件数（クラスタのディスクキャッシュに格納するデータ件数）	説明欄 参照	補助合意結果は古い結果から自動削除しますが、トランザクション処理が完了する前に補助合意結果が削除されると分散合意の一貫性が保てなくなるため、トランザクション処理が完了する前に補助合意結果が削除されないよう十分な件数を指定してください。一方で件数が大き過ぎるとディスク使用量が増えるためシステム要件に合わせて指定してください。 具体的には下記計算式で見積もってください。 TPS（件/秒）×MAX※1(最大トランザクション実行時間（秒）＋許容する障害時のトランザクション実行遅延時間（秒）, 1)×安全係数※2
1 レンジ当たりのデータ件数	説明欄 参照	補助合意結果保持件数×余裕値係数※3÷EADS サーバの台数
1EADS サーバへのデータの更新操作のスループット（単位：件/秒）	説明欄 参照	TPS（件/秒）÷EADS サーバの台数
1EADS サーバの 1 時間当たりの更新回数（単位：件/時）	説明欄 参照	1EADS サーバへのデータの更新操作のスループット（件/秒）×3600（秒/時）
コンパクションの実施間隔（単位：秒）	600	左記の値(600)はデフォルト値です。 コンパクションの実施間隔を変更したい場合は「(18)　EADS のセットアップ（通常版限定)」で、eztool compaction コマンドを定期的

パラメタ	値	説明
		に実行する際の eads-command-cronjob-chart.tgz で指定する schedule フィールドの値を変更してください。 マニュアル「Elastic Application Data Store ユーザーズガイド」の見積もり式の「コンパクションの実施間隔（単位：時)」には、この値を 3600（秒/時）で割った値を設定してください。

注※1

例えば MAX(A, B)であった場合、A と B のどちらか大きい方の値を返します。

注※2

安全係数は 2 以上を推奨します。

注※3

余裕値係数の見積もり方法の詳細は、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照してください。

(1) EADS サーバのパラメタ

設定が必要な EADS サーバのプロパティファイルのパラメタ値について説明します。

(a) EADS サーバチャート用 values.yaml ファイル

EADS サーバ用のパラメタ指定値は、EADS サーバチャート用 values.yaml ファイルで設定します。values.yaml のサンプルを次に示します。

values.yaml ファイルでの設定は、EADS サーバのデプロイ時にプロパティファイルに展開されます。

EADS サーバチャートで使用する values.yaml の例

```
release:
  service: "eads-service"
  server: "eads-server"
namespace: " my-namespace"
serverImage:
  repository: "eads-server-image"
  tag: ""
  imageOverride: ""
  pullSecrets: []
  pullPolicy: "IfNotPresent"
commandImage:
  repository: "eads-command-image"
  tag: ""
  imageOverride: ""
  pullSecrets: []
  pullPolicy: "IfNotPresent"
cluster:
  servers: 3
  replicationFactor: 2
caches:
- name: "hmppctoCache"
  type: "Disk"
  pvc:
    capacity: "258Mi"
```

```
    storageClassName: ""
  properties:
  - eads.cache.disk.filenum=8
  - eads.cache.disk.filesize=16
  - eads.cache.eviction.policy=FIFO
  - eads.cache.eviction.keyCount=1024
service:
  server: 24600
  operation: 24620
  heartbeat: 24630
  failureDetector: 24631
  transfer: 24632
  replication: 24633
logs:
  capacity: "1Gi"
  storageClassName: ""
store:
  capacity: "500Mi"
  storageClassName: ""
resources:
  limits:
    cpu: "2"
  requests:
    memory: "1664Mi"
properties:
  server:
  - eads.server.maxConnections=60
  - eads.server.cache.maxExecuteThreads=60
  - eads.java.nio.maxCachedBufferSize=8192
  - eads.logger.message.console.enable=true
  - eads.server.connection.keepAlive.timeout=70
  - eads.failureDetector.heartbeat.timeout=10000
  - eads.failureDetector.connection.timeout=2500
  - eads.failureDetector.read.timeout=2500
  command: []
  shared:
  - eads.java.heapsize=768
  - eads.java.external.heapsize=7
  - eads.replication.external.heapsize=6
  - eads.cache.key.maxsize=164
  - eads.cache.keyCount=1025
  cluster: []
  functions:
    eads: []
volumes: []
index: 0
args: "status"
schedule: "*/10 * * * *"
```

(b) eads_server.properties（サーバ定義ファイル）

eads_server.properties のパラメタについて説明します。

values.yaml の.properties.server.～で設定します。

表 8-15　eads_server.properties のパラメタ

パラメタ	説明
eads.server.maxConnections	補助合意の時に EADS サーバに同時にアクセスする最大同時接続数を指定してください。EADS サーバに対してこの値を超過する接続があった場合、その接続は拒否されます。 推奨値は、トランザクションに参加する Participant 数×eads.client.connectionPool.poolsize×2（安全係数※）となります。
eads.server.cache.maxExecuteThreads	eads.server.maxConnections に指定した値をこのプロパティにも指定してください。
eads.java.nio.maxCachedBufferSize	8192 を指定してください。
eads.logger.message.console.enable	EADS サーバによって出力されるメッセージログの、標準出力への出力を有効にするか、無効にするかを指定します。HMP-PCTO のトラブルシュート機能で EADS サーバのログを監視する場合は、true を指定してください。
eads.server.connection.keepAlive.timeout	SQL-Participant、TCC-Participant が再起動した場合、EADS サーバが検知してからコネクションを切断するまでの時間を指定します。 推奨値は、70 秒です。
eads.failureDetector.heartbeat.timeout	ハートビートのタイムアウト時間をミリ秒単位で指定します。 10000 を指定してください。ただし、実行環境に影響を受けるため、通信遅延が原因でタイムアウトが頻繁に発生する場合は、長めの値に調整してください。 このパラメタの指定値を変更すると、次のパラメタに影響します。 • eads.failureDetector.connection.timeout • eads.failureDetector.read.timeout
eads.failureDetector.connection.timeout	生存確認の接続タイムアウト時間をミリ秒単位で指定します。 2500 を指定してください。なお、ハートビートのタイムアウト時間（eads.failureDetector.heartbeat.timeout）を変更した場合は、指定値を見直してください。 このパラメタの指定値は、ハートビートのタイムアウト時間の 1/4 を目安としてください。
eads.failureDetector.read.timeout	生存確認の受信タイムアウト時間をミリ秒で指定します。 2500 を指定してください。なお、ハートビートのタイムアウト時間（eads.failureDetector.heartbeat.timeout）を変更した場合は、指定値を見直してください。 このパラメタの指定値は、ハートビートのタイムアウト時間の 1/4 を目安としてください。

注※

Participant が再起動すると、一時的に EADS サーバに同時にアクセスする接続数が倍になる場合があります。そのための安全係数です。

Participant 再起動から、eads.server.connection.keepAlive.time 経過後にコネクションが切断されます。
eads.server.connection.keepAlive.time 間に 2 回以上 Participant 再起動を考慮する場合は、安全係数を Participant 再起動回数+1 としてください。

(c) eads_command.properties（コマンド定義ファイル）

このファイルでは、設定が必要な EADS サーバのプロパティファイルのパラメタ値はありません。

(d) eads_shared.properties（共通設定ファイル）

eads_shared.properties のパラメタについて説明します。

values.yaml の.properties.shared.～で設定します。

表 8-16 eads_shared.properties のパラメタ

パラメタ	説明
eads.replication.factor （values.yaml の「.properties.cluster.replicationFactor」です）	EADS の多重度を指定してください。 多重度は EADS サーバの障害許容台数+1 の値となります。 最低は 2 です。 2 の場合、EADS サーバが 1 台故障しても、補助合意の継続が保証されます。
eads.replication.external.heapsize	見積もり方法の詳細については、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照ください。 「表 8-14 EADS パラメタに依存しないパラメタ」に記載したパラメタで見積もった値を指定してください。
eads.java.heapsize	見積もり方法の詳細については、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照ください。 「表 8-14 EADS パラメタに依存しないパラメタ」に記載したパラメタで見積もった値を指定してください。
eads.java.external.heapsize	見積もり方法の詳細については、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照ください。 「表 8-14 EADS パラメタに依存しないパラメタ」に記載したパラメタで見積もった値を指定してください。
eads.cache.key.maxsize	「表 8-14 EADS パラメタに依存しないパラメタ」の「EADS サーバに格納される key の最大サイズ」を指定してください。
eads.cache.keyCount	見積もり方法の詳細については、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照ください。 「表 8-14 EADS パラメタに依存しないパラメタ」に記載したパラメタで見積もった値を指定してください。

(e) eads_cluster.properties（クラスタ定義ファイル）

eads_cluster.properties のパラメタについて説明します。

values.yaml の.cluster.～で設定します。

表 8-17　eads_cluster.properties のパラメタ

パラメタ	説明
eads.node.＜ EADS サーバ ID ＞.address （values.yaml の.cluster.servers です）	EADS サーバの台数を指定してください。 最低は 3 台です。 台数を増やすと 1 台当たりのメモリ・ディスク容量を減らすことができます。

(f)　eads_cache.＜キャッシュ名＞.properties（キャッシュ定義ファイル）

eads_cache.＜キャッシュ名＞.properties のパラメタについて説明します。

values.yaml の.caches.properties～で設定します。

表 8-18　eads_cache.＜キャッシュ名＞.properties のパラメタ

パラメタ	説明
eads.cache.type （values.yaml の.caches.type～です）	EADS のキャッシュタイプを指定します。 Disk を指定してください。
eads.cache.eviction.policy	EADS に格納された補助合意結果の、自動削除ポリシーを指定します。 FIFO（古いデータから自動削除）を指定してください。
eads.cache.eviction.keyCount	自動削除でデータを削除するための閾値として使用する 1 レンジ当たりのデータ件数を指定します。 「表 8-14　EADS パラメタに依存しないパラメタ」に記載したパラメタの「1 レンジ当たりのデータ件数」を指定してください。
eads.cache.disk.filesize	キャッシュデータファイルのファイルサイズ(MB)を指定します。 見積もり方法の詳細は、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照ください。 「表 8-14　EADS パラメタに依存しないパラメタ」に記載したパラメタで見積もった値を指定してください。
eads.cache.disk.filenum	キャッシュデータファイルのファイル数を指定します。 見積もり方法の詳細は、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照ください。 「表 8-14　EADS パラメタに依存しないパラメタ」に記載したパラメタで見積もった値を指定してください。

(g)　eads_function.properties（ファンクション定義ファイル）

このファイルでは、設定が必要な EADS サーバのプロパティファイルのパラメタ値はありません。

(h)　注意事項

必要なリソース（メモリ使用量やディスク使用量など）については、マニュアル「Elastic Application Data Store ユーザーズガイド」を参照して、見積もってください。

(2) EADS クライアントのパラメタ

(a) EADS クライアント定義ファイル

EADS クライアント定義ファイルおよびパラメタの詳細についてはマニュアル「Elastic Application Data Store ユーザーズガイド」を参照してください。

ここでは、HMP-PCTO で使用する場合に指定が必要なパラメタについて説明します。次の表で示すパラメタ以外は、デフォルトとしてください。

EADS クライアント定義は、ユーザ責務で作成します。

EADS クライアント定義は、Participant の Kubernetes マニフェストで EADS クライアント定義ファイルに設定するように定義します。「2.6.4 SQL-Participant の Kubernetes マニフェストの作成（SQL-Participant 限定)」、または「2.6.5 TCC-Participant の Kubernetes マニフェストの作成（TCC-Participant 限定)」を参照してください。

表 8-19 EADS クライアント定義ファイル

パラメタ	説明
eads.client.connect.servers	任意の名称を指定してください。
eads.client.＜接続先 EADS サーバ＞.address	EADS サーバの DNS 名※を指定してください。
eads.client.＜接続先 EADS サーバ＞.port	EADS サーバのポート番号を指定してください。 EADS サーバのポート番号は EADS サーバチャート用 values.yaml の.service.server に指定した値です。
eads.client.logger.dir	"eads-log"を指定してください。
eads.client.connectionPool.poolsize	1 を指定してください。

注※

＜ EADS サーバの DNS 名＞：＜ Service リリース名＞-eads-service-＜複製インデックス＞.＜ Namespace 名＞

・＜ Service リリース名＞：EADS サーバデプロイ時の Service リリース名

・＜複製インデックス＞：EADS サーバデプロイ時の StatefulSet が複製した EADS サーバ Pod の名前（＜ StatefulSet の名前＞-[0-9]+）の末尾に割り当てられた数値です。0 からシーケンシャルに割り当てられます。

・＜ Namespace 名＞：EADS サーバデプロイ時の Namespace 名

EADS クライアント定義ファイルの指定例を次に示します。

EADS クライアント定義ファイルの指定例

```
eads.client.connect.servers=sv1,sv2,sv3
eads.client.sv1.address=eads-service-eads-service-0.my-namespace
eads.client.sv1.port=24600
eads.client.sv2.address=eads-service-eads-service-1.my-namespace
eads.client.sv2.port=24600
eads.client.sv3.address=eads-service-eads-service-2.my-namespace
eads.client.sv3.port=24600
```

```
eads.client.logger.dir=eads-log
eads.client.connectionPool.poolsize=1
```

8.3.4 Ext-Cons（トライアル版限定）

Ext-Cons のパラメタを説明します。トライアル版 HMP-PCTO を使用する場合に設定します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

表 8-20 Ext-Cons の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Ext-Cons の Namespace 名を設定してください。	String	−	my-namespace
image	Ext-Cons のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。
imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。 次のように指定します。 imagePullSecrets: - name: regcred 資格情報が不要な場合は設定不要です。	List<Object>	−	[] （半角左角括弧と半角右角括弧です）
imagePullPolicy	Ext-Cons のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.image	「3.4.5 Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	−	なし dependencyCheck.enabled に true を設定した場合は、必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30

パラメタ	説明	型	設定範囲	デフォルト値
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.targets	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのターゲット Kubernetes オブジェクト群を設定してください。 指定方法の詳細は「6.5.1　デプロイ依存関係チェックスクリプト（dependency_check.sh）」を参照してください。	List<String>	－	• - my-namespace/deployments/logstash • - my-namespace/deployments/jaeger-query • - my-namespace/deployments/jaeger-collector • - my-namespace/deployments/prometheus • - my-namespace/deployments/metricbeat • - my-namesp

パラメタ	説明	型	設定範囲	デフォルト値
				ace/daemonsets/filebeat
http.port	Ext-Cons の http ポート番号を設定してください。 動的ポートなどほかのポートと重ならないよう設定してください。	Integer	1024-65535	20556
timezone	Ext-Cons コンテナのタイムゾーンを指定します。ユーザ責務の Kubernetes アプリケーションと同じタイムゾーンに設定することを推奨します。 このパラメタに指定した値は環境変数 TZ ユーザに設定します。 このパラメタに指定した値がタイムゾーンとして不正な文字列である場合、Ext-Cons コンテナのタイムゾーンは協定世界時(UTC)となります。	String	－	Asia/Tokyo
java.options	Ext-Cons の Java プロセス起動時に Java VM に渡すオプションを指定します。 半角スペース区切りで複数のオプションを指定できます。 このパラメタには「-XX:+UseG1GC」オプションを必ず含めてください。「-XX:+UseG1GC」オプションを含めなかった場合、「-XX:+UseG1GC」オプションを含めた場合と比較して、Ext-Cons の実行性能およびメモリ性能が低下することがあります。	String	－	-XX:+UseG1GC

（凡例）－：該当しません。

8.3.5 Filebeat

Filebeat のパラメタを説明します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

パラメタの値に、Integer 型、Long 型で 7 桁以上の数値を指定すると、Helm の仕様によって、指数表示に変換されます。このため、7 桁以上の値を指定する場合は、ダブルクォーテーションで囲んでください。

次の表に記載されていないパラメタは設定を変更しないでください。

表 8-21　Filebeat の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Filebeat の Namespace 名を設定してください。	String	－	my-namespace
requests.cpu	コンテナが最低限要求する CPU コア数を設定してください。	String※3	－※1、※3	1000m 必ず設定してください。
requests.memory	コンテナが最低限要求するメモリを設定してください。	String※3	－※1、※3	2Gi 必ず設定してください。
limits.cpu	コンテナで上限の CPU コア数を設定してください。	String※3	－※1、※3	1000m 必ず設定してください。
limits.memory	コンテナで上限のメモリを設定してください。	String※3	－※1、※3	2Gi 必ず設定してください。
image	Filebeat のコンテナイメージを設定してください。	String	－	なし 必ず設定してください。
imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。 次のように指定します。 imagePullSecrets: - name: regcred 資格情報が不要な場合は設定不要です。	List<Object>	－	[] (半角左角括弧と半角右角括弧です)
imagePullPolicy	コンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	10
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
livenessProbe.successThreshold	Liveness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1

パラメタ	説明	型	設定範囲	デフォルト値
readinessProbe.initialDelaySeconds	Readiness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	10
readinessProbe.periodSeconds	Readiness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
readinessProbe.failureThreshold	Readiness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
readinessProbe.timeoutSeconds	Readiness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
readinessProbe.successThreshold	Readiness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	30
nodes.hostnames	HMP-PCTO の pod を割り当てる Kubernetes ノードのホスト名を配列で設定してください。	List<String>	−	[] (半角左角括弧と半角右角括弧です)
hmpPcto.namespaces	HMP-PCTO で使用している Pod の、標準出力の収集対象とする namespace を指定します。 namespace 名をシーケンス（配列）で設定してください。次のように指定します。 ["my-namespace1", "my-namespace2"]	List<String>	−	[] (半角左角括弧と半角右角括弧です)
dependencyCheck.enabled	Dependency-Checker を有効にするかどうかを設定してください。	Boolean	true | false	true
dependencyCheck.image	「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	−	なし dependencyCheck.enabled に true を設定した場合は、必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10

パラメタ	説明	型	設定範囲	デフォルト値
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
filebeat.queue.maxSize	Filebeat の内部キューとして使用する容量の最大値を設定してください。 設定した値は次のパラメタとして有効になります。 • Filebeat の queue.disk.max_size	String※3	－※2、※3	10GB 必ず設定してください。
filebeat.queue.readAhead	Filebeat の内部キューから、Logstash への出力のためにメモリに読み取る必要のあるイベント数を設定してください。 設定した値は次のパラメタとして有効になります。 • Filebeat の queue.disk.read_ahead	String※3	－※2、※3	512 必ず設定してください。
filebeat.output.maxRetries	Logstash への送信に失敗した際に実施する再送の試行回数を設定してください。 設定した値は次のパラメタとして有効になります。 • Filebeat の output.logstash.max_retries	String※3	－※2、※3	3 必ず設定してください。

（凡例）

－：該当しません。

注※1

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

注※2

Filebeat への設定値であるため、最大値、最大文字数および許容される文字パターンは、Filebeat に依存します。詳細は Filebeat ドキュメントを参照してください。

注※3

Helm の仕様によって、7 桁以上の Integer、Long 型の入力は指数形式（例：「1e+06」）に自動で変換されるため、String 型の仕様とします。設定する際は指定値をダブルクォーテーションで囲んでください。

8.3.6 Jaeger-query

Jaeger-query のパラメタを説明します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

パラメタの値に、Integer 型、Long 型で 7 桁以上の数値を指定すると、Helm の仕様によって、指数表示に変換されます。このため、7 桁以上の値を指定する場合は、ダブルクォーテーションで囲んでください。

次の表に記載されていないパラメタは設定を変更しないでください。

表 8-22　Jaeger-query の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Jaeger-query の Namespace 名を設定してください。	String	−	my-namespace
requests.cpu	コンテナが最低限要求する CPU コア数を設定してください。	String※3	−※1、※3	500m 必ず設定してください。
requests.memory	コンテナが最低限要求するメモリを設定してください。	String※3	−※1、※3	1Gi 必ず設定してください。
limits.cpu	コンテナで上限の CPU コア数を設定してください。	String※3	−※1、※3	500m 必ず設定してください。
limits.memory	コンテナで上限のメモリを設定してください。	String※3	−※1、※3	1Gi 必ず設定してください。
image	Jaeger-query のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。
imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。 次のように指定します。 imagePullSecrets: - name: regcred 資格情報が不要な場合は設定不要です。	List<Object>	−	[] (半角左角括弧と半角右角括弧です)
imagePullPolicy	コンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	30
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	15
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	10

パラメタ	説明	型	設定範囲	デフォルト値
livenessProbe.successThreshold	Liveness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
readinessProbe.initialDelaySeconds	Readiness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	30
readinessProbe.periodSeconds	Readiness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	5
readinessProbe.failureThreshold	Readiness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
readinessProbe.timeoutSeconds	Readiness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	4
readinessProbe.successThreshold	Readiness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	30
http.port	Jaeger の Web UI アクセス用ポート番号を設定してください。 動的ポートなど、ほかのポートと重ならないよう設定してください。	Integer	1024-65535	なし 必ず指定してください。
jaegerQuery.maxSpanAge	Elasticsearch から分散トレースを取得する際、検索の範囲とする時間を設定してください。 設定した値は次のパラメタとして有効になります。 • Jaeger-collector の--es.max-span-age	String	－※2	168h0m0s 必ず指定してください。
dependencyCheck.enabled	Dependency-Checker を有効にするかどうかを設定してください。	Boolean	true | false	true
dependencyCheck.image	「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	－	なし dependencyCheck.enabled に true を設定した場合は、必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30

パラメタ	説明	型	設定範囲	デフォルト値
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
tls.enabled	Elasticsearch と暗号化通信をするかどうかを指定します。 true 暗号化通信をします。 false 暗号化通信をしません。 Elasticsearch の tls.enabled を true にした場合は true を指定してください	Boolean	true|false	false
tls.secretName	Elasticsearch の TLS 通信シークレットの名前を設定してください。	String	－	なし tls.enabled に true を設定した場合は、必ず設定してください。
elasticUserSecretName	tls.enabled パラメタに true を設定した場合、Elasticsearch の elastic ユーザアカウントに設定するパスワードが設定されたシークレットの名前を設定してください。 シークレットの data.password にパスワードを設定してください。 この設定が行われていない場合、elastic ユーザアカウントのパスワードは"changeme"となります。	String	－	なし

（凡例）

－：該当しません。

注※1

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

注※2

Jaeger-query への設定値であるため、最大値、最大文字数および許容される文字パターンは、Jaeger-query に依存します。詳細は Jaeger-query ドキュメントを参照してください。

注※3

Helm の仕様によって、7 桁以上の Integer、Long 型の入力は指数形式（例：「1e+06」）に自動で変換されるため、String 型の仕様とします。設定する際は指定値をダブルクォーテーションで囲んでください。

8.3.7 Jaeger-collector

Jaeger-collector のパラメタを説明します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

パラメタの値に、Integer 型、Long 型で 7 桁以上の数値を指定すると、Helm の仕様によって、指数表示に変換されます。このため、7 桁以上の値を指定する場合は、ダブルクォーテーションで囲んでください。

次の表に記載されていないパラメタは設定を変更しないでください。

表 8-23 Jaeger-collector の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Jaeger-collector の Namespace 名を設定してください。	String	−	my-namespace
replicas	起動する Jaeger-collector 数を設定してください。 設定した値は次のパラメタとして有効になります。 • Jaeger-collector の--es.num-replicas	Integer	1 以上	1 必ず設定してください。
requests.cpu	コンテナが最低限要求する CPU コア数を設定してください。	String※3	−※1、※3	1000m 必ず設定してください。
requests.memory	コンテナが最低限要求するメモリを設定してください。	String※3	−※1、※3	500Mi 必ず設定してください。
limits.cpu	コンテナで上限の CPU コア数を設定してください。	String※3	−※1、※3	1000m 必ず設定してください。
limits.memory	コンテナで上限のメモリを設定してください。	String※3	−※1、※3	500Mi 必ず設定してください。
image	Jaeger-collector のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。
imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。 次のように指定します。 imagePullSecrets: - name: regcred	List<Object>	−	[] (半角左角括弧と半角右角括弧です)

パラメタ	説明	型	設定範囲	デフォルト値
	資格情報が不要な場合は設定不要です。			
imagePullPolicy	コンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	300
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
livenessProbe.successThreshold	Liveness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
readinessProbe.initialDelaySeconds	Readiness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	60
readinessProbe.periodSeconds	Readiness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
readinessProbe.failureThreshold	Readiness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
readinessProbe.timeoutSeconds	Readiness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
readinessProbe.successThreshold	Readiness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	3
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	30
collector.queueSize	Jaeger-collector の内部キューのサイズを設定してください。 設定した値は次のパラメタとして有効になります。 • Jaeger-collector の--collector.queue-size パラメタ	String※3	－※2、※3	10000
collector.numWorkers	Jaeger-collector の内部で動作するワーカの数を設定してください。 設定した値は次のパラメタとして有効になります。 • Jaeger-collector の--collector.num-workers	String※3	－※2、※3	50

パラメタ	説明	型	設定範囲	デフォルト値
collector.numShards	Jaeger-collector が Elasticsearch 上に作成するインデックスのシャード数を設定してください。 設定した値は次のパラメタとして有効になります。 • Jaeger-collector の--es.num-shards	String※3	―※2、※3	5
dependencyCheck.enabled	Dependency-Checker を有効にするかどうかを設定してください。	Boolean	true | false	true
dependencyCheck.image	「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	－	なし dependencyCheck.enabled に true を設定した場合は、必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
tls.enabled	Elasticsearch と暗号化通信をするかどうかを指定します。 true 　暗号化通信をします。 false 　暗号化通信をしません。 Elasticsearch の tls.enabled を true にした場合は true を指定してください	Boolean	true|false	false
tls.secretName	Elasticsearch の TLS 通信シークレットの名前を設定してください。	String	－	なし tls.enabled に true を設

パラメタ	説明	型	設定範囲	デフォルト値
				定した場合は、必ず設定してください。
elasticUserSecretName	tls.enabled パラメタに true を設定した場合、Elasticsearch の elastic ユーザアカウントに設定するパスワードが設定されたシークレットの名前を設定してください。 シークレットの data.password にパスワードを設定してください。 この設定が行われていない場合、elastic ユーザアカウントのパスワードは"changeme"となります。	String	－	なし。

（凡例）

－：該当しません。

注※1

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

注※2

Jaeger-collector に対する設定値であるため、最大値、最大文字数および許容される文字パターンについては、Jaeger-collector のドキュメントを参照してください。

注※3

Helm の仕様によって、7 桁以上の Integer、Long 型の入力は指数形式（例：「1e+06」）に自動で変換されるため、String 型の仕様とします。設定する際は指定値をダブルクォーテーションで囲んでください。

8.3.8 Prometheus

Prometheus のパラメタを説明します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

パラメタの値に、Integer 型、Long 型で 7 桁以上の数値を指定すると、Helm の仕様によって、指数表示に変換されます。このため、7 桁以上の値を指定する場合は、ダブルクォーテーションで囲んでください。

次の表に記載されていないパラメタは設定を変更しないでください。

表 8-24　Prometheus の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Prometheus の Namespace 名を設定してください。	String	−	my-namespace
requests.cpu	コンテナが最低限要求する CPU コア数を設定してください。	String※3	−※1、※3	500m 必ず設定してください。
requests.memory	コンテナが最低限要求するメモリを設定してください。	String※3	−※1、※3	500Mi 必ず設定してください。
limits.cpu	コンテナで上限の CPU コア数を設定してください。	String※3	−※1、※3	500m 必ず設定してください。
limits.memory	コンテナで上限のメモリを設定してください。	String※3	−※1、※3	500Mi 必ず設定してください。
image	Prometheus のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。
imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。 次のように指定します。 imagePullSecrets: - name: regcred 資格情報が不要な場合は設定不要です。	List<Object>	−	[] (半角左角括弧と半角右角括弧です)
imagePullPolicy	コンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	30
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	15
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	10
livenessProbe.successThreshold	Liveness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
readinessProbe.initialDelaySeconds	Readiness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	30

パラメタ	説明	型	設定範囲	デフォルト値
readinessProbe.periodSeconds	Readiness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	5
readinessProbe.failureThreshold	Readiness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
readinessProbe.timeoutSeconds	Readiness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	4
readinessProbe.successThreshold	Readiness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	300
http.port	Prometheus の Web UI アクセス用ポート番号を設定してください。 動的ポートなど、ほかのポートと重ならないよう設定してください。	Integer	1024-65535	なし。 必ず指定してください。
scrapeNamespace.orchestrator	Orchestrator の pod の Namespace 名を設定してください。 複数の Namespace 名を指定する場合は、正規表現（RE2 構文）を使用して設定してください。	String	−	なし。 必ず指定してください。
scrapeNamespace.sqlparticipant	SQL-Participant の pod の Namespace 名を設定してください。 複数の Namespace 名を指定する場合は、正規表現（RE2 構文）を使用して設定してください。 SQL-Participant が存在しない場合は、値の設定は不要です。	String	−	なし。
scrapeNamespace.tccparticipant	TCC-Participant の pod の Namespace 名を設定してください。 複数の Namespace 名を指定する場合は、正規表現（RE2 構文）を使用して設定してください。 TCC-Participant が存在しない場合は、値の設定は不要です。	String	−	なし。
scrapeNamespace.tp1bridge	TP1-Bridge の pod の Namespace 名を設定してください。 複数の Namespace 名を指定する場合は、正規表現（RE2 構文）を使用して設定してください。 TP1-Bridge が存在しない場合は、値の設定は不要です。	String	−	なし。
scrapeNamespace.entity	Entity-service の pod の Namespace 名を設定してください。 複数の Namespace 名を指定する場合は、正規表現（RE2 構文）を使用して設定してください。	String	−	なし。 必ず指定してください。

パラメタ	説明	型	設定範囲	デフォルト値
scrapeNamespace.mediator	Mediator の pod の Namespace 名を設定してください。 複数の Namespace 名を指定する場合は、正規表現（RE2 構文）を使用して設定してください。	String	−	なし。 必ず指定してください。
scrapeNamespace. jaegercollector	Jaeger-collector の pod の Namespace 名を設定してください。 複数の Namespace 名を指定する場合は、正規表現（RE2 構文）を使用して設定してください。	String	−	なし。 必ず指定してください。
scrapePort.orchestrator	Orchestrator の pod の http ポート番号を設定してください。 単一のポート番号を指定する場合は、Integer 型で設定してください。 複数のポート番号を指定する場合は、正規表現（RE2 構文）を使用して String 型で設定してください。	Integer/ String	−	なし。 必ず指定してください。
scrapePort.sqlparticipant	SQL-Participant の pod の http ポート番号を設定してください。 単一のポート番号を指定する場合は、Integer 型で設定してください。 複数のポート番号を指定する場合は、正規表現（RE2 構文）を使用して String 型で設定してください。 SQL-Participant が存在しない場合は、値の設定は不要です。	Integer/ String	−	なし。
scrapePort.tccparticipant	TCC-Participant の pod の http ポート番号を設定してください。 単一のポート番号を指定する場合は、Integer 型で設定してください。 複数のポート番号を指定する場合は、正規表現（RE2 構文）を使用して String 型で設定してください。 TCC-Participant が存在しない場合は、値の設定は不要です。	Integer/ String	−	なし。
scrapePort.tp1bridge	TP1-Bridge の pod の http ポート番号を設定してください。 単一のポート番号を指定する場合は、Integer 型で設定してください。 複数のポート番号を指定する場合は、正規表現（RE2 構文）を使用して String 型で設定してください。 TP1-Bridge が存在しない場合は、値の設定は不要です。	Integer/ String	−	なし。

パラメタ	説明	型	設定範囲	デフォルト値
scrapePort.entity	Entity-service の pod の http ポート番号を設定してください。 単一のポート番号を指定する場合は、Integer 型で設定してください。 複数のポート番号を指定する場合は、正規表現（RE2 構文）を使用して String 型で設定してください。	Integer/ String	−	なし。 必ず指定してください。
scrapePort.mediator	Mediator の pod の http ポート番号を設定してください。 単一のポート番号を指定する場合は、Integer 型で設定してください。 複数のポート番号を指定する場合は、正規表現（RE2 構文）を使用して String 型で設定してください。	Integer/ String	−	なし。 必ず指定してください。
prometheus.scrape.interval	Prometheus がメトリクスの取得対象から情報を取得する間隔を設定してください。 設定した値は次のパラメタとして有効になります。 • Prometheus の global.scrape_interval	String	−※2	15s
prometheus.storage.retentionTime	Prometheus 内部で収集したメトリクスを保持する期間を設定してください。 設定した値は次のパラメタとして有効になります。 • Prometheus の--storage.tsdb.retention.time	String	−※2	7d
prometheus.remoteWrite.queue.capacity	Prometheus から Metricbeat に対する送信処理を行うために Prometheus 内部で作成されるキューのサイズを設定してください。 設定した値は次のパラメタとして有効になります。 • Prometheus の remote_write.queue_coinfig.capacity	String※3	−※2、※3	10000
prometheus.remoteWrite.queue.maxShards	Metricbeat に送信するデータの最大サイズを設定してください。 この値は、送信が遅延した場合などに有効となる最大サイズを指定します。 設定した値は次のパラメタとして有効になります。 • Prometheus の remote_write.queue_coinfig.max_shards	String※3	−※2、※3	50
prometheus.remoteWrite.queue.maxSamplesPerSend	Metricbeat に対して一度にまとめて送信するデータのサイズを設定してください。 設定した値は次のパラメタとして有効になります。 • Prometheus の remote_write.queue_coinfig.max_samples_per_send	String※3	−※2、※3	2000

パラメタ	説明	型	設定範囲	デフォルト値
dependencyCheck.enabled	Dependency-Checker を有効にするかどうかを設定してください。	Boolean	true | false	true
dependencyCheck.image	「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	－	なし dependencyCheck.enabled に true を設定した場合は、必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10

（凡例）

－：該当しません。

注※1

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

注※2

Prometheus への設定値であるため、最大値、最大文字数および許容される文字パターンは、Prometheus に依存します。詳細は Prometheus ドキュメントを参照してください。

注※3

Helm の仕様によって、7 桁以上の Integer、Long 型の入力は指数形式（例：「1e+06」）に自動で変換されるため、String 型の仕様とします。設定する際は指定値をダブルクォーテーションで囲んでください。

8.3.9　Metricbeat

Metricbeat のパラメタを説明します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

パラメタの値に、Integer 型、Long 型で 7 桁以上の数値を指定すると、Helm の仕様によって、指数表示に変換されます。このため、7 桁以上の値を指定する場合は、ダブルクォーテーションで囲んでください。

次の表に記載されていないパラメタは設定を変更しないでください。

表 8-25 Metricbeat の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Metricbeat の Namespace 名を設定してください。	String	−	my-namespace
replicas	起動する Metricbeat の数を設定してください。	Integer	1 以上	1 必ず設定してください。
requests.cpu	コンテナが最低限要求する CPU コア数を設定してください。	String※3	−※1、※3	1000m 必ず設定してください。
requests.memory	コンテナが最低限要求するメモリを設定してください。	String※3	−※1、※3	2Gi 必ず設定してください。
limits.cpu	コンテナで上限の CPU コア数を設定してください。	String※3	−※1、※3	1000m 必ず設定してください。
limits.memory	コンテナで上限のメモリを設定してください。	String※3	−※1、※3	2Gi 必ず設定してください。
image	Metricbeat のコンテナイメージを設定してください。	String	−	なし 必ず設定してください。
imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。 次のように指定します。 imagePullSecrets: - name: regcred 資格情報が不要な場合は設定不要です。	List<Object>	−	[] (半角左角括弧と半角右角括弧です)
imagePullPolicy	コンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	10

パラメタ	説明	型	設定範囲	デフォルト値
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
livenessProbe.successThreshold	Liveness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
readinessProbe.initialDelaySeconds	Readiness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	10
readinessProbe.periodSeconds	Readiness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
readinessProbe.failureThreshold	Readiness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
readinessProbe.timeoutSeconds	Readiness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
readinessProbe.successThreshold	Readiness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	30
metricbeat.queue.maxSize	Metricbeat の内部キューとして使用する容量の最大値を設定してください。 設定した値は次のパラメタとして有効になります。 • Metricbeat の queue.disk.max_size	String※3	―※2、※3	10GB 必ず設定してください。
metricbeat.queue.readAhead	Metricbeat の内部キューから、Elasticsearch への出力のためにメモリに読み取る必要のあるイベント数を設定してください。 設定した値は次のパラメタとして有効になります。 • Metricbeat の queue.disk.read_ahead	String※3	―※2、※3	512 必ず設定してください。
metricbeat.output.maxRetries	Elasticsearch への送信に失敗した際に実施する再送の試行回数を設定してください。 設定した値は次のパラメタとして有効になります。 • Metricbeat の output.elasticsearch.max_retries	String※3	―※2、※3	3 必ず設定してください。
dependencyCheck.enabled	Dependency-Checker を有効にするかどうかを設定してください。	Boolean	true | false	true
dependencyCheck.image	「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	―	なし dependencyCheck.en

パラメタ	説明	型	設定範囲	デフォルト値
				abled に true を設定した場合は、必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
tls.enabled	Elasticsearch と暗号化通信をするかどうかを指定します。 true 暗号化通信をします。 false 暗号化通信をしません。 Elasticsearch の tls.enabled を true にした場合は true を指定してください	Boolean	true|false	false
tls.secretName	Elasticsearch の TLS 通信シークレットの名前を設定してください。	String	−	なし tls.enabled に true を設定した場合は、必ず設定してください。
elasticUserSecretName	tls.enabled パラメタに true を設定した場合、Elasticsearch の elastic ユーザアカウントに設定するパスワードが設定されたシークレットの名前を設定してください。 シークレットの data.password にパスワードを設定してください。 この設定が行われていない場合、elastic ユーザアカウントのパスワードは"changeme"となります。	String	−	なし。

（凡例）

－：該当しません。

注※1

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

注※2

Metricbeat への設定値であるため、最大値、最大文字数および許容される文字パターンは、Metricbeat に依存します。詳細は Metricbeat ドキュメントを参照してください。

注※3

Helm の仕様によって、7 桁以上の Integer、Long 型の入力は指数形式（例：「1e+06」）に自動で変換されるため、String 型の仕様とします。設定する際は指定値をダブルクォーテーションで囲んでください。

8.3.10 Elasticsearch

Elasticsearch のパラメタを説明します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

パラメタの値に、Integer 型、Long 型で 7 桁以上の数値を指定すると、Helm の仕様によって、指数表示に変換されます。このため、7 桁以上の値を指定する場合は、ダブルクォーテーションで囲んでください。

次の表に記載されていないパラメタは設定を変更しないでください。

表 8-26 Elasticsearch の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Elasticsearch の Namespace 名を設定してください。	String	－	my-namespace
replicas	起動する Elasticsearch 数を設定してください。	Integer	1 以上	1 必ず設定してください。
image	Elasticsearch のコンテナイメージを設定してください。	String	－	なし 必ず設定してください。
imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。 次のように指定します。 imagePullSecrets: - name: regcred 資格情報が不要な場合は設定不要です。	List<Object>	－	[] （半角左角括弧と半角右角括弧です）

パラメタ	説明	型	設定範囲	デフォルト値
imagePullPolicy	コンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
role	デプロイする Elasticsearch のロールを設定してください。 最小機器構成の場合はデフォルト値をそのまま使用してください。 Elasticsearch をクラスタ構成で構築する場合は、マスターノード用 values.yaml には「master」、データノード用 values.yaml には「data」を設定してください。	String	−	なし
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	300
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
livenessProbe.successThreshold	Liveness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
readinessProbe.initialDelaySeconds	Readiness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	60
readinessProbe.periodSeconds	Readiness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
readinessProbe.failureThreshold	Readiness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
readinessProbe.timeoutSeconds	Readiness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
readinessProbe.successThreshold	Readiness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	3
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	120
affinity	Pod の割り当てノードの制御をします。	Object	−	podAntiAffinity: requiredDuringSchedulingIgnoredDuringExecution:

パラメタ	説明	型	設定範囲	デフォルト値
				- labelSelector: matchExpressions: - key: app.kubernetes.io/name operator: In values: - elasticsearch-master topologyKey: kubernetes.io/hostname クラスタ構成で、データノードとして起動させる場合は、elasticsearch-master を elasticsearch-data に変更してください。
http.port	Elasticsearch REST API 用ポート番号を設定してください。 動的ポートなど、ほかのポートと重ならないよう設定してください。	Integer	1024-65535	なし。 必ず指定してください。
data.storageClassName	「3.6.3　Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で作成する Elasticsearch の Pod に割り当てるリポジトリディレクトリの永続ボリュームのストレージクラス名を設定してください。 クラスタ構成で構築する場合、マスターノード用 values.yaml の場合はマスターノード用のストレージクラス名、データノード用 values.yaml の場合はデータノード用のストレージクラスを設定してください。	String	−	なし。 必ず指定してください。

パラメタ	説明	型	設定範囲	デフォルト値
data.storageSize	「3.6.3 Elasticsearch データディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で作成する Elasticsearch の Pod に割り当てるデータディレクトリの永続ボリュームのサイズを設定してください。	String	−	なし。 必ず指定してください。
repo.storageClassName	「3.6.4 Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で作成する Elasticsearch の Pod に割り当てるリポジトリディレクトリの永続ボリュームのストレージクラス名を設定してください。	String	−	なし。 必ず指定してください。
repo.storageSize	「3.6.4 Elasticsearch リポジトリディレクトリの永続ボリュームおよびストレージクラスの Kubernetes マニフェストの作成」で作成する Elasticsearch の Pod に割り当てるリポジトリディレクトリの永続ボリュームのサイズを設定してください。	String	−	なし。 必ず指定してください。
sysctlVmMaxMapCount	Elasticsearch コンテナのカーネルパラメタ vm.max_map_count の値です。Elasticsearch の Pod がダウンする場合、この値を 262144 より大きくしてください。	Integer	−	262144
requests.cpu	コンテナが最低限要求する CPU コア数を設定してください。	String※2	−※1、※2	2000m 必ず指定してください。
requests.memory	コンテナが最低限要求するメモリを設定してください。	String※2	−※1、※2	4Gi 必ず指定してください。
limits.cpu	コンテナで上限の CPU コア数を設定してください。	String※2	−※1、※2	2000m 必ず指定してください。
limits.memory	コンテナで上限のメモリを設定してください。	String※2	−※1、※2	4Gi 必ず指定してください。
tls.enabled	Elasticsearch と暗号化通信をするかどうかを指定します。 true 暗号化通信をします。 false 暗号化通信をしません。	Boolean	true|false	false
tls.secretName	Elasticsearch の TLS 通信シークレットの名前を設定してください。	String	−	なし

パラメタ	説明	型	設定範囲	デフォルト値
				tls.enabled に true を設定した場合は、必ず設定してください。
elasticUserSecretName	tls.enabled パラメタに true を設定した場合、Elasticsearch の elastic ユーザアカウントに設定するパスワードが設定されたシークレットの名前を設定してください。 シークレットの data.password にパスワードを設定してください。 この設定が行われていない場合、elastic ユーザアカウントのパスワードは"changeme"となります。	String	－	なし。

（凡例）

－：該当しません。

注※1

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

注※2

Helm の仕様によって、7 桁以上の Integer、Long 型の入力は指数形式（例：「1e+06」）に自動で変換されるため、String 型の仕様とします。設定する際は指定値をダブルクォーテーションで囲んでください。

8.3.11 Logstash

Logstash のパラメタを説明します。

(1) Helm チャートのパラメタ

Helm チャートの各パラメタの値は設定範囲に従って設定してください。

パラメタの値に、Integer 型、Long 型で 7 桁以上の数値を指定すると、Helm の仕様によって、指数表示に変換されます。このため、7 桁以上の値を指定する場合は、ダブルクォーテーションで囲んでください。

次の表に記載されていないパラメタは設定を変更しないでください。

表 8-27 Logstash の Helm チャートのパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
namespace	Logstash の Namespace 名を設定してください。	String	－	my-namespace
replicas	起動する Logstash の数を設定してください。	Integer	1 以上	1

パラメタ	説明	型	設定範囲	デフォルト値
				必ず設定してください。
requests.cpu	コンテナが最低限要求する CPU コア数を設定してください。	String※3	―※1、※3	1000m 必ず設定してください。
requests.memory	コンテナが最低限要求するメモリを設定してください。	String※3	―※1、※3	3Gi 必ず設定してください。
limits.cpu	コンテナで上限の CPU コア数を設定してください。	String※3	―※1、※3	1000m 必ず設定してください。
limits.memory	コンテナで上限のメモリを設定してください。	String※3	―※1、※3	3Gi 必ず設定してください。
image	Logstash のコンテナイメージを設定してください。	String	－	なし 必ず設定してください。
imagePullSecrets	コンテナイメージを Pull する際の資格情報を設定してください。 次のように指定します。 imagePullSecrets: - name: regcred 資格情報が不要な場合は設定不要です。	List<Object>	－	[] (半角左角括弧と半角右角括弧です)
imagePullPolicy	コンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
logstash.pipeline.batchSize	Logstash が Input 処理によって収集するイベントの最大数を設定してください。 設定した値は次のパラメタとして有効になります。 • Logstash の pipeline.batch.size	String※3	―※2、※3	125
logstash.queue.event.enabled	Logstash の内部キューで、イベント数によるキューサイズ設定を有効にするかどうかを設定してください。 logstash.queue.event.enabled と logstash.queue.byte.enabled が false に設定された場合、内部キューの最大値に対して無制限が設定され、Logstash は起動します。	Boolean	true | false	true 必ず設定してください。
logstash.queue.event.maxEvents	Logstash の内部キューでの未読イベントの最大数を設定してください。 logstash.queue.event.enabled が true に設定された場合、このパラメタの設定値は有効になります。	String※3	―※2、※3	0（無制限）

パラメタ	説明	型	設定範囲	デフォルト値
	logstash.queue.byte.maxBytes が有効な場合、この設定も有効になります。 設定した値は次のパラメタとして有効になります。 • Logstash の queue.max_events			
logstash.queue.byte.enabled	Logstash の内部キューで、Byte によるキューサイズ設定を有効にするかどうかを設定してください。 logstash.queue.event.enabled と logstash.queue.byte.enabled が false に設定された場合、内部キューの最大値に対して無制限が設定され、Logstash は起動します。	Boolean	true | false	false 必ず設定してください。
logstash.queue.byte.maxBytes	Logstash の内部キューの最大容量を設定してください。 logstash.queue.byte.enabled が true に設定された場合、このパラメタの設定値は有効になります。 logstash.queue.event.maxEvents が有効な場合、この設定は無効となります。 設定した値は次のパラメタとして有効になります。 • Logstash の queue.max_bytes	String※3	―※2、※3	なし
livenessProbe.initialDelaySeconds	Liveness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	300
livenessProbe.periodSeconds	Liveness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
livenessProbe.failureThreshold	Liveness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
livenessProbe.timeoutSeconds	Liveness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
livenessProbe.successThreshold	Liveness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	1
readinessProbe.initialDelaySeconds	Readiness Probe の初回実行待機秒数を設定してください。	Integer	0 以上※1	60
readinessProbe.periodSeconds	Readiness Probe の実行間隔秒数を設定してください。	Integer	1 以上※1	10
readinessProbe.failureThreshold	Readiness Probe の最大失敗回数を設定してください。	Integer	1 以上※1	3
readinessProbe.timeoutSeconds	Readiness Probe のタイムアウト秒数を設定してください。	Integer	1 以上※1	5
readinessProbe.successThreshold	Readiness Probe の最小連続成功数を設定してください。	Integer	1 以上※1	3

パラメタ	説明	型	設定範囲	デフォルト値
terminationGracePeriodSeconds	Pod の安全な停止待機秒数を設定してください。	Integer	0-2147483647	120
dependencyCheck.enabled	Dependency-Checker を有効にするかどうかを設定してください。	Boolean	true | false	true
dependencyCheck.image	「3.4.5 Dependency-Checker コンテナイメージの作成（通常版限定）」で作成した Dependency-Checker のコンテナイメージを設定してください。	String	−	なし dependencyCheck.enabled に true を設定した場合は、必ず設定してください。
dependencyCheck.imagePullPolicy	Dependency-Checker のコンテナイメージを Pull する際のポリシー設定をしてください。	String	IfNotPresent|Always|Never	IfNotPresent
dependencyCheck.retryCount	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ回数を設定してください。	Integer	0-1080	30
dependencyCheck.retryIntervalSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトのリトライ間隔秒数を設定してください。	Integer	0-360	10
dependencyCheck.connectTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの接続タイムアウト秒数を設定してください。	Integer	1-360	10
dependencyCheck.readTimeoutSeconds	初期化コンテナで実行するデプロイ依存関係チェックスクリプトの応答タイムアウト秒数を設定してください。	Integer	1-360	10
tls.enabled	Elasticsearch と暗号化通信をするかどうかを指定します。 true 暗号化通信をします。 false 暗号化通信をしません。 Elasticsearch の tls.enabled を true にした場合は true を指定してください。	Boolean	true|false	false
tls.secretName	Elasticsearch の TLS 通信シークレットの名前を設定してください。	String	−	なし tls.enabled に true を設定した場合は、必ず設定してください。

パラメタ	説明	型	設定範囲	デフォルト値
elasticUserSecretName	tls.enabled パラメタに true を設定した場合、Elasticsearch の elastic ユーザアカウントに設定するパスワードが設定されたシークレットの名前を設定してください。 シークレットの data.password にパスワードを設定してください。 この設定が行われていない場合、elastic ユーザアカウントのパスワードは"changeme"となります。	String	－	なし。

（凡例）

－：該当しません。

注※1

コンテナオーケストレーションに設定する値であるため、最大値、最大文字数および許容される文字パターンはコンテナオーケストレーションに依存します。詳細は Kubernetes ドキュメントを参照してください。

注※2

Logstash への設定値であるため、最大値、最大文字数および許容される文字パターンは、Logstash に依存します。詳細は Logstash ドキュメントを参照してください。

注※3

Helm の仕様によって、7 桁以上の Integer、Long 型の入力は指数形式（例：「1e+06」）に自動で変換されるため、String 型の仕様とします。設定する際は指定値をダブルクォーテーションで囲んでください。

8.4　各 RDBMS の使用に必要なパラメタ（SQL-Participant 限定）

次に示す RDBMS を使用する際に必要なパラメタについて説明します。

- PostgreSQL
- SQL Server
- HiRDB

8.4.1　PostgreSQL（SQL-Participant 限定）

PostgreSQL を使用する際に必要なパラメタは、HMP-PCTO のパラメタではなく、PostgreSQL のサーバ側のパラメタです。HMP-PCTO を意識して設定してください。各パラメタのデフォルト値などを含めた詳細については、PostgreSQL のマニュアルを参照してください。

表 8-28　PostgreSQL のパラメタ

パラメタ	説明
max_connections	PostgreSQL サーバに同時接続する最大数を設定します。sql.pool.max-pool-size に指定した値より大きい値を指定してください。 sql.pool.max-pool-size については、「(2)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。
max_prepared_transactions	同時にプリペアド状態にできるトランザクションの最大数を設定します。アプリケーションのサービスの同時実行数以上の値とするとともに、障害時にコミット、ロールバックのリトライを行うトランザクションの実行数を考慮した十分な値を指定してください。
idle_in_transaction_session_timeout	PostgreSQL サーバが開いているトランザクション内でアイドル状態になっている（クライアントのクエリを待っている）セッションを終了する時間をミリ秒で指定します。 PostgreSQL ではトランザクションタイムアウト時間を SQL-Participant から設定できないため、このパラメタの設定が必要です。トランザクションブランチの処理時間（業務要件）に応じたタイマ監視時間である sql.transaction-termination.timeout-in-millis のパラメタより大きい値を指定してください。

8.4.2　SQL Server（SQL-Participant 限定）

SQL Server を使用する際に必要なパラメタは、HMP-PCTO のパラメタではなく、SQL Server のサーバ側のオプションです。HMP-PCTO を意識して設定してください。各オプションのデフォルト値などを含めた詳細については、SQL Server のマニュアルを参照してください。

また、HMP-PCTO は XA インタフェースを使用して SQL Server に接続します。そのため、SQL Server 側で XA 分散トランザクションコンポーネントを有効化して、必要なロールを設定してください。

表 8-29　SQL Server のオプション

オプション	説明
user connections	SQL Server に同時接続する最大数を設定します。sql.pool.max-pool-size に指定した値より大きい値を指定してください。 sql.pool.max-pool-size については、「(2)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。

8.4.3　HiRDB（SQL-Participant 限定）

HiRDB を使用する際に必要なパラメタは、HMP-PCTO のパラメタではなく、HiRDB のサーバ側のサーバ定義のオペランドです。HMP-PCTO を意識して設定してください。

表 8-30　HiRDB のサーバ定義のオペランド

オペランド	説明
pd_max_users	HiRDB に同時接続する最大数を設定します。sql.pool.max-pool-size に指定した値より大きい値を指定してください。 sql.pool.max-pool-size については、「(2)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。

また、HiRDB サーバと接続するために、HiRDB のクライアント環境定義ファイル（HiRDB.ini）を作成する必要があります。HiRDB サーバに接続し、データベースを更新するために必要な定義を次に示します。

表 8-31　HiRDB のクライアント環境定義

クライアント環境定義	説明
PDHOST	接続する HiRDB サーバのホスト名を設定してください。
PDNAMEPORT	接続する HiRDB サーバのポート番号を設定してください。
PDSRVTYPE	接続する HiRDB サーバのサーバ種別を設定してください。
PDCONTYPE	HiRDB の接続時の接続方式を設定します。 ACTIVE を指定してください。
PDUSER	認可識別子およびパスワードを設定してください。
PDSWAITTIME	HiRDB サーバが HiRDB クライアントからの要求に対する応答を返してから，次に HiRDB クライアントから要求が来るまでの HiRDB サーバの最大待ち時間を設定します。 HiRDB を使用する場合、トランザクションタイムアウト時間を SQL-Participant から設定できないため、このパラメタの設定が必要です。トランザクションブランチの処理時間（業務要件）に応じたタイマ監視時間の sql.transaction-termination.timeout-in-millis パラメタより大きい値を指定してください。

HiRDB のサーバ定義および HiRDB のクライアント環境定義の各定義のデフォルト値などを含めた詳細については、マニュアル「HiRDB システム定義」を参照してください。

8.5 TP1 連携時の連携先 OpenTP1 に設定が必要なパラメタ（通常版かつ TP1-Bridge 限定）

TP1 連携機能を使用する際に必要なパラメタは、HMP-PCTO のパラメタではなく、連携先の OpenTP1 の次の定義のオペランドです。HMP-PCTO を意識して設定してください。

- rap リスナーサービス定義
- システム共通定義
- トランザクションサービス定義
- ユーザサービス定義

各定義のオペランドのデフォルト値などを含めた詳細については、マニュアル「OpenTP1 システム定義」を参照してください。

8.5.1 rap リスナーサービス定義（通常版かつ TP1-Bridge 限定）

連携先の OpenTP1 の rap リスナーサービス定義に設定が必要なオペランドを次に示します。

表 8-32 連携先の OpenTP1 の rap リスナーサービス定義のオペランド

オペランド	説明
rap_listen_port	rap リスナーのポート番号を設定します。 TP1-Bridge の tp1.cltj.dcrapport の設定値と同じ値を指定してください。
rap_recovery_serve	XA リソースサービスを使用する場合、クライアントからのリカバリ要求用待機 rap サーバ数を設定します。 rap サーバに接続する TP1-Bridge の pod 数を指定してください。
rap_parallel_server	rap サーバ数を設定します。 rap_recovery_server の設定値に、次の値を加算した値以上の値を設定してください。 • TP1-Bridge の tp1.pool.maxPoolSize の設定値 なお、複数の TP1-Bridge から同じ rap リスナー経由で rap サーバに接続する場合は、rap_recovery_server の設定値に、次の値を加算した値以上の値を設定してください。 • 各 TP1-Bridge の tp1.pool.maxPoolSize の設定値の合計
rap_connection_assign_type	ダイナミックコネクションスケジュール機能を使用するかどうかを設定します。 TP1-Bridge での常設コネクションのスケジュール方法は、スタティックコネクションスケジュールモードを前提としているため、static を指定してください。

8.5.2 システム共通定義（通常版かつ TP1-Bridge 限定）

連携先の OpenTP1 のシステム共通定義に設定が必要なオペランドを次に示します。

表 8–33　連携先の OpenTP1 のシステム共通定義のオペランド

オペランド	説明
rpc_max_message_size	RPC コールで送受信できるメッセージ（ユーザ電文）の最大長を設定します。 TP1-Bridge の tp1.cltj.dccltrpcmaxmsgsize の設定値と同じ値を指定してください。 このパラメタの設定値を変更する場合は、TP1-Bridge の server.tomcat.maxSwallowSize の設定値が、AP が TP1-Bridge に送信するリクエストボディの最大サイズ以上の値となっているか見直してください。

8.5.3　トランザクションサービス定義（通常版かつ TP1-Bridge 限定）

連携先の OpenTP1 のトランザクションサービス定義に設定が必要なオペランドを次に示します。

表 8–34　連携先の OpenTP1 のトランザクションサービス定義のオペランド

オペランド	説明
trn_xar_use	XA リソースサービスを使用するかどうかを設定します。 XA リソースサービスを使用するため、Y を指定してください。

8.5.4　HMP-PCTO のトランザクションに参加する SPP のユーザサービス定義（通常版かつ TP1-Bridge 限定）

連携先の OpenTP1 の HMP-PCTO のトランザクションに参加する SPP のユーザサービス定義に設定が必要なオペランドを次に示します。

表 8–35　連携先の OpenTP1 のユーザサービス定義のオペランド

オペランド	説明
atomic_update	サービスグループのプロセスでトランザクションを発生させるかどうかを設定します。 トランザクションの範囲に入れるため、Y を指定してください。

8.6 uCosminexus Application Server に設定が必要な TP1-Converter のパラメタ（通常版かつ TP1-Bridge 限定）

TP1-Converter のパラメタは、次の 2 つのファイルに設定してください。

- uCosminexus Application Server の usrconf.properties（J2EE サーバ用ユーザプロパティファイル）
- uCosminexus Application Server の usrconf.cfg（J2EE サーバ用オプション定義ファイル）

jp.co.Hitachi.soft.hmppcto のパラメタ以外は、uCosminexus Application Server のパラメタです。uCosminexus Application Server のパラメタの詳細については、マニュアル「Cosminexus アプリケーションサーバ リファレンス 定義編(サーバ定義)」を参照してください。

ファイルの格納場所は、「/opt/Cosminexus/CC/server/usrconf/ejb/TP1Converter/」です。

8.6.1 usrconf.properties（通常版かつ TP1-Bridge 限定）

(1) jp.co.Hitachi.soft.hmppcto のパラメタ

パラメタの接頭辞は、jp.co.Hitachi.soft.hmppcto です。

表 8–36 usrconf.properties の jp.co.Hitachi.soft.hmppcto のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
tp1.data-conversion.definition-file.dir	データ変換で使用する定義ファイルを格納するディレクトリの絶対パス名を設定してください。	String	−	なし 必ず設定してください。

(2) webserver.connector のパラメタ

表 8–37 usrconf.properties の webserver.connector のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
webserver.connector.nio_http.max_threads	TP1-Bridge のデータ変換機能を実行するリクエスト処理スレッドの最大数を設定します。 トランザクションの同時実行数+1 以上の値を設定してください。+1 は、キャッシュの参照およびリフレッシュの実行で使用される分です。 複数の TP1-Bridge が同一の TP1-Converter を使用する場合は、それぞれの TP1-Bridge のトランザクションの同時実行数の合計+1 以上の値を設定してください。	uCosminexus Application Server の仕様に依存します。		

パラメタ	説明	型	設定範囲	デフォルト値
webserver.connector.nio_http.max_servlet_execute_threads	webserver.connector.nio_http.max_threads のうち、Web コンテナが同期サーブレットを実行するスレッドの最大数を設定します。 TP1-Converter は、同期サーブレットとして実装されています。そのため、webserver.connector.nio_http.max_threads と同じ値を設定してください。			
webserver.connector.nio_http.port	TP1-Bridge のデータ変換機能に対するリクエストを受信する uCosminexus Application Server の NIO HTTP サーバのポート番号を設定します。 TP1-Bridge の converter.port と同じ値を設定してください。			
webserver.connector.nio_http.limit.max_request_body	HTTP リクエストのリクエストボディの最大サイズを設定します。 上限値なしを意味する-1（デフォルト値）を設定してください。			
webserver.connector.nio_http.keep_alive.max_requests	TCP コネクションを持続したままの連続接続回数の上限を設定します。 TP1-Bridge と TP1-Converter の接続は持続しないため、1 を設定してください。			

(3) ejbserver.application のパラメタ

表 8–38 usrconf.properties の ejbserver.application のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
ejbserver.application.userlog.loggers	TP1-Converter がメッセージの出力に使用するロガーを設定します。 「jp.co.Hitachi.soft.hmppcto.converter」を設定してください。	uCosminexus Application Server の仕様に依存します。		
ejbserver.application.userlog.Logger.jp.co.Hitachi.soft.hmppcto.converter.handlers	ロガーが使用するハンドラを設定します。 「com.hitachi.software.ejb.application.userlog.CJMessageFileHandler;conf1」を設定してください。			
ejbserver.application.userlog.CJLogHandler.conf1.level	ハンドラが出力するメッセージのレベルを設定します。 「INFO」を設定してください。			

パラメタ	説明	型	設定範囲	デフォルト値
ejbserver.application.userlog.Logger.jp.co.Hitachi.soft.hmppcto.converter.level	ロガーが出力するメッセージのレベルを設定します。 「INFO」を設定してください。			

8.6.2 usrconf.cfg（通常版かつ TP1-Bridge 限定）

(1) add のパラメタ

表 8–39 usrconf.cfg の add のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
add.class.path	uCosminexus Application Server のデータ変換 API のライブラリの絶対パス名を設定します。 「/opt/Cosminexus/CSC/lib/cscdt_api.jar」を設定してください。	uCosminexus Application Server の仕様に依存します。		

9 メッセージ

9.1　メッセージの概要

9.1.1　メッセージの記述形式

メッセージログの記述形式を記載します。

KFSG$N_1N_2N_3N_4N_5$-X (Y)

メッセージテキスト

メッセージの説明や要因コードなど、個別出力項目の説明を示します。

(S)システムの動作を示します。

(O)オペレータの処置を示します。

〔対策〕

システム管理者の処置を示します。

メッセージ中の、オペレータの処置、または対策で「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、システム管理者が当社問い合わせ窓口へ連絡することを示します。

メッセージ ID の各項目についての説明を記載します。

KFSG：メッセージプレフィックス

$N_1N_2N_3N_4N_5$：コード

X：インジケータ

メッセージのレベルを表します。インジケータの表記と、対応するログレベル、内容を次の表に示します。

表 9-1　インジケータの表記と対応する内容

表記	ログレベル	内容
E	ERROR	稼働に支障のあるメッセージです。 参考例 • 環境設定、パラメタ誤りによって稼働できない • 稼働に支障のある障害発生
W	WARN	継続稼働しますが、注意が必要なメッセージです。 参考例 • 環境設定、パラメタ誤りのうち継続稼働可能なケースの警告 • メモリの使用状況などの警告 • ロールバックの通知
I	INFO	稼働状況変化（正常動作）を示すメッセージです。

表記	ログレベル	内容
		参考例 • 起動、終了

Y：出力先

メッセージログの出力先を示します。複数の出力先に出力する場合は、複数記載します。

表 9-2　メッセージログの出力先

表記	出力先
M	標準出力（標準エラー出力を含む）
J	例外メッセージ ユーザ責務の Kubernetes アプリケーションのユーザコーディング処理に例外の詳細メッセージとして渡します。
D	Kubernetes のイベントリソース※1
C	TP1-Converter が出力する uCosminexus Service Platform の J2EE アプリケーションのユーザログ※2

注※1

Kubernetes のコマンドで Kubernetes のイベントリソースへ出力されたメッセージログを参照できるのは、コンテナのライフサイクルイベントで実行した処理がエラー終了した場合だけです。コンテナのライフサイクルイベントで実行した処理が正常終了した場合は、Kubernetes のコマンドで Kubernetes のイベントリソースを参照してもメッセージログはありません。

注※2

uCosminexus Service Platform の J2EE アプリケーションのユーザログは、「*xx*user_log.log」のファイル名で次のディレクトリに出力されます。なお、ファイル名の *xx* は、1 から 16 の整数です。

/opt/Cosminexus/CC/server/public/ejb/TP1Converter/logs/user

9.1.2　メッセージ系統

メッセージ ID はコードごとに系統が分かれています。メッセージの系統を次の表に示します。

表 9-3　メッセージ系統

メッセージコード	メッセージの系統
00001～09999	不使用
10000～19999	不使用
20000～29999	不使用
30000～39999	不使用
40000～49999	不使用
50000～59999	不使用

メッセージコード	メッセージの系統
60000～69999	エラーメッセージ全般、警告メッセージも含む
70000～79999	トラブルシュート機能のメッセージ
80000～89999	正常に動作しているときのメッセージ
90000～99999	コマンド実行時のメッセージ

9.1.3　メッセージの説明

(1)　メッセージテキストの形式

メッセージテキストは英文です。次の形式が基本となります。

メッセージ主文 (詳細情報名 ＝ 詳細情報, 詳細情報名 ＝ 詳細情報...)

【メッセージテキスト例】

The aaa server ended abnormally. (information 1 ＝ aa....aa, information 2 ＝ bb....bb, information 3 ＝ cc....cc)

詳細情報は動的にテキストとして埋め込まれます。

(2)　詳細情報の出力要否

障害発生時には、値が取得可能な場合、次の情報を詳細情報として出力します。

- ブランチ XID(未採番の場合はルート XID)：分散トレースとの紐づけに使用します。Base64 エンコードした値を出力します。

【メッセージテキスト例】

Test message. (branchxid ＝ aa....aa)

aa....aa：ブランチ XID

Test message. (rootxid ＝ aa....aa)

aa....aa：ルート XID

次の情報は、メッセージログ出力処理で付与するため、詳細情報での出力は不要です。

- 出力日時
- ログレベル
- ホスト名

- スレッド名
- クラス名
- プロセス ID

(3) 例外の扱い

処理中に例外を throw する場合、次のとおりメッセージを出力します。

- 新規に例外オブジェクトを作成（例：throw new Exception("KSFG10001-E test message.");）し、ユーザの対処が必要な例外詳細メッセージを付与する場合、KFSG64001-E メッセージを出力します。
- 初回に異常系の例外を throw する場合、java.lang.Throwable.printStackTrace()でスタックトレースを出力します。スタックトレース出力はメッセージ出力 method 内で実行します。

(4) 記載例 KFSG10000 の場合

メッセージの記載例を次に示します。

KFSG10000-E (Y)M

The aaa server ended abnormally. (information 1 = aa....aa, information 2 = bb....bb)

aaa サーバを異常終了します。

aa....aa：情報 1

bb....bb：情報 2

(S)aaa サーバを異常終了します。

(O)システム管理者に連絡してください。

［対策］

- xxx メッセージログファイルにこのメッセージが出力された場合
  aaaaaaaa コマンドを実行してトラブルシュート情報を取得してから、保守員に連絡してください。
- yyy メッセージログファイルにこのメッセージが出力された場合
  yyy のバックアップを取得してから、保守員に連絡してください。

(5) メッセージで出力するパラメタ名の形式

メッセージ中のパラメタ名はケバブケースで出力します。

パラメタが Mediator のパラメタの場合はキャメルケースに置き換えてパラメタの確認を行ってください。

例：(parameter = jp.co.Hitachi.soft.participant.sender.queue-size)

(parameter = jp.co.Hitachi.soft.participant.sender.queue-size)の場合、キャメルケースでのパラメタ名はパッケージ名"jp.co.Hitachi.soft"を除いた"participant.sender.queueSize"となります。

9.2 メッセージ一覧

9.2.1 KFSG60000～KFSG69999

KFSG60000～KFSG69999 のメッセージを次に示します。

KFSG60000-E (Y)M

Transaction control failed to access the Resource Manager. (branchxid = aa....aa, information 1 = bb....bb, information 2 = cc....cc, information 3 = dd....dd, information 4 = ee....ee, information 5 = ff....ff)

リソースマネージャへのトランザクションマネージャのアクセスが失敗しました。

aa....aa：ブランチ XID

bb....bb：エラーが発生したリソースマネージャへのアクセス処理の種別

cc....cc：リソースマネージャが返却したエラーコード

dd....dd：接続先リソースマネージャのリソースマネージャ名
　指定されていない場合は*

ee....ee：接続先リソースマネージャのサーバ名
　指定されていない場合は*

ff....ff：接続先リソースマネージャのポート番号
　指定されていない場合は*

(S)処理の種別が 004 の場合、Participant の起動を中止します。それ以外の場合、トランザクションブランチ処理を中断し、Participant はサービス閉塞します。

(O)システム管理者に連絡してください。

〔対策〕
　リソースマネージャのデータの整合性を確認してデータ修復後、Participant のサービス閉塞を解除するため再起動してください。解決しない場合は、保守員に連絡してください。

KFSG60001-E (Y)M

The gRPC server failed to start. (parameter name = aa....aa)

gRPC サーバの起動に失敗しました。

aa....aa：エラーが発生したポート番号のパラメタ名

- jp.co.Hitachi.soft.hmppcto.sql.app.port
- jp.co.Hitachi.soft.hmppcto.sql.mediator.port
- jp.co.Hitachi.soft.hmppcto.tcc.mediator.port
- jp.co.Hitachi.soft.hmppcto.tp1.mediator.port
- jp.co.Hitachi.soft.hmppcto.grpc.port

(S)gRPC サーバの起動処理を終了します。

(O)直後に出力するスタックトレースを参照し、gRPC サーバが起動しなかった要因を除いたあと、Pod を再起動してください。

[対策]

なし。

KFSG60002-E (Y)M

An error occurred in the initialization of the Participant.

Participant の初期化時にエラーが発生しました。

(S)Participant を異常終了します。

(O)システム管理者に連絡してください。

[対策]

リソースマネージャで異常が発生していないか確認してください。このメッセージの直前に出力されたメッセージを参照して原因を取り除き、Participant を再起動してください。解決しない場合は、保守員に連絡してください。

KFSG60003-W (Y)M

Transaction control failed to access the Resource Manager. (branchxid = aa....aa, information 1 = bb....bb, information 2 = cc....cc, information 3 = dd....dd, information 4 = ee....ee, information 5 = ff....ff)

リソースマネージャへのトランザクションマネージャのアクセスが失敗しました。

aa....aa：ブランチ XID

トランザクション処理以外からのアクセスの場合は*

bb....bb：エラーが発生したリソースマネージャへのアクセス処理の種別

cc....cc：リソースマネージャが返却したエラーコード

dd....dd：接続先リソースマネージャのリソースマネージャ名

指定されていない場合は*

ee....ee：接続先リソースマネージャのサーバ名

指定されていない場合は*

ff....ff：接続先リソースマネージャのポート番号

指定されていない場合は*

(S)エラーが発生したリソースマネージャへのアクセス処理の種別が 001 の場合はトランザクションブランチをロールバックします。それ以外の場合はトランザクションブランチ処理を続行します。

コミットまたはロールバック処理でこのメッセージが出力された場合、コミットまたはロールバックのリトライを開始します。リトライ中は同じメッセージの出力を 60 秒間抑止します。コミットまたはロールバックが成功すると、KFSG80007-I または KFSG80008-I メッセージを出力します。

(O)システム管理者に連絡してください。

〔対策〕

- エラーコードが-7 の場合はリソースマネージャが起動しているか確認し、起動していなければ起動してください。ネットワーク障害が発生している場合は原因を特定し対策してください。パラメタ jp.co.Hitachi.soft.hmppcto.sql.db-type に PostgreSQL を指定している場合は PostgreSQL のパラメタ max_prepared_transactions の値を確認してください。
- エラーが発生したリソースマネージャへのアクセス処理の種別が 005、リソースマネージャが返却したエラーコードが XA_RB*（100～107）の場合は、リソースマネージャで例外が発生しましたが、ロールバック処理を続行するため対策は不要です。
- 上記以外の場合は保守員に連絡してください。

KFSG60006-E (Y)M

The specified property cannot be set. (property name = aa....aa)

指定したプロパティは設定できません。

aa....aa：プロパティ名

(S)SQL-Participant の起動処理を中止します。

(O)パラメタ jp.co.Hitachi.soft.hmppcto.sql.datasource.class-name で指定したデータソースは、プロパティ名のプロパティをサポートしていません。パラメタ jp.co.Hitachi.soft.hmppcto.sql.datasource.property.values に指定したプロパティ名を見直してください。そのあと、SQL-Participant を再起動してください。

〔対策〕

なし。

KFSG60007-E (Y)M

An error occurred in the processing to set the transaction timeout for the database. (transaction timeout = aa....aa)

データベースのトランザクションタイムアウトの設定でエラーが発生しました。

aa....aa：トランザクションタイムアウトの値（パラメタ jp.co.Hitachi.soft.hmppcto.sql.pool.transaction-timeout-in-sec）

(S)SQL-Participant の起動処理を中止します。

(O)システム管理者に連絡してください。

［対策］

出力した値がデータベースで有効かどうかを確認し値を見直してください。有効である場合は、直後に出力されるスタックトレースを参考に、データベースで異常が発生していないか確認してください。データベースの異常を取り除いたあと、SQL-Participant を再起動してください。解決しない場合は、保守員に連絡してください。

KFSG60008-E (Y)M

An error occurred in the processing to set properties for the data source. (property name = aa....aa)

データソースにプロパティを設定するときにエラーが発生しました。

aa....aa：設定しようとしたプロパティ名称

(S)SQL-Participant の起動処理を中止します。

(O)直後に出力するスタックトレースを参照し、パラメタ jp.co.Hitachi.soft.hmppcto.sql.datasource.property.values に指定した値を見直してください。そのあと、再起動してください。

［対策］

なし。

KFSG60009-W (Y)M

An error occurred in the processing to release the database connection. (information 1 = aa....aa, information 2 = bb....bb, information 3 = cc....cc)

データベース接続の解放でエラーが発生しました。

aa....aa：接続先データベースのデータベース名。指定されていない場合は*

bb....bb：接続先データベースのサーバ名。指定されていない場合は*

cc....cc：接続先データベースのポート番号。指定されていない場合は*

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

データベースが起動していない場合は起動してください。ネットワーク障害が発生している場合は障害原因を取り除いてください。そのあと、データベースを確認し、SQL-Participant との接続数がパラメタ jp.co.Hitachi.soft.hmppcto.sql.pool.max-pool-size を超えている場合、データベースを再起動してください。解決しない場合は、保守員に連絡してください。

KFSG60010-W (Y)M+J

A new transaction branch cannot be started because the Participant is blocked. (branchxid = aa....aa, information = bb....bb)

Participant が閉塞しているため新規トランザクションブランチの開始はできません。

aa....aa：ブランチ XID

bb....bb：サービス閉塞した Participant の識別子

- SQL-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.xa-server-name の値
- TP1-Bridge を使用する場合：tp1.xaServerName の値

(S)アプリケーションからの要求を無視して処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

このメッセージの直前に出力された KFSG60014-E を参照して原因を取り除き、再起動してください。解決しない場合は保守員に連絡してください。

KFSG60011-E (Y)M+J

The transaction branch could not be started. (branchxid = aa....aa, information 1 = bb....bb, information 2 = cc....cc, information 3 = dd....dd, information 4 = ee....ee)

トランザクションブランチを開始できません。

aa....aa：ブランチ XID

bb....bb：リソースマネージャが返却したエラーコードを示します。

cc....cc：接続先リソースマネージャのリソースマネージャ名
　指定されていない場合は*

dd....dd：接続先リソースマネージャのサーバ
　指定されていない場合は*

ee....ee：接続先リソースマネージャのポート番号
　指定されていない場合は*

(S)アプリケーションからの要求を無視してトランザクションをロールバックします。

(O)システム管理者に連絡してください。

［対策］
　エラーコードが-7 の場合はリソースマネージャが起動しているか確認し、起動していなければ起動してください。ネットワーク障害が発生している場合は原因を特定し対策してください。解決しない場合は保守員に連絡してください。

KFSG60012-W (Y)M+J

The transaction branch has already been run. (branchxid = aa....aa)

トランザクションブランチは Prepare 実行済みです。

aa....aa：ブランチ XID

(S)アプリケーションからの要求を無視して処理を続行します。

(O)システム管理者に連絡してください。

［対策］
　ユーザアプリケーションの処理を見直してください。解決しない場合は、保守員に連絡してください。

KFSG60013-W (Y)M+J

There are insufficient connections to between the Resource Manager stored in the connection pool. (branchxid = aa....aa)

コネクションプールに保管しているリソースマネージャとの間のコネクションが不足しました。

aa....aa：ブランチ XID

(S)アプリケーションからの要求を無視して処理を続行します。

(O)次のパラメタの値を見直し、必要であれば再起動してください。

- SQL-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.pool.max-pool-size
- TP1-Bridge を使用する場合：tp1.pool.maxPoolSize

〔対策〕

なし。

KFSG60014-E (Y)M

The Participant has been blocked. (branchxid = aa....aa, information = bb....bb)

Participant が閉塞しました。

aa....aa：ブランチ XID

bb....bb：サービス閉塞した Participant の識別子

- SQL-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.xa-server-name の値
- TP1-Bridge を使用する場合：tp1.xaServerName の値

(S)Participant は新規にトランザクションブランチを開始しません。仕掛中のトランザクションブランチはロールバックします。

(O)システム管理者に連絡してください。

〔対策〕

これより前に出力された KFSG60000-E および KFSG60011-E メッセージについて対策してください。そのあと、Participant を再起動してください。解決しない場合は、保守員に連絡してください。

KFSG60015-W (Y)M

The transaction termination timer timed out. (branchxid = aa....aa, transaction termination timeout = bb....bb)

トランザクション決着監視タイマがタイムアウトしました。

aa....aa：ブランチ XID

bb....bb：トランザクション決着監視タイマの値

- SQL-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.timeout-in-millis の値
- TP1-Bridge を使用する場合：tp1.transactionTermination.timeoutInMillis の値

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

- ログや分散トレースを参照し、タイムアウトの原因となっている問題点があるかどうか確認し、確認結果に従い対応してください。

- トランザクション決着監視タイマのタイマ監視時間のパラメタの値が十分な値になっていないおそれがあります。「3.10.2　タイマ」を参照し、タイマのチューニングパラメタの指定値を見直してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG60016-E (Y)M

A heuristic error occurred in the commit processing. (branchxid = aa....aa, errorCode = bb....bb, learninglog = cc....cc)

Commit 処理でヒューリスティックエラーが発生しました。

aa....aa：ブランチ XID

bb....bb：リソースマネージャが返却したエラーコード

cc....cc：LearningLog の情報（Mediator の IP、Mediator のポート番号、投票結果、ログ作成時刻）

(S)Commit 処理を終了し、サービス閉塞を行います。

(O)システム管理者に連絡してください。

[対策]

リソースマネージャで異常が発生していないか確認してください。リソースマネージャの異常を取り除いたあと、Participant を再起動してください。解決しない場合は、保守員に連絡してください。

KFSG60017-E (Y)M

A heuristic error occurred in the rollback processing. (branchxid = aa....aa, errorCode = bb....bb, learninglog = cc....cc)

Rollback 処理でヒューリスティックエラーが発生しました。

aa....aa：ブランチ XID

bb....bb：リソースマネージャが返却したエラーコード

cc....cc：LearningLog の情報（Mediator の IP、Mediator のポート番号、投票結果、ログ作成時刻）

(S)Rollback 処理を終了し、サービス閉塞を行います。

(O)システム管理者に連絡してください。

[対策]

リソースマネージャで異常が発生していないか確認してください。リソースマネージャの異常を取り除いたあと、Participant を再起動してください。解決しない場合は、保守員に連絡してください。

KFSG60018-W (Y)M+J

The transaction branch has not been started. (branchxid = aa....aa)

トランザクションブランチは開始していません。

aa....aa：ブランチ XID

(S)次のどれかの状態のため、アプリケーションからの該当するトランザクションブランチに対する要求に対してエラー応答して、処理を続行します。

- Participant の再起動によってトランザクションブランチの情報が破棄された状態
- 先行プリペア機能によるプリペア処理で、リソースマネージャとの接続障害などのエラーが発生し、リソースマネージャとのコネクションが使用できない状態
- トランザクションのキャンセル指示、トランザクション決着監視タイマのタイムアウト、またはトランザクション分散合意リプレイインターバルタイマのタイムアウトによるロールバック処理で、リソースマネージャとの接続障害などのエラーが発生し、リソースマネージャとのコネクションが使用できない状態
- トランザクションのキャンセル指示、トランザクション決着監視タイマのタイムアウト、またはトランザクション分散合意リプレイインターバルタイマのタイムアウトによるロールバック処理で、ロールバックが成功し、リソースマネージャとのコネクションの使用を終了した状態

(O)必要ありません。

［対策］

新しいトランザクションブランチとして、アプリケーションからの SQL を再実行してください。

KFSG60019-E (Y)M

An invalid state was detected in a transaction branch. (branchxid = aa....aa)

トランザクションブランチで不正な状態を検知しました。

aa....aa：ブランチ XID

(S)このメッセージのあとに KFSG60002-E が出力された場合は、Participant を起動しません。KFSG60024-E が出力された場合は、トランザクション決着を中止します。それ以外の場合はロールバックとして処理を続行します。

(O)システム管理者に連絡してください。

［対策］

このメッセージが出力される場合は Participant として問題が発生しているため、保守員に連絡してください。

KFSG60020-E (Y)M

An invalid state was detected in a transaction branch. (branchxid = aa....aa, consensus type = bb....bb)

トランザクションブランチで不正な状態を検知しました。

aa....aa：ブランチ XID

bb....bb：合意種別。"Commit"または"Rollback"。

(S)トランザクション決着を中止して処理を続行します。

(O)システム管理者に連絡してください。

[対策]

このメッセージが出力される場合は Participant として問題が発生しているため、保守員に連絡してください。

KFSG60021-E (Y)M

An error occurred in the replay processing of the transaction termination timer. (branchxid = aa....aa)

トランザクション決着監視タイマの Replay 投票処理でエラーが発生しました。

aa....aa：ブランチ XID

(S)Replay 投票をスキップします。そのあと、トランザクション分散合意リプレイインターバルタイマによって再投票が行われます。

(O)システム管理者に連絡してください。

[対策]

このメッセージが出力される場合は Participant として問題が発生しているため、保守員に連絡してください。

KFSG60022-E (Y)M

An error occurred in the replay processing of the transaction distributed-consensus replay interval timer. (branchxid = aa....aa)

トランザクション分散合意リプレイインターバルタイマの Replay 投票処理でエラーが発生しました。

aa....aa：ブランチ XID

(S)Replay 投票をスキップします。そのあと、トランザクション分散合意リプレイインターバルタイマによって再投票が行われます。

(O)システム管理者に連絡してください。

〔対策〕

このメッセージが出力される場合は Participant として問題が発生しているため、保守員に連絡してください。

KFSG60023-E (Y)M

An error occurred in the initialization processing of the timer. (information = aa....aa)

タイマ制御機能の初期化処理でエラーが発生しました。

aa....aa：保守情報

お問い合わせの際に連絡してください。

(S)Participant の起動処理を中止します。

(O)直後に出力するスタックトレースを参照し、要因を除いたあと、Pod を再起動してください。

〔対策〕

なし。

KFSG60024-E (Y)M

The transaction branch could not be settled. (rootxid = aa....aa)

トランザクションブランチの決着を行えませんでした。

aa....aa：ルート XID

(S)トランザクション決着処理を中止します。

(O)システム管理者に連絡してください。

〔対策〕

このメッセージが出力される場合は Participant として問題が発生しているため、保守員に連絡してください。

KFSG60025-W (Y)M

The Participant is not running yet.

Participant が起動状態になっていません。

(S)要求を無視します。

(O)このメッセージより前に出力されている KFSG60001-E メッセージを参照し、要因を除いたあと、Pod を再起動してください。

〔対策〕

なし。

KFSG60026-W (Y)M

Transaction control failed to access the Resource Manager. (branchxid = aa....aa, information 1 = bb....bb, information 2 = cc....cc, information 3 = dd....dd, information 4 = ee....ee, information 5 = ff....ff)

リソースマネージャへのトランザクションマネージャのアクセスが失敗しました。

aa....aa：ブランチ XID

トランザクション処理以外からのアクセスの場合は*

bb....bb：エラーが発生したリソースマネージャへのアクセス処理の種別

cc....cc：リソースマネージャが返却したエラーコード

dd....dd：接続先リソースマネージャのデータベース名

指定されていない場合は*

ee....ee：接続先リソースマネージャのサーバ名

指定されていない場合は*

ff....ff：接続先リソースマネージャのポート番号

指定されていない場合は*

(S)トランザクション完了通知の送信処理を中止します。Participant が Mediator からのトランザクション完了確認通知を受信すると、中止したトランザクション完了通知の送信処理をします。

(O)システム管理者に連絡してください。

〔対策〕

エラーコードが-7 の場合はリソースマネージャが起動しているか確認し、起動していなければ起動してください。ネットワーク障害が発生している場合は原因を特定し対策してください。それ以外の場合は保守員に連絡してください。

KFSG60027-E (Y)M

An error occurred in the processing to communicate with the participant.

Participant への通信処理でエラーが発生しました。

(S)トランザクション完了確認通知の送信処理を中止します。

(O)システム管理者に連絡してください。

〔対策〕

ホスト名から IP アドレスの解決ができない状態のため、直後に出力するスタックトレースを参照し、要因を取り除いたあと、Mediator を再起動してください。解決しない場合は、保守員に連絡してください。

KFSG60028-W (Y)M

The number of proposal log entries has exceeded the threshold. (threshold = aa....aa, proposal log size = bb....bb)

Proposal Log の件数が閾値を超えました。

aa....aa：jp.co.Hitachi.soft.hmppcto.mediator.judge.proposal-log-limit-num パラメタの値

bb....bb：閾値を超えた時の Proposal Log の件数

(S)トランザクションが完了済みの Proposal Log を削除します。トランザクションが完了していない Proposal Log は Participant へトランザクション完了確認通知を送信し、トランザクションの完了を確認します。

(O)このメッセージが頻繁に出力される場合は、jp.co.Hitachi.soft.hmppcto.mediator.judge.proposal-log-limit-num パラメタの値を見直してください。

〔対策〕

なし。

KFSG60029-W (Y)M+J

The SQL statement could not be executed, because the transaction termination timer timed out and rollback processing was performed. (branchxid = aa....aa, information 1 = bb....bb)

トランザクション決着監視タイマのタイムアウトによりロールバックしたため SQL を実行できません。

aa....aa：ブランチ XID

bb....bb：保守情報

お問い合わせの際に連絡してください。

(S)SQL を実行せずトランザクションブランチをロールバックします。

(O)システム管理者に連絡してください。

［対策］

メッセージ KFSG62014-W が出力されていないか確認してください。出力されている場合は、メッセージ KFSG62014-W に従って対策してください。出力されていない場合は、保守員に連絡してください。

KFSG60030-W (Y)M+J

Failed to obtain connection to database. (branchxid = aa....aa)

データベースとのコネクション取得に失敗しました。

aa....aa：ブランチ XID

(S)アプリケーションからの要求の場合、要求を無視して処理を続行します。リトライはしません。

コミットまたはロールバック処理でこのメッセージが出力された場合、コミットまたはロールバックのリトライを開始します。リトライ中は同じメッセージの出力を 60 秒間抑止します。コミットまたはロールバックが成功すると、KFSG80007-I または KFSG80008-I メッセージを出力します。

(O)システム管理者に連絡してください。

［対策］

データベースが起動しているか確認してください。データベースが起動している場合は、直後に出力されるスタックトレースを参考に、データベースで異常が発生していないか確認してください。

KFSG60031-E (Y)M

Failed to obtain connection to database.

データベースとのコネクション取得に失敗しました。

(S)SQL-Participant を異常終了します。

(O)システム管理者に連絡してください。

［対策］

データベースが起動しているか確認してください。データベースが起動している場合は、直後に出力されるスタックトレースを参考に、データベースで異常が発生していないか確認してください。

KFSG60032-W (Y)M

Failed to obtain connection to database.

データベースとのコネクション取得に失敗しました。

(S)トランザクション完了通知の送信処理を中止します。SQL-Participant が Mediator からのトランザクション完了確認通知を受信すると、中止したトランザクション完了通知の送信処理をします。

(O)システム管理者に連絡してください。

〔対策〕

データベースが起動しているか確認してください。データベースが起動している場合は、直後に出力されるスタックトレースを参考に、データベースで異常が発生していないか確認してください。

KFSG60033-W (Y)M

There are insufficient connections to between the Resource Manager stored in the connection pool. (branchxid = aa....aa)

コネクションプールに保管しているリソースマネージャとの間のコネクションが不足しました。

aa....aa：ブランチ XID

(S)Commit 処理または Rollback 処理をリトライします。

(O)次のパラメタの値を見直し、必要であれば再起動してください。

- SQL-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.pool.max-pool-size
- TP1-Bridge を使用する場合：tp1.pool.maxPoolSize

〔対策〕

なし。

KFSG61001-E (Y)M

Initialization of the EADs client failed.

EADS クライアントの初期化に失敗しました。

メッセージの次行から EADS の Exception 情報を出力します。

(S)Participant の初期化が失敗します。

(O)Exception 情報を確認して対策を行うか、管理者に連絡してください。

〔対策〕

Exception 情報を確認して、EADS クライアントの初期化失敗要因を取り除いてください。解決しない場合は、保守員に連絡してください。

KFSG61002-E (Y)M

An error occurred in the sending of a gRPC request to the mediator. (host = aa....aa, port = bb....bb, rootxid = cc....cc, information = dd....dd)

Mediator への gRPC 通信の送信でエラーが発生しました。

メッセージの次行から gRPC 通信の Exception 情報を出力します。

aa....aa：Mediator のホスト名または IP アドレス

bb....bb：Mediator のポート番号

cc....cc：ルート XID

dd....dd：保守情報
　お問い合わせの際に連絡してください。

(S)トランザクションは、ほかの Mediator との通信によって処理を継続します。

(O)Exception 情報を確認して対策を行うか、Mediator などに問題が発生していないかログおよびトレースを確認してください。

〔対策〕
　Exception やログ、およびトレースの確認結果に従い対応してください。解決しない場合は、保守員に連絡してください。

KFSG61004-W (Y)M

External consensus failed. (rootxid = aa....aa, consensus value = bb....bb)

補助合意に失敗しました。補助合意は一定時間後にリトライします。

メッセージの次行から EADS の Exception 情報を出力します。

aa....aa：補助合意を行うルート XID

bb....bb：補助合意の提案内容
　"Commit"または"Rollback"

(S)処理を続行します。

(O)このメッセージが継続して出力される場合は、補助合意用の EADS に問題が発生しているおそれがあるため、管理者に連絡してください。

〔対策〕
　EADS サーバのログを確認し、マニュアル「Elastic Application Data Store ユーザーズガイド」に従い対応してください。解決しない場合は、保守員に連絡してください。

KFSG61005-W (Y)M

External consensus failed. (rootxid = aa....aa, consensus value = bb....bb, reason = cc....cc)

補助合意に失敗しました。補助合意は一定時間後にリトライします。

aa....aa：補助合意を行うルート XID

bb....bb：補助合意の提案内容
　"Commit"または"Rollback"

cc....cc：失敗した REST-API の HTTP エラーコード

(S)処理を続行します。

(O)このメッセージが継続して出力される場合は補助合意用の Ext-Cons サーバに問題が発生しているおそれがあるため、必要ならば管理者に連絡してください。

[対策]
　Ext-Cons サーバの状態を確認し、必要ならば Ext-Cons サーバを再起動してください。

KFSG61006-E (Y)M

An error occurred in the processing to create a timer. (parameter = aa....aa)

タイマの作成に失敗しました。

メッセージの次行から Exception 情報を出力します。

aa....aa：タイマの作成に失敗した対象のパラメタ名

(S)Participant の初期化が失敗します。

(O)システム管理者に連絡してください。

[対策]
　Exception の確認結果に従い対応してください。解決しない場合は、保守員に連絡してください。

KFSG61007-W (Y)M

The timer value was rounded up. (parameter = aa....aa, original value = bb....bb, rounded up value = cc....cc)

タイマ値の切り上げを行いました。

aa....aa：タイマ値の切り上げを行った対象のパラメタ名

bb....bb：タイマ値の切り上げ前の値

cc....cc：タイマ値の切り上げ後の値

(S)処理を続行します。

(O)タイマ値の切り上げ後の値が妥当でない場合、パラメタの指定値を見直して変更してください。

〔対策〕

なし。

KFSG61008-E (Y)M

The thread pool queue overflowed. (aa....aa = bb....bb, cc....cc = dd....dd)

スレッドプールのキューあふれが発生しました。

aa....aa：キューあふれが発生したスレッドプールのスレッド数を指定したパラメタ名

bb....bb：キューあふれが発生したスレッドプールのスレッド数を指定したパラメタの値

cc....cc：キューあふれが発生したスレッドプールのスレッド割り当て待ちキューサイズを指定したパラメタ名

dd....dd：キューあふれが発生したスレッドプールのスレッド割り当て待ちキューサイズを指定したパラメタの値

(S)処理を続行します。

SQL-Participant のスレッドプールの場合：

- jp.co.Hitachi.soft.hmppcto.sql.participant-processing.queue-size：トランザクションがロールバック決着します。
- jp.co.Hitachi.soft.hmppcto.mediator-communication.queue-size：トランザクション分散合意リプレイインターバルタイマまたはトランザクション決着監視タイマでタイムアウトが発生します。

TCC-Participant のスレッドプールの場合：

- jp.co.Hitachi.soft.hmppcto.tcc.participant-processing.queue-size：トランザクションがロールバック決着します。
- jp.co.Hitachi.soft.hmppcto.mediator-communication.queue-size：トランザクション分散合意インターバルタイマまたはトランザクション決着監視タイマでタイムアウトが発生します。

TP1-Bridge のスレッドプールの場合：

- jp.co.Hitachi.soft.hmppcto.tp1.participant-processing.queue-size：トランザクションがロールバック決着します。
- jp.co.Hitachi.soft.hmppcto.mediator-communication.queue-size：トランザクション分散合意リプレイインターバルタイマまたはトランザクション決着監視タイマでタイムアウトが発生します。

Mediator のスレッドプールの場合：

SQL-Participant/TP1-Bridge の場合は、トランザクション分散合意リプレイインターバルタイマまたはトランザクション決着監視タイマでタイムアウトが発生します。TCC-Participant の場合

は、トランザクション分散合意インターバルタイマまたはトランザクション決着監視タイマでタイムアウトが発生します。

(O)スレッドプールのスレッド数およびスレッド割り当て待ちキューサイズのパラメタの指定値を見直し、必要であれば再起動してください。

SQL-Participant のスレッドプールの場合は、SQL-Participant がコネクションプールにプールする DB コネクションの数のパラメタ（jp.co.Hitachi.soft.hmppcto.sql.pool.max-pool-size）の指定値も見直してください。

TP1-Bridge のスレッドプールの場合は、TP1-Bridge がコネクションプールにプールする OpenTP1 とのコネクションの数のパラメタ（tp1.pool.maxPoolSize）の指定値も見直してください。

〔対策〕

なし

KFSG61101-E (Y)M

An error occurred in the initialization of the mediator.

Mediator の初期化時にエラーが発生しました。

(S)Mediator を異常終了します。

(O)このメッセージの前に出力されるメッセージを参照して障害を調査してください。

〔対策〕

なし。

KFSG61102-W (Y)M

Participation in consensus was not possible, because the status of the consensus log file is incorrect. (rootxid = aa....aa)

ConsensusLog の永続化に失敗したため、分散合意に参加しませんでした。

aa....aa：ルート XID

(S)処理を続行します。

(O)このメッセージに先行して出力される KFSG61105-E、KFSG61106-E、KFSG61107-E または、KFSG61108-E に従い対処してください。

〔対策〕

なし。

KFSG61103-E (Y)M

An exception occurred in the warm start processing.

Mediator の再起動処理で例外が発生しました。

メッセージの次行から Exception 情報を出力します。

(S)Mediator の初期化が失敗します。

(O)システム管理者に連絡してください。

〔対策〕

Mediator の再起動時に ConsensusLog ファイルの障害で再起動に失敗した場合、Mediator を再起動するためには ConsensusLog ファイルを削除する必要があります。

しかし、ConsensusLog ファイルを削除し、該当する Mediator を再起動すると分散合意の一貫性が維持できなくなります。

このため、ConsensusLog ファイルを削除して Mediator を再起動しないでください。

再起動時に ConsensusLog ファイルの読み込みに失敗した Mediator を復旧させたい場合は、Orchestrator、Entity-Service、Participant、Mediator を停止し、ConsensusLog ファイルを削除し、Orchestrator、Entity-Service、Participant、Mediator を再起動してください。

KFSG61104-E (Y)M

Creation of a consensus log file failed. (file name = aa....aa, reason = bb....bb)

ConsensusLog ファイルの作成に失敗しました。

aa....aa：ConsensusLog ファイル名

bb....bb：ConsensusLog ファイルの作成に失敗した理由

disk capacity shortage：ディスクの容量不足

occurred io error：IO エラーが発生

(S)Mediator の初期化が失敗します。

(O)Reason の内容に従い対処を行い、Mediator を再起動してください。

〔対策〕

- disk capacity shortage の場合
  Mediator に紐づける PersistentVolume の容量を拡張してください。
- occurred io error の場合
  このメッセージと併せて出力される KFSG61105-E もしくは、KFSG61107-E メッセージに従い対処してください。

KFSG61105-E (Y)M

File I/O processing for the consensus log file failed. (rootxid = aa....aa, file name = bb....bb, function = cc....cc, errno = dd....dd)

ConsensusLog ファイルへのファイル IO に失敗しました。

aa....aa：ルート XID

bb....bb：ConsensusLog ファイル名

cc....cc：システムコール名（pread, pwrite, open, close）

dd....dd：システムコールの errno

(S)処理を続行します。

(O) システム管理者に連絡してください。

〔対策〕

errno を基に Mediator に紐づけた PersistentVolume に異常がないか確認してください。異常を取り除いたあと Mediator を再起動してください。解決しない場合は、保守員に連絡してください。

KFSG61106-E (Y)M

A shortage of space in the consensus log file occurred in the processing to make the log persistent. (rootxid = aa....aa, max = bb....bb)

ConsensusLog の永続化時、ConsensusLog ファイルの容量不足が発生しました。

aa....aa：ルート XID

bb…bb：jp.co.Hitachi.soft.hmppcto.consensus-log.size の値

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

パラメタ jp.co.Hitachi.soft.hmppcto.consensus-log.size を見直し、必要なサイズに変更したあと Mediator を再起動してください。

KFSG61107-E (Y)M

A timeout occurred in the I/O processing. (time in seconds = aa....aa, rootxid = bb....bb, information = cc....cc)

ConsensusLog ファイルへのファイル IO 中にタイムアウトが発生しました。

aa....aa：IO 監視時間（パラメタ jp.co.Hitachi .soft.hmppcto.consensus-log.io.watch-in-seconds）

bb....bb：ルート XID

cc....cc：保守情報
お問い合わせの際に連絡してください。

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕
Mediator に紐づけた PersistentVolume に異常がないか確認してください。異常を取り除いたあと、Mediator を再起動してください。解決しない場合は、保守員に連絡してください。

KFSG61108-E (Y)M

Allocation of an I/O area failed. (size = aa....aa)

メモリ容量不足のため ConsensusLog ファイルのファイル IO 用メモリ領域の確保に失敗しました。

aa....aa：必要なサイズ

(S)Mediator の初期化が失敗します。

(O)システム管理者に連絡してください。

〔対策〕
Mediator に割り当てるメモリ容量を大きくして、Mediator を再起動してください。

KFSG61109-E (Y)M

This file system does not support direct I/O.

Mediator に紐づける PersistentVolume がダイレクト IO に対応していません。

(S)Mediator の初期化が失敗します。

(O)システム管理者に連絡してください。

〔対策〕
Mediator に紐づける PersistentVolume をダイレクト IO に対応した PersistentVolume に変更して Mediator を再起動してください。

KFSG61110-E (Y)M

Shrinking of the consensus log file failed. (file name = aa....aa, shrink size = bb....bb, required size= cc....cc)

ConsensusLog ファイルの縮小に失敗しました。

aa....aa：ConsensusLog ファイル名

bb....bb：jp.co.Hitachi.soft.hmppcto.consensus-log.size の値

cc....cc：必要なサイズ

(S)Mediator の初期化が失敗します。

(O)システム管理者に連絡してください。

〔対策〕

パラメタ jp.co.Hitachi.soft.hmppcto.consensus-log.size を見直し、必要なサイズに変更したあと Mediator を再起動してください。

KFSG61111-W (Y)M

The maximum number of participants in the transaction was exceeded. (rootxid = aa....aa)

トランザクションへ参加した Participant 数が上限数を超えました。

aa....aa：ルート XID

(S)処理を続行します。

(O)トランザクションへ参加した Participant 数が上限数を超えると、トランザクションは Commit 決着できません。このため、トランザクション処理を見直してください。

〔対策〕

なし。

KFSG61112-E (Y)M

An attempt to read the ConsensusLog file failed. (file name = aa....aa, reason = bb....bb)

ConsensusLog ファイルの読み込みに失敗しました。

aa....aa：ConsensusLog ファイル名

bb....bb：ConsensusLog ファイルの読み込みに失敗した理由

file corrupted：ConsensusLog ファイルの破壊を検出

invalid format：ConsensusLog ファイルのフォーマット不正を検出

(S)Mediator の初期化が失敗します。

(O)システム管理者に連絡してください。

［対策］

Mediator を再起動するためには ConsensusLog ファイルを削除する必要があります。しかし、ConsensusLog ファイルを削除し、該当する Mediator を再起動すると分散合意の一貫性が維持できなくなります。このため、ConsensusLog ファイルを削除した Mediator の再起動はしないでください。再起動時に Mediator を復旧させたい場合は、Orchestrator、Entity-Service、Participant、Mediator を停止し、ConsensusLog ファイルを削除し、Orchestrator、Entity-Service、Participant、Mediator を再起動してください。

KFSG61113-W (Y)M

The transaction termination timer timed out. (timeout in millis = aa....aa, branchxid = bb....bb)

トランザクション決着監視タイマがタイムアウトしました。

aa....aa：トランザクション決着監視タイマのタイマ監視時間(jp.co.Hitachi.soft.hmppcto.tcc.trn-termination.timeout-in-millis パラメタ)

bb....bb：ブランチ XID

(S)処理を続行します。

(O)システム管理者に連絡してください。

［対策］

- ログや分散トレースを参照し、タイムアウトの原因となっている問題点があるかどうか確認し、確認結果に従い対応してください。
- トランザクション決着監視タイマのタイマ監視時間のパラメタの値が十分な値になっていないおそれがあります。「3.10.2　タイマ」を参照し、タイマのチューニングパラメタの指定値を見直してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG61114-E (Y)M

All of the records in the consensus log file have become unusable. (file name = aa....aa)

ConsensusLog ファイルの全レコードが使用できなくなりました。

aa....aa：ConsensusLog ファイル名

(S)Mediator の停止処理を行います。

(O)システム管理者に連絡してください。

〔対策〕

Mediator の再起動時に ConsensusLog ファイルの障害で再起動に失敗した場合、Mediator を再起動するためには ConsensusLog ファイルを削除する必要があります。

しかし、ConsensusLog ファイルを削除し、該当する Mediator を再起動すると分散合意の一貫性が維持できなくなります。

このため、ConsensusLog ファイルを削除して Mediator を再起動しないでください。

再起動時に ConsensusLog ファイルの読み込みに失敗した Mediator を復旧させたい場合は、Orchestrator、Entity-Service、Participant、Mediator を停止し、ConsensusLog ファイルを削除し、Orchestrator、Entity-Service、Participant、Mediator を再起動してください。

KFSG61115-W (Y)M

Some of the records in the consensus log file have become unusable. (file name = aa....aa, number of records = bb....bb)

ConsensusLog ファイルの一部のレコードが使用できなくなりました。

aa....aa：ConsensusLog ファイル名

bb....bb：使用可能なレコード数

(S)処理を続行します。

(O)このメッセージに先行して出力される KFSG61105-E、KFSG61107-E に従い対処してください。

〔対策〕なし。

KFSG62000-E (Y)M

An error occurred in the initialization of the TCC participant.

TCC-Participant の初期化時にエラーが発生しました。

(S)Participant を異常終了します。

(O)システム管理者に連絡してください。

〔対策〕

このメッセージの前に出力されるメッセージを参照して障害を調査してください。

KFSG62001-E (Y)M

An error occurred in the initialization of the thread pool.

スレッドプールの初期化時にエラーが発生しました。

(S)Pod を異常終了します。

(O)システム管理者に連絡してください。

〔対策〕

直後に出力するスタックトレースを参照し、スレッドプールの初期化でエラーが発生した要因を除いたあと、Pod を再起動してください。

上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62002-E (Y)M

Retrieval of transactional branch information failed.

トランザクションブランチ情報の取得に失敗しました。

(S)トランザクションをロールバックします。

(O)システム管理者に連絡してください。

〔対策〕

アプリケーション側でトランザクション境界の設定、トランザクションの伝搬、トランザクションブランチの属性の設定が正しく実装できているかどうかを確認および修正してください。

上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62003-E (Y)M

The class annotated with @Tcc does not inherit the TccService interface. (class name = aa....aa)

@Tcc アノテーションを付与しているクラスが TccService インタフェースを継承していません。

aa....aa：Try 前処理で実行しようとしたインスタンスのクラス名

(S)トランザクションをロールバックします。

(O)システム管理者に連絡してください。

〔対策〕

@Tcc アノテーションを付与しているクラスが TccService インタフェースを継承するようアプリケーションを修正してください。

KFSG62004-E (Y)M

The recovery processing failed. (operationid = aa....aa)

未決着トランザクションの Recovery 処理が失敗しました。

aa....aa：OperationId

(S)Participant を異常終了します。

(O)システム管理者に連絡してください。

［対策］

直後に出力するスタックトレースを参照し、TccService#executeRecover()メソッドでエラーが発生した要因を除いたあと、Pod を再起動してください。

KFSG62006-W (Y)M

An error occurred in the processing to vote on whether to update API resources. (branchxid = aa....aa)

API リソース更新可否の投票処理でエラーが発生しました。

aa....aa：ブランチ XID

(S)API リソース更新可否の投票を中止します。そのあと、タイマ制御機能によってロールバックします。

(O)システム管理者に連絡してください。

［対策］

このメッセージが出力される場合は Participant として問題が発生しているため、保守員に連絡してください。

KFSG62007-E (Y)M

An error occurred in the confirmation processing. (branchxid = aa....aa, operationid = bb....bb)

Confirm 実行処理で例外が発生しました。

aa....aa：ブランチ XID

bb....bb：OperationId

(S)トランザクションを終了します。

(O)システム管理者に連絡してください。

［対策］

Pod を再起動すると、未決着トランザクションの Recovery 機能によって executeConfirm()メソッドを再実行し、トランザクションが決着に向かいます。

KFSG62008-E (Y)M

An error occurred in the cancellation processing. (branchxid = aa....aa, operationid = bb....bb)

Cancel 実行処理で例外が発生しました。

aa....aa：ブランチ XID

bb....bb：OperationId

(S)トランザクションを終了します。

(O)システム管理者に連絡してください。

［対策］

Pod を再起動すると、未決着トランザクションの Recovery 機能によって executeCancel ()メソッドを再実行し、トランザクションが決着に向かいます。

KFSG62009-W (Y)M

An error occurred in the replay processing of the timer. (branchxid = aa....aa)

タイマ制御機能による Replay 投票処理でエラーが発生しました。

aa....aa：ブランチ XID

(S)処理を続行します。そのあと、再投票します。

(O)システム管理者に連絡してください。

［対策］

このメッセージが出力される場合は Participant として問題が発生しているため、保守員に連絡してください。

KFSG62010-E (Y)M

The XID lock processing failed. (xid = aa....aa)

トランザクション状態を更新するための排他ロック処理に失敗しました。

aa....aa：XID

(S)処理状況に応じてコミットもしくはロールバックします。

(O)システム管理者に連絡してください。

［対策］

このメッセージが出力される場合は Participant として問題が発生しているため、保守員に連絡してください。

KFSG62011-W (Y)M

The thread pool executor has not finished.

スレッドプールの非同期タスクが終了していません。

(S)処理を続行します。

(O)システム管理者に連絡してください。

[対策]

未決着トランザクションを完了させたい場合など、必要に応じて Pod を再起動してください。

KFSG62012-E (Y)M

An error occurred in the getOperationId processing. (class name = aa....aa)

getOperationId 実行処理で例外が発生しました。

aa....aa：TccService インタフェースを実装したクラス名

(S)Participant を異常終了します。

(O)システム管理者に連絡してください。

[対策]

直後に出力するスタックトレースを参照し、TccService#getOperationId ()メソッドでエラーが発生した要因を除いたあと、Pod を再起動してください。

KFSG62013-E (Y)M

The TccService#executeTry() method for the same OperationId cannot be called multiple times within the same transaction. (branchxid = aa....aa, operationid = bb....bb)

同一 OperationId の TccService#executeTry()メソッドは同一トランザクション内で複数回呼び出すことはできません。

aa....aa：ブランチ XID

bb....bb：OperationId

(S)トランザクションをロールバックします。

(O)システム管理者に連絡してください。

[対策]

同一 OperationId の TccService#executeTry()メソッドは同一トランザクション内で複数回呼び出さないようにアプリケーションを修正してください。

KFSG62014-W (Y)M

Rollback/Cancel processing is performed because the transaction termination timer timed out. (timeout in millis = aa....aa, branchxid = bb....bb)

トランザクション決着監視タイマのタイムアウトによって Rollback/Cancel 処理を実行します。

aa....aa：トランザクション決着監視タイマのタイマ監視時間

- SQL-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.timeout-in-millis の値
- TCC-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.tcc.trn-termination.timeout-in-millis の値
- TP1-Bridge を使用する場合：tp1.transactionTermination.timeoutInMillis の値

bb....bb：ブランチ XID

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

- ログや分散トレースを参照し、タイムアウトの原因となっている問題点があるかどうか確認し、確認結果に従い対応してください。
- トランザクション決着監視タイマのタイマ監視時間のパラメタの値が十分な値になっていないおそれがあります。「3.10.2　タイマ」を参照し、タイマのチューニングパラメタの指定値を見直してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62015-W (Y)M

Rollback/Cancel processing is performed because the transaction consensus interval timer timed out. (interval in millis = aa....aa, branchxid = bb....bb)

トランザクション分散合意インターバルタイマのタイムアウトによって Rollback/Cancel 処理を実行します。

aa....aa：トランザクション分散合意処理をリトライする間隔

- SQL-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.transaction-termination.consensus.replay-interval-in-millis の値
- TCC-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.tcc.consensus.interval-in-millis の値
- TP1-Bridge を使用する場合：tpl.transactionTermination.consensus.replayIntervalInMillis の値

bb....bb：ブランチ XID

(S)処理を続行します。

(O)システム管理者に連絡してください。メッセージの出力が止まらない場合は、次のパラメタを見直してください。

- SQL-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.pod.service-name
- TCC-Participant を使用する場合：jp.co.Hitachi.soft.hmppcto.tcc.pod.service-name
- TP1-Bridge を使用する場合：tp1.pod.serviceName

〔対策〕

- ログや分散トレースを参照し、タイムアウトの原因となっている問題点があるかどうか確認し、確認結果に従い対応してください。
- トランザクション分散合意処理をリトライする間隔のパラメタの値が十分な値になっていないおそれがあります。「3.10.2　タイマ」を参照し、タイマのチューニングパラメタの指定値を見直してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62016-E (Y)M

The constructor of the TccResponse class cannot be called outside of the transaction.

TccResponse クラスのコンストラクタはトランザクション外で呼び出すことはできません。

(S)Participant を異常終了します。

(O)システム管理者に連絡してください。

〔対策〕

アプリケーションで TccResponse クラスのコンストラクタを呼び出すタイミングがトランザクション開始前またはトランザクション完了後である場合、このエラーメッセージを出力します。「(a)　未決着状態かつ自身の TccService が登録した永続化したデータの一覧を返す」に記載されているアプリケーション実装要領に従い、アプリケーションを修正してください。

KFSG62017-E (Y)M

The return value of the TccService#executeTry() method is invalid. (branchxid = aa....aa, operationid = bb....bb, reason = cc....cc)

TccService#executeTry()メソッドの戻り値が不正です。

aa....aa：ブランチ XID

bb....bb：OperationId

cc....cc：戻り値を不正と判断した理由を示します。

Null: 戻り値が null である

The transactionKey is null: 戻り値の TccResponse クラスのインスタンスの transactionKey フィールドが null である

(S)トランザクションをロールバックします。

(O)システム管理者に連絡してください。

〔対策〕

このエラーメッセージに含まれている reason の内容を基に、アプリケーションの TccService#executeTry()メソッドを修正してください。

KFSG62018-E (Y)M

The return value of the TccService#executeRecover() method is invalid. (operationid = aa....aa, reason = bb....bb)

TccService#executeRecover()メソッドの戻り値が不正です。

aa....aa：OperationId

bb....bb：戻り値を不正と判断した理由を示します。

Null: 戻り値が null である

Contains null: 戻り値の java.util.Set に null が含まれている

The transactionKey is null: 戻り値の java.util.Set に transactionKey フィールドが null である TccResponse クラスのインスタンスが含まれている

(S) Participant を異常終了します。

(O)システム管理者に連絡してください。

〔対策〕

このエラーメッセージに含まれている reason の内容を基に、アプリケーションの TccService#executeRecover()メソッドを修正してください。

KFSG62019-E (Y)M

Null cannot be specified as the return value of the TccService#getOperationId() method. (class name = aa....aa)

TccService#getOperationId()メソッドの戻り値に null を指定することはできません。

aa....aa：TccService インタフェースを実装したクラス名

(S) Participant を異常終了します。

(O)システム管理者に連絡してください。

〔対策〕

アプリケーションの TccService#getOperationId()メソッドを修正してください。

KFSG62020-E (Y)M

The length of the return value of the TccService#getOperationId() method is outside the valid range. (class name = aa....aa, length = bb....bb)

TccService#getOperationId()メソッドの戻り値の長さが範囲外です。

aa....aa：TccService インタフェースを実装したクラス名

bb....bb：戻り値の OperationId の長さ(バイト)

(S) Participant を異常終了します。

(O)システム管理者に連絡してください。

〔対策〕

アプリケーションの TccService#getOperationId()メソッドを修正してください。

KFSG62021-E (Y)M

The same operation ID cannot be assigned to multiple TCC services. (operationid = aa....aa)

同一 OperationId を複数の TccService に割り当てることはできません。

aa....aa：OperationId

(S) Participant を異常終了します。

(O)システム管理者に連絡してください。

〔対策〕

アプリケーションの各 TccService の getOperationId()メソッドについて、OperationId がそれぞれ一意となるように修正してください。

KFSG62022-E (Y)M

An error occurred during initialization of the thread pool that is used for communication with the mediator.

Mediator と通信するために使用するスレッドプールの初期化時にエラーが発生しました。

(S)Pod を異常終了します。

(O)システム管理者に連絡してください。

〔対策〕

直後に出力するスタックトレースを参照し、スレッドプールの初期化でエラーが発生した要因を除いたあと、Pod を再起動してください。

上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62023-W (Y)M

The executor of the thread pool that is used for communication with the mediator has not ended.

Mediator と通信するために使用するスレッドプールの非同期タスクが終了していません。

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

未決着トランザクションを完了させたい場合など、必要に応じて Pod を再起動してください。

KFSG62024-E (Y)M

An error occurred in the confirmation preprocessing. (branchxid = aa....aa, operationid = bb....bb)

Confirm の前処理で例外が発生しました。

aa....aa：ブランチ XID

bb....bb：operationid

(S)トランザクションを終了します。

(O)システム管理者に連絡してください。

〔対策〕

アプリケーション側でトランザクションを Confirm 実行処理後の本来の結果に修復してください。

また、原因調査のため保守員に連絡してください。

KFSG62025-E (Y)M

An error occurred in the cancellation preprocessing. (branchxid = aa....aa, operationid = bb....bb)

Cancel の前処理で例外が発生しました。

aa....aa：ブランチ XID

bb....bb：operationid

(S)トランザクションを終了します。

(O)システム管理者に連絡してください。

〔対策〕

アプリケーション側でトランザクションを Cancel 実行処理後の本来の結果に修復してください。

また、原因調査のため保守員に連絡してください。

KFSG62100-E (Y)D

One or more command arguments are invalid. (command line = aa....aa)

Usage: health_check.sh <retry-count> <retry-interval-seconds> <connect-timeout-seconds> <read-timeout-seconds> <http-port>

health_check.sh --version

トランザクションヘルスチェックスクリプトのコマンド引数が不正です。

aa....aa：実行されたコマンドライン

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

〔対策〕

コマンドライン引数を修正したあと、Pod を再起動してください。

KFSG62101-E (Y)D

One or more command arguments are invalid. (command line = aa....aa)

Usage: wait_transactions.sh <retry-count> <retry-interval-seconds> <connect-timeout-seconds> <read-timeout-seconds> <http-port>

wait_transactions.sh --version

未決着トランザクション待機スクリプトのコマンド引数が不正です。

aa....aa：実行されたコマンドライン

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

〔対策〕

コマンドライン引数を修正したあと、Pod を再起動してください。

KFSG62102-W (Y)D+M

An error occurred in the HTTP request. (error code = aa....aa)

HTTP リクエストに失敗しました。

aa....aa：curl コマンドのエラーコード

(S)処理を続行します。

(O)なし。

〔対策〕

- エラーが発生しているスクリプトが「トランザクションヘルスチェックスクリプト」または「未決着トランザクション待機スクリプト」の場合：

  ・直前に出力されている KFSG82100-I もしくは KFSG82102-I のメッセージに含まれている HTTP ポート番号が正しいかどうかを確認し、誤りがある場合はコマンドライン引数の HTTP ポート番号を修正したあと、Pod を再起動してください。

  ・「kubectl logs <Pod 名> -c <コンテナ名>」コマンドを実行します。<コンテナ名>は、コンテナライフサイクル設定で「トランザクションヘルスチェックスクリプト」および「未決着トランザクション待機スクリプト」の実行を定義しているコンテナを指定してください。コンテナのコンソール出力内容にエラーメッセージが出力されている場合は、該当するメッセージの原因に従い対策を実施してください。そのあと、Pod を再起動してください。

  ・上記対策を実施しても解決しない場合は、保守員に連絡してください。
- エラーが発生しているスクリプトが「デプロイ依存関係チェックスクリプト」の場合：

  Kubernetes コントロールプレーンに Kubernetes API サーバ（kube-apiserver）の Pod が正常に起動していることを確認してください。正常に起動していない場合は Kubernetes API サーバ（kube-apiserver）の Pod を再起動するなどの対処を実施してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62103-W (Y)D+M

The HTTP status code indicates an error. (status code = aa....aa)

HTTP ステータスコードがエラーを示しています。

aa....aa：HTTP ステータスコード

(S)処理を続行します。

(O)なし。

〔対策〕

- エラーが発生しているスクリプトが「トランザクションヘルスチェックスクリプト」または「未決着トランザクション待機スクリプト」の場合：

  ・「kubectl logs <Pod 名> -c <コンテナ名>」コマンドを実行します。<コンテナ名>は、コンテナライフサイクル設定で「トランザクションヘルスチェックスクリプト」および「未決着トランザクション待機スクリプト」の実行を定義しているコンテナを指定してください。コンテナのコンソール出力内容にエラーメッセージが出力されている場合は、該当するメッセージの原因に従い対策を実施してください。そのあと、Pod を再起動してください。

  ・保守員に連絡してください。

- エラーが発生しているスクリプトが「デプロイ依存関係チェックスクリプト」の場合：

  ・Kubernetes コントロールプレーンに Kubernetes API サーバ（kube-apiserver）の Pod が正常に起動していることを確認してください。正常に起動していない場合は Kubernetes API サーバ（kube-apiserver）の Pod を再起動するなどの対処を実施してください。

  ・HTTP ステータスコードが「403」の場合、デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストが正しくデプロイできているかどうかを確認し、デプロイできていない場合はデプロイしたあと、Pod を再起動してくだい。

  ・HTTP ステータスコードが「403」の場合、デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの記述内容が正しいかどうかを確認し、誤りがある場合は記述内容を修正したあと、Pod を再起動してください（デプロイ依存関係チェック機能用ロールおよびロールバインディングの Kubernetes マニフェストの記述内容については、各 Pod の Namespace に応じてリソースの種類を「Role＆RoleBinding」と「ClusterRole＆ClusterRoleBinding」のどちらにする必要があるかについても確認してください)。

  ・HTTP ステータスコードが「404」の場合、直前に出力されている KFSG82104-I のメッセージに含まれているターゲット Kubernetes オブジェクト群の指定内容が正しいかどうかを確認し、誤りがある場合は指定内容を修正したあと、Pod を再起動してください。

- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62104-W (Y)D+M

The JSON response body is invalid. (response body = aa....aa)

JSON レスポンスボディが不正です。

aa....aa：返却された HTTP レスポンスボディの内容（改行などの特殊文字エスケープ済み）

(S)処理を続行します。

(O)なし。

〔対策〕

- エラーが発生しているスクリプトが「トランザクションヘルスチェックスクリプト」または「未決着トランザクション待機スクリプト」の場合：

・直前に出力されている KFSG82100-I もしくは KFSG82102-I のメッセージに含まれている HTTP ポート番号が正しいかどうかを確認し、誤りがある場合はコマンドライン引数の HTTP ポート番号を修正したあと、Pod を再起動してください。

・「kubectl logs <Pod 名> -c <コンテナ名>」コマンドを実行します。<コンテナ名>は、コンテナライフサイクル設定で「トランザクションヘルスチェックスクリプト」および「未決着トランザクション待機スクリプト」の実行を定義しているコンテナを指定してください。コンテナのコンソール出力内容にエラーメッセージが出力されている場合は、該当するメッセージの原因に従い対策を実施してください。そのあと、Pod を再起動してください。

・上記対策を実施しても解決しない場合は、保守員に連絡してください。

- エラーが発生しているスクリプトが「デプロイ依存関係チェックスクリプト」の場合：

  Kubernetes コントロールプレーンに Kubernetes API サーバ（kube-apiserver）の Pod が正常に起動していることを確認してください。正常に起動していない場合は Kubernetes API サーバ（kube-apiserver）の Pod を再起動するなどの対処を実施してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62105-W (Y)D

The service is not ready yet.

サービスがまだ新規トランザクションを受け付けられる状態になっていません。

(S)処理を続行します。

(O)なし。

〔対策〕

- 直後に KFSG82101-I (Y)D もしくは KFSG82103-I のメッセージが出力されている場合は、リトライによって処理が成功しているため、対策は不要です。
- 直後に KFSG62107-E (Y)D もしくは KFSG62108-E のメッセージが出力されている場合は、リトライオーバーしているため、KFSG62107-E (Y)D、または KFSG62108-E の対策を参照してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62106-W (Y)D

The service still has active transactions. (number of the active transactions = aa....aa)

サービスにまだ未決着トランザクションが存在します。

aa....aa：未決着トランザクション件数

(S)処理を続行します。

(O)なし。

〔対策〕

- 直後に KFSG82101-I (Y)D もしくは KFSG82103-I のメッセージが出力されている場合は、リトライによって処理が成功しているため、対策は不要です。
- 直後に KFSG62107-E (Y)D もしくは KFSG62108-E のメッセージが出力されている場合は、リトライオーバーしているため、KFSG62107-E (Y)D もしくは KFSG62108-E の対策を参照してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62107-E (Y)D

The specified number of retries for the transaction health-check processing was exceeded.

トランザクションヘルスチェック処理でリトライオーバーが発生しました。

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

〔対策〕

- 直前に Mediator/Participant のエラーメッセージが出力されている場合は、該当するメッセージの原因に従い対策を実施したあと、Pod を再起動してください。
- 直前に Mediator/Participant のエラーメッセージが出力されていない場合は、トランザクションヘルスチェックスクリプトもしくは未決着トランザクション待機スクリプトのコマンド引数に指定しているリトライ回数もしくはリトライ間隔秒数が、リトライに十分な値になっていないおそれがあります。リトライ回数もしくはリトライ間隔秒数の指定値を見直したあと、Pod を再起動してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62108-E (Y)D

The specified number of retries for the processing that waits for active transactions was exceeded.

未決着トランザクション待機処理でリトライオーバーが発生しました。

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

〔対策〕

- 直前に Mediator/Participant のエラーメッセージが出力されている場合は、そのメッセージの原因に従い対策を実施してください。そのあと、未決着トランザクションを完了させたい場合など、必要に応じて Pod を再起動してください。

- 直前に Mediator/Participant のエラーメッセージが出力されていない場合は、トランザクションヘルスチェックスクリプトもしくは未決着トランザクション待機スクリプトのコマンド引数に指定しているリトライ回数もしくはリトライ間隔秒数が、リトライに十分な値になっていないおそれがあります。リトライ回数もしくはリトライ間隔秒数の指定値を見直してください。そのあと、未決着トランザクションを完了させたい場合など、必要に応じて Pod を再起動してくだい。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62109-E (Y)D+M

One or more command arguments are invalid. (command line = aa....aa)
Usage: dependency_check.sh <retry-count> <retry-interval-seconds> <connect-timeout-seconds> <read-timeout-seconds> {<namespace>/<resource>/[<name>|<label-selector>]} [...]
dependency_check.sh --version

デプロイ依存関係チェックスクリプトのコマンド引数が不正です。

aa....aa：実行されたコマンドライン

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

[対策]
コマンドライン引数を修正したあと、Pod を再起動してください。

KFSG62110-W (Y)D+M

The target Kubernetes object was not found. (target = aa....aa)

ターゲット Kubernetes オブジェクトが存在していません。

aa....aa：ターゲット Kubernetes オブジェクト

(S)処理を続行します。

(O)なし。

[対策]

- 直後に KFSG82105-I のメッセージが出力されている場合は、リトライによって処理が成功しているため、対策は不要です。
- 直後に KFSG62112-E のメッセージが出力されている場合は、リトライオーバーしているため、KFSG62112-E の対策を参照してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62111-W (Y)D+M

The target Kubernetes object is not ready yet. (target = aa....aa)

ターゲット Kubernetes オブジェクトがまだ起動完了していません。

aa....aa：ターゲット Kubernetes オブジェクト

(S)処理を続行します。

(O)なし。

〔対策〕

- 直後に KFSG82105-I のメッセージが出力されている場合は、リトライによって処理が成功しているため、対策は不要です。
- 直後に KFSG62112-E のメッセージが出力されている場合は、リトライオーバーしているため、KFSG62112-E の対策を参照してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62112-E (Y)D+M

The specified number of retries for the dependency check processing was exceeded.

デプロイ依存関係チェック処理でリトライオーバーが発生しました。

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

〔対策〕

- 直前に KFSG62102-W、KFSG62103-W、KFSG62104-W のエラーメッセージが出力されている場合は、該当するメッセージの原因に従い対策を実施したあと、Pod を再起動してください。
- 直前に KFSG62110-W のエラーメッセージが出力されている場合は、該当する Pod が起動していないため、Pod を起動してください。
- 直前に KFSG62111-W のエラーメッセージが出力されている場合は、デプロイ依存関係チェックスクリプトのコマンド引数に指定しているリトライ回数もしくはリトライ間隔秒数が、リトライに十分な値になっていないおそれがあります。リトライ回数もしくはリトライ間隔秒数の指定値を見直したあと、Pod を再起動してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62200-E (Y)J+M

Failed to read the kubeconfig file.

kubeconfig ファイルの読み込みに失敗しました。

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

[対策]

- 該当するコンテナで環境変数 KUBECONFIG を定義している場合は、環境変数 KUBECONFIG に設定している kubeconfig ファイルのパスおよび kubeconfig ファイルの内容が正しいことを確認してください。直後に出力するスタックトレースを参照し、設定内容に誤りがあれば、設定内容を見直したあと、Pod を再起動してください。
- 該当するコンテナに$HOME/.kube/config ファイルが存在している場合は、この kubeconfig ファイルの内容が正しいことを確認してください。直後に出力するスタックトレースを参照し、設定内容に誤りがあれば、設定内容を見直したあと、Pod を再起動してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62201-E (Y)J+M

The request to the Kubernetes API server timed out. (timeout in seconds = aa....aa)

Kubernetes API サーバへのリクエストがタイムアウトしました。

aa....aa：Kubernetes からのサービス一覧取得の待ち時間（jp.co.Hitachi.soft.hmppcto.lb.list-timeout-in-seconds パラメタ）

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

[対策]

- 該当するコンテナで「6.4　各 Kubernetes アプリケーションの作成要件」に示すとおり Pod に対するロールベースアクセス制御（RBAC）が設定されていることを確認してください。設定内容に誤りがあれば、設定内容を見直したあと、Pod を再起動してください。
- Kubernetes コントロールプレーンに Kubernetes API サーバ（kube-apiserver）の Pod が正常に起動していることを確認してください。正常に起動していない場合は Kubernetes API サーバ（kube-apiserver）の Pod を再起動するなどの対処を実施してください。
- jp.co.Hitachi.soft.hmppcto.lb.list-timeout-in-seconds パラメタに指定している Kubernetes からのサービス一覧取得の待ち時間が十分な値になっていないおそれがあります。値を見直したあと、Pod を再起動してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG62202-E (Y)J+M

Failed to read the Kubernetes namespace file.

Kubernetes Namespace ファイルの読み込みに失敗しました。

(S)要求を無視します。

(O)システム管理者に連絡してください。

〔対策〕

- 直後に出力するスタックトレースを参照し、該当するコンテナに Kubernetes Namespace ファイルがマウントされていない場合や、ファイルの状態が不正であるなどの場合は、Pod を再起動してください。
- 上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG63000-E (Y)J+M

The method is not supported by the JDBC proxy driver.

JDBC Proxy Driver がサポートしていないメソッドです。

(S)要求を無視します。

(O)システム管理者に連絡してください。

〔対策〕

サポート対象のメソッドを使用するようアプリケーションを修正してください。

KFSG63001-E (Y)J+M

The format of the URL specified in the configuration file is incorrect. (information 1 = aa....aa)

設定ファイルに記載している URL の形式が不正です。

aa....aa：設定ファイルに記載している URL

(S)要求を無視します。

(O)管理者に連絡してください。

〔対策〕

Spring Boot の設定ファイル（application.properties、application.yml など）に設定している次の項目の設定値を修正してください。

- spring.datasource.url

KFSG63002-E (Y)J+M

Failed to communicate with the participant. (branchxid = aa....aa, requet-type = bb....bb, host = cc....cc, port = dd....dd)

Participant との通信に失敗しました。

aa....aa：ブランチ XID

bb....bb：通信に失敗した契機（Participant への要求）を示します。

SQL：SQL の実行要求

Prepare：先行 Prepare 機能使用時の Prepare 要求

cc....cc：Participant のホスト名

dd....dd：Participant のポート番号

(S)aa....aa が SQL の場合、トランザクションをロールバックします。Prepare の場合は、トランザクションの決着には影響しないで続行します。

(O)Entity-Service の Spring Boot の設定ファイル（application.properties、application.yml など）の次の項目に、Participant が gRPC を受信する URL が正しく設定されているか確認してください。正しくない場合は修正してください。

- spring.datasource.url

正しく設定されている場合にこのメッセージが出力される場合は Participant が起動しているか確認してください。起動していない場合は、システム管理者に連絡してください。

〔対策〕

Participant を起動してください。Participant が障害によって停止している場合は原因を調査してから起動してください。

KFSG63003-E (Y)J+M

The JDBC proxy driver was run outside the scope of a transaction.

JDBC Proxy Driver がトランザクションの範囲外で実行されました。

(S)要求を無視します。

(O) JDBC Proxy Driver はトランザクションの範囲内で実行するようプログラムを修正してください。

〔対策〕

なし

KFSG63005-E (Y)J+M

Execution of the method failed, because the instance generated by the JDBC driver was lost. (branchxid = aa....aa, class name = bb....bb)

Participant がダウンした場合 JDBC Driver が生成したインスタンスが失われているため、メソッドの実行に失敗しました。

aa....aa：ブランチ XID

bb....bb：実行しようとしたインスタンスのクラス名

(S)トランザクションをロールバックします。

(O)システム管理者に連絡してください。

［対策］

Participant で障害が発生していないか確認してください。

KFSG63006-E (Y)J+M

Processing to read or write data failed during execution of the JDBC Proxy Driver.

JDBC Proxy Driver の実行時にデータの読み込みまたは書き込みに失敗しました。

(S)要求を無視します。

(O)このメッセージのあとに出力される例外情報を参照して障害の原因を調査してください。

［対策］

なし。

KFSG63007-E (Y)J+M

An instance of an unsupported class was specified as an instance of the Object class. (branchxid = aa....aa, class name = bb....bb)

サポート対象外のクラスのインスタンスが、Object クラスのインスタンスとして指定されました。

aa....aa：ブランチ XID

bb....bb：指定されたインスタンスのクラス名

(S)要求を無視します。

(O)システム管理者に連絡してください。

〔対策〕

サポート対象の Object クラスのインスタンスを指定するようアプリケーションを修正してください。

KFSG63008-E (Y)J+M

An unsupported class was specified as a class representing a Java data type. (class name = aa....aa)

サポート対象外のクラスが、Java データ型を表すクラスとして指定されました。

aa....aa：指定された Java データ型を表すクラス

(S)要求を無視します。

(O)システム管理者に連絡してください。

〔対策〕

Java データ型を表すクラスにはサポート対象のクラスを指定するようアプリケーションを修正してください。

KFSG63009-E (Y)J+M

A value corresponding to an unsupported JDBC type was specified for the sqlType parameter, which specifies the JDBC type. (sqlType = aa....aa)

サポート対象外の JDBC 型を示す値が、JDBC 型を指定するパラメタ sqlType に指定されました。

aa....aa：指定された sqlType の値

(S)要求を無視します。

(O)システム管理者に連絡してください。

〔対策〕

サポート対象の JDBC 型を指定するようアプリケーションを修正してください。

KFSG63100-E (Y)M

In the processing to initialize the orchestrator, the specified number of retries for waiting for the mediator to start was exceeded. (expected value = aa....aa, counted mediator = bb....bb)

Orchestrator の初期化処理で Mediator の起動の待ち合わせのリトライ回数が設定値を超えました。

aa....aa：期待する Mediator 起動数(mediator.size 設定値)

bb....bb：現在の Mediator 起動数

(S)Orchestrator を異常終了します。

(O)Mediator を mediator.size 設定値以上起動してください。Mediator が mediator.size 設定値以上起動している場合にこのメッセージが出力される場合は、mediator.name、mediator.name.suffix の設定値に誤りがないか確認してください。誤っていた場合は修正してください。

〔対策〕

なし。

KFSG63101-E (Y)M

A failure occurred in the processing to send a transaction settlement notification from the orchestrator to the mediator. (rootxid = aa....aa, host = bb....bb, port = cc....cc)

Orchestrator から Mediator へのトランザクションの決着通知の送信で障害が発生しました。

aa....aa：ルート XID

bb....bb：Mediator のホスト名

cc....cc：Mediator のポート番号

(S)処理を続行します。

(O)このメッセージのあとに出力される例外情報を参照して障害の原因を調査してください。

〔対策〕

なし。

KFSG63102-E (Y)M

In the processing to initialize the orchestrator, a failure occurred in the processing to send a transaction cancellation notification to mediator. (host = aa....aa, port = bb....bb)

Orchestrator の初期化処理で Mediator へのトランザクションのキャンセル通知の送信で障害が発生しました。

aa....aa：Mediator のホスト名

bb....bb：Mediator のポート番号

(S)処理を続行します。

(O)このメッセージのあとに出力される例外情報を参照して障害の原因を調査してください。

〔対策〕

なし。

KFSG63103-E (Y)J+M

The transaction could not be started. (reason = aa....aa, expected value = bb....bb, counted mediator = cc....cc)

トランザクションを開始できませんでした。

aa....aa：トランザクションを開始できなかった理由

Lack of mediator to participate in the transaction：トランザクションに参加する Mediator の不足

bb....bb：期待する Mediator 起動数(mediator.minimum-start-size 設定値)

cc....cc：現在の Mediator 起動数

(S)トランザクションを開始せずに呼び出し元に例外を送出します。

(O)Mediator の動作状況を確認して期待する Mediator 起動数以上の Mediator を起動してください。

[対策]

なし。

KFSG63104-E (Y)J+M

A failure occurred in the processing to send a transaction settlement notification to all mediators participating in the transaction. (rootxid = aa....aa)

トランザクションに参加するすべての Mediator へのトランザクションの決着通知の送信で障害が発生しました。トランザクションの決着結果は不明です。

aa....aa：ルート XID

(S)処理を続行します。

(O)このメッセージの前に出力されている KFSG63101-E の例外情報を参照して通信障害の原因を確認してください。

トランザクションの決着結果は、HMP-PCTO が出力する分散トレースの次の Span から確認できます。

オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」の Tags: consensus-value に COMMIT、ROLLBACK のどちらかを出力します。

Span は、Jaeger の Web UI で表示して確認してください。

Tags に"root-xid"="< KFSG63104-E の aa....aa 出力値>"を入力して検索することで、該当トランザクションの分散トレースを表示します。

該当トランザクションの分散トレースから GrpcNotifyTerminationResultRequestSender で検索することでオペレーション名:j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender の Tags の内容を確認してください。

〔対策〕

なし。

KFSG63105-W (Y)J+M

The transaction was settled as a rollback. (rootxid = aa....aa, reason = bb....bb)

トランザクションがロールバック決着しました。

aa....aa：ルート XID

bb....bb：トランザクションがロールバック決着した理由を示します。

Communication failure with Entity-Service：Entity-Service との通信障害

Rollback request from Entity-Service：Entity-Service からのロールバック要求

Rollback target exception occurs：Rollback 対象の例外が発生

Rollback settlement notification from mediator：Mediator からの Rollback 決着通知

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

下表を参照してトランザクションがロールバックした原因を調査して、必要に応じて対策してください。

bb....bb の出力値	原因の調査方法
Communication failure with Entity-Service	AP が Entity-Service との通信時に出力している情報から通信障害の原因を調査してください。
Rollback request from Entity-Service	Entity-Service で出力している KFSG63107-W を参照して AP がロールバック対象のステータスコードを設定した原因を調査してください。
Rollback target exception occurs	このメッセージが出力している例外情報中の「Suppressed:」以降が AP が送出した例外の情報です。この情報や AP が出力している情報から、AP がロールバック対象の例外を送出した原因を調査してください。
Rollback settlement notification from mediator	Mediator、または Participant で次のメッセージが出力されていないか確認してください。出力されている場合は、これらのメッセージからロールバックした原因を調査してください。出力されていない場合は、保守員に連絡してください。 KFSG60003-W KFSG60011-E KFSG60014-E KFSG60018-W KFSG60019-E KFSG60021-E KFSG60022-E

bb....bb の出力値	原因の調査方法
	KFSG60029-W KFSG61008-E KFSG62002-E KFSG62003-E KFSG62006-W KFSG62009-W KFSG62010-E KFSG62013-E KFSG62014-W KFSG62015-W KFSG62017-E KFSG63005-E

KFSG63107-W (Y)M

The transaction branch was marked as a rollback. (branchxid = aa....aa, reason = bb....bb, status code = cc....cc)

トランザクションブランチをロールバックにマークしました。

aa....aa：ブランチ XID

bb....bb：トランザクションをロールバックにマークした理由

Communication failure with Entity-Service：Entity-Service との通信障害

Participant failure：Participant の障害

The HTTP status of the rollback target：Rollback 対象の HTTP ステータス

cc....cc：HTTP ステータスコード（出力する情報がない場合は「null」を表示します）

(S)トランザクションブランチをロールバックにマークして処理を続行します。トランザクションはロールバックします。

(O)システム管理者に連絡してください。

［対策］

下表を参照してトランザクションがロールバックした原因を調査して、必要に応じて対策してください。

bb....bb の出力値	原因の調査方法
Communication failure with Entity-Service	AP が Entity-Service との通信時に出力している情報から通信障害の原因を調査してください。
Participant failure	Participant、または Entity-Service で次のメッセージが出力されていないか確認してください。出力されている場合は、これらのメッセージからロールバックした原因を調査してください。出力されていない場合は、保守員に連絡してください。

bb....bb の出力値	原因の調査方法
	KFSG60010-W KFSG60011-E KFSG60012-E KFSG60013-W KFSG60016-E KFSG63002-E KFSG63005-E
The HTTP status of the rollback target	AP がロールバック対象の HTTP ステータスコードを設定した原因を調査してください。

KFSG63108-E (Y)M

An error occurred in the initialization of the orchestrator. Initialization of the orchestrator failed.

Orchestrator の初期化時にエラーが発生しました。Orchestrator の初期化に失敗しました。

(S)Orchestrator を異常終了します。

(O)このメッセージの前に出力されているメッセージを参照して障害を調査してください。

〔対策〕

なし。

KFSG63109-E (Y)M

An error occurred in the initialization of the entity service. Initialization of the entity service failed.

Entity-Service の初期化時にエラーが発生しました。Entity-Service の初期化に失敗しました。

(S)Entity-Service を異常終了します。

(O)このメッセージの前に出力されているメッセージを参照して障害を調査してください。

〔対策〕

なし。

KFSG63200-E (Y)M

An error occurred in the communication process with TP1-Converter. (branchxid = aa....aa, URL = bb....bb)

TP1-Converter との通信処理でエラーが発生しました。

aa....aa：ブランチ XID

bb....bb：通信先の TP1-Converter へのリクエスト URL

(S)OpenTP1 のサービスの呼び出し処理を中止して、TP1-Bridge を呼び出した AP にエラー応答を返します。

(O)このメッセージのあとに出力されている例外情報、および TP1-Converter のログを参照して障害を調査してください。

〔対策〕

なし。

KFSG63201-E (Y)M

Failed to obtain connection to TP1. (reason = aa....aa)

OpenTP1 とのコネクション取得に失敗しました。

aa....aa：コネクション取得に失敗した理由コード

TIMEOUT：コネクションプールからの割り当て待ち時間のタイムアウト

(S)OpenTP1 のサービスの呼び出し処理を中止して、TP1-Bridge を呼び出した AP にエラー応答を返します。

(O)システム管理者に連絡してください。

〔対策〕

次のパラメタの設定値を見直してください。

- jp.co.Hitachi.soft.hmppcto.tp1.pool.max-pool-size
- tp1.pool.allocationWaitTimeInMillis

そのあと、TP1-Bridge を再起動してください。

KFSG63204-E (Y)M

TP1-Bridge was run outside the scope of a transaction.

TP1-Bridge がトランザクションの範囲外で実行されました。

(S)OpenTP1 のサービスの呼び出し処理を中止して、TP1-Bridge を呼び出した AP にエラー応答を返します。

(O)システム管理者に連絡してください。

〔対策〕

TP1-Bridge を呼び出した AP を、トランザクションの範囲内で TP1-Bridge を呼び出すように修正してください。

KFSG63205-W (Y)M

Failed to return connection to TP1.

OpenTP1 のコネクションの返却に失敗しました。

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

保守員に連絡してください。

KFSG63207-E (Y)M

Execution of the TP1/Client/J method failed. (method = aa....aa)

TP1/Client/J のメソッドの実行に失敗しました。

aa....aa：実行に失敗した OpenTP1/Client/J のメソッド名

(S)TP1-Bridge の開始時にこのメッセージを出力した場合は、TP1-Bridge を異常終了します。TP1-Bridge の開始時以外でこのメッセージを出力した場合は、処理を続行します。

(O)このメッセージの前後に出力されている例外情報を参照して障害を調査してください。

〔対策〕

なし。

KFSG63208-E (Y)M

TP1 service called could not be executed, because the transaction termination timer timed out and rollback processing was performed. (branchxid = aa....aa, information 1 = bb....bb)

トランザクション決着監視タイマのタイムアウトによってロールバックしたため、OpenTP1 のサービスの呼び出しができませんでした。

aa....aa：ブランチ XID

bb....bb：保守情報

お問い合わせの際に連絡してください。

(S)OpenTP1 のサービスの呼び出し処理を中止して、TP1-Bridge を呼び出した AP にエラー応答を返します。

(O)システム管理者に連絡してください。

〔対策〕

このメッセージの前に KFSG62014-W メッセージが出力されているかどうかを確認してください。出力されている場合は、KFSG62014-W メッセージに従って対策してください。出力されていない場合は、保守員に連絡してください。

KFSG63209-E (Y)M

Failed to call TP1 service. (service_group_name = aa....aa, service_name = bb....bb)

OpenTP1 のサービスの呼び出しに失敗しました。

aa....aa：呼び出し先の OpenTP1 のサービスグループ名

bb....bb：呼び出し先の OpenTP1 のサービス名

(S)OpenTP1 のサービスの呼び出し処理を中止して、TP1-Bridge を呼び出した AP にエラー応答を返します。

(O)このメッセージの前後に出力されている例外情報を参照して障害を調査してください。

〔対策〕

なし。

KFSG63300-E (Y)C

An error occurred in the initialization of the tp1 converter. Initialization of the tp1 converter failed.

TP1-Converter の初期化時にエラーが発生しました。TP1-Converter の初期化に失敗しました。

(S)TP1-Converter を異常終了します。

(O)このメッセージの前に出力されているメッセージを参照して障害を調査してください。

〔対策〕

なし。

KFSG63301-E (Y)C

A parameter is outside the valid range or was not specified. (parameter = aa....aa, value = bb....bb, reason = cc....cc)

パラメタに設定範囲外の値が指定されている、またはパラメタが指定されていません。

aa....aa：パラメタ名

複数のパラメタが関係する場合は、「;」で区切られて複数表示されます。

bb....bb：パラメタの指定値

指定されていない場合は、「null」が表示されます。

複数のパラメタが関係する場合は、「;」で区切られて複数表示されます。

cc....cc：理由コード

VALUE：パラメタの指定値が設定範囲外です。

NULL：必須パラメタが指定されていません。

(S)TP1-Converter を異常終了します。

(O)パラメタの指定値を見直して、正しい値を指定してください。そのあと、TP1-Converter を再起動してください。

［対策］

なし。

KFSG63302-E (Y)C

The service group is not registered in the cache. (service_group_name = aa....aa)

サービスグループがキャッシュに登録されていません。

aa....aa：登録されていないサービスグループ名

(S)データ変換処理を終了し、TP1-Bridge にエラー応答を返します。

(O)このメッセージのあとに出力されている例外情報を参照して障害を調査してください。

［対策］

なし。

KFSG63303-E (Y)C

Failed to read request data.

リクエストデータの読み取りに失敗しました。

(S)データ変換処理を終了し、TP1-Bridge にエラー応答を返します。

(O)このメッセージのあとに出力されている例外情報を参照して障害を調査してください。

〔対策〕

なし。

KFSG63304-E (Y)C

The definition file storage directory does not exist. (path = aa....aa)

定義ファイル格納ディレクトリが存在しません。

aa....aa：定義ファイル格納ディレクトリのパス名

(S)TP1-Converter 起動時に発生した場合は、TP1-Converter の起動を停止します。リフレッシュ処理実行時に発生した場合は、リフレッシュ処理の実行元にエラー応答を返します。

(O)tp1.data-conversion.definition-file.dir パラメタの設定値が正しいかどうかを確認してください。

- 設定値が誤っていた場合：正しい値に変更してください。
- 設定値のディレクトリが存在していない場合：ディレクトリを作成してください。

そのあと、TP1-Converter を再起動してください。

〔対策〕

なし。

KFSG63305-E (Y)C

Failed to load subdirectories of the definition file storage directory. (path = aa....aa)

定義ファイル格納ディレクトリのサブディレクトリの読み込みに失敗しました。

aa....aa：定義ファイル格納ディレクトリのパス名

(S)TP1-Converter 起動時に発生した場合は、TP1-Converter の起動を停止します。リフレッシュ処理実行時に発生した場合は、リフレッシュ処理の実行元にエラー応答を返します。

(O)このメッセージのあとに出力されている例外情報を参照して障害を調査してください。

〔対策〕

なし。

KFSG63306-W (Y)C

The definition files required for data conversion are not available. (path = aa....aa, service = bb....bb)

データ変換に必要な定義ファイルが揃っていません。

aa....aa：定義ファイル格納ディレクトリ下のサービスグループ名のディレクトリのパス名

bb....bb：サービス名

サービスグループ名のディレクトリ内に定義ファイルが 1 つもない場合は、「null」が表示されます。

(S)キャッシュ登録処理を続行します。

(O)必要な定義ファイルを作成して定義ファイル格納ディレクトリ内に格納してください。そのあと、キャッシュのリフレッシュ処理を実行してください。

〔対策〕

なし。

KFSG63307-E (Y)C

JSON conversion failed.

JSON 変換に失敗しました。

(S)キャッシュ参照処理を終了し、キャッシュ参照処理の実行元にエラー応答を返します。

(O)このメッセージのあとに出力されている例外情報を参照して障害を調査してください。

〔対策〕

なし。

KFSG63308-E (Y)C

Failed to create message. (message ID = aa....aa, details = bb....bb, exception = cc....cc)

メッセ—ジの作成に失敗しました。

aa....aa：入力されたメッセージ ID

指定されていない場合は、「null」が表示されます。

bb....bb：詳細情報

指定されていない場合は、「null」が表示されます。

複数の詳細情報が関係する場合は、「;」で区切られて複数表示されます。

cc....cc：例外情報

(S)処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

保守員に連絡してください。

KFSG63309-E (Y)C

Failed to load the list of definition files or load the definition files. (path = aa....aa)

定義ファイルの一覧または定義ファイルの読み込みに失敗しました。

aa....aa：読み込みに失敗したパス名

- 定義ファイルの一覧の読み込みに失敗した場合：定義ファイルを格納しているディレクトリのパス名
- 定義ファイルの読み込みに失敗した場合：定義ファイルのパス名

(S)キャッシュ登録処理を続行します。

(O)このメッセージのあとに出力されている例外情報を参照して障害を調査してください。

[対策]
なし。

KFSG63310-E (Y)C

A new exception was thrown. (message = aa....aa)

処理中に例外が発生しました。

aa....aa：発生した例外の詳細メッセージ

(S)TP1-Converter を異常終了するか、または呼び出し元にエラー応答を返します。

(O)システム管理者に連絡してください。

[対策]
保守員に連絡してください。

KFSG63311-W (Y)C

The definition file name does not have _jtob or _btoj. (path = aa....aa, file name = bb....bb)

定義ファイルのファイル名に、「_jtob」または「_btoj」がありません。

aa....aa：定義ファイル格納ディレクトリ下のサービスグループ名のディレクトリのパス名

bb....bb：定義ファイル名

(S)キャッシュ登録処理を続行します。

(O)出力された定義ファイルのファイル名を修正してください。そのあと、キャッシュのリフレッシュ処理を実行してください。

〔対策〕

なし。

KFSG63312-E (Y)C

Cache registration of definition file failed. (message = aa....aa)

定義ファイルのキャッシュ登録に失敗しました。

aa....aa：発生した例外の詳細メッセージ

(S)キャッシュ登録処理を続行します。

(O)キャッシュ登録に失敗した定義ファイルを修正して、キャッシュのリフレッシュ処理を実行してください。

定義ファイルのファイル名の形式を次に示します。

<キャッシュ名>.<拡張子>

<キャッシュ名>は、次のどちらかです。

- 発生した例外の詳細メッセージ中の「キャッシュ名」
- 「cacheName」

<拡張子>は、次のどれかです。

- 発生した例外の詳細メッセージ中の「種別」
- 「formatDataKind」が「FDX」の場合：fdx
- 「formatDataKind」が「XSLT」の場合：xsl

〔対策〕

なし。

KFSG64001-E (Y)M

A new exception was thrown. (message = aa....aa)

処理中に例外が発生しました。

aa....aa：発生した例外の詳細メッセージ

(S)処理を続行します。

(O)例外の詳細メッセージまたは直後に出力されたメッセージに従ってください。

〔対策〕

なし。

KFSG64002-E (Y)M

A parameter is outside the valid range or was not specified. (parameter = aa....aa, value = bb....bb, reason = cc....cc)

パラメタに設定範囲外の値が指定されている、またはパラメタが指定されていません。

aa....aa：パラメタ名

複数のパラメタが関係する場合は、「;」で区切られて複数表示されます。

bb....bb：パラメタの指定値

指定されていない場合は、「null」が表示されます。複数のパラメタが関係する場合は、「;」で区切られて複数表示されます。

cc....cc：理由コード

VALUE：パラメタの指定値が設定範囲外です。

NULL：必須パラメタが指定されていません。

(S)処理を終了します。

(O)パラメタの指定値を見直し正しい値に変更したあと、再起動してください。

〔対策〕

なし。

KFSG64601-E (Y)M

The processing to initialize encryption failed.

暗号化の初期化処理に失敗しました。

メッセージの次行から暗号化の初期化処理で発生した Exception 情報を出力します。

(S)Pod の起動を中止します。

(O)システム管理者に連絡してください。

〔対策〕

Exception やログおよびトレースの確認結果に従い対応してください。

次に、出力される Exception 例と原因を示します。

- java.security.cert.CertificateExpiredException：NotAfter: 日付
  原因：証明書ファイルの有効期限が切れている。
- java.lang.IllegalArgumentException：File does not contain valid private key：証明書の秘密鍵ファイル名
  原因：秘密鍵ファイルのフォーマット不正、証明書ファイルに対応していない、など

KFSG64602-E (Y)M

An error occurred in the sending of a gRPC request to the participant. (service name = aa....aa, port = bb....bb, rootxid = cc....cc, information = dd....dd)

Participant への gRPC 通信の送信でエラーが発生しました。

メッセージの次行から gRPC 通信の Exception 情報を出力します。

aa....aa：Participant の<Service 名>.<ネームスペース名>

bb....bb：Participant のポート番号

cc....cc：ルート XID

トランザクション処理以外の gRPC 通信の場合は*

dd....dd：保守情報

お問い合わせの際に連絡してください。

(S)処理を継続します。

(O) Exception 情報を確認して対策を行うか、Participant などに問題が発生していないかログおよびトレースを確認してください。また、次の Participant のパラメタの設定値を見直してください。

- SQL-Participan を使用する場合：jp.co.Hitachi.soft.hmppcto.sql.pod.service-name
- TCC-Participan を使用する場合：jp.co.Hitachi.soft.hmppcto.tcc.pod.service-name
- TP1-Bridge を使用する場合：tp1.pod.serviceName

〔対策〕

Exception やログおよびトレースの確認結果に従い対応してください。

9.2.2 KFSG70000～KFSG79999

KFSG70000～KFSG79999 のメッセージを次に示します。

KFSG74001-E (Y)M

A new exception was thrown. (message = aa....aa)

処理中に例外が発生しました。

aa....aa：発生した例外の詳細メッセージ。

(S)処理を続行します。

(O)例外の詳細メッセージまたは直後に出力されたメッセージに従ってください。

〔対策〕

なし。

KFSG74002-E (Y)M

A parameter is outside the valid range or was not specified. (parameter = aa....aa, value = bb....bb, reason = cc....cc)

パラメタに設定範囲外の値が指定されている、またはパラメタが指定されていません。

aa....aa：パラメタ名

複数のパラメタが関係する場合は、「;」で区切られて複数表示されます。

bb....bb：パラメタの指定値

指定されていない場合は、「null」が表示されます。複数のパラメタが関係する場合は、「;」で区切られて複数表示されます。

cc....cc：理由コード

VALUE：パラメタの指定値が設定範囲外です。

NULL：必須パラメタが指定されていません。

(S)処理を終了します。

(O)パラメタの指定値を見直し正しい値に変更したあと、再起動してください。

〔対策〕

なし。

KFSG74100-I (Y)M

Initialization of the message logging function will now start.

メッセージログ機能の初期化を開始します。

(S)処理を続行します。

(O)必要ありません。

〔対策〕

なし。

KFSG74101-I (Y)M

The message logging function will now start.

メッセージログ機能を開始します。

(S)処理を続行します。

(O)必要ありません。

〔対策〕

なし。

KFSG74102-E (Y)M

The message logging function failed.

メッセージログ機能の開始に失敗しました。

(S)Pod の起動処理を終了します。

(O)管理者に連絡してください。

〔対策〕

直後に出力するスタックトレースを参照し、メッセージログ機能が起動しなかった要因を除いたあと、Pod を再起動してください。

上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG74103-E (Y)M

The logger could not find the message property file.

ロガーがメッセージプロパティファイルを取得できませんでした。

(S)Pod の起動処理を終了します。

(O)管理者に連絡してください。

〔対策〕

直後に出力するスタックトレースを参照し、メッセージログ機能が起動しなかった要因を除いたあと、Pod を再起動してください。

上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG74104-E (Y)M

Creation of a message failed. (message ID = aa....aa, detail(s) = bb....bb, exception = cc....cc)

メッセ—ジの作成に失敗しました。

aa....aa：入力されたメッセージ ID

指定されていない場合は「null」を表示します。

bb....bb：詳細情報

指定されていない場合は「null」を表示します。複数の詳細情報が関係する場合は「;」で区切って複数表示します。

cc....cc：例外情報

指定されていない場合は「null」を表示します。

(S)処理を続行します。

(O)管理者に連絡してください。

〔対策〕

Pod を再起動してください。

上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG74200-I (Y)M

Initialization of the trace function will now start.

分散トレース機能の初期化を開始します。

(S)処理を続行します。

(O)必要ありません。

〔対策〕

なし。

KFSG74201-I (Y)M

The trace function will now start.

分散トレース機能を開始します。

(S)処理を続行します。

(O)必要ありません。

〔対策〕

なし。

KFSG74202-E (Y)M

The trace function failed.

分散トレース機能の開始に失敗しました。

(S)Pod の起動処理を終了します。

(O)管理者に連絡してください。

〔対策〕

直後に出力するスタックトレースを参照し、メッセージログ機能が起動しなかった要因を除いたあと、Pod を再起動してください。

上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG74203-I (Y)M

The processing to stop the trace function will now start.

分散トレース機能の終了処理を開始します。

(S)処理を続行します。

(O)必要ありません。

〔対策〕

なし。

KFSG74204-I (Y)M

The trace function will now stop.

分散トレース機能を終了します。

(S)処理を続行します。

(O)必要ありません。

〔対策〕

なし。

KFSG74205-W (Y)M

The trace function is disabled.

分散トレース機能が無効です。

(S)処理を続行します。

(O)管理者に連絡してください。

〔対策〕

テスト環境などで Jaeger を使用できない場合などを除き、分散トレース機能を有効にしてください。

パラメタ jp.co.Hitachi.soft.hmppcto.tracing.trace-use を true を設定してください。そのあと、再起動してください。

KFSG74206-I (Y)M

The trace will be sent to jaeger. (instance-id = aa....aa, URL = bb....bb)

分散トレースを Jaeger に送信します。

aa....aa：パラメタ jp.co.Hitachi.soft.hmppcto.tracing.instance-id の設定値
未設定の場合は自動生成された ID です。

bb....bb：Jaeger の endpoint の URL

(S)処理を続行します。

(O)必要ありません。

[対策]
なし。

KFSG74401-I (Y)M

The metrics function is running.

メトリクス機能の起動中です。

(S)処理を続行します。

(O)必要ありません。

[対策]
なし。

KFSG74402-I (Y)M

The metrics function will now start.

メトリクス機能を開始します。

(S)処理を続行します。

(O)必要ありません。

[対策]
なし。

KFSG74403-I (Y)M

The metrics function will now stop.

メトリクス機能を終了します。

(S)メトリクス機能が停止（終了）します。

(O)必要ありません。

〔対策〕

なし。

KFSG74404-I (Y)M

Initialization of the metrics collector timer will now start. (component = aa....aa)

メトリクスコレクタタイマの初期化を開始します。

aa....aa：初期化を実施するコンポーネント名

- Mediator
- SQL-Participant
- TCC-Participant
- TP1-Bridge

(S)処理を続行します。

(O)必要ありません。

〔対策〕

なし。

KFSG74405-I (Y)M

Initialization of the metrics collector timer finished. (component = aa....aa)

メトリクスコレクタタイマの初期化が完了しました。

aa....aa：初期化を実施するコンポーネント名

- Mediator
- SQL-Participant
- TCC-Participant
- TP1-Bridge

(S)処理を続行します。

(O)必要ありません。

〔対策〕

なし。

KFSG74406-I (Y)M

The metrics collector timer was stopped. (component = aa....aa)

メトリクスコレクタタイマを終了しました。

aa....aa：初期化を実施するコンポーネント名

- Mediator
- SQL-Participant
- TCC-Participant
- TP1-Bridge

(S)メトリクス機能が停止（終了）します。

(O)必要ありません。

〔対策〕

なし。

KFSG74407-E (Y)M

Creation of a metrics collector timer failed. (component = aa....aa)

メトリクスコレクタタイマの初期化に失敗しました。

aa....aa：初期化を実施するコンポーネント名

- Mediator
- SQL-Participant
- TCC-Participant
- TP1-Bridge

(S)Pod の起動処理を終了します。

(O)管理者に連絡してください。

［対策］

直後に出力するスタックトレースを参照し、メトリクスコレクタタイマが初期化できなかった要因を除いたあと、Pod を再起動してください。

上記対策を実施しても解決しない場合は、保守員に連絡してください。

KFSG74408-W (Y)M

Failed to clear the metrics collector timer. (component = aa....aa)

メトリクスコレクタタイマの破棄に失敗しました。

aa....aa：初期化を実施するコンポーネント名

- Mediator
- SQL-Participant
- TCC-Participant
- TP1-Bridge

(S)処理を続行します。

(O)必要ありません。

［対策］

なし。

9.2.3 KFSG80000～KFSG89999

KFSG80000～KFSG89999 のメッセージを次に示します。

KFSG80000-I (Y)M

The gRPC server will now start. (port = aa....aa)

gRPC サーバを起動します。

aa....aa：gRPC サービスが使用するポート番号

(S)gRPC サーバの起動処理を実行します。

(O)必要ありません。

［対策］

なし。

KFSG80001-I (Y)M

The processing to initialize the Participant will now start.

Participant の初期化処理を開始します。

(S)Participant の初期化処理を行います。

(O)必要ありません。

〔対策〕

なし。

KFSG80002-I (Y)M

The processing to initialize the Participant finished.

Participant の初期化処理が完了しました。

(S)Participant の初期化処理を終了します。

(O)必要ありません。

〔対策〕

なし。

KFSG80003-I (Y)M

Shutdown of the gRPC server will now start. (port = aa....aa)

gRPC サーバの停止を開始します。

aa....aa：gRPC サーバのポート番号

(S)gRPC サーバを停止します。

(O)必要ありません。

〔対策〕

なし。

KFSG80004-I (Y)M

The Participant has started.

Participant が起動しました。

(S)Participant が起動しました。

(O)必要ありません。

〔対策〕

なし。

KFSG80005-I (Y)M

Shutdown of the gRPC server finished. (port = aa....aa)

gRPC サーバの停止が完了しました。

aa....aa：gRPC サーバのポート番号

(S)Pod を終了します。

(O)必要ありません。

〔対策〕

なし。

KFSG80006-I (Y)M

The SQL-Participant will now shut down.

SQL-Participant の停止処理を行います。

(S)SQL-Participant の停止処理を行います。

(O)必要ありません。

〔対策〕

なし。

KFSG81101-I (Y)M

The processing to initialize the mediator will now start.

Mediator の初期化処理を開始します。

(S)Mediator の初期化処理を行います。

(O)必要ありません。

〔対策〕

なし。

KFSG81102-I (Y)M

The processing to initialize the mediator finished.

Mediator の初期化処理が完了しました。

(S)Mediator の初期化処理を終了します。

(O)必要ありません。

〔対策〕

なし。

KFSG81105-I (Y)M

The mediator will now shutdown.

Mediator の停止処理を行います。

(S)Mediator の停止処理を行います。

(O)必要ありません。

〔対策〕

なし。

KFSG81107-I (Y)M

A consensus log file was created. (file name = aa....aa)

ConsensusLog ファイルの作成を行いました。

aa....aa：ConsensusLog ファイル名

(S)処理を継続します。

(O)必要ありません。

〔対策〕

なし

KFSG81108-I (Y)M

The consensus log file already exists. (file name = aa....aa)

ConsensusLog ファイルが存在しました。

aa....aa：ConsensusLog ファイル名

(S)処理を継続します。

(O)必要ありません。

[対策]
　なし

KFSG81109-I (Y)M

The consensus log file was closed. (file name = aa....aa)

Mediator の停止によって、ConsensusLog ファイルをクローズしました。

aa....aa：ConsensusLog ファイル名

(S)処理を続行します。

(O)必要ありません。

[対策]
　なし。

KFSG82000-I (Y)M

The processing to initialize the TCC participant will now start.

TCC-Participant の初期化処理を開始します。

(S)処理を続行します。

(O)なし。

[対策]
　なし。

KFSG82002-I (Y)M

The processing to initialize the thread pool will now start. (thread pool size = aa....aa)

スレッドプールの初期化処理を開始します。

aa....aa：スレッドプールサイズ

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG82003-I (Y)M

The processing to initialize the TCC participant finished.

TCC-Participant の初期化処理が完了しました。

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG82004-I (Y)M

The TCC participant has started.

TCC-Participant が起動しました。

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG82005-I (Y)M

The TCC participant will now shut down.

TCC-Participant の停止処理を行います。

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG82006-I (Y)M

The initialization process for the thread pool that is used for communication with the mediator will now start. (jp.co.Hitachi.soft.hmppcto.mediator-process-management.thread-pool-size =

aa....aa, jp.co.Hitachi.soft.hmppcto.mediator-communication.thread-pool-size = bb....bb, jp.co.Hitachi.soft.hmppcto.mediator-communication.queue-size = cc....cc)

Mediator と通信するために使用するスレッドプールの初期化処理を開始します。

aa....aa：Mediator へ送信を行うスレッドを管理するスレッドが使用するスレッドプールサイズ

bb....bb：Mediator へのトランザクション決着要求スレッドおよび Mediator 受信スレッドが使用するスレッドプールサイズ

cc....cc：Mediator へのトランザクション決着要求スレッドおよび Mediator 受信スレッドが使用するキューサイズ

(S)処理を続行します。

(O)なし。

[対策]
なし。

KFSG82007-I (Y)M

The initialization process for the thread pool will now start. (aa....aa = bb....bb, cc....cc = dd....dd)

スレッドプールの初期化処理を開始します。

aa....aa：スレッドプールサイズを指定したパラメタ名

- SQL-Participant の場合：jp.co.Hitachi.soft.hmppcto.sql.participant-processing.thread-pool-size
- TCC-Participant の場合：jp.co.Hitachi.soft.hmppcto.tcc.participant-processing.thread-pool-size
- TP1-Bridge の場合：jp.co.Hitachi.soft.hmppcto.tp1.participant-processing.thread-pool-size
- Mediator の場合：jp.co.Hitachi.soft.hmppcto.grpc.server.thread-pool-size、jp.co.Hitachi.soft.hmppcto.mediator.judge.thread-pool-size、jp.co.Hitachi.soft.hmppcto.consensus-log.io.thread-pool-size または jp.co.Hitachi.soft.hmppcto.participant.sender.thread-pool-size

bb....bb：スレッドプールサイズを指定したパラメタの値

cc....cc：キューサイズを指定したパラメタ名

- SQL-Participant の場合：jp.co.Hitachi.soft.hmppcto.sql.participant-processing.queue-size
- TCC-Participant の場合：jp.co.Hitachi.soft.hmppcto.tcc.participant-processing.queue-size
- TP1-Bridge の場合：jp.co.Hitachi.soft.hmppcto.tp1.participant-processing.queue-size

- Mediator の場合：jp.co.Hitachi.soft.hmppcto.grpc.server.queue-size、
  jp.co.Hitachi.soft.hmppcto.mediator.judge.queue-size、
  jp.co.Hitachi.soft.hmppcto.consensus-log.io.queue-size または
  jp.co.Hitachi.soft.hmppcto.participant.sender.queue-size

dd....dd：キューサイズを指定したパラメタの値

(S)処理を続行します。

(O)なし。

〔対策〕
　なし。

KFSG82100-I (Y)D

The transaction health check processing will now start. (retry count = aa....aa, retry interval seconds = bb....bb, connect timeout seconds = cc....cc, read timeout seconds = dd....dd, http port = ee....ee)

トランザクションヘルスチェック処理を開始します。

aa....aa：リトライ回数

bb....bb：リトライ間隔秒数

cc....cc：接続タイムアウト秒数

dd....dd：応答タイムアウト秒数

ee....ee：HTTP ポート番号

(S)処理を続行します。

(O)なし。

〔対策〕
　なし。

KFSG82101-I (Y)D

The transaction health check processing finished.

トランザクションヘルスチェック処理が完了しました。

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG82102-I (Y)D

The processing that waits for active transactions will now start. (retry count = aa....aa, retry interval seconds = bb....bb, connect timeout seconds = cc....cc, read timeout seconds = dd....dd, http port = ee....ee)

未決着トランザクション待機処理を開始します。

aa....aa：リトライ回数

bb....bb：リトライ間隔秒数

cc....cc：接続タイムアウト秒数

dd....dd：応答タイムアウト秒数

ee....ee：HTTP ポート番号

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG82103-I (Y)D

The processing that waits for active transactions finished.

未決着トランザクション待機処理が完了しました。

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG82104-I (Y)D+M

The dependency check processing will now start. (retry count = aa....aa, retry interval seconds = bb....bb, connect timeout seconds = cc....cc, read timeout seconds = dd....dd, targets = ee....ee)

デプロイ依存関係チェック処理を開始します。

aa....aa：リトライ回数

bb....bb：リトライ間隔秒数

cc....cc：接続タイムアウト秒数

dd....dd：応答タイムアウト秒数

ee....ee：ターゲット Kubernetes オブジェクト群

(S)処理を続行します。

(O)なし。

[対策]
　なし。

KFSG82105-I (Y)D+M

The dependency check processing finished.

デプロイ依存関係チェック処理が完了しました。

(S)処理を続行します。

(O)なし。

[対策]
　なし。

KFSG83100-I (Y)M

The orchestrator initialization processing is waiting for a mediator to start. (retry-count = aa....aa, interval-in-millis = bb....bb)

Orchestrator の初期化処理で Mediator の起動を待ち合わせます。

aa....aa：Mediator が起動しているかのチェックをリトライする回数（mediator.discovery-retry-count 設定値）

bb....bb：Mediator が起動しているかのチェックをリトライする間隔（単位：ミリ秒）（mediator.discovery-interval-in-millis 設定値）

(S)Mediator が起動しているかのチェックを aa....aa の回数、bb....bb の間隔でリトライします。

(O)なし。

〔対策〕

なし。

KFSG83101-I (Y)M

Initialization of the orchestrator finished.

Orchestrator の初期化が終了しました。

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG83102-I (Y)M

Initialization of the entity service finished.

Entity-Service の初期化が終了しました。

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG83103-I (Y)C

Initialization of the tp1 converter finished.

TP1-Converter の初期化が終了しました。

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

10 システム運用に必要となる機能

10.1　概要

10.1.1　取得する情報および各情報の紐づけ方法

HMP-PCTO のトラブルシュート機能や連携製品で取得する情報、およびこれらの情報の紐づけ方法について説明します。

(1)　取得する情報

次の要件を満たすため、HMP-PCTO は最低限必要な情報取得先として次の表の情報を取得します。

- ユーザによる障害検知ができること
- どこで問題が起こっているか障害個所を調査できること
- 何が起こっているかを調査できること

表 10–1　HMP-PCTO の情報取得先

項番	情報取得先 名称	説明
1	メッセージログ	ユーザが障害検知するため、また、何が起こっているかを調査するため、詳細な情報を出力します。発生したイベントをタイムスタンプとともにテキストで取得します。
2	分散トレース	障害個所（エラー個所およびレスポンス遅延個所）の特定を行うため、サービス要求や製品機能間の通知に関するアクティビティを、関連づけ可能な形で取得します。各サービスでの処理時間、呼び出し関係と、付随する情報（種別など）を取得します。 分散トレースのイメージについては次の個所を参照してください。 • SQL-Participant 使用時：「図 10–1　分散トレースのイメージ（SQL-Participant を使用する場合）」 • TCC-Participant 使用時：「図 10–2　分散トレースのイメージ（TCC-Participant を使用する場合）」 • TP1-Bridge 使用時：「図 10–3　分散トレースのイメージ（TP1-Bridge を使用する場合）」
3	メトリクス	レスポンス遅延が発生しているときに何が起こっているかを調査するため、システムの利用状況（CPU 使用率など）、Java VM の利用状況（メモリ使用量など）、および HMP-PCTO の性能を評価するための指標となる数値（トランザクション数など）を取得します。

HMP-PCTO における「ログ」は、ユーザに何が起こっているかを知らせる情報を文章として出力し、「トレース」はユーザにどこで問題が発生しているかを追跡するための情報を出力します。

分散トレースのイメージを次の図に示します。この図は分散トレースのイメージを表しているため、実際の処理シーケンスとは異なります。

図 10–1　分散トレースのイメージ（SQL-Participant を使用する場合）

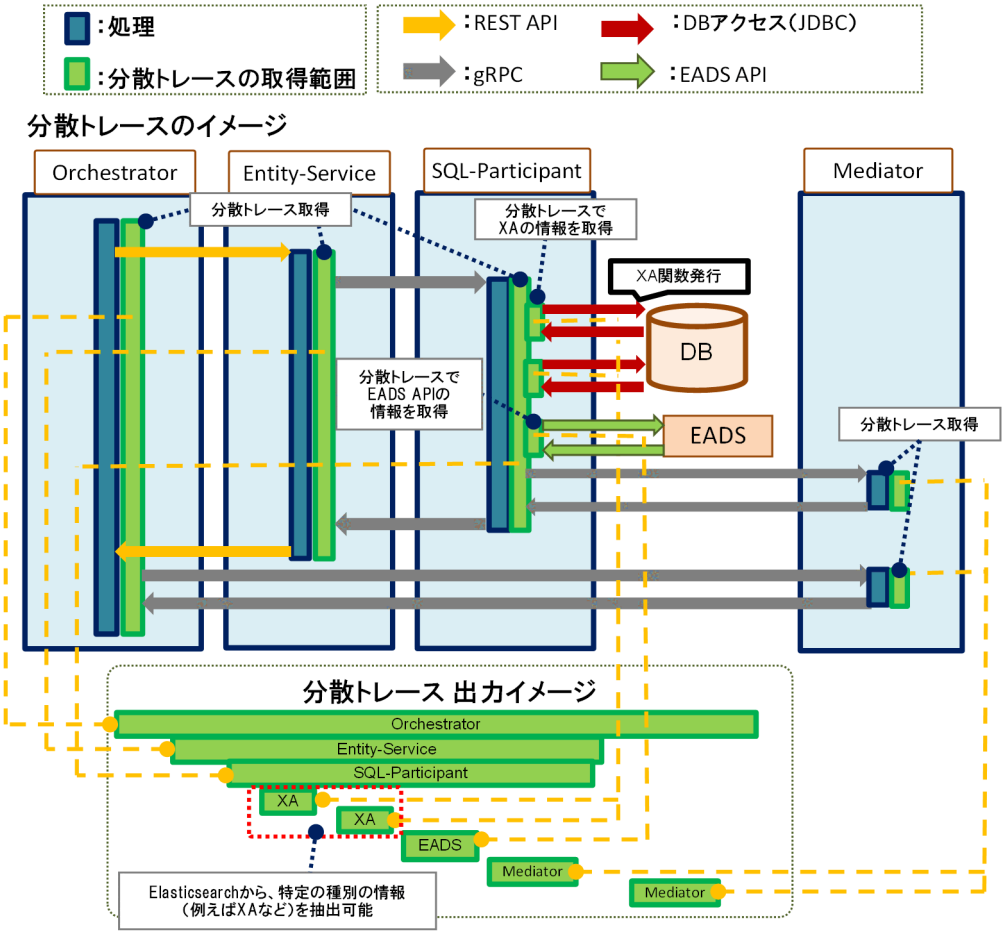


図 10–2　分散トレースのイメージ（TCC-Participant を使用する場合）

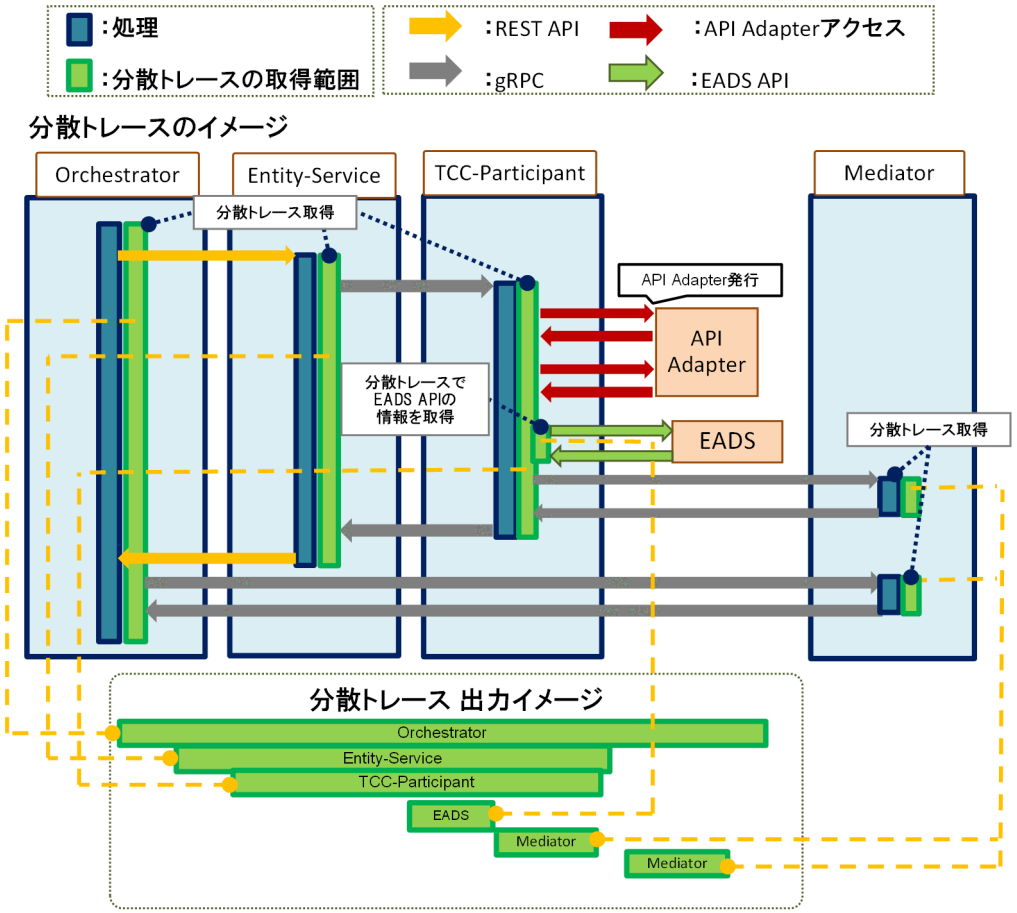
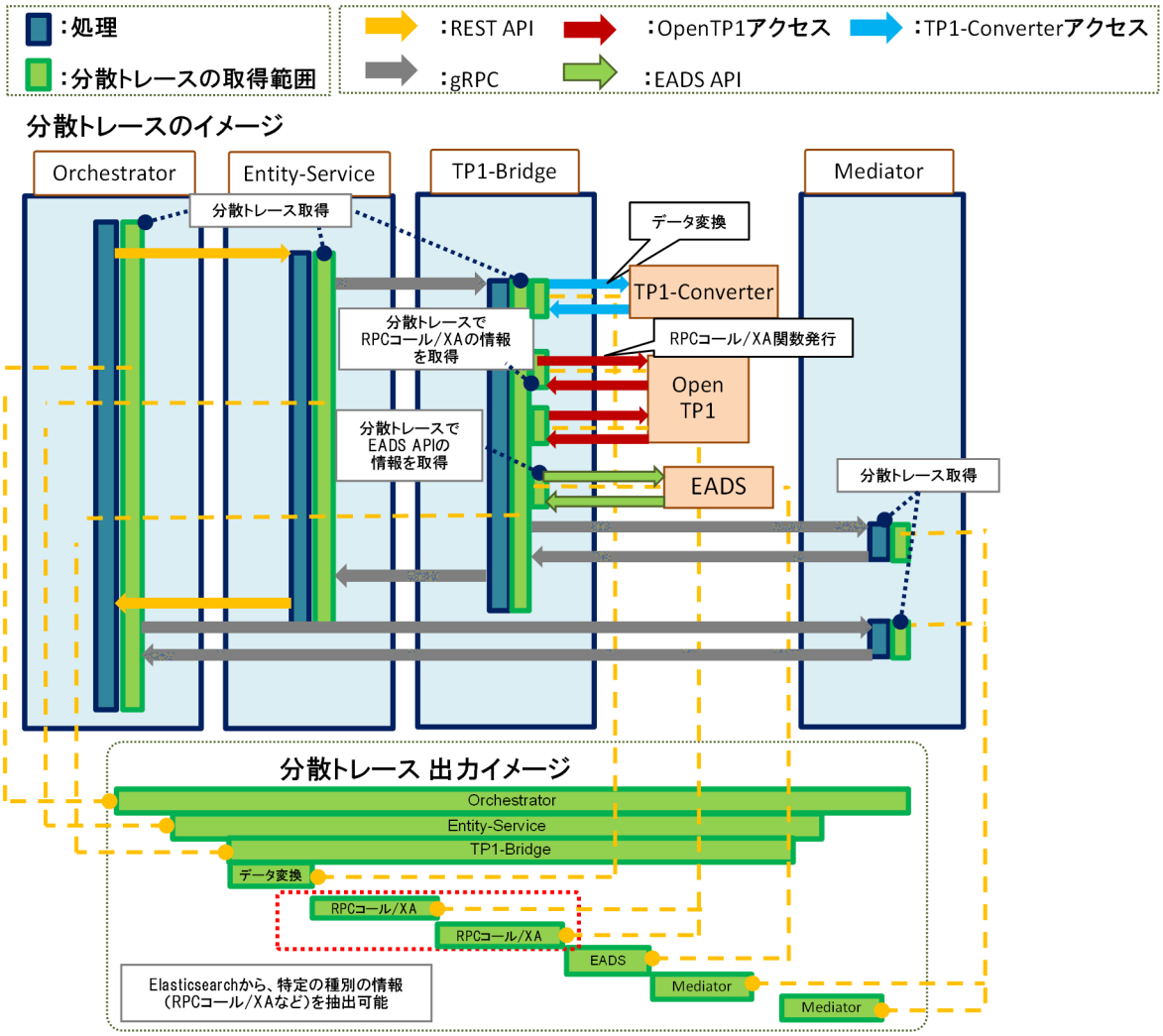


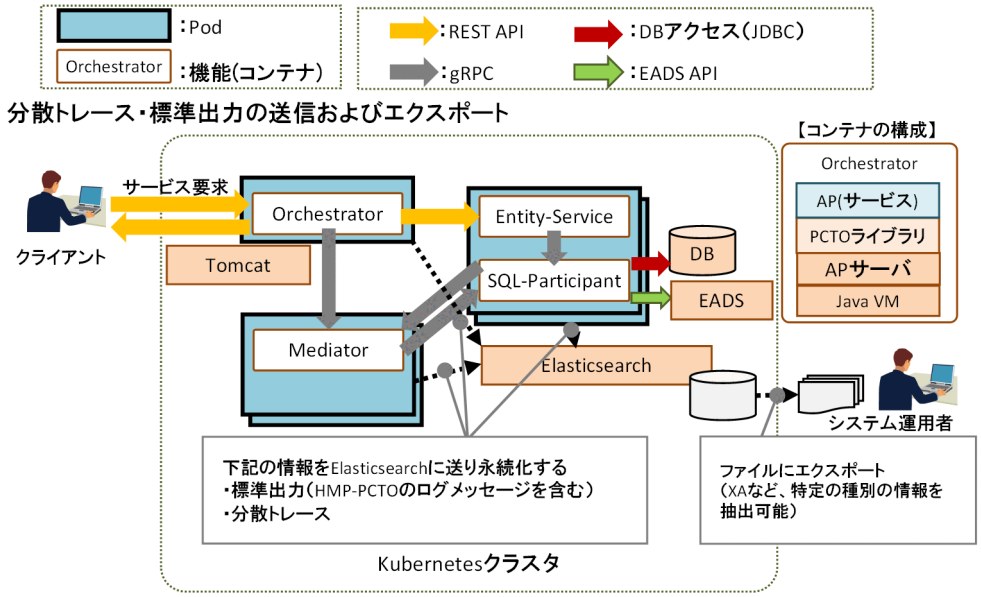
図 10–3　分散トレースのイメージ（TP1-Bridge を使用する場合）



要件「情報を永続化できること」を満たすため、HMP-PCTO は分散トレースおよび標準出力の情報を Elasticsearch に送り永続化します。また、メトリクスの情報を永続化します。メトリクスの情報についても Elasticsearch に送り永続化します。障害が発生するケースに備えて、Elasticsearch はクラスタ構成を可能とします。クラスタは Elasticsearch クラスタを制御するマスターノード群と情報を保持するデータノード群で構成されます。本バージョンではこの構成だけをサポートします。

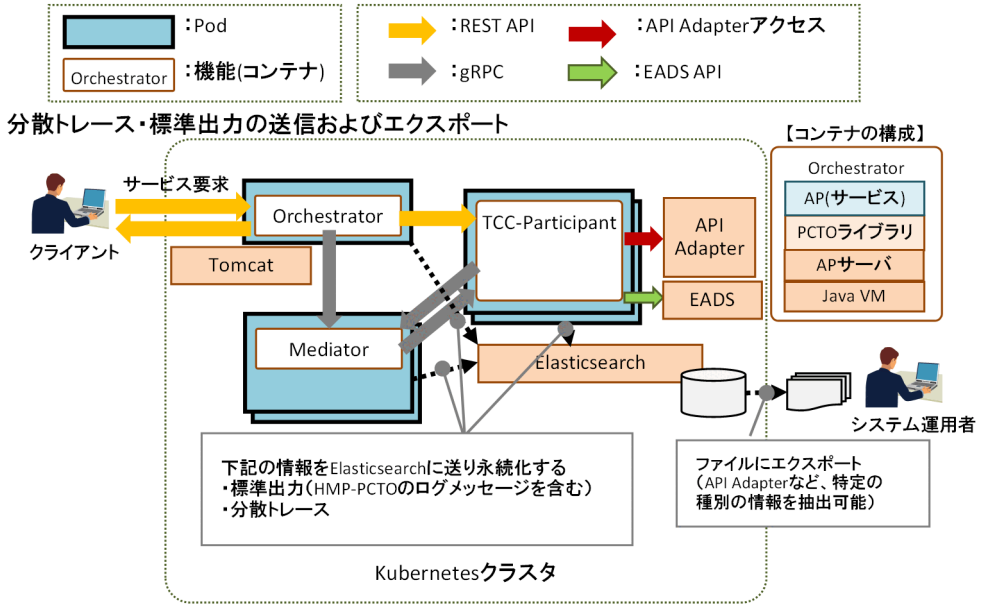
分散トレースの情報は、ファイルにエクスポートする際に特定の種別の情報を抽出できます。例えば、SQL-Participant 使用時の場合、XA 関数発行時のトレースに絞って出力することで、1 トランザクション当たりの XA 関数の処理時間の合計を抽出できます。SQL-Participant 使用時の場合の、HMP-PCTO 分散トレース・標準出力の送信およびエクスポートのイメージを次の図に示します。

図 10–4　HMP-PCTO 分散トレース・標準出力の送信およびエクスポート（SQL-Participant を使用する場合）



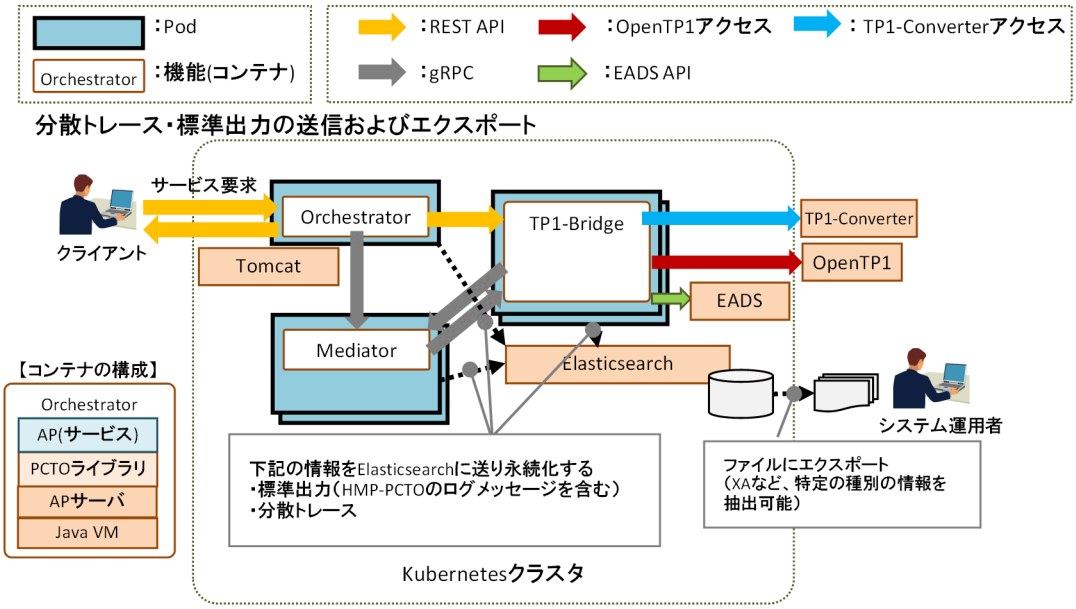
TCC-Participant 使用時の場合、API Adapter 発行時のトレースに絞って出力することで、1 トランザクション当たりの処理時間の合計を抽出できます。TCC-Participant を使用する場合の、HMP-PCTO 分散トレース・標準出力の送信およびエクスポートイメージを次の図に示します。

図 10–5　HMP-PCTO 分散トレース・標準出力の送信およびエクスポート（TCC-Participant を使用する場合）



TP1-Bridge 使用時の場合、RPC コール/XA 関数発行時のトレースに絞って出力することで、1 トランザクション当たりの処理時間の合計を抽出できます。TP1-Bridge を使用する場合の、HMP-PCTO 分散トレース・標準出力の送信およびエクスポートイメージを次の図に示します。

図 10-6　HMP-PCTO 分散トレース・標準出力の送信およびエクスポート（TP1-Bridge を使用する場合）



要件「システムへのリソースおよび性能の影響を最小限にすること」を満たすため、「表 10-1　HMP-PCTO の情報取得先」以外の情報は、連携製品側で出力される情報を使用します。HMP-PCTO のトラブルシュートで利用する連携製品の情報は次のとおりです。

なお、各連携製品側の名称や機能などの最新の情報については、各製品のマニュアルを参照してください。

表 10-2　トラブルシュートで利用する連携製品の情報

項番	名称	説明
1	PostgreSQL のログ情報	XA 関数に対応した Postgre のログ情報をトラブルシュートに利用します。
2	HiRDB のトレース・ログ情報	XA 関数に対応した HiRDB のトレース・ログ情報をトラブルシュートに利用します。
3	SQL Server のトレース・ログ情報	SQL Server のトレース・ログ情報をトラブルシュートに利用します。
4	EADS の各種トラブルシュートファイル	EADS サーバでレスポンスの低下が起こっている場合、ログ情報のエラーメッセージや統計情報をトラブルシュートに利用します。トライアル版では利用できません。
5	Tomcat のアクセスログ	Orchestrator、Entity-Service または Entity-Module の REST API 通信についてヘッダ情報をトラブルシュートに利用します。
6	uCosminexus Application Runtime with Java for Spring Boot のログ	Tomcat に問題（スローダウンによるレスポンス遅延など）が発生した際にトラブルシュートに利用します。
7	uCosminexus Service Platform のトレース・ログ情報	TP1 連携機能の使用時に、TP1-Converter、および uCosminexus Service Platform のトラブルシュートに利用します。
8	OpenTP1 のトレース・ログ情報	TP1 連携機能の使用時に、OpenTP1 のトラブルシュートに利用します。

項番	名称	説明
9	TP1/Client/J のトレース情報	TP1 連携機能の使用時に、TP1/Client/J のトラブルシュートに利用します。

HMP-PCTO の取得先ごとに、調査に必要な情報は次のとおりです。

表 10–3　取得先ごとに調査で必要な情報

項番	情報取得先 名称	取得する情報
1	メッセージログ	• エラー、警告の情報 • スタックトレースの情報 ＜例＞ 　• EADS API エラー時のキー名・実行結果の情報 　• ConsensusLog アクセス時のエラーの情報 　• gRPC 通信失敗時の情報
2	分散トレース	各サービスでの処理時間、呼び出し関係と、付随する情報（種別など） ＜例＞ 　• XA 関数名、処理結果、DB のログと紐づけするための情報（ブランチ XID） 　• EADS へのアクセスで遅延発生した場合の処理時間、処理範囲 　• ConsensusLog へのアクセス時に遅延発生した場合の処理時間、処理範囲
3	メトリクス	• システムの利用状況 • Java VM の利用状況 • HMP-PCTO の性能を評価するための指標 ＜例＞ 　• CPU 使用率 　• メモリ使用量 　• トランザクション数
4	PostgreSQL のログ情報	ログ情報
5	HiRDB のトレース・ログ情報	• ログ情報 • HiRDB サーバのトランザクションの状態表示の結果 • SQL トレース
6	SQL Server のトレース・ログ情報	• クエリトレース • ログ情報
7	EADS の各種トラブルシュートファイル	• ログ情報（異常時） • 統計情報（レスポンス低下時） 上記のファイルは、トライアル版では取得できません。
8	Tomcat のアクセスログ	REST API 通信のヘッダ情報（HMP-PCTO の情報と紐づけ可能なブランチ XID の情報が記録されている）
9	uCosminexus Application	uCosminexus Application Runtime with Java for Spring Boot のトレース情報（HTTP リクエストのトレース）

項番	情報取得先 名称	取得する情報
	Runtime with Java for Spring Boot のログ	uCosminexus Application Runtime with Java for Spring Boot のトレース情報として HTTP リクエスト URI が出力されますが、HMP-PCTO の情報との紐づけは、Tomcat のアクセスログの情報を用いて行います。
10	uCosminexus Service Platform のトレース・ログ情報	• J2EE アプリケーションのユーザログ • NIO HTTP サーバのアクセスログ • Web サーブレットログ • JavaVM スタックトレース情報 • JavaVM の GC ログ • JavaVM ログ（JavaVM ログファイル）
11	OpenTP1 のトレース・ログ情報	リアルタイム出力機能
12	TP1/Client/J のトレース情報	• デバッグトレース • エラートレース • メソッドトレース • UAP トレース • データトレース エラートレース、メソッドトレース、UAP トレース、およびデータトレースはデフォルトでは出力されません。アプリケーションのデバッグ時や、HMP-PCTO が出力するメッセージだけでは問題の解決が困難な場合に、トレースを取得してください。

(2) 各情報の紐づけ方法

サポート部署が調査する上で障害原因の究明を行うために、各情報を紐づける必要があります。

HMP-PCTO で取得するメッセージログ・分散トレース、および連携製品で取得する情報の紐づけ情報として、ブランチ XID の情報を使用します（ブランチ XID にはルート XID の情報を含んでいます）。

それぞれの紐づけ方法について説明します。

分散トレースの情報から障害（エラーおよびレスポンス遅延）個所を切り分け、その後、何が起こっていたかをメッセージログなどの情報から抽出するための紐づけ情報として、次の分散トレース情報でブランチ XID を取得します。

- HMP-PCTO の各コンポーネント間の REST API 通信、および gRPC 通信

なお、Orchestrator から Mediator への通信、および Mediator から Participant への通信（分散合意結果通知）にはブランチ XID がないため取得できるのはルート XID だけとなりますが、ブランチ XID にはルート XID の情報を含むため、ルート XID の情報で紐づけできます。

ブランチ XID は、次に示す表の項番 1～3 の情報でも取得します。また項番 4 以降は、連携製品側の情報として記録されています。

表 10-4　ブランチ XID を取得する情報、および連携製品のトラブルシュート情報の紐づけ

項番	情報			説明
1	ブランチ XID を取得する情報	HMP-PCTO のメッセージログ（エラー時）		異常発生時に HMP-PCTO のメッセージログに出力されるエラーメッセージに、ブランチ XID を含めます。
2		HMP-PCTO で XA 関数発行時に取得する分散トレース情報		XA 関数とブランチ XID の情報を取得します。
3		HMP-PCTO で EADS API 発行時に取得する分散トレース情報		EADS API のキー情報と ConsensusID を取得します。ConsensusID はルート XID の 16 進数表記のため、ブランチ XID と紐づけできます。 なお、トライアル版では EADS API 発行時に取得する分散トレース情報は取得できません。
4	連携製品のトラブルシュート情報（共通）	Tomcat アクセスログ		REST API 通信とブランチ XID の情報が記録されています。
5		EADS トラブルシュート情報		ブランチ XID の情報などのキーとなる情報は記録されません。このため、時刻をキーにして EADS トラブルシュート情報の調査をします。 なお、トライアル版では EADS トラブルシュート情報は取得できません。
6	連携製品のトラブルシュート情報（SQL-Participant 使用時）	PostgreSQL のログ情報		メッセージに XA 関数とブランチ XID の情報が記録されています。
7		HiRDB のトレース・ログ情報	ログ情報	トランザクションの障害時に出力されるメッセージにブランチ XID の情報が記録されています。
8			HiRDB サーバのトランザクションの状態表示の結果	トランザクション識別子として、ブランチ XID が記録されています。
9			SQL トレース	トランザクション識別子として、ブランチ XID が記録されています。
10		SQL Server のトレース・ログ情報	クエリトレース	ブランチ XID の情報などのキーとなる情報は記録されません。
11			ログ情報	ブランチ XID の情報などのキーとなる情報は記録されません。
12	連携製品のトラブルシュート情報（TCC-Participant 使用時）	API Adapter アクセスログ情報		API Adapter アクセスとブランチ XID の情報が記録されています。
13	連携製品のトラブルシュート情報（TP1-Bridge 使用時）	uCosminexus Service Platform のトレース・ログ情報	J2EE アプリケーションのユーザログ	TP1-Converter が出力するメッセージ内に、ブランチ XID の情報が記録されています。

項番	情報			説明
14			NIO HTTP サーバのアクセスログ	ログ中の時刻情報をキーに、そのほかのトラブルシュート情報と紐づけます。
15			Web サーブレットログ	ログ中の時刻情報をキーに、そのほかのトラブルシュート情報と紐づけます。
16			JavaVM スタックトレース情報	スタックトレース情報を参照し、例外が発生した要因および例外の発生個所を調査します。
17			JavaVM の GC ログ	統計情報の確認のために取得する情報です。通信遅延などの現象が発生した場合、その現象が発生した時刻およびログの時刻情報を基にトラブルシュートを実施します。
18			JavaVM ログ (JavaVM ログファイル)	ログ中の時刻情報をキーに、そのほかのトラブルシュート情報と紐づけます。
19		OpenTP1 のトレース・ログ情報	リアルタイム出力機能	時刻情報をキーにし、そのほかのトラブルシュート情報と紐づけます。
20		TP1/Client/J のトレース情報	デバッグトレース	トレース情報は、TP1-Brdige Pod の標準出力に出力されます。HMP-PCTO のメッセージ、メッセージが出力された時刻情報、およびトレースの時刻情報を基に、トラブルシュートを実施します。
21			エラートレース	時刻情報およびスレッド名をキーにし、そのほかのトラブルシュート情報と紐づけます。
22			メソッドトレース	時刻情報およびスレッド名をキーにし、そのほかのトラブルシュート情報と紐づけます。
23			UAP トレース	時刻情報およびスレッド名をキーにして、そのほかのトラブルシュート情報と紐づけます。
24			データトレース	時刻情報およびスレッド名をキーにし、そのほかのトラブルシュート情報と紐づけます。

「表 10-4　ブランチ XID を取得する情報、および連携製品のトラブルシュート情報の紐づけ」で述べた、分散トレースとそのほかの情報との紐づけのイメージについて次の図に示します。図中の番号は、表の項番に対応しています。

図 10–7　HMP-PCTO 各情報の紐づけのイメージ（SQL-Participant を使用する場合）

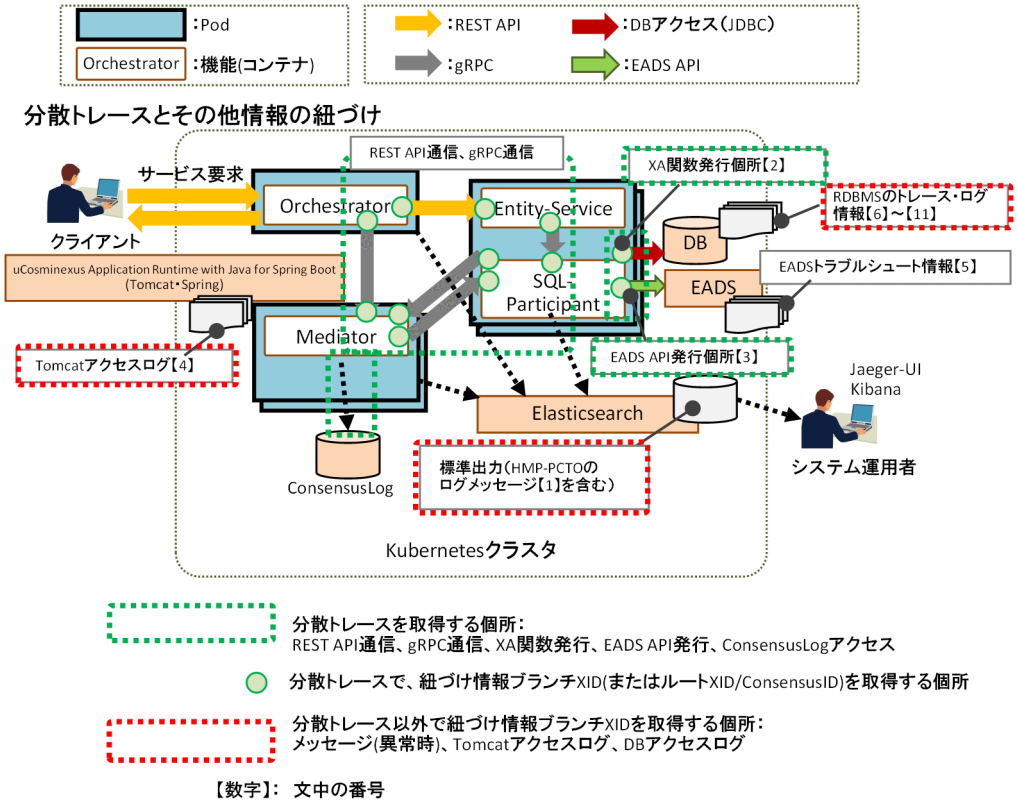


図 10–8　HMP-PCTO 各情報の紐づけのイメージ（TCC-Participant を使用する場合）

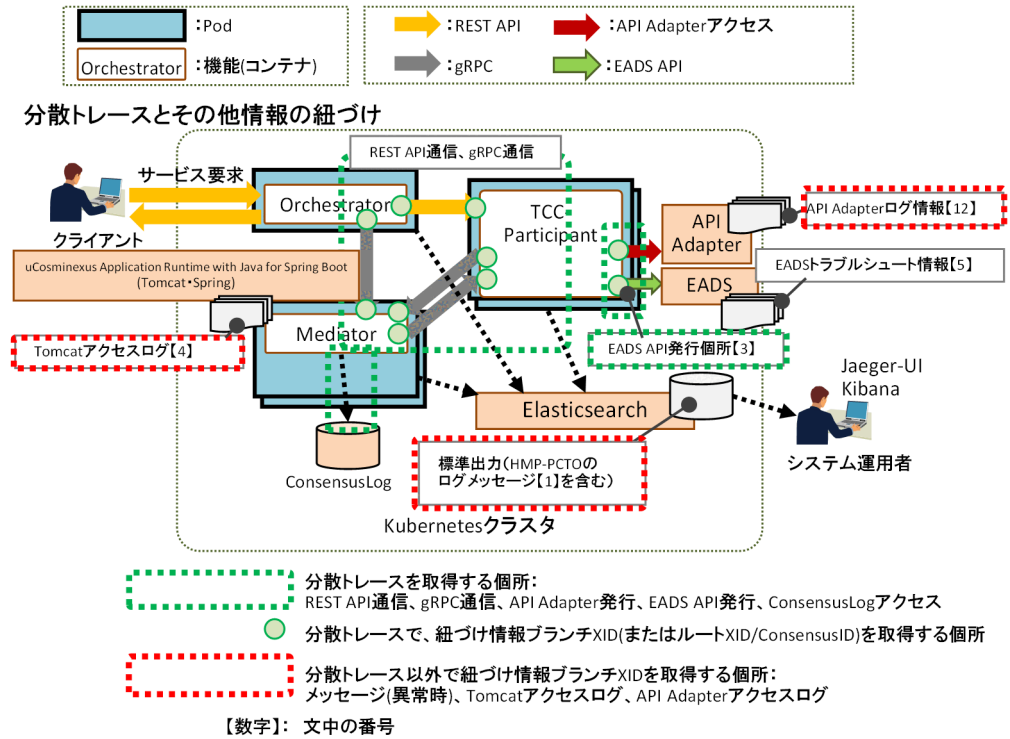
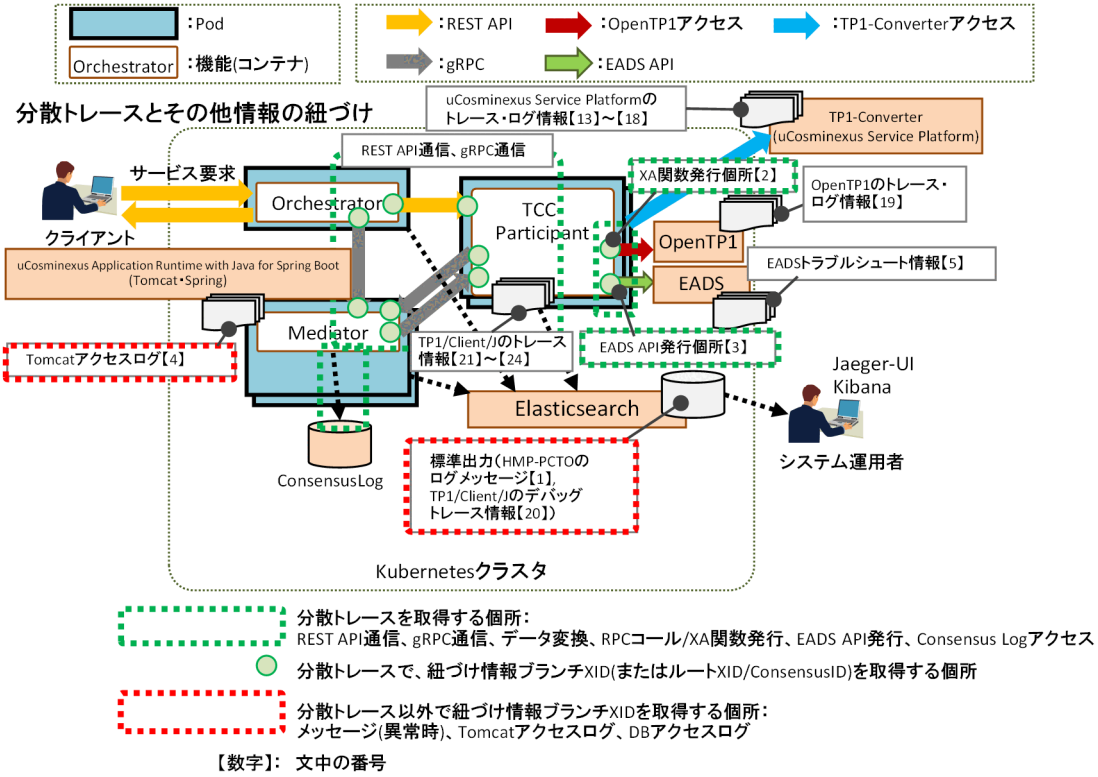


図 10–9　HMP-PCTO 各情報の紐づけのイメージ（TP1-Bridge を使用する場合）



ユーザは、HMP-PCTO が取得する分散トレースの情報と、連携製品の情報をブランチ XID で紐づけて調査できます。なお、HMP-PCTO が取得する各分散トレース情報（「図 10–7　HMP-PCTO 各情報の紐づけのイメージ（SQL-Participant を使用する場合）」、「図 10–8　HMP-PCTO 各情報の紐づけのイメージ（TCC-Participant を使用する場合）」、および「図 10–9　HMP-PCTO 各情報の紐づけのイメージ（TP1-Bridge を使用する場合）」の緑枠部分）は、分散トレース固有の紐づけ情報（トレース ID）によって関連づけられるため、ユーザは Jaeger の Web UI などで一貫した情報として見ることができます。

(3)　性能評価の対象と取得する情報

HMP-PCTO を使用した分散システムで、ユーザが性能を評価する対象は次のとおりです。

表 10–5　性能評価の対象

項番	評価対象		説明
1	システムの利用状況		Kubernetes などのコンテナ環境における性能 ＜例＞ CPU 使用率
2	Java VM の利用状況		Java プロセス固有の性能 ＜例＞ Java Heap、GC
3	HMP-PCTO の性能を評価するための指標	gRPC 通信の性能	HMP-PCTO の各コンポーネント間の gRPC 通信の性能 ＜例＞ gRPC 通信の処理時間
4		非同期処理の性能	別スレッドで実行する非同期タスクの性能

項番	評価対象		説明
			＜例＞非同期タスクの実行回数・時間、未実行の（キューイングされている）数
5		機能固有の稼働統計情報	HMP-PCTO のコンポーネントごとに固有で取得する稼働統計情報 ＜例＞ Mediator の例：ProposalLog、ConsesnusLog の状態ごとの件数 Participant の例：トランザクションの件数・時間 TP1-Bridge の例：Spring Boot 組み込み Tomcat のスレッド数

要件「ユーザが HMP-PCTO の性能を評価するための情報を取得すること」を満たすため、HMP-PCTO は、最低限必要な情報取得先として次の表の情報を取得します。

表 10–6　HMP-PCTO の情報取得先

項番	情報取得先 名称	説明
1	メトリクス	HMP-PCTO を使用した分散システムでユーザが性能を評価するため、システムの利用状況（CPU 使用率など）、Java VM の利用状況（メモリ使用量など）、および HMP-PCTO の性能を評価するための指標となる数値（トランザクション数など）を取得します。

10.2　メッセージログ

10.2.1　概要

HMP-PCTO は、稼働状況管理、障害検知、および障害発生時の切り分けなどを目的としてメッセージログを出力します。メッセージログ出力の契機は、アプリケーションの稼働状況の変化、障害発生などです。

HMP-PCTO は、ロギングライブラリを使用して、メッセージログを標準出力します。ユーザは、HMP-PCTO が出力するメッセージログを監視し、出力されたメッセージログに従って適切に対応する必要があります。

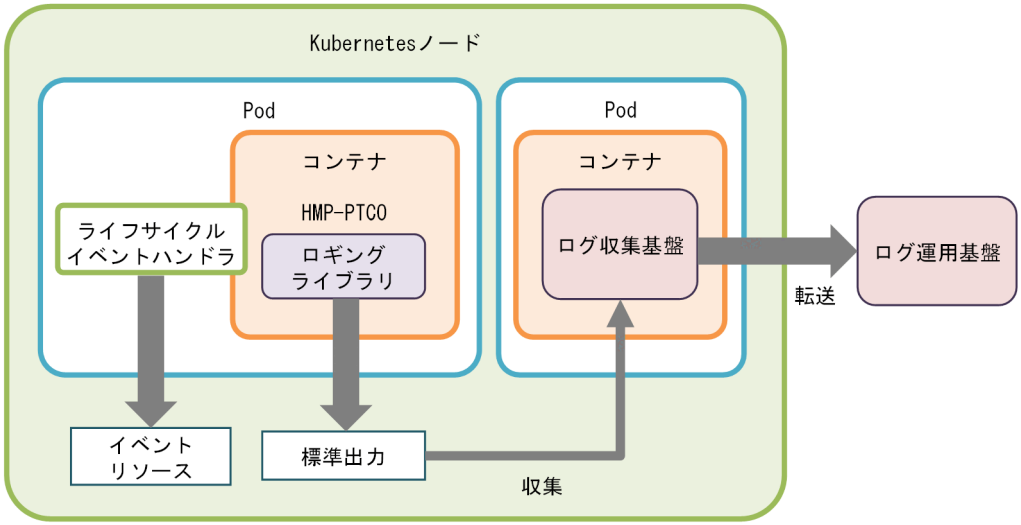
標準出力の内容は、ノード上にログ収集基盤を使用したコンテナを作成し、標準出力の内容を収集します。収集したログは、ログ運用基盤などに転送され、永続化および可視化されます。障害発生時は、必要に応じて保存されたログをエクスポートし、障害調査者に送付してください。詳細については、「10.6　HMP-PCTO の障害調査」を参照してください。

また、Kubernetes のコンテナのライフサイクルイベントハンドラでは Mediator の起動・停止に関するイベントがメッセージログとして出力されます。出力された内容は Kubernetes のイベントリソースに保存されます。ユーザは、必要に応じてイベントリソースのメッセージログを確認し、適切に対応する必要があります。

なお、ログ収集基盤は、イベントソースに保存されたメッセージログを収集しません。必要に応じて、永続化する必要があります。詳細については、「10.6　HMP-PCTO の障害調査」を参照してください。

メッセージログの構成を次の図に示します。なお、メッセージログの出力形式は、各コンポーネントで共通です。

図 10-10　メッセージログの構成



HMP-PCTO では、次に示すロギングライブラリを使用します。

- Logback

重要

- ユーザアプリケーションで使用するロギングライブラリ

  ユーザアプリケーションでロギングライブラリを使用する場合、HMP-PCTO の使用するロギングライブラリと記載が一致している必要があります。一致していない場合は、メッセージログが正常な出力フォーマットで出力されないか、またはメッセージログが出力されないことがあります。

- HMP-ADIF、ユーザアプリケーションのログ出力形式

  HMP-ADIF、またはユーザアプリケーションでログを出力する場合、ログの確認、永続化、および可視化の利便性の観点から、HMP-PCTO のメッセージログと同様に、標準出力への出力を推奨します。

HMP-PCTO では、次に示すログ収集基盤をサポートしています。

- Filebeat

また、HMP-PCTO では、次に示すログ運用基盤を推奨しています。

- ElasticStack（Logstash および Elasticsearch）

ログ運用基盤の設定・運用方法については、次の説明を参照してください。

- 「3.5　共有ファイルシステムでの作業」
- 「3.6　Kubernetes マニフェスト作成」
- 「3.7　Helm チャートのカスタマイズ」
- 「3.8　HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションの運用」
- 「3.9　ログの取得および監視の設定」

10.2.2　取得方針

HMP-PCTO は、アプリケーションの稼働状況の変化、障害発生などを契機に、メッセージ ID を付与したメッセージを出力します。

メッセージは、ログレベル「ERROR」、「WARN」、および「INFO」で出力されます。メッセージの説明および対策については、「9.　メッセージ」を参照してください。

10.3　分散トレース

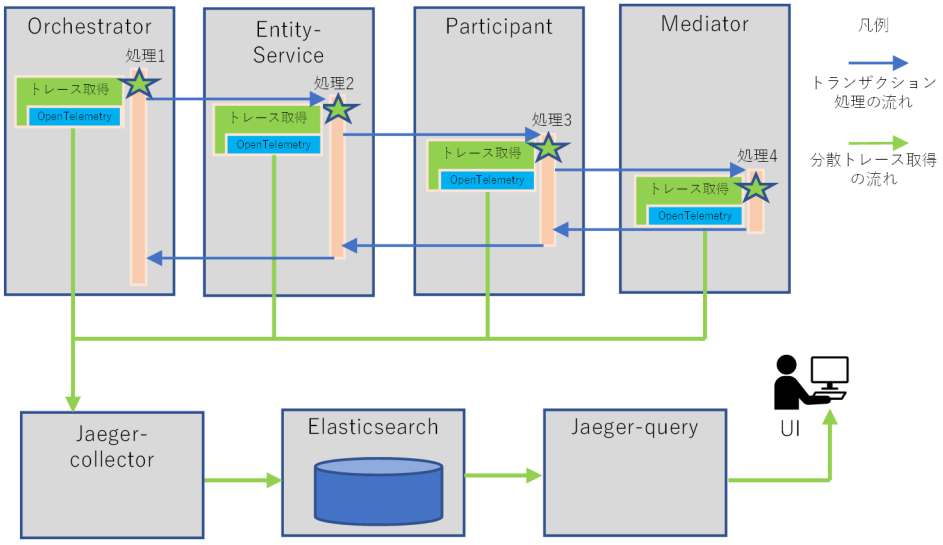
10.3.1　概要

HMP-PCTO は 1 つのトランザクション内で、Orchestrator、Mediator、Entity-Service または Entity-Module、Participant などのコンポーネントがお互いに通信し動作しており、処理の流れが複雑です。HMP-PCTO は、トランザクションの開始から終了までの間の各コンポーネントでの処理の情報（開始時刻や実行時間など）を関連づけて、分散トレース情報として取得します。障害発生時に分散トレース情報を確認することでトランザクションの処理の複雑な流れを把握でき、問題のあった処理の特定を容易化します。

分散トレースは HMP-PCTO が OpenTelemetry を使用して取得し、分散トレース基盤である Jaeger に送信します。Jaeger 単体ではデータの永続化の機能を持たないため、データの永続化が必要です。データを永続化しない場合、分散トレースの情報が失われ、HMP-PCTO 利用者およびサポート部署共に分散トレースを使用しての調査ができません。そのため分散トレースの構成は次のようになります。

分散トレースの永続化手段として Elasticsearch を使います。HMP-PCTO の各コンポーネントの処理で OpenTelemetry を使用してトレース情報を取得し、Jaeger-collector にトレース情報を送信します。Jaeger-collector は Elasticsearch にデータを送信しデータを永続化します。分散トレースを参照する際は Jaeger の Web UI 上で参照します。この時 Jaeger-query が Elasticsearch からトレース情報を取得し、Jaeger の Web UI 上に表示します。Jaeger の Web UI 上でトランザクションの一覧、およびトランザクションごとの処理の流れを確認できます。次に示す図は SQL-Participant を使用する場合です。TCC-Participant を使用する場合、Entity-Service を Entity-Module と読み替えてください。

図 10–11　分散トレースの概要



重要

分散トレース機能で付与するヘッダ

HMP-PCTO では、Span の親子関係やサンプリング有無などの情報を各コンポーネントに伝搬させるために、ほかのコンポーネントへのリクエストに特定のヘッダを付与することで、分散トレーシングを実現します。

なお、ヘッダの情報が変更されると、正しい情報が伝搬できなくなります。そのため、ほかの製品で同様のヘッダが付与されないようにしてください。

付与するヘッダを次に示します。

- ヘッダ名が traceparent の場合
  トレース ID、スパン ID、およびサンプリング有無が付与されます。
- ヘッダ名が tracestate の場合
  OpenTelemetry 固有のデータが付与されます。

(1) 分散トレースの送信に失敗したとき

(a) 分散トレース送信失敗時の検知、および分散トレース機能の抑止について

HMP-PCTO で、分散トレースを定期的に分散トレース基盤へ送信しますが、エンドポイントが機能していない、送信先エンドポイントのパラメタ指定値が誤っている、などの要因で送信に失敗した場合、OpenTelemetry は警告メッセージを標準出力に出力します。このため、標準出力には定期的に警告メッセージが出力されることになります。

ユーザの運用での対処として、分散トレース基盤への送信失敗時の警告メッセージが、標準出力に出力されているかどうかを確認します。警告メッセージが出力されていた場合は、送信先定義を見直し、分散トレース基盤が正常起動していることを確認してください。詳細については、「10.6 HMP-PCTO の障害調査」を参照してください。

(b) 分散トレース送信失敗時、トレース情報を一時的に別の出力先に変更することについて

「(a) 分散トレース送信失敗時の検知、および分散トレース機能の抑止について」にあるとおり、ユーザが運用で対処して分散トレース送信失敗を回復する必要があるため、HMP-PCTO ではトレース情報の出力先を別の出力先に切り替えません。

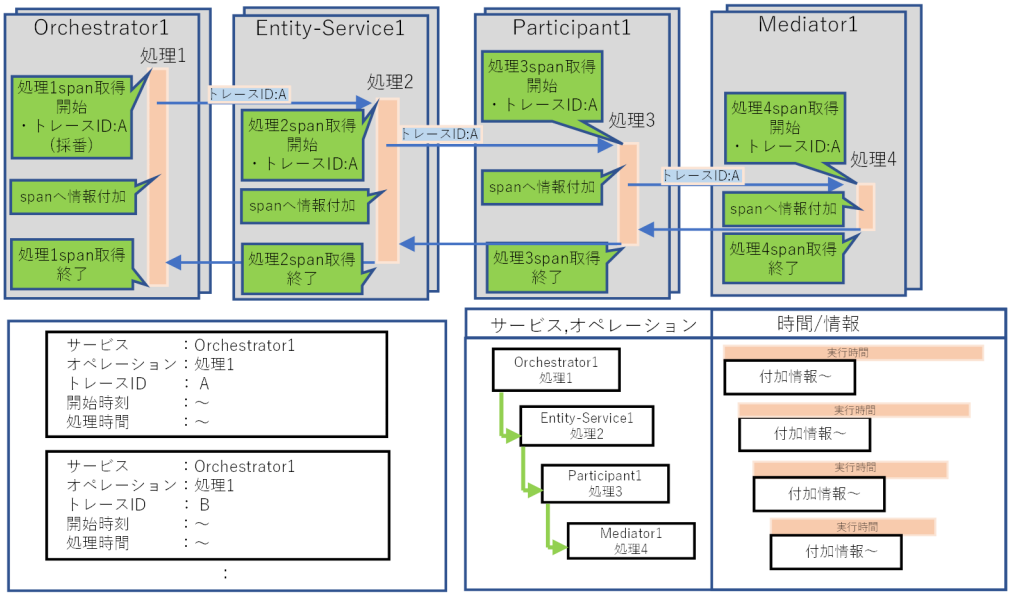
そのため、ユーザが運用で対処するまでは分散トレースが Jaeger に送信されず、トレース情報がロストするため注意してください。

10.3.2 トレースの単位

HMP-PCTO は分散トレースとして Span を取得します。Span はトレースの単位です。HMP-PCTO では、各コンポーネント（Orchestrator、Entity-Service、Entity-Module、Participant、Mediator）のプロセスをそれぞれ 1 つのマイクロサービスと考え、各コンポーネント内の特定処理（オペレーション）の

開始時に Span 取得を開始し、特定処理の終了時に Span 取得を終了します。Span を取得することで、特定処理の開始時刻と実行時間を取得できます。HMP-PCTO は Span の取得開始から終了までの間に、Span に特定処理に関する情報（ブランチ XID など）を付加し取得します。次に示す図は SQL-Participant を使用する場合です。TCC-Participant を使用する場合、Entity-Service を Entity-Module と読み替えてください。

図 10–12　トレースの単位



トレースID一覧イメージ図　　特定トレースIDのspan情報表示イメージ図

HMP-PCTO は処理の起点（トランザクション開始時など）でトレース ID を採番します。トレース ID はグローバルに一意な識別子であり、同一トランザクションのすべてのコンポーネントにわたって同じトレース ID をコンポーネント間の通信に埋め込み伝搬し、各 Span に付与することで Span をグループ化できます。Span はネストできます。呼び出し元の処理が親 Span、呼び出し先の処理が子 Span となります。

ユーザは Jaeger の Web UI へアクセスし、トレース ID の一覧、およびトレース ID 単位での親 Span、子 Span の呼び出し関係を確認することで、処理の流れを確認できます。

未決着トランザクションの Recovery 機能が実行された場合、SQL-Participant または TCC-Participant の再起動前のトレース ID と再起動後のトレース ID は別の値となります。そのため、処理の流れを確認する場合は、同時に出力されているブランチ XID を基に処理の流れを確認する必要があります。

10.3.3　Span へ付加する情報

Span に付加する情報の形式について説明します。付加する情報の形式は 3 パターンあります。

(1) Tags

次の key（情報名）、value（情報値）の形式で情報を取得します。Jaeger の Web UI で Tags の情報名、情報値を指定することで表示するトレース ID をフィルタできます。時間情報を必要としない情報の場合に、Tags に設定します。

例：受信した要求電文に含まれていたブランチ XID など

図 10–13 Tags

(2) Logs

次の Logs 形式 1、2 どちらかの形式で情報を取得します。Span 内での処理の実施時刻を取得でき、Span 内で遅延した処理の確認などに使用できます。時間情報を必要とする場合に Logs に設定します（例：I/O 開始、終了時刻など）。message に対し、処理ごとの情報説明を設定します。情報説明には、対象情報がどのようなイベントかを設定します（ファイル I/O 開始など）。

図 10–14 Logs 形式 1

情報説明に加えて key（情報名）、value（情報値）の形式で情報追加したい場合は、Logs 形式 2 で情報を取得します。追加情報が不要であれば、Logs 形式 1 で取得します。

図 10–15　Logs 形式 2

情報名と情報値を設定する場合に、Tags と Logs 形式 2 の使い分けについて説明します。

対象イベントだけに関わる情報の場合は Logs 形式 2 へ、対象 Span 全体に関わる情報の場合は Tags に設定します。例えば、ある Span 内でファイル I/O する場合に、スレッド名やブランチ XID など Span 全体に関わる情報は Tags に、ファイル I/O の結果は Logs 形式 2 に設定します。

(3)　Process

次の key（情報名）、value（情報値）の形式でプロセスに関する情報を取得します。Jaeger の Web UI で Process の情報名、情報値を指定することで表示するトレース ID をフィルタできます。

図 10-16　Process

10.3.4　取得方針

(1)　基本

Span の基本の取得方針について説明します。EADS、XA については、「(2)　EADS（通常版限定）」および「(3)　XA 関数（SQL-Participant 限定）」を参照してください。

取得個所、取得情報、および情報を取得する Span について説明します。

(a)　取得個所方針

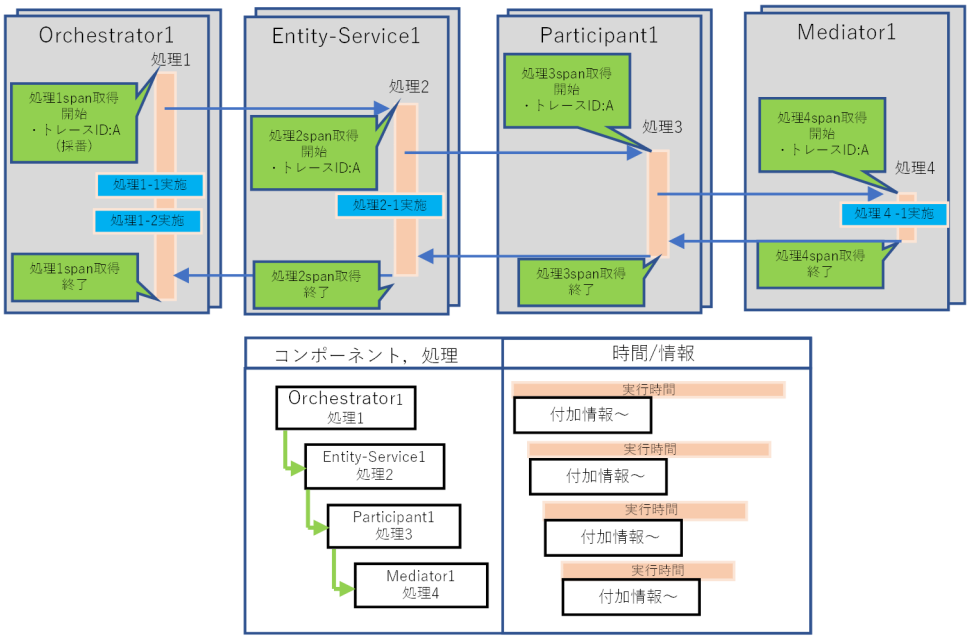
HMP-PCTO では Span を一般的な取得個所と、製品保守観点での取得個所で取得します。

一般的な取得個所での Span と、製品保守観点での取得個所での Span を、Span に付与するオペレーション名で区別できます。オペレーション名は、各コンポーネント内の特定処理（オペレーション）を特定するため、製品の Java クラス名やメソッド名などを組み合わせた文字列です。一般的な取得個所の Span は、オペレーション名冒頭に「Recv」を付与します。なお、ここで示す図は SQL-Participant を使用する場合です。TCC-Participant を使用する場合、Entity-Service を Entity-Module と読み替えてください。

一般的な取得個所

分散トレースは一般的に、マイクロサービス間の情報を把握するために 1 回のサービス呼び出しで 1Span 取得します。HMP-PCTO でも、各コンポーネントでの処理開始時の上位処理で Span を取得します。上位処理から呼び出す下位処理では別途 Span は取得しません。例外として、「製品保守観点での取得個所」に示す個所で Span を取得します。

図 10-17　取得イメージ図



特定トレースIDのspan情報表示イメージ図

製品保守観点での取得個所

一般的な Span の取得個所で情報を取得できない処理について、製品処理の調査のため別途 Span を取得します。

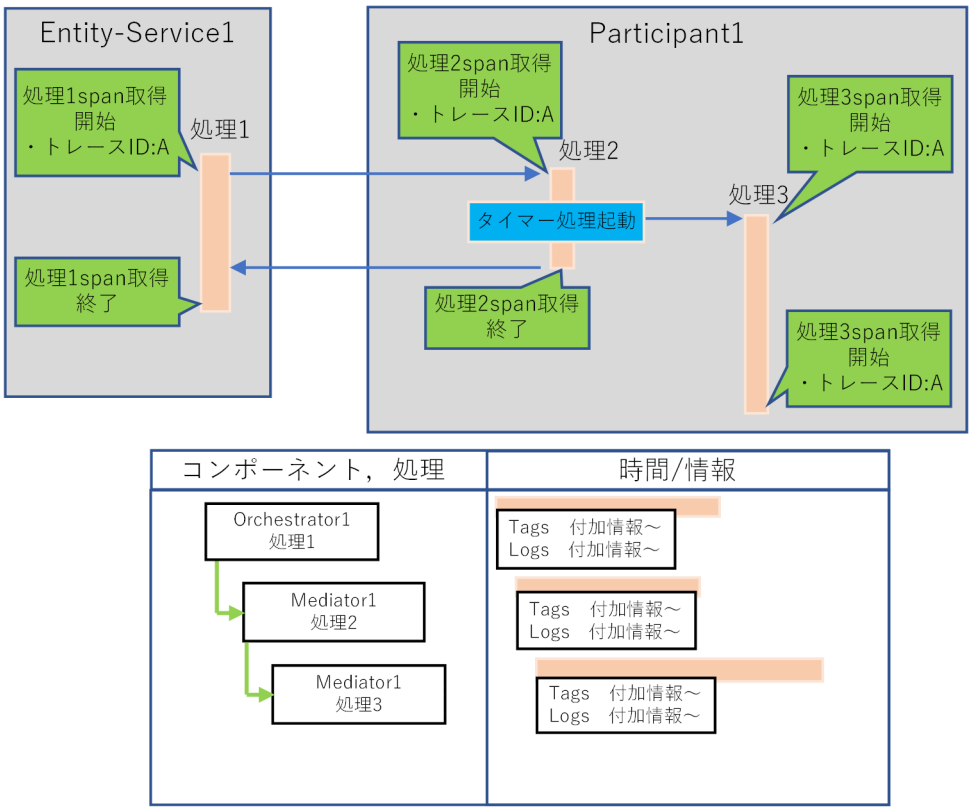
上位処理とは非同期に処理を実行する場合

タイマ処理など、他コンポーネントから要求の延長で実施した処理から登録し、別スレッドで実施する処理があります。これらの処理は登録元の処理が完了しても処理は続行するため、処理の終了時などの情報を、登録元の Span へ登録できません。

そのため、コンポーネント内処理に応じて、次のどれかで取得します。

- 処理途中および終了時の情報を取得する場合（他コンポーネントや他製品と連携する場合など）
  登録元処理の Span を親 Span とした新たな Span を作成します。
- 起動時の情報だけを取得する場合（処理途中、終了時情報が不要の場合）
  登録元処理の Span に、情報を付加します。

図 10-18　例 1 タイマ処理

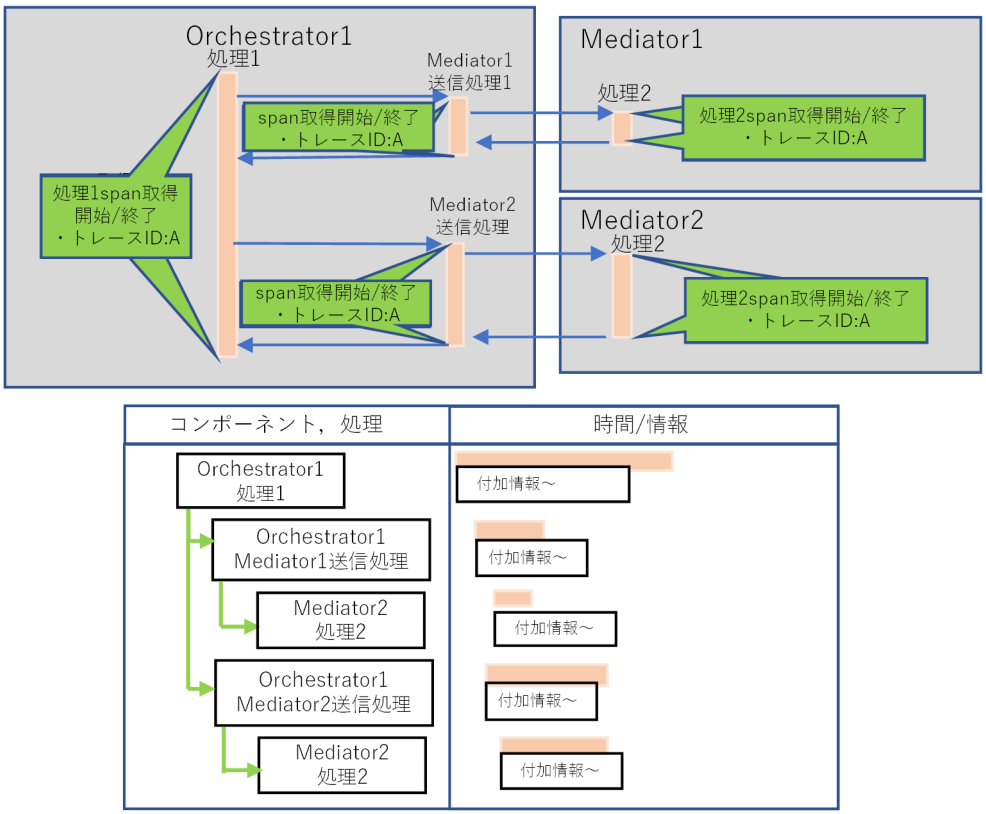


特定トレースIDのspan情報表示イメージ図

電文送信処理

HMP-PCTO では、複数の通信相手へ複数スレッドで同時送信する処理があります（Orchestrator から Mediator へのトランザクション決着要求など）。送信先の情報だけでは、ネットワーク遅延などが起きた場合に原因調査が難しいため、送信元での電文送受信時の情報（宛先、送受信時刻など）を取得したいため、送信先数分の情報取得が必要です。それらを 1 つの Span で管理すると、宛先ごとの情報の確認がしづらくなります。そのため、送信先ごとに子 Span を取得し、送信先ごとの情報を子 Span で取得します。

図 10-19　通信処理



特定トレースIDのspan情報表示イメージ図

(b)　取得する付加情報方針

HMP-PCTO が取得する付加情報について説明します。

次の観点で付加情報を取得します。

- 他トラブルシュート情報との紐づけ情報（ルート XID/ブランチ XID など）
- 対象処理がどこで実施した処理かの識別情報（ホスト名、スレッド名など）
- 通信先の情報（送信先の IP アドレス、ポート番号など）
- 通信データ（他コンポーネントへの指示内容（Commit/rollback 要求）など）
- 対象処理の結果（commit/rollback など）
- 実行時間情報（ユーザアプリ、ファイル I/O など開始終了時間など）

付加する情報の一覧を次に示します。

コンポーネント共通

HMP-PCTO のコンポーネントが、共通で Span に付加する情報の一覧を次に示します。

- Tags

表 10-7　コンポーネント共通の付加情報（Tags）

項番	情報名	設定する情報値の説明
1	root-xid	ルート XID を base64 エンコードした値を設定します。 ルート XID とブランチ XID のうち、ルート XID を使用する処理の場合に設定します。
2	branch-xid	ブランチ XID を base64 エンコードした値を設定します。 ルート XID とブランチ XID のうち、ブランチ XID を使用する処理の場合に設定します。
3	thread-name	スレッド名を設定します。
4	otel.library.name	Span が取得されたライブラリの名称を設定します。 HMP-PCTO のライブラリで取得された Span の場合、「HMP-PCTO」が設定されます。
5	otel.library.version	Span が取得されたライブラリのバージョンを設定します。

- Logs

表 10-8　コンポーネント共通の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	start:operation	Span で取得する処理が開始したことを意味します。 親処理と異なるスレッドで実行する処理で、スレッド割り当てなどが完了し、実際の処理開始時に取得します。	情報なし	－
2	End:operation	Span で取得する処理の終了を意味します。	情報なし	－

（凡例）－：該当しません。

- Process

表 10-9　コンポーネント共通の付加情報（Process）

項番	情報名	設定する情報値の説明
1	hostname	コンポーネントのホスト名を設定します。
2	ip	コンポーネントの IP アドレスを設定します。
3	jaeger.version	HMP-PCTO が使用している OpenTelemetry のライブラリを設定します。 「opentelemetry-java」が設定されます。
4	otel.library.name	Span が取得されたライブラリの名称を設定します。 HMP-PCTO のライブラリで取得された Span の場合、「HMP-PCTO」が設定されます。
5	otel.library.version	Span が取得されたライブラリのバージョンを設定します。
6	service.name	HMP-PCTO の jp.co.Hitachi.soft.hmppcto.tracing.instance-id パラメタに設定した値が設定されます。

項番	情報名	設定する情報値の説明
7	telemetry.sdk.language	HMP-PCTO が使用している OpenTelemetry のライブラリの言語を設定します。 「java」が設定されます。
8	telemetry.sdk.name	HMP-PCTO が使用している OpenTelemetry のライブラリの名称を設定します。 「opentelemetry」が設定されます。
9	telemetry.sdk.version	HMP-PCTO が使用している OpenTelemetry のバージョンを設定します。 HMP-PCTO 01-01 では、「1.38.0」が設定されます。

REST 通信

コンポーネント共通の情報に加えて、REST 通信を実施する際に Span に付加する情報の一覧を次に示します。

- Tags

表 10–10　REST 通信の付加情報（Tags）

項番	項目名	説明
1	http-method	リクエストの HTTP メソッド。 例：POST、GET など
2	http-url	リクエストの URL。
3	http-status-code	レスポンスの HTTP ステータスコード 通信障害でレスポンスが受信できなかった場合、この項目は設定されません。

- Logs

表 10–11　REST 通信の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	start:rest	REST 通信の開始（リクエストの送信）を意味します。	情報なし	－
2	end:rest	REST 通信の終了（レスポンスの受信）を意味します。	情報なし	－

（凡例）－：該当しません。

- Process

  付加情報はありません。

gRPC 通信

コンポーネント共通の情報に加えて、gRPC 通信を実施する際に Span に付加する情報の一覧を次に示します。

- Tags

表 10–12　gRPC 通信の付加情報（Tags）

項番	項目名	説明
1	network.peer.address	ネットワーク接続先の IP アドレスを設定します。
2	network.peer.port	ネットワーク接続先のポート番号を設定します。
3	network.type	ネットワークの種別を設定します。 「ipv4」を設定します。
4	rpc.grpc.status_code	gRPC のステータスコードを設定します。
5	rpc.method	gRPC のメソッドを設定します。
6	rpc.service	gRPC のサービスを設定します。
7	rpc.system	RPC の種別を設定します。 「grpc」が設定されます。
8	server.address	gRPC サーバの IP アドレスまたはホスト名を設定します。
9	server.port	gRPC サーバのポート番号を設定します。
10	span.kind	Span 種別を設定します。 • gRPC サーバの場合：「server」が設定されます。 • gRPC クライアントの場合：「client」が設定されます。
11	otel.scope.name	Span が取得されたライブラリの名称を設定します。 「io.opentelemetry.grpc-1.6」が設定されます。
12	otel.scope.version	Span が取得されたライブラリのバージョンを設定します。

- Logs

表 10–13　gRPC 通信の付加情報（Logs）

項番	項目名	説明	追加情報名	設定する情報値の説明
1	message	gRPC のメッセージ送信が行われたことを意味します。	message.id	メッセージのシーケンス番号
2			message.type	メッセージ送信時は「SENT」が設定されます。メッセージ受信時は「RECEIVED」が設定されます。
3	sent message(hmppcto)	gRPC のメッセージ送信が行われたことを意味します。	json-convert	JSON 形式での出力か、protobuf 形式での出力かを示します。 true: JSON 形式 false: protobuf 形式
4			message-value	電文内容。 電文の先頭から 256 文字の情報を表示します。 256 文字を超過する電文内容の場合、超過した部分は、分散トレースに登録しません。

項番	項目名	説明	追加情報名	設定する情報値の説明
				また、電文内容がすべて出力されていないことを表すため、分散トレースに登録される電文内容の終端 3 文字を"…"へと置換します。
5	received message(hmppcto)	gRPC のメッセージ受信が行われたことを意味します。	json-convert	JSON 形式での出力か、protobuf 形式での出力かを示します。 true: JSON 形式 false: protobuf 形式
6			message-value	電文内容。 電文の先頭から 256 文字の情報を表示します。 256 文字を超過する電文内容の場合、超過した部分は、分散トレースに登録しません。 また、電文内容がすべて出力されていないことを表すため、分散トレースに登録される電文内容の終端 3 文字を"…"へと置換します。

- Process

  付加情報はありません。

Orchestrator

コンポーネント共通の情報に加えて、Orchestrator が Span に付加する情報の一覧を次に示します。

- Tags

表 10–14　Orchestrator の付加情報（Tags）

項番	情報名	設定する情報値の説明
1	result	グローバルトランザクション決着結果 • Commit コミット • Rollback ロールバック • Failure トランザクション決着失敗

- Logs

表 10–15　Orchestrator の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	start:service	Orchestrator のアプリケーション開始を意味します。	情報なし	−

項番	情報説明名	意味	追加情報名	設定する情報値の説明
2	end:service	Orchestrator のアプリケーション終了を意味します。	情報なし	－

（凡例）－：該当しません。

- Process

  付加情報はありません。

Entity-Service または Entity-Module

コンポーネント共通の情報に加えて、Entity-Service または Entity-Module が Span に付加する情報の一覧を次に示します。

- Tags

表 10–16　Entity-Service または Entity-Module の付加情報（Tags）

項番	情報名	設定する情報値の説明
1	result	Orchestrator へのトランザクション結果 • Commit コミット • Rollback ロールバック
2	prepare	先行 prepare の実施有無 • Y 実行 • N 未実行

- Logs

表 10–17　Entity-Service または Entity-Module の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	start:service	Entity-Service または Entity-Module のアプリケーション開始を意味します。	情報なし	－
2	end:service	Entity-Service または Entity-Module のアプリケーション終了を意味します。	情報なし	－

（凡例）－：該当しません。

- Process

  付加情報はありません。

SQL-Participant

コンポーネント共通の情報に加えて、SQL-Participant で Span に付加する情報の一覧を次に示します。

- Tags

表 10–18　SQL-Participant の付加情報（Tags）

項番	情報名	設定する情報値の説明
1	consensus-value	分散合意結果 • Commit コミット • Rollback ロールバック
2	external-consensus	補助合意実施有無 • Y 補助合意実施 • N 補助合意未実施

- Logs

表 10–19　SQL-Participant の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	learning log size m/n	LearningLog（Mediator から受信した合意）の数、m:受信数、n:Mediator 数です。	情報なし	－
2	fast consensus	高速合意の実施を意味します。	情報なし	－
3	skip consensus	分散合意条件を満たせないため、分散合意をスキップしたことを意味します。	情報なし	－
4	external consensus	補助合意の実施を意味します。	情報なし	－
5	consensus finished	タイムアウト前に分散合意が完了していたことを意味します。	情報なし	－
6	start:jdbc	Participant での JDBC メソッド開始を意味します。	情報なし	－
7	end:jdbc	Participant での JDBC メソッド終了を意味します。	情報なし	－

（凡例）－：該当しません。

- Process

付加情報はありません。

TCC-Participant

コンポーネント共通の情報に加えて、TCC-Participant で Span に付加する情報の一覧を次に示します。

- Tags

表 10–20　TCC-Participant の付加情報（Tags）

項番	情報名	設定する情報値の説明
1	consensus-value	分散合意結果 • Commit コミット • Rollback ロールバック
2	external-consensus	補助合意実施有無 • Y 補助合意実施 • N 補助合意未実施

- Logs

表 10–21　TCC-Participant の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	learning log size m/n	LearningLog（Mediator から受信した合意）の数、m:受信数、n:Mediator 数です。	情報なし	−
2	fast consensus	高速合意の実施を意味します。	情報なし	−
3	skip consensus	分散合意条件を満たせないため、分散合意をスキップしたことを意味します。	情報なし	−
4	external consensus	補助合意の実施を意味します。	情報なし	−
5	consensus finished	タイムアウト前に分散合意が完了していたことを意味します。	情報なし	−
6	start:try	Try 開始を意味します。	情報なし	−
7	end:try	Try 終了を意味します。	情報なし	−
8	start:confirm	confirm 開始を意味します。	情報なし	−
9	end:confirm	confirm 終了を意味します。	情報なし	−
10	start:cancel	cancel 開始を意味します。	情報なし	−
11	end:cancel	cancel 終了を意味します。	情報なし	−

（凡例）−：該当しません。

- Process

  付加情報はありません。

TP1-Bridge

コンポーネント共通の情報に加えて、TP1-Bridge で Span に付加する情報の一覧を次に示します。

- Tags

表 10–22　TP1-Bridge の付加情報（Tags）

項番	情報名	設定する情報値の説明
1	consensus-value	分散合意結果 • Commit コミット • Rollback ロールバック
2	external-consensus	補助合意実施有無 • Y 補助合意実施 • N 補助合意未実施

- Logs

表 10–23　TP1-Bridge の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	learning log size m/n	LearningLog（Mediator から受信した合意）の数、m:受信数、n:Mediator 数です。	情報なし	－
2	fast consensus	高速合意の実施を意味します。	情報なし	－
3	skip consensus	分散合意条件を満たせないため、分散合意をスキップしたことを意味します。	情報なし	－
4	external consensus	補助合意の実施を意味します。	情報なし	－
5	consensus finished	タイムアウト前に分散合意が完了していたことを意味します。	情報なし	－
6	start:<TP1/Client/J のメソッド名> または start:TP1-Converter	TP1/Client/J のメソッドの呼び出し開始、または TP1-Converter の呼び出し開始を意味します。	情報なし	－
7	end:<TP1/Client/J のメソッド名> または end:TP1-Converter	TP1/Client/J のメソッドの呼び出し終了、または TP1-Converter の呼び出し終了を意味します。	情報なし	－

（凡例）－：該当しません。

- Process

付加情報はありません。

Mediator

コンポーネント共通の情報に加えて、Mediator が Span に付加する情報の一覧を次に示します。

- Tags

表 10–24　Mediator の付加情報（Logs）

項番	情報名	設定する情報値の説明
1	consensus-value	Participant での分散合意結果 • Commit コミット • Rollback ロールバック
2	consensus-id	ConsensusId の値を設定します。

- Logs

表 10–25　Mediator の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	start:consensuslog	ConsensusLog へのアクセス(I/O)開始を意味します。	情報なし	−
2	end:consensuslog	ConsensusLog へのアクセス(I/O)終了を意味します。	情報なし	−
3	start:proposallog-add-propose	Proposal Log へのアクセス（投票内容保存）の開始を意味します。	propose-values	Proposal Log に保存した投票内容の一覧を設定します。 • COMMIT_VOTE Commit 投票 • ROLLBACK_VOTE Rollback 投票 一覧は上記の値を","（コンマ）区切りで設定します。
4	end:proposallog-add-propose	Proposal Log へのアクセス（投票内容保存）の終了を意味します。	情報なし	−
5	start:proposallog-add-propose-participant	Proposal Log へのアクセス（投票を受信した Participant 情報保存）の開始を意味します。	participants-value	Proposal Log に保存した投票を受信した Participant 情報（<Service 名>.<ネームスペース名>）の一覧を設定します。 一覧は<Service 名>.<ネームスペース名>を","（コンマ）区切りで設定します。
6	end:proposallog-add-propose-participant	Proposal Log へのアクセス（投票を受信した Participant 情報保存）の終了を意味します。	情報なし	−

項番	情報説明名	意味	追加情報名	設定する情報値の説明
7	start:proposallog-add-all-participants	Proposal Log へのアクセス（トランザクションに参加した Participant 情報（<Service 名>.<ネームスペース名>））の一覧の保存の開始を意味します。	all-participants	Proposal Log に保存されたトランザクションに参加した Participant 情報（<Service 名>.<ネームスペース名>）の一覧を設定します。 一覧は<Service 名>.<ネームスペース名>を","（コンマ）区切りで設定します。
8	end:proposallog-add-all-participants	Proposal Log へのアクセス（トランザクションに参加した Participant 情報（<Service 名>.<ネームスペース名>））の一覧の保存の終了を意味します。	情報なし	－

（凡例）－：該当しません。

- Process

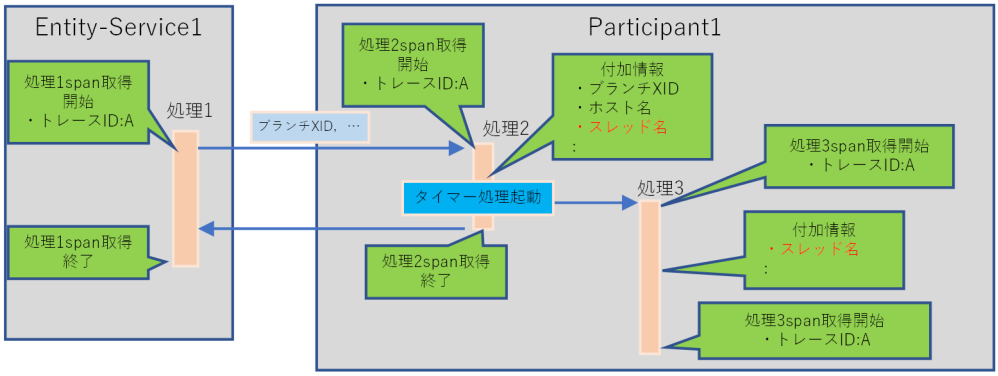
  付加情報はありません。

(c)　情報を取得する Span の方針

付加情報をどの Span で取得するかについて説明します。

Span に付加する情報は、全 Span 共通で付加するわけではなく、他コンポーネントから伝達された情報（ブランチ XID など）やプロセスに関わる情報（ホスト名）などは、他コンポーネントから呼び出された先頭処理（主に受信処理）の Span で取得します。先頭処理から生成/呼び出した処理は、先頭 Span と差異のある情報（スレッド名など）を取得します。次に示す図は SQL-Participant を使用する場合です。TCC-Participant を使用する場合、Entity-Service を Entity-Module と読み替えてください。

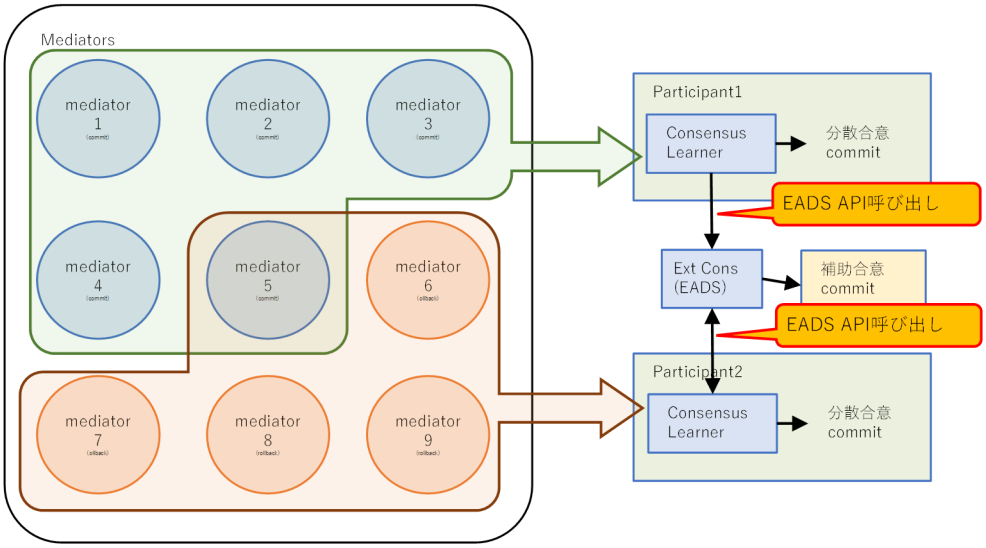
図 10–20　取得する情報



(2)　EADS（通常版限定）

EADS API 向け分散トレースは、分散合意取得機能の補助合意に使用される EADS で、次の図に示すような API を発行する Participant と EADS 間のデータ処理が妥当かどうかを判別するために Span を取得します。なお、トライアル版では EADS は使用できません。

図 10-21　EADS 向けトレース



この Span によって、EADS へのアクセス時間、実行内容、実行結果を分析可能とし、障害発生時にトラブルシュートが可能になります。

(a)　取得個所

EADS API の create および get 向けに Participant から補助合意のために EADS API を呼び出す時に Span を取得します。

(b)　取得する情報

コンポーネント共通の情報に加えて、EADS API 向けの Span で取得する情報を次に示します。

- Tags
  なし。
- Logs

表 10-26　EADS の付加情報（Logs）

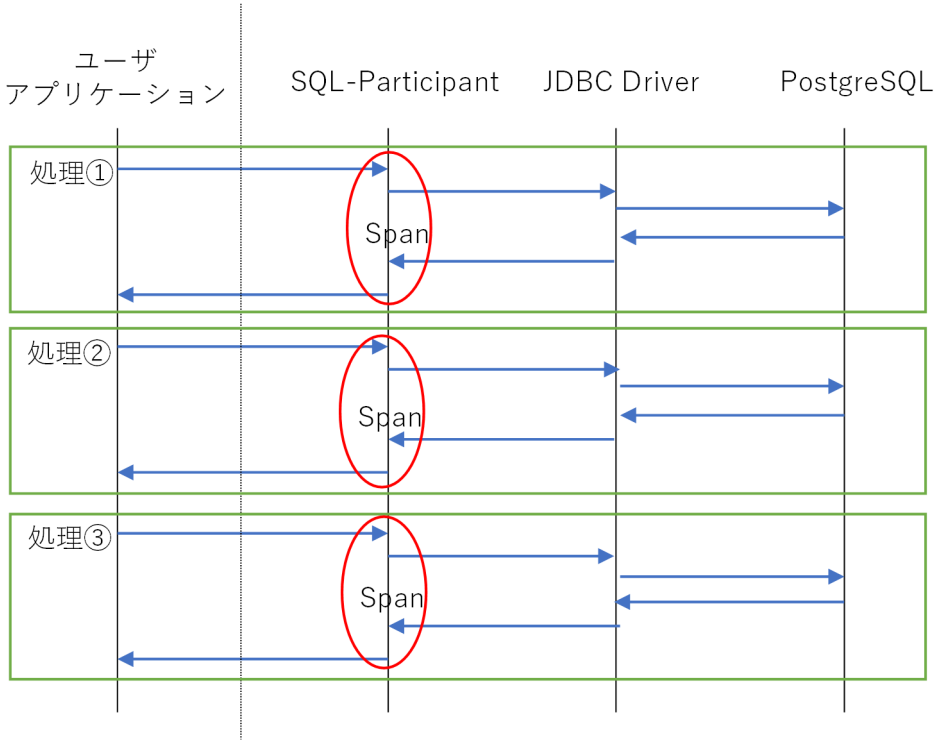
項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	start:EADS API	EADS の API の呼び出しが行われたことを意味します。	API-name	呼び出す EADS API 名 • create create の EADS API • get get の EADS API
2			consensus	補助合意提案値。create の場合だけ出力する。 • Commit コミット • Rollback ロールバック

項番	情報説明名	意味	追加情報名	設定する情報値の説明
3	end:EADS API	EADS の API の実行が終わったことを意味します。	API-name	実行が終わった EADS API 名 • create create の EADS API • get get の EADS API
4			ret	取得した補助合意提案値。get の場合だけ出力する。 • Commit コミット • Rollback ロールバック
5	failed:EADS API	EADS の API の実行中に例外が発生したことを意味します。	API name	例外が発生した EADS API 名 • create create の EADS API • get get の EADS API
6			message	例外の詳細メッセージ文字列

(3) XA 関数（SQL-Participant 限定）

XA 関数向け分散トレースは、SQL-Participant と DB 間のデータ処理が妥当かどうかを判別するために Span を取得します。トランザクションを対象とした Span のため、次の図に示すような、処理 1、処理 2、処理 3 の個別な情報を取得するだけではなく、各処理が 1 つのトランザクションであることを把握できるデータも合わせて出力を行い、トランザクション全体での妥当性も確認します。この Span によって、XA 関数の実行順序、実行内容、実行結果を分析し、障害発生時にトラブルシュートが可能になります。

図 10–22　XA 関数向け Span



(a)　取得個所

1. Participant で、Entity-Service から gRPC 通信に初回の SQL 実行が要求された時に Span を取得します。
2. Participant 初期化で、prepare 済みのトランザクションを取得する時に Span を取得します。
3. Participant で、Commit もしくは Rollback の通知の gRPC 通信を受信した時の Span に Log を追加します。

(b)　取得する情報

コンポーネント共通の情報に加えて、XA 関数向けの Span で取得する情報を次に示します。

- Tags

  なし。

- Logs

表 10–27　XA 関数の付加情報（Logs）

項番	情報説明名	意味	追加情報名	設定する情報値の説明
1	start: xa function	XA 関数の呼び出しが行われたことを意味します。	xa-name	呼び出す XA 関数名 • xa_start xa_start 関数 • xa_end xa_end 関数

項番	情報説明名	意味	追加情報名	設定する情報値の説明
				• xa_prepare xa_prepare 関数 • xa_commit xa_commit 関数 • xa_rollback xa_rollback 関数 • xa_recover xa_recover 関数
2	end: xa function	XA 関数の実行が終わったことを意味します。	xa-name	実行が終わった XA 関数名 • xa_start xa_start 関数 • xa_end xa_end 関数 • xa_prepare xa_prepare 関数 • xa_commit xa_commit 関数 • xa_rollback xa_rollback 関数 • xa_recover xa_recover 関数
3	failed:xa function	XA 関数の実行中に例外が発生したことを意味します。	xa-name	例外が発生した XA 関数名 • xa_start xa_start 関数 • xa_end xa_end 関数 • xa_prepare xa_prepare 関数 • xa_commit xa_commit 関数 • xa_rollback xa_rollback 関数 • xa_recover xa_recover 関数
4			message	例外の詳細メッセージ文字列

(4) パラメタ

次のパラメタを使用します。

- jp.co.Hitachi.soft.hmppcto.tracing.instance-id

- jp.co.Hitachi.soft.hmppcto.tracing.url
- jp.co.Hitachi.soft.hmppcto.tracing.trace-use
- jp.co.Hitachi.soft.hmppcto.tracing.queue-size
- jp.co.Hitachi.soft.hmppcto.tracing.delay-time
- jp.co.Hitachi.soft.hmppcto.tracing.batch-size

パラメタの詳細については、「8. パラメタ」を参照してください。

10.4　メトリクス

ここでは、トラブルシュート機能および性能チューニング計測機能で使用するメトリクスの機能について説明します。

10.4.1　概要

HMP-PCTO が動作するシステムでサービスのレスポンス遅延が発生したときに、パフォーマンスの傾向を調査するため、また、HMP-PCTO の性能評価を行うため、システムのパフォーマンスの指標となるメトリクスを取得します。

(1)　メトリクスの評価対象

メトリクスの評価対象を次の表に示します。システム（Kubernetes の場合、コンテナ環境）の利用状況、Java VM の利用状況、および HMP-PCTO の性能を評価するための指標となる数値を取得することで、HMP-PCTO が動作しているシステムのパフォーマンスの傾向を調査できます。

表 10–28　メトリクスの評価対象

項番	評価対象		説明
1	システムの利用状況		Kubernetes などのコンテナ環境における性能 ＜例＞ CPU 使用率
2	Java VM の利用状況		Java プロセス固有の性能 ＜例＞ • Java Heap • GC
3	HMP-PCTO の性能を評価するための指標	gRPC 通信の性能	HMP-PCTO の各機能間の gRPC 通信の性能 ＜例＞ gRPC 通信の処理時間
4		送信処理の性能	HMP-PCTO 内部で実施される非同期送信処理の性能 ＜例＞ • 実行回数・時間 • 未実行の（キューイングされている）数
5		機能固有の稼働統計情報	HMP-PCTO のコンポーネント（Mediator など）で固有に取得する稼働統計情報 ＜例＞ • Mediator の例：ProposalLog、ConsesnusLog の状態ごとの件数 • Participant の例：トランザクションブランチ数・処理時間 • TP1-Bridge の例：Spring Boot 組み込み Tomcat のスレッド数

項番	評価対象		説明
			• Jaeger-collector の例：受信した Span 数、内部キューに登録されたデータ数

上記の評価対象ごとに取得する項目を次に示します。項目の詳細については、「(3)　出力内容」を参照してください。

各コンポーネントで「表 10–28　メトリクスの評価対象」の項番 1～4 の情報を取得します。

項番 5「機能固有の稼働統計情報」については、機能ごとに次の表のパフォーマンスを評価します。

表 10–29　機能固有の稼働統計情報の評価対象

項番	コンポーネント	評価対象
1	Orchestrator、Entity-Service、Entity-Module	なし。
2	Mediator	ConsensusLog アクセス所要時間
3	SQL-Participant TCC-Participant TP1-Bridge	• 処理中のトランザクションブランチ数 • 完了したトランザクションブランチ数 • 完了したトランザクションブランチの処理時間 • EADS API の所要時間（トライアル版では評価できません）
4	TP1-Bridge	Spring Boot の組み込み Tomcat で管理されるスレッド数
5	Jaeger-collector	• Jaeger-collector の内部キュー内の Span 数 • Jaeger-collector が受信した Span 数

上記で示したメトリクスの評価対象以外にも、次の表に示す Exporter などを用いてメトリクスを取得することを推奨します。Prometheus によって収集することで、HMP-PCTO 各コンポーネントや Kubernetes クラスタのパフォーマンス監視や保守を行うための情報を取得できます。

表 10–30　取得を推奨するメトリクスを出力する Exporter

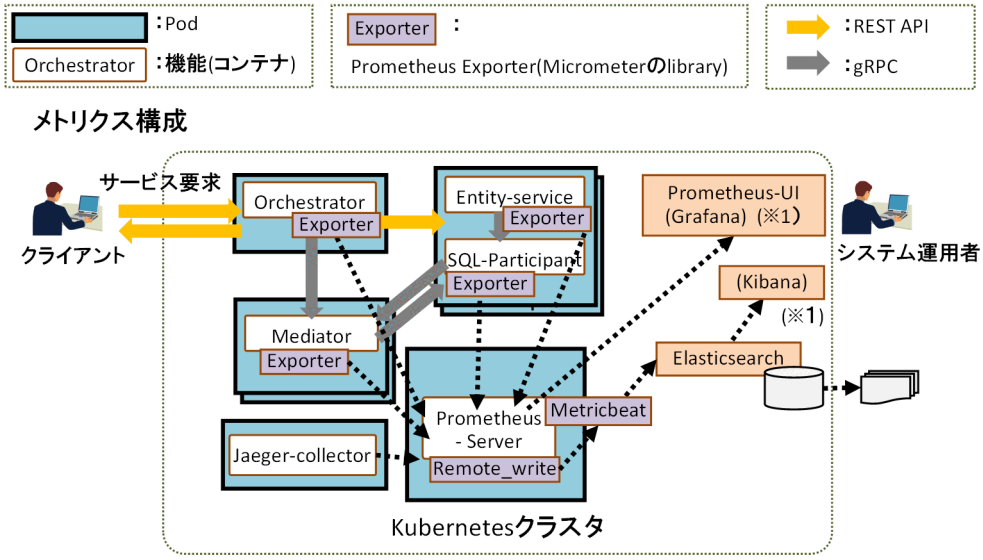
項番	Exporter/コンポーネント	概要
1	Node-Exporter	各 Kubernetes ノード上で実行し、各マシンの CPU,メモリの使用量などのマシンレベルのメトリクスを出力します。
2	kube-state-metrics	Kubernetes API より、クラスタ上のリソース情報を収集し、メトリクスとして出力を行います。
3	cAdvisor	cgroup についてのメトリクスを収集する Exporter。Kubernetes 上で稼働するコンテナのパフォーマンスに関する情報を出力します。

(2)　メトリクスの取得手段

HMP-PCTO では、各コンポーネントで Micrometer のライブラリを使用してメトリクスの情報を取得します。メトリクスデータを収集するサーバ Prometheus（Prometheus-Server）は、HMP-PCTO が

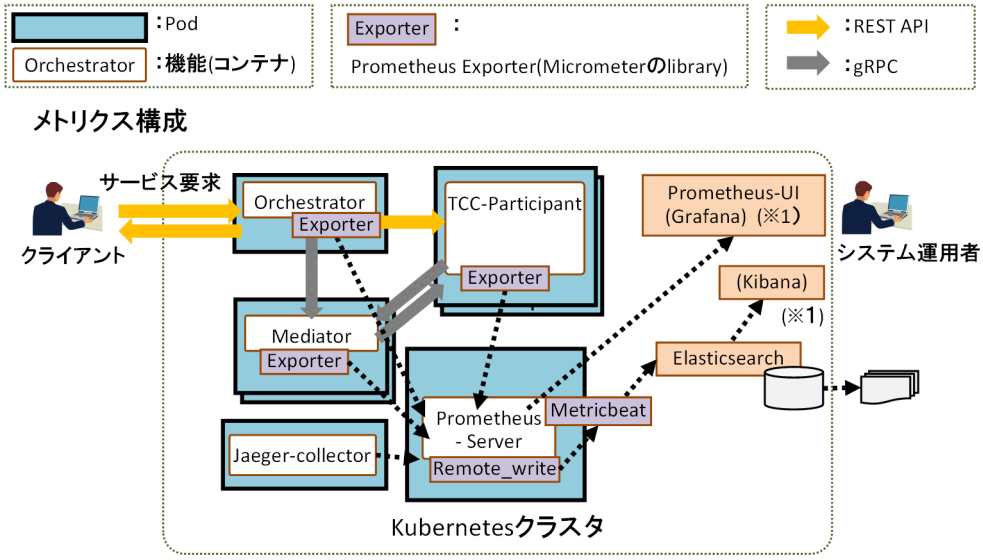
Micrometer で取得したメトリクスデータを収集します。HMP-PCTO では、送信先である Prometheus に対応した Exporter を Micrometer で実装することで Prometheus にメトリクスデータを送信できます。次に示す図は SQL-Participant を使用する場合です。TCC-Participant を使用する場合、Entity-Service を Entity-Module と読み替えてください。

図 10–23　HMP-PCTO メトリクスの構成（SQL-Participant の場合）



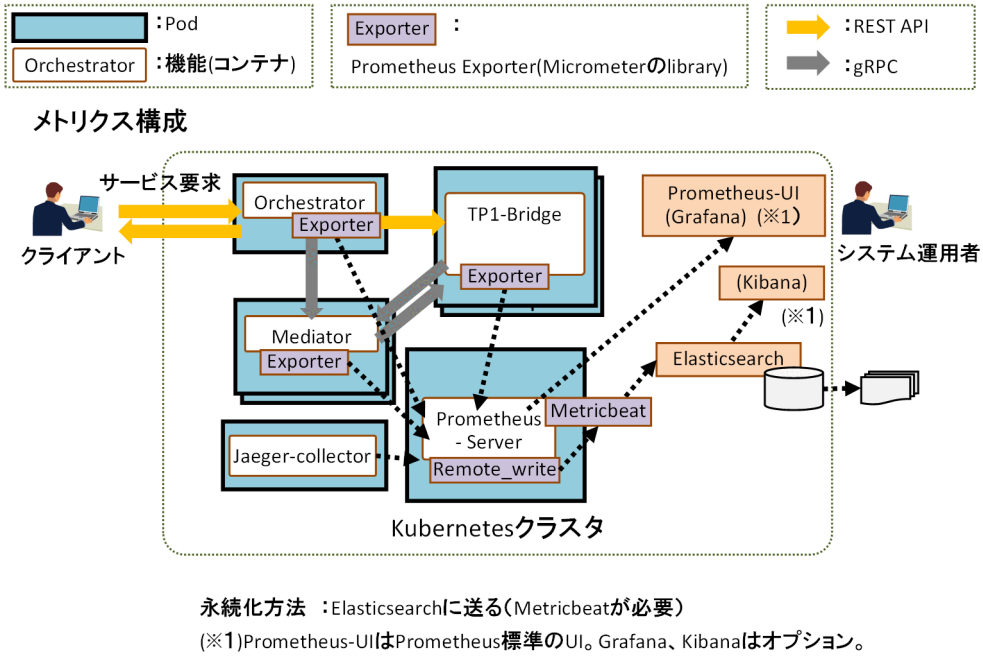
永続化方法　：Elasticsearchに送る（Metricbeatが必要）
(※1)Prometheus-UIはPrometheus標準のUI。Grafana、Kibanaはオプション。

図 10–24　HMP-PCTO メトリクスの構成（TCC-Participant の場合）



永続化方法　：Elasticsearchに送る（Metricbeatが必要）
(※1)Prometheus-UIはPrometheus標準のUI。Grafana、Kibanaはオプション。

図 10–25　HMP-PCTO メトリクスの構成（TP1-Bridge の場合）



評価対象ごとの取得手段を次の表に示します。

表 10–31　メトリクスの取得手段

項番	評価対象		取得手段
1	システムの利用状況		Micrometer が標準で用意するライブラリで Prometheus-Server に送信するメトリクスデータを生成します。
2	Java VM の利用状況		Micrometer が標準で用意するライブラリで Prometheus-Server に送信するメトリクスデータを生成します。
3	HMP-PCTO の性能を評価するための指標	gRPC 通信の性能	Micrometer が標準で用意するライブラリで Prometheus-Server に送信するメトリクスデータを生成します。
4		送信処理の性能	HMP-PCTO 内部で実施される非同期送信処理に関する項目について、Micrometer を使用して独自に作成し、この取得項目について Prometheus-Server に送信するメトリクスデータを生成します。
5		機能固有の稼働統計情報	• HMP-PCTO が各コンポーネント固有の稼働統計情報について、Micrometer を使用して独自に作成し、この取得項目について Prometheus-Server に送信するメトリクスデータを生成します。 • Jaeger-collector は、Jaeger-collector の機能によってメトリクス情報を生成します。
6		Spring Boot の組み込み Tomcat の情報（TP1-Bridge 使用時)	Spring Boot の組み込み Tomcat に関する情報について、Tomcat Web サーバに関する情報を独自に作成し、Micrometer が標準で用意するライブラリを用いて、Prometheus-Server に送信するメトリクスデータを生成します。

重要

メトリクスデータの取得条件

Micrometer のライブラリによって取得されるメトリクスデータには、取得条件を満たした場合だけ記録されるため、Prometheus の Web UI で閲覧できないことがあります。

メトリクスデータの取得条件については、Micrometer のライブラリのドキュメントを参照してください。

Prometheus-Server 自体は永続化の手段を持たないため、さらに永続化を目的として外部に送信を行います。次の送信先があります。

- Elasticsearch

  Prometheus-Server は Remote_write 機能を使用して Metricbeat に送信し、Metricbeat が Elasticsearch に送信します。

Prometheus-Server が収集したメトリクスデータを閲覧するには、Prometheus の Web UI を使用します。詳細については、「10.5.3　メトリクス」を参照してください。

10.4.2　パラメタ

パラメタの詳細については、「8.　パラメタ」を参照してください。

10.5　データおよびファイル

10.5.1　メッセージログ

(1)　出力先

(a)　HMP-PCTO が出力するメッセージログ

HMP-PCTO は、メッセージログをログレベルに応じて標準出力および標準エラー出力に出力します。メッセージログのログレベルと出力先の対応を次の表に示します。

表 10–32　ログレベルと出力先

ログレベル	出力先
ERROR	標準エラー出力
WARN	標準出力
INFO	標準出力

出力内容は、次の方法で確認できます。

- Kubernetes の kubectl logs コマンド

  出力内容は、Kubernetes の kubectl logs コマンドで確認できます。詳細については、Kubernetes のリファレンスを参照してください。

- Kibana などの可視化ツール

  Elasticsearch に格納された情報を、Kibana などの可視化ツールで確認できます。詳細については、Kibana などの可視化ツールのリファレンスを参照してください。

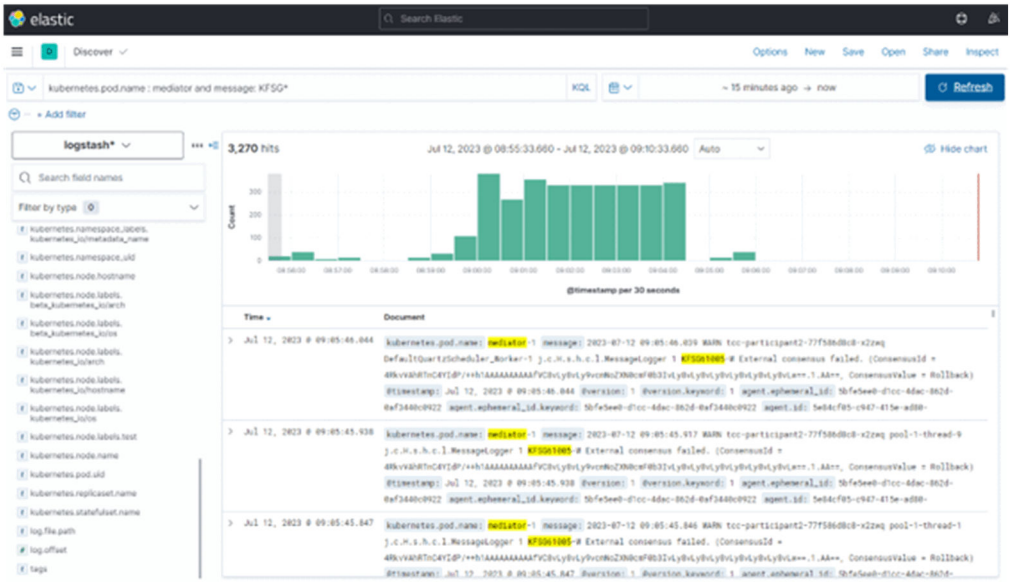
  Kibana の Discover 機能で HMP-PCTO のメッセージを抽出するためのフィールドを次の表に示します。このフィールドは Filebeat によって設定されています。

  表 10–33　検索対象のフィールド

フィールド名	説明
@timestamp	Filebeat で付与された時刻情報。 検索時に対象時刻の絞り込みに使用します。
kubernetes.container.name	ログが出力されたコンテナ名。 サイドカーとして起動した Filebeat で収集したログでは、「-」として表されます。
kubernetes.deployment.name	Pod を定義した Deployment リソースの名前。 サイドカーとして起動した Filebeat で収集したログでは、「-」として表されます。 Deployment で作成した Pod ではない場合は、「-」として表されます。
kubernetes.daemonset.name	Pod を定義した Daemonset リソースの名前。

フィールド名	説明
	サイドカーとして起動した Filebeat で収集したログでは、「-」として表されます。 Daemonset で作成した Pod ではない場合は、「-」として表されます。
kubernetes.labels.*	Pod に設定したラベルに設定した値。 * には、ラベルの key に相当する値が入ります。 サイドカーとして起動した Filebeat で収集したログでは、「-」として表されます。
kubernetes.namespace	ログが出力された Pod が作成されているネームスペース名。 サイドカーとして起動した Filebeat で収集したログでは、Pod 名を付与するために設定が必要です。設定内容については、「3.6　Kubernetes マニフェスト作成」に記載のマニフェスト例を参照してください。
kubernetes.node.name	ログが出力された Pod が起動している Kubernetes ノードの名前。 サイドカーとして起動した Filebeat で収集したログでは、Pod 名を付与するために設定が必要です。設定内容については、「3.6　Kubernetes マニフェスト作成」に記載のマニフェスト例を参照してください。
kubernetes.pod.name	ログが出力された Pod 名。 サイドカーとして起動した Filebeat で収集したログでは、Pod 名を付与するために設定が必要です。設定内容については、「3.6　Kubernetes マニフェスト作成」に記載のマニフェスト例を参照してください。 検索時には、メッセージを抽出したい HMP-PCTO のコンポーネントを指定します。 （例）「mediator」
kubernetes.statefulset.name	Pod を定義した Statefulset リソースの名前。 サイドカーとして起動した Filebeat で収集したログでは、「-」として表されます。 Statefulset で作成した Pod ではない場合は、「-」として表されます。
log.file.path	ログが出力されたファイルパス。
message	ログ本文。 検索時には、「KFSG*」と指定して、HMP-PCTO が出力したメッセージを抽出します。

図 10–26　検索の例



(b)　ライフサイクルイベントハンドラが出力するメッセージログ

Kubernetes のコンテナのライフサイクルイベントハンドラは、メッセージログを標準出力および標準エラー出力します。出力内容は、Kubernetes のイベントリソースに保存されます。Kubernetes の kubectl describe コマンドまたは kubectlget events コマンドで確認できます。

詳細については、Kubernetes のリファレンスを参照してください。

(2)　出力フォーマット

(a)　HMP-PCTO が出力するメッセージログ

メッセージの出力フォーマットを次に示します。

```
yyyy-mm-dd hh:mm:ss.SSS level hostname thread classname PID KFSGN1N2N3N4N5-X message
```

表 10–34　出力情報

出力情報	説明
yyyy-mm-dd hh:mm:ss.SSS	出力日時 年月日 時分秒ミリ秒
level	ログレベル
hostname	ホスト名の値を取得し、出力します。 値が取得できない場合は、null を表示します。
thread	スレッド名
classname	メッセージ出力元クラス名 短縮名で表示します。

出力情報	説明
	＜例＞ 正式名称:jp.co.Hitachi.soft.hmppcto.common.Myclass 短縮名:j.c.H.s.h.c.Myclass
PID	プロセス ID
KFSGN1N2N3N4N5-X	メッセージ ID
message	メッセージテキスト

メッセージログ出力例を次に示します。

```
2023-03-09 18:49:08.347 INFO myhost Main j.c.H.s.h.c.Myclass 31 KSFG10000-E test message. in
formation = test information
```

(b) ライフサイクルイベントハンドラが出力するメッセージログ

メッセージの出力フォーマットを次に示します。

```
KFSGN1N2N3N4N5-X message
```

(3) メッセージログ機能が参照するファイル

(a) ロギングライブラリの設定ファイル

HMP-PCTO は、メッセージログを出力するためのロギングライブラリとして Logback を使用します。その設定はロギングライブラリの設定ファイルを参照しています。ファイルは HMP-PCTO のライブラリファイル内に格納されています。

重要

ユーザ責務の Kubernetes アプリケーションで使用するロギングライブラリの設定ファイル

ユーザ責務の Kubernetes アプリケーションで Logback を使用し、その設定をリソースフォルダの直下の logback-spring.xml に記載する場合、次の記載が必要です。記載しない場合は、メッセージログが正常な出力フォーマットで出力されなくなります。また、メッセージログが出力されないことがあります。ユーザ責務の Kubernetes アプリケーションで、ロギングライブラリの設定を変更しない場合、次の記載は不要です。HMP-PCTO と同様の出力フォーマットでログが出力されます。

記載個所

次のどちらかに記載します。

記載個所 1

```
<configuration>
記載個所
</configuration>
```

記載個所 2

```
<springProfile name={プロファイル名}>
記載個所
</springProfile >
```

記載内容

logback-spring.xml への記載内容を次に示します。

```
<include resource="hmppcto-include.xml"/>
<logger name="PCTO_STD" level="INFO" additivity="false">
        <appender-ref ref="PCTO_STD" />
</logger>
<logger name="PCTO_ERR" level="ERROR" additivity="false">
        <appender-ref ref="PCTO_ERR" />
</logger>
```

重要

HMP-ADIF と連携する場合

HMP-ADIF のロギング機能は、Logback を使用します。その設定は logback-spring.xml に定義します。ユーザ責務の Kubernetes アプリケーションで HMP-ADIF と連携し、HMP-ADIF のロギングパターンで HMP-ADIF のログを出力したい場合は、次の対応が必要です。対応しない場合は、HMP-PCTO と同様の出力フォーマットで HMP-ADIF のログが出力されます。

Logback の設定方法を次に示します。詳細については、Logback や Spring Framework のドキュメントを参照してください。

1. 次の表に示すとおりに HMP-ADIF の JAR ファイルを展開し、HMP-ADIF のデフォルト設定の logback-spring.xml ファイルを取り出す

表 10-35 HMP-ADIF のデフォルト設定の logback-spring.xml ファイルが含まれている JAR ファイル

項目	仕様
ファイル名	msa-common-core-<*version*>.jar
ファイル格納パス	<*HMP-ADIF インストール先パス*>¥org¥globallogic¥microservice¥msa-common-core¥<*version*>
logback-spring.xml ファイル格納パス	<*JAR ファイル展開パス*>¥logback-spring.xml

2. logback-spring.xml ファイルに次の記載を追加する

記載個所

```
<configuration>
記載個所
</configuration>
```

記載内容

```
<include resource="hmppcto-include.xml"/>
<logger name="PCTO_STD" level="INFO" additivity="false">
        <appender-ref ref="PCTO_STD" />
</logger>
<logger name="PCTO_ERR" level="ERROR" additivity="false">
        <appender-ref ref="PCTO_ERR" />
</logger>
```

3. カスタマイズした logback-spring.xml ファイルを次の表に示す仕様に従って格納する

表 10–36　logback-spring.xml ファイルの格納仕様

項目	仕様
ファイル名	logback-spring.xml
ファイル格納パス	Spring アプリケーションのリソースフォルダの直下

(b)　メッセージプロパティファイル

HMP-PCTO は、メッセージログを作成するためにメッセージプロパティファイルを参照しています。ファイルは HMP-PCTO のライブラリファイル内に格納されています。

重要

ユーザ責務の Kubernetes アプリケーションで使用するプロパティファイル

ユーザ責務の Kubernetes アプリケーションでは、リソースフォルダの直下に次プロパティファイルを作成しないでください。作成した場合は、メッセージログが正常な出力フォーマットで出力されなくなります。

プロパティファイル名：hmppcto_messages.properties

10.5.2　分散トレース

(1)　出力先

Jaeger と連携した Elasticsearch に分散トレースを出力します。

(2)　参照方法

収集された分散トレースは、Jaeger の Web UI の次の URL へアクセスすることで参照できます。

```
http://A...A:B...B
```

A...A：Kubernetes ワーカーノードの IP アドレス

B...B：ポート番号(「4.2.8 Jaeger-query」の http.port に指定した値)

(3)　出力形式

Jaeger の Web UI 上での表示内容について説明します。この製品で使用する 2 つの画面について説明します。それ以外の画面に関しては Jaeger の公式ページを参照してください。

(a)　Search Page

分散トレースの検索画面です。検索条件に合致したトレース ID の一覧を表示します。

図 10–27　Search Page

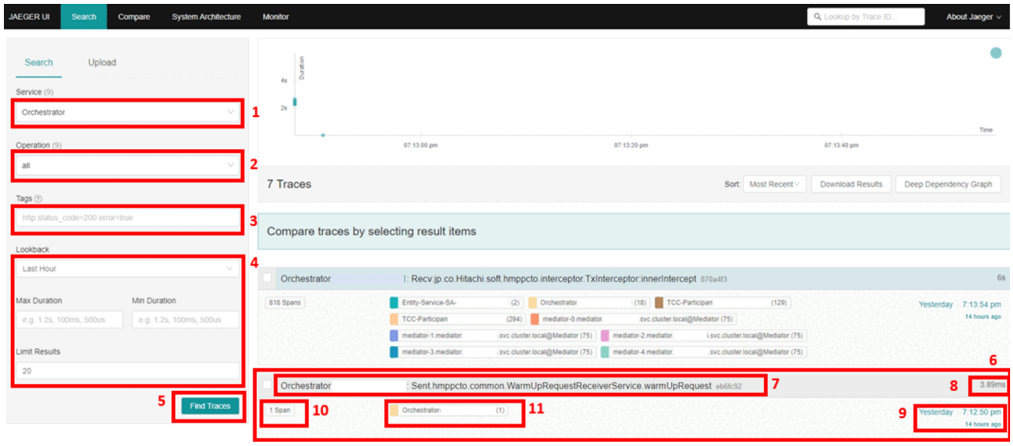


表 10–37　Search Page 項目

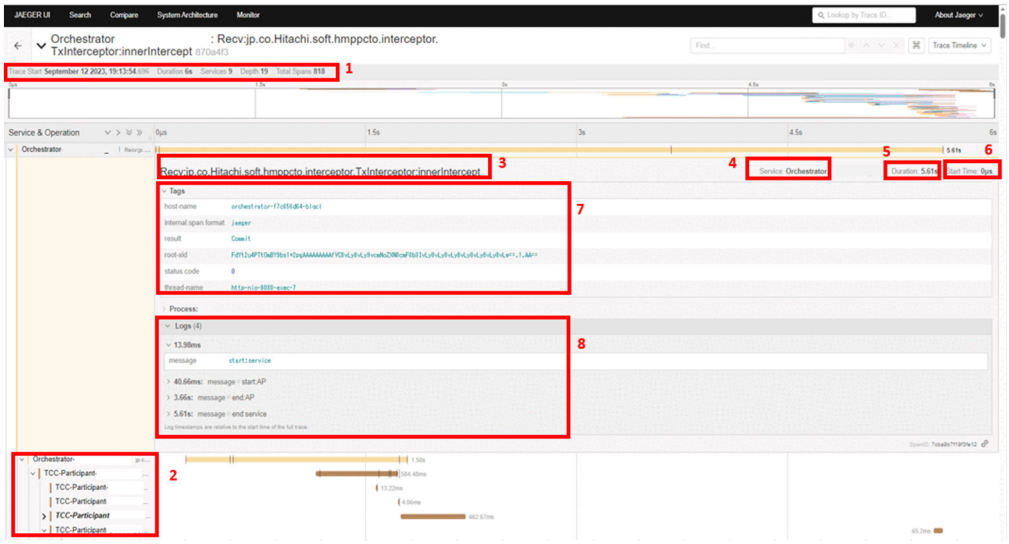
図中番号	項目名	説明
1	service	HMP-PCTO のコンポーネントの service 名を選択します。この項目は選択必須です。 service 名の形式は、jp.co.Hitachi.soft.hmppcto.tracing.instance-id パラメタの説明を参照してください。
2	Operation	HMP-PCTO のオペレーション名です。デフォルトは all です。 Operation の詳細については、「(b)　Trace Page」を参照してください。
3	Tags	Span の Tags を指定します。Tags の詳細については、「(b)　Trace Page」を参照してください。
4	−	検索対象範囲を指定します。詳細は Jaeger の公式情報を参照してください。

図中番号	項目名	説明
5	Find trace	検索を実行します。
6	トレース情報	トレース ID 1 件分の情報を表示します。このエリアをクリックすると、「(b)　Trace Page」へ遷移します。
7	−	次の項目を表示します。 親 Span の Service 名：親 Span の Operation 名：トレース ID
8	−	該当トレース ID での実行時間を表示します。
9	−	該当トレース ID での開始時刻を表示します。
10	−	該当トレース ID で取得した Span 数を表示します。
11	−	該当トレース ID に含む Span の service ごとの Span 数を表示します。

(b)　Trace Page

「(a)　Search Page」で選択したトレース ID のトレース情報を表示します。

図 10–28　Trace Page



一般的な取得個所の Span は、オペレーション名冒頭に「Recv」を付与します。

表 10–38　Trace Page 項目

図中番号	項目名	説明
1	−	トレース ID の情報を表示します。 • 該当トレース ID での開始時刻 • 該当トレース ID での実行時間 • 該当トレース ID で取得した Span の service 数 • 該当トレース ID で最も深い親 Span と子 Span の階層数

図中番号	項目名	説明
		• 該当トレース ID に含む Span 数
2	−	親 Span、子 Span を階層構造で表示します。該当する Span をクリックすると、Span の詳細を表示します。 「V」：クリックすると、子 Span が非表示になります。 「>」：クリックすると、子 Span が表示されます。
3	Operation	選択 Span を取得した HMP-PCTO のオペレーション名です。 一般的な取得個所の Span は、オペレーション名冒頭に「Recv」を付与します。
4	Service	HMP-PCTO のコンポーネントの service 名です。詳細は「(a)　Search Page」を参照してください。
5	Duration	選択 Span の実行時間です。
6	Starttime	該当するトレース ID を開始してから、選択した Span のトレースを開始するまでの経過時間です。
7	Tags	選択 Span での付加情報です。次の形式で表示します。 `情報名　情報値` 次の情報は、全 Span で取得します。 • internal.span.format jaeger • status.code 製品で取得する Span：0 grpc で取得する Span：grpc の status code を設定します。 • status.message status.code が 0 以外の場合に、status.code に応じたメッセージを設定します。
8	Logs	選択 Span での付加情報です。次の形式で表示します。 `経過時間 message 情報` 経過時間は、該当するトレース ID を開始してから、選択した Span のトレースを開始するまでの経過時間です。
9	Process	選択 Span での付加情報です。次の形式で表示します。 `情報名　情報`

10.5.3　メトリクス

(1)　出力先

HMP-PCTO の各コンポーネントは、Micrometer を使用してメトリクスデータを取得し、Prometheus 向けの Exporter を使って、テキストのデータを Prometheus サーバに送ります。

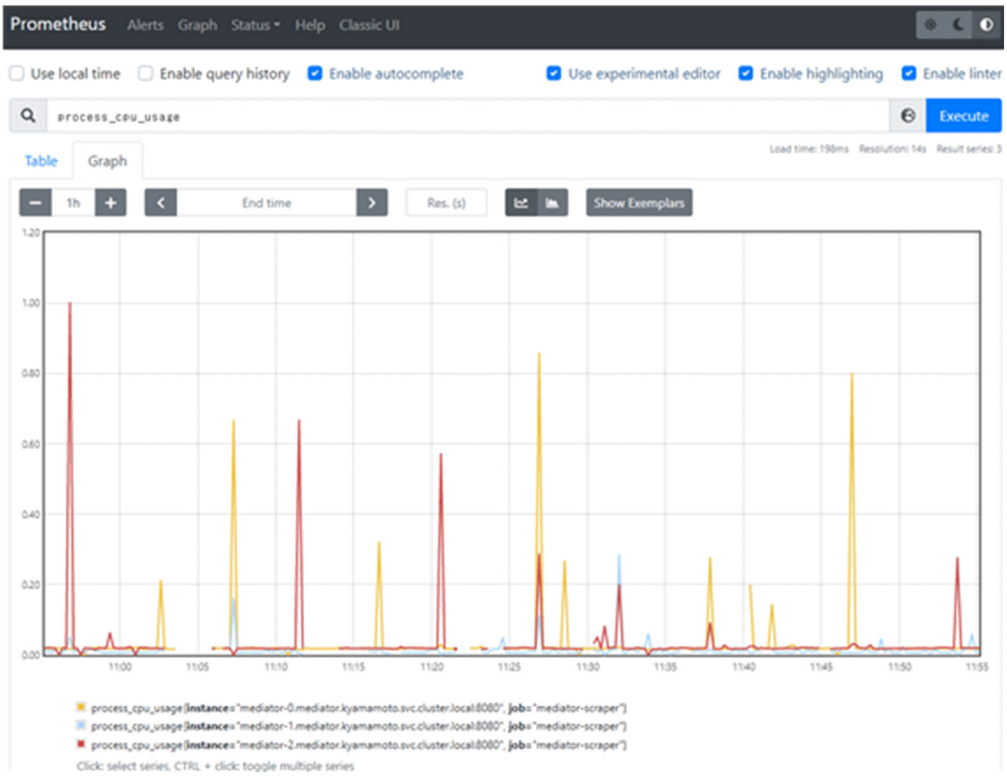
また、Jaeger-collector から出力されるメトリクスについても Prometheus サーバで収集します。

(2)　出力形式

(a)　Prometheus サーバの Web UI による参照

Prometheus サーバの標準の Web UI にアクセスすると、次の画面が表示されます。

図 10–29　Prometheus サーバの標準の Web UI



虫眼鏡マークの右にあるテキストフィールドに PromQL の構文でクエリを入力し、Execute ボタンを押すことでクエリの結果のメトリクス情報が画面に出力されます。

(b)　Kibana などの可視化ツールによる参照

Elasticsearch に保存されたメトリクスは Kibana などの可視化ツールで確認できます。詳細は Kibana などの可視化ツールのリファレンスを参照してください。

Kibana を例として、HMP-PCTO のメトリクスを抽出するためのフィールドを指定する例を次に示します。

```
prometheus.metrics.<メトリクス取得項目※1> : *※2
```

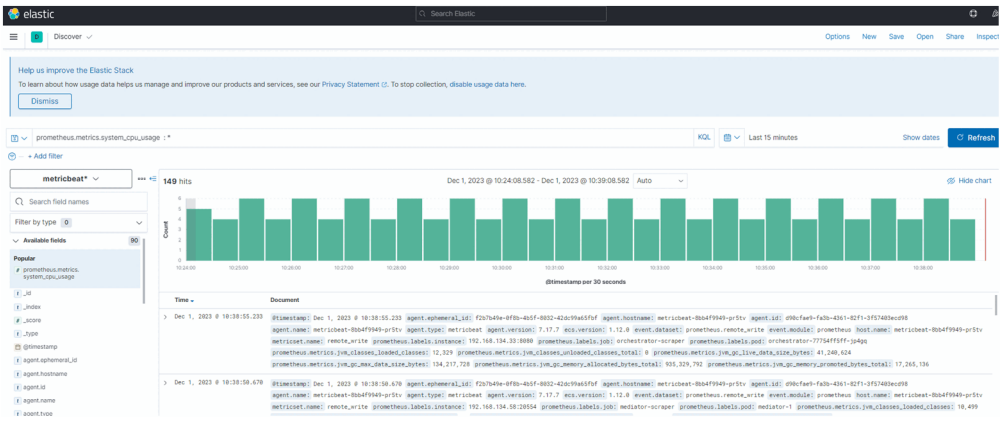
注※1

メトリクス取得項目に関しては、「(3)　出力内容」の項目を参照してください。

注※2

数値に関して特に条件がない場合は、": *"を指定することで、該当するメトリクスを取得した項目の絞り込みができます。

図 10–30　Kibana によるメトリクス参照例



(3)　出力内容

メトリクスデータとして出力する、HMP-PCTO のメトリクス取得項目を次に示します。

なお、ユーザアプリケーション（業務処理）部分は、ユーザがメトリクスを取得する必要があります。

表 10–39　システムの利用状況、および Java VM の利用状況

メトリクス取得項目	説明
system_cpu_usage	システム（Kubernetes の場合、コンテナ環境）の CPU 使用率です。
process_cpu_usage	Java VM プロセスの CPU 使用率です。
jvm_memory_max_bytes	Java VM で使用できるメモリの最大量です。
jvm_memory_used_bytes	Java VM の使用メモリ量です。
process_files_max_files	ファイル記述子の最大数です。
process_files_open_files	開いているファイル記述子の数です。
jvm_gc_memory_allocated_bytes_total	ある GC 実施後から、次の GC の間での、Java ヒープ New 世代領域のメモリプールサイズの増加量です。
jvm_gc_memory_promoted_bytes_total	ある GC 実施前と GC 実施後での、Java ヒープ Old 世代領域のメモリプールサイズの増加量です。
executor_active_threads	アクティブにタスクを実行しているユーザのスレッドのおおよその数です。
executor_queued_tasks	実行待ちのタスクのおおよその数です。

表 10-40　HMP-PCTO の性能を評価するための指標

メトリクス取得項目	説明	対象コンポーネント
grpc_server_processing_duration_seconds	gRPC サーバが呼び出し完了に掛かった時間	すべて
grpc_client_processing_duration_seconds	gRPC クライアントが通信完了するまでに掛かった合計時間	すべて
hmppcto_participant_transactions_inProcess_operations	処理中のトランザクションブランチ数	SQL-Participant TCC-Participant TP1-Bridge
hmppcto_participant_transactions_terminated_operations	完了したトランザクションブランチ数	SQL-Participant TCC-Participant TP1-Bridge
hmppcto_participant_transaction_duration_seconds_*	完了したトランザクションブランチの処理時間の統計情報	SQL-Participant TCC-Participant TP1-Bridge
hmppcto_asynctask_duration_seconds_*	HMP-PCTO 内部で実施される非同期送信処理の所要時間	すべて
hmppcto_asynctask_operations_total	HMP-PCTO 内部で実施される非同期送信処理の数	すべて
hmppcto_asynctask_queued_operations	キュー内にあり、まだ実行されていない非同期タスクの数	すべて
hmppcto_mediator_consensus_duration_*	ConsensusLog アクセス所要時間	Mediator
hmppcto_participant_eads_duration_*	EADS API の所要時間（トライアル版では出力できません）	SQL-Participant TCC-Participant TP1-Bridge
tomcat_threads_busy_threads	Spring Boot 組み込み Tomcat で、ビジー状態のスレッド数	TP1-Bridge

表 10-41　Jaeger-collector が出力するメトリクス

メトリクス取得項目	説明	出力するコンポーネント
jaeger_collector_queue_length	Jaeger-collector の内部キュー内に存在するアイテム数	Jaeger-collector
jaeger_collector_spans_dropped_total	Jaeger-collector の内部キューに存在し、Elasticsearch に送信されることなく、上書きされてしまった Span の数	Jaeger-collector

(4) 参照方法

(a) Prometheus の Web UI 参照方法

Prometheus サーバに収集されたメトリクスデータは、Prometheus サーバが公開している Web UI に、次の URL へアクセスすることで参照できます。

```
http://A...A:B...B
```

A...A：Prometheus の Web UI の DNS 名または IP アドレス

B...B：Prometheus の Web UI のポート番号

(b) エンドポイント

HMP-PCTO のメトリクスデータと、Mediator での処理中のトランザクションブランチ数、処理が完了したトランザクションブランチ数を参照可能とするために、各コンポーネントでエンドポイントを次の形式で用意しています。

```
http://A...A/B...B
```

A...A：各コンポーネントの Pod の DNS 名

B...B：メトリクスのエンドポイント

次に HMP-PCTO の各コンポーネントにおけるエンドポイントの詳細を示します。

Orchestrator

Orchestrator のエンドポイントを次に示します。

表 10–42 Orchestrator のエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/orchestrator/metrics	Prometheus にメトリクスを公開するためのエンドポイント

Entity-Service または Entity-Module

Entity-Service または Entity-Module のエンドポイントを次に示します。

表 10–43 Entity-Service または Entity-Module のエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/entity/metrics	Prometheus にメトリクスを公開するためのエンドポイント

SQL-Participant

SQL-Participant のエンドポイントを次に示します。

表 10–44　SQL-Participant のエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/sqlparticipant/metrics	Prometheus にメトリクスを公開するためのエンドポイント

TCC-Participant

TCC-Participant のエンドポイントを次に示します。

表 10–45　TCC-Participant のエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/tccparticipant/metrics	Prometheus にメトリクスを公開するためのエンドポイント

Mediator

Mediator のエンドポイントを次に示します。

表 10–46　Mediator のエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/mediator/metrics	Prometheus にメトリクスを公開するためのエンドポイント

TP1-Bridge

TP1-Bridge のエンドポイントを次に示します。

表 10–47　TP1-Bridge のエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/tp1bridge/metrics	Prometheus にメトリクスを公開するためのエンドポイント

10.5.4　定義するデータ形式

(1)　XID の形式

HMP-PCTO では、トランザクションを一意に識別する情報として XID を使用します。XID をコンポーネント間で引き渡して、トランザクションの制御をします。

XID の構造は、javax.transaction.xa インタフェースで定義された構造（X/Open トランザクション識別子 XID 構造の Java マッピング）とします。XID の構造と内容を次の表に示します。

表 10–48　XID の構造

名称	説明
GlobalTransactionId （グローバルトランザクション識別子）	トランザクションザクションを一意に識別するための ID。
FormatId	XID の形式を示す数値。

名称	説明
(形式識別子/フォーマット ID)	現時点の HMP-PCTO ではフォーマット ID=1 の一形式。 将来的に、HMP-PCTO のバージョン、使用する機能などによって形式が変わる場合は、この値で形式を区別します。
BranchQualifier (トランザクションブランチ識別子)	トランザクションブランチを一意に識別するための ID。

(2) フォーマット ID=1 の XID 形式

(a) GlobalTransactionId

64 バイトの byte 配列です。内容を次に示します。

表 10-49 GlobalTransactionId の内容

名称	オフセット (単位：バイト)	長さ (単位：バイト)	説明
UUID (ユニバーサル固有識別子)	0	16	UUID を byte 配列に変換した値です。Java の標準クラスでタイプ 4 (擬似ランダム生成) UUID を生成して格納します。 BIG ENDIAN です。
MediatorNumbersMap (Mediator 番号)	16	8	トランザクションに参加する Mediator の番号に対応するビットマップです。 次のようにマッピングします。 • 番号 0 を 2^0 ビットへマッピング • 番号 63 を 2^63 ビットへマッピング BIG ENDIAN です。
consensusType (分散合意種別)	24	1	Orchestrator のパラメタ jp.co.Hitachi.soft.hmppcto.learner.consensus-type の設定値に対応した 1 バイトの文字です。 • T：Three-Quarters • P：Privileged-Value
filler (空き領域)	25	7	未使用領域です。 「’ /’ 」で埋めます。
instanceId (インスタンス識別子)	32	32	どの Orchestrator のトランザクションか判断するための識別子です。 Orchestrator のパラメタ jp.co.Hitachi.soft.hmppcto.instance-id 設定値を byte 配列に変換した値です。 設定値が 32 バイト未満の場合は、「’ /’ 」で埋めます。

(b) FormatId

int 型の 1 です。

(c) BranchQualifier

32 バイトの byte 配列です。

Entity-Service もしくは Entity-Module を呼び出す Orchestrator、または Entity-Service もしくは Entity-Module のパラメタ jp.co.Hitachi.soft.hmppcto.branch.qualifier.list に設定した識別子を byte[] に変換した値です。設定値が 32 バイト未満の場合は、「#」で埋めます。

ルート XID の場合、BranchQualifier は意味はありません。0x00（1 バイト）が設定されますが、使用されません。

10.6　HMP-PCTO の障害調査

HMP-PCTO で障害が発生した場合に取得する情報と、その収集方法について説明します。

10.6.1　取得する情報

ユーザ（システム運用者）が調査するため、また、システム運用者がサポート部署へ資料を送付するために取得する情報を次の表に示します。

表 10–50　取得する情報

項番	情報の種類	取得する情報
1	標準出力の情報 （HMP-PCTO のメッセージを含む）	Elasticsearch に格納された標準出力の情報（HMP-PCTO のメッセージを含む）です。 HMP-PCTO は、各 pod の標準出力を Elasticsearch に送信します。 各コンテナの標準出力には、HMP-PCTO と同一コンテナ上で動作している次の項目が出力したメッセージなどを含みます。Elasticsearch に送信することで、どの項目が出力した情報なのかを検索したり、フィルタリングしたりできます。 • ユーザ責務の Kubernetes アプリケーションが出力したメッセージ • HMP-PCTO の前提プログラム/連携製品 ・HMP-ADIF ・Spring ・Java VM ・Java ランタイム ・TP1/Client/J
2	分散トレースの情報	Elasticsearch に格納された分散トレース情報です。 HMP-PCTO は、各 pod の分散トレース情報を Elasticsearch に送信します。
3	メトリクスの情報	Prometheus-Server に格納されたメトリクス情報です。 HMP-PCTO は、各 pod のメトリクス情報を Prometheus-Server に送信します。Prometheus-Server はメトリクス情報を Elasticsearch に送信します。
4	Kubernetes の情報	Kubernetes 上の pod の詳細情報です。 Pod で問題が発生していないかどうか、発生している場合は Event 情報で、どんな問題が発生しているかを確認します。 この Event 情報にはライフサイクルイベントハンドラで実行した HMP-PCTO のシェルスクリプトの標準出力の情報も出力されます。
5	連携製品が出力したトラブルシュート情報	連携製品が出力した次のトラブルシュート情報のファイルです。 次の情報については、コンテナのボリュームにファイルが出力されます。 • PostgreSQL のログ情報 • EADS のトラブルシュート情報（トライアル版では取得できません）

項番	情報の種類	取得する情報
		• Tomcat のアクセスログ • TP1/Client/J のトレース情報 次の情報については、連携製品が稼働するマシン上で収集します。 出力されるファイルのパス、ファイルの出力方法については、各連携製品のマニュアルを参照してください。 • HiRDB のトレース・ログ情報 • SQL Server のトレース・ログ情報 • uCosminexus Service Platform のトレース・ログ情報 • OpenTP1 のトレース・ログ情報

ユーザ（システム運用者）が調査する方法を次の表に示します。

表 10–51　ユーザの調査方法

項番	情報の種類	調査方法
1	標準出力の情報 （HMP-PCTO のメッセージを含む）	Elasticsearch に送られた標準出力の情報を、Kibana などの可視化ツールの UI で確認します。
2	分散トレースの情報	Elasticsearch に送られた分散トレースの情報を Jaeger の Web UI で確認します。
3	メトリクスの情報	• Prometheus-Server に送られたメトリクスの情報を Prometheus の Web UI（または Grafana などの可視化ツール）で確認します。 • Elasticsearch に送られたメトリクスの情報を Kibana などの可視化ツールの UI で確認します。
4	Kubernetes の情報	コマンド「kubectl describe」で確認します。
5	連携製品が出力したトラブルシュート情報	• uCosminexus Application Runtime with Java for Spring Boot のスナップショットログ収集機能を使用して収集したものを確認します。 • 各 Pod のトラブルシュート情報のファイルの内容を、サイドカーの Filebeat コンテナから Logstash に出力し、Elasticsearch に送信した内容を確認します。 • 連携製品が稼働するマシン上に出力されたトラブルシュート情報を確認します。

ユーザが調査しても原因不明だった場合、システム運用者がサポート部署へ資料を送付するために収集する資料を次の表に示します。

表 10–52　情報の収集方法

項番	情報の種類	収集する情報	収集方法
1	標準出力の情報 （HMP-PCTO のメッセージを含む）	Elasticsearch に格納された標準出力の情報	Elasticsearch のスナップショット機能で収集します。
2	分散トレースの情報	Elasticsearch に格納された分散トレース情報をエクスポートしたファイル	Elasticsearch のスナップショット機能で収集します。

項番	情報の種類	収集する情報	収集方法
3	メトリクスの情報	Elasticsearch からメトリクス情報をエクスポートしたファイル	Elasticsearch のスナップショット機能で収集します。
4	Kubernetes の情報	Kubernetes 上の pod の詳細情報	コマンド「kubectl describe」で出力された情報をファイルに出力します。
5	連携製品が出力したトラブルシュート情報	連携製品が出力した、次のトラブルシュート情報のファイル • PostgreSQL のログ情報 • EADS のトラブルシュート情報（トライアル版では取得できません） • Tomcat のアクセスログ • TP1/Client/J のトレース情報	• uCosminexus Application Runtime with Java for Spring Boot のスナップショットログ収集機能を使用して収集します。 • Elasticsearch のスナップショット機能で収集します。
6		連携製品が出力した、次のトラブルシュート情報のファイル • HiRDB のトレース・ログ情報 • SQL Server のトレース・ログ情報 • uCosminexus Service Platform のトレース・ログ情報 • OpenTP1 のトレース・ログ情報	連携製品が稼働するマシン上でトラブルシュート情報の収集・出力を実施します。 収集・出力方法については、連携製品のマニュアルを参照してください。

10.6.2　情報の収集方法

サポート部署へ資料を送付する目的で、収集方法を次に示します。

(1)　Elasticsearch に格納された情報の収集方法

Elasticsearch のスナップショット機能で収集します。収集先として、不揮発のストレージに出力してください。

Elasticsearch に HMP-PCTO が出力する情報の index 名は、次の文字で始まります。

表 10–53　Elasticsearch の index 名のプレフィックス

項番	情報の種類	Elasticsearch の index 名のプレフィックス
1	• 標準出力の情報 （HMP-PCTO のメッセージを含む） • 連携製品が出力したトラブルシュート情報	「logstash」
2	分散トレースの情報	「jaeger」
3	メトリクスの情報	「metricbeat」

Elasticsearch のスナップショット機能で収集する場合、Kubernetes コントロールプレーン上で次のコマンド例でバックアップを行ってください（Elasticsearch の http.port が 30202、elastic ユーザのパスワードが password の場合）。Elasticsearch リポジトリディレクトリに情報が格納されます。

```
# curl --cacert ca.crt --key elastic.key --cert elastic.crt -u elastic:password -XPUT 'https
://localhost:30202/_snapshot/backup/snapshot1'
```

Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリで情報が作成されていることを確認してください。

```
# ls /elasticsearch-repo-volume
index  indices  meta-snapshot-2023.07.11.dat
```

サポート部署へ送付するなどの目的で Elasticsearch リポジトリディレクトリの情報を 1 つのファイルにまとめる場合は、次の例を参考にしてください。

＜ Elasticsearch リポジトリディレクトリの情報を repo.tgz というファイルにまとめる例＞

```
# cd /elasticsearch-repo-volume
# tar cvfz /repo.tgz *
```

Elasticsearch リポジトリディレクトリの情報を見るためにリストアする場合は、Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリに Elasticsearch リポジトリディレクトリの情報を展開し、Kubernetes コントロールプレーン上で次のコマンド例でリストアを行ってください（Elasticsearch の http.port が 30202、elastic ユーザのパスワードが password の場合）。

```
# curl --cacert ca.crt --key elastic.key --cert elastic.crt -u elastic:password -XPOST 'http
s://localhost:30202/_snapshot/backup/snapshot1/_restore'
```

(2) uCosminexus Application Runtime for Spring Boot のスナップショットログ収集機能での収集方法

uCosminexus Application Runtime with Java for Spring Boot の「スナップショットログ収集機能」によって、コンテナ内の連携製品が出力したトラブルシュート情報のファイルを収集し、不揮発のストレージに出力してください。

コンテナ内に出力されるトラブルシュート情報を uCosminexus Application Runtime with Java for Spring Boot の機能で収集したい場合は、uCosminexus Application Runtime with Java for Spring Boot の設定ファイル（config.properties）を定義した ConfigMap のマニフェストファイルに、次のパラメタを追加してください。このパラメタに収集対象とするファイルが出力されるファイルパスを指定することで、スナップショットログの収集対象を追加できます。

```
snapshot.include.paths.<n>※ = <追加収集するファイルパス>
```

注※

<n>には、1 以上の整数を指定してください。

スナップショットログ収集機能についての詳細は、マニュアル「uCosminexus Application Runtime for Spring Boot ユーザーズガイド」を参照してください。

(3) 連携製品が稼働するマシン上のトラブルシュート情報の収集方法

マシン上で直接起動する連携製品のトラブルシュート情報の出力・収集は、ユーザの設定に依存します。そのため、各製品のマニュアルを参照して、トラブルシュート情報を出力・収集してください。

10.6.3 障害原因の調査方法

HMP-PCTO で発生する障害と、原因の調査方法について説明します。

(1) 障害発生時の流れ

ユーザは、標準出力（標準エラー出力を含む）に出力されたメッセージログのエラーメッセージ（警告メッセージを含む)、または分散トレースの出力情報によって、障害発生を検知します。

HMP-PCTO の障害発生時に、ユーザ（システム運用者）およびサポート部署が実施する障害調査の流れは次のとおりです。

1. 資料取得
2. 主なトラブルと調査個所
3. 障害個所の特定
4. 障害原因の調査
5. 障害対策

次に、それぞれのフェーズでユーザ（システム運用者）およびサポート部署が実施する内容を示します。

図 10–31　HMP-PCTO 障害発生時の流れ（SQL-Participant を使用する場合）

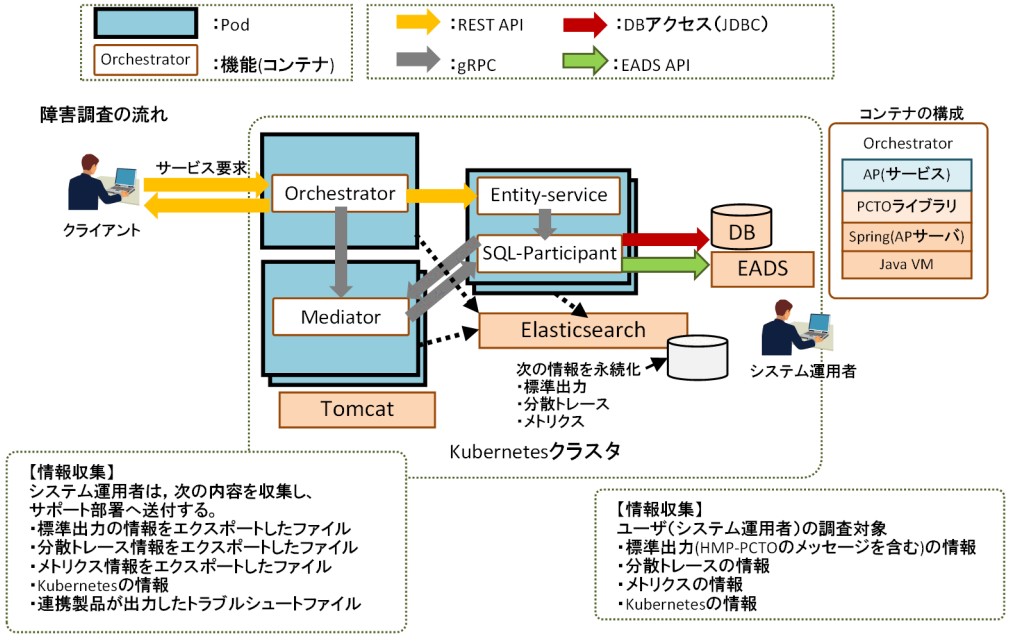


図 10–32　HMP-PCTO 障害発生時の流れ（TCC-Participant を使用する場合）

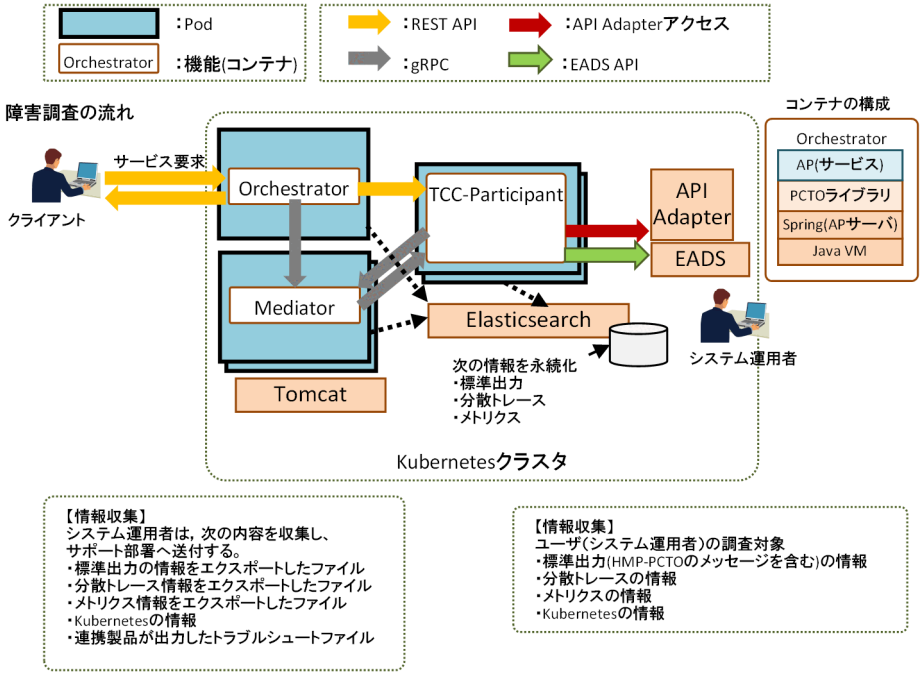
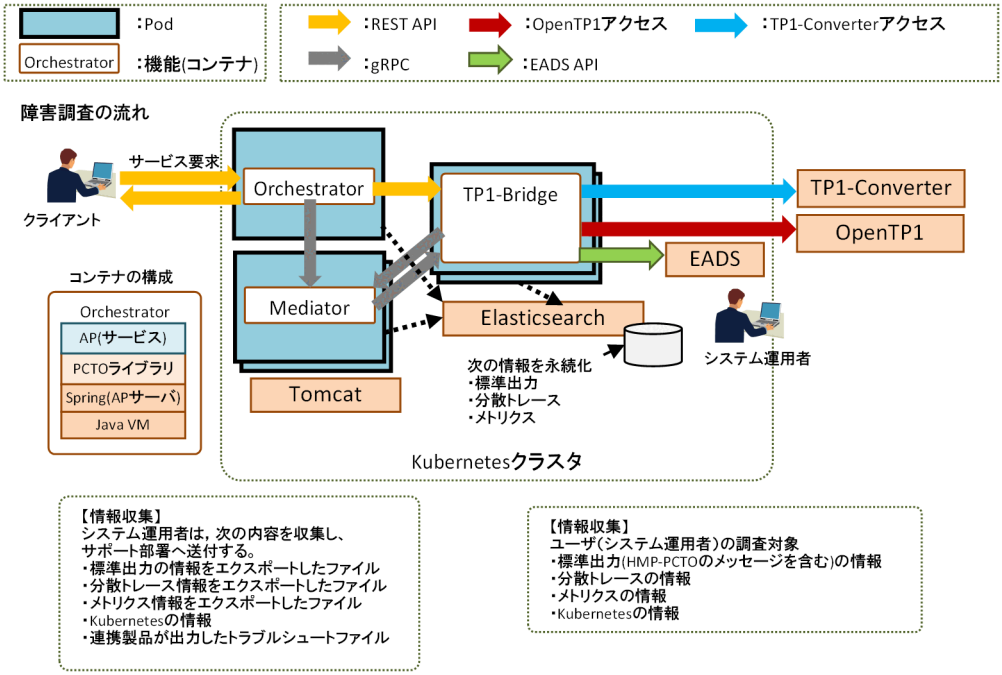


図 10-33　HMP-PCTO 障害発生時の流れ（TP1-Bridge を使用する場合）



(2)　資料取得

トラブルシュートを行うために必要な資料を収集する方法は、「10.6.2　情報の収集方法」を参照してください。

(3)　主なトラブルと調査個所

トラブル（障害）が発生するフェーズは次のとおりです。

- 環境構築時および起動時
  HMP-PCTO の各コンポーネントで使用する Pod・コンテナを構築し、起動した際に障害となった場合
- システム稼働中
  HMP-PCTO の各コンポーネントが稼働中に障害となった場合

障害の種類と調査個所について、次の表に示します。

表 10-54　障害の種類と調査個所

項番	障害の種類	現象	主な原因	調査個所
1	通信エラー	エラーメッセージ出力	サービス要求や各コンポーネント間の通信に異常が起こる	• 標準出力を Kibana などの可視化ツールで確認 • 分散トレースを Jaeger の Web UI で確認
2	通信レスポンス遅延	応答遅延	サービス要求や各コンポーネント間の通信でレスポンスが遅延する	• 分散トレースを Jaeger の Web UI で確認

項番	障害の種類	現象	主な原因	調査個所
				• メトリクスを Prometheus の Web UI などの可視化ツールで確認
3	連携製品のエラー	エラーメッセージ出力	HMP-PCTO から連携製品（DB など）へのアクセス時のエラー	• 標準出力を Kibana などの可視化ツールで確認 • 分散トレースを Jaeger の Web UI で確認
4	連携製品のレスポンス遅延	応答遅延	HMP-PCTO から連携製品（DB など）へのアクセスでレスポンスが遅延する	• 分散トレース • メトリクスを Prometheus の Web UI などの可視化ツールで確認
5	製品内部のエラー	エラーメッセージ出力	HMP-PCTO 各コンポーネントの内部で、製品処理に起因しない要因でエラー ＜例＞ ConsensusLog アクセス時にエラーが返る	• 標準出力を Kibana などの可視化ツールで確認 • 分散トレースを Jaeger の Web UI で確認
6	製品内部のレスポンス遅延	応答遅延	HMP-PCTO 各コンポーネントの内部で、製品処理に起因しない要因でレスポンスが遅延する ＜例＞ ConsensusLog 出力に時間が掛かりレスポンスが遅延する	• 分散トレースを Jaeger の Web UI で確認 • メトリクスを Prometheus の Web UI などの可視化ツールで確認
7	Pod 障害	エラーメッセージ出力	Pod が異常終了する	• 標準出力を Kibana などの可視化ツールで確認 • Kubernetes の詳細情報
8	メモリ不足	エラーメッセージ出力	HMP-PCTO 各コンポーネントが使用できるメモリが不足する	• 標準出力を Kibana などの可視化ツールで確認 • メトリクスを Prometheus の Web UI などの可視化ツールで確認
9	パラメタ設定ミス	エラーメッセージ出力	パラメタに誤りがあり HMP-PCTO が動作しない	• 標準出力を Kibana などの可視化ツールで確認
10	HTTP ヘッダ設定ミス	・HTTP ステータスコード 3xx/4xx/5xx	HTTP ヘッダに誤りがありサービスからエラーが返る	• 標準出力を Kibana などの可視化ツールで確認 • 分散トレースを Jaeger の Web UI で確認
11	製品不良	エラーメッセージ出力	HMP-PCTO 製品に問題があり正常に動作しない ＜例＞ 通信データ不正	• 標準出力を Kibana などの可視化ツールで確認 • 分散トレースを Jaeger の Web UI で確認

また、障害の種類と、ユーザの対策方法について次の表に示します。

表 10–55　障害の種類と対策

項番	障害の種類	現象	主な原因	調査個所	対策
1	HMP-PCTO の Pod が終了しない	KFSG82103-I（未決着トランザクション待機処理が完了）が出ず、Pod が終了しない	未決着トランザクションの情報が取得できない。	Kubernetes の詳細情報（Event 情報）	preStop フックで実行する未決着トランザクション待機スクリプトのリトライ回数、リトライ間隔の設定を見直してください。

(4)　障害個所の特定、原因の調査・対策

ユーザ（システム運用者）、およびサポート部署が行う、障害個所の特定、原因の調査・対策についてそれぞれ次に示します。

(a)　ユーザ（システム運用者）の調査、対策

ユーザ（システム運用者）は、次のそれぞれの情報について調査、および対策を行ってください。

調査順序

ユーザは、次の順序で障害を調査します。

1. Jaeger の Web UI で、分散トレースの情報にエラー、または処理の遅延がないかどうかを確認します。エラーまたは処理の遅延があった場合、発生した個所や時刻を確認します。
2. Kibana などの可視化ツールで発生時間帯の標準出力を確認し、HMP-PCTO のメッセージなどエラーの情報があるかどうか確認します。
3. Prometheus の Web UI などの可視化ツールでメトリクスの情報を確認し、処理の遅延や、スレッドのキューへの滞留があるかどうかを確認します。
4. Pod の障害、または Pod が終了しない事象があった場合は、Kubernetes 上の pod の詳細情報を取得して、Pod の状態や Event 情報を確認します。

調査個所、対策

ユーザは、それぞれ次の個所を調査し、対策を行います。

- 標準出力の情報

  標準出力を Kibana などの可視化ツールで確認してください。標準出力に HMP-PCTO が出力したエラーメッセージがあったときは、「9.　メッセージ」を参照してください。メッセージに対策が記載されている場合は、それに従って対策を行ってください。

  HMP-PCTO が出力したエラーメッセージに出力されたブランチ XID（またはルート XID）の情報から、分散トレースや、連携製品の Tomcat や DB のログのトラブルシュート情報に、同じブランチ XID（またはルート XID）の情報があるかどうか、あった場合はその情報から、サービス依頼元や DB に問題がないかを合わせて確認してください。

  パラメタの誤りがある場合は、修正して再度 HMP-PCTO を起動してください。

- 分散トレースの情報

  分散トレースを Jaeger の Web UI で確認してください。処理結果がエラーとなっている個所、または、処理が遅延している個所を確認してください。また、その処理が HMP-PCTO 製品内部の処理かどうかも確認してください。

  分散トレースに出力されたブランチ XID（またはルート XID）の情報から、連携製品の Tomcat や DB のログのトラブルシュート情報に、同じブランチ XID（またはルート XID）の情報があるかどうか、あった場合はその情報から、サービス依頼元や DB に問題がないかを合わせて確認してください。

  HMP-PCTO 製品内部の処理でエラー、または遅延が発生している場合は、「6.5.1 取得する情報」に記載の情報を収集し、サポート部署へ調査を依頼してください。

  HMP-PCTO の処理ではない場合は、連携製品でのエラー、遅延であるため連携製品のサポートに調査を依頼してください。

- メトリクスの情報

  メトリクスを Prometheus の Web UI などの可視化ツールで確認してください。CPU 使用率・メモリ使用量などの情報から、処理が遅延している個所を確認してください。必要に応じて CPU コアやメモリの増強を行ってください。また、スレッドのキューへの滞留数を示している「executor_queued_tasks」「hmppcto_asynctask_queued_operations」が常時 1 以上になっている場合、同時実行可能なスレッド数を増やしてください。

- Kubernetes の情報

  Kubernetes 上の pod の詳細情報を取得し、環境不正または pod の操作ミスがないか確認してください。また、詳細情報の中の Event 情報から、ライフサイクルイベントハンドラで実行する HMP-PCTO のシェルスクリプトが出力したエラーメッセージがあったときは、「9. メッセージ」を参照してください。メッセージに対策が記載されている場合は、それに従って対策を行ってください。

(b) サポート部署の調査、対策

ユーザから調査依頼を受け、ユーザから資料を入手したサポート部署が調査する内容は次のとおりです。

- 標準出力の情報をエクスポートしたファイル

  HMP-PCTO のエラーメッセージ、またはスタックトレースが出力されていないか確認します。

  また、次に示す、そのほかの連携製品から出力された標準出力の情報からエラーメッセージが出力されていないか確認します。

  - HMP-ADIF
  - Spring
  - Java VM
  - TP1/Client/J
  - SQL Server
  - HiRDB
  - Kubernetes の情報

サポート部署は、ユーザに pod の情報収集した出力結果ファイルを依頼し、その結果から状態を確認します。

- 分散トレース情報をエクスポートしたファイル
  サービス呼び出し関連が知りたい場合や、遅延が発生していて遅延個所を特定したい場合は、分散トレースを確認します。分散トレースでは、トランザクション内の次の実行時間を確認できます。
  - トランザクション（ルート XID の割り当て～合意）
  - サービス（Orchestrator/Entity-Service または Orchestrator/Entity-Module）の実行時間
  - DB アクセス時間
  - EADS API アクセス時間（トライアル版では取得できません）
  - ConsensusLog アクセス時間
  - REST API 通信時間
  - gRPC 通信時間

連携製品側のトラブルシュートが必要になった場合、関連部署へ調査依頼を行います。

- HMP-ADIF
- EADS（トライアル版では不要です）
- uCosminexus Application Runtime with Java for Spring Boot
- TP1/Client/J
- uCosminexus Service Platform
- OpenTP1
- HiRDB

OSS でのトラブルシュートが必要になった場合、関連部署（OSS サポートチーム）へ調査依頼を行います。

- Spring
- PostgreSQL
- SQL Server

通信データの確認

製品不良による電文設定ミス、ユーザの HTTP ヘッダミス（SQL-Participant を使用する場合の先行プリペア設定もれなど）のおそれがある場合は、次の情報を確認します。

- Tomcat アクセスログ
  ヘッダ情報から、REST API 通信の内容を確認します。
- 分散トレース
  gRPC 通信の内容を確認します。

連携製品の調査

DB 障害や EADS 障害などの連携製品については、連携製品側のトラブルシュート情報を取得し、連携製品のサポート部署に調査を依頼します。

- HiRDB のトレース・ログ情報
- EADS のトラブルシュートファイル（トライアル版では取得できません）
- uCosminexus Service Platform のトレース・ログ情報
- OpenTP1 のトレース・ログ情報
- TP1/Client/J のトレース情報

次の情報を、連携製品調査の補助資料として必要に応じて連携製品側へ情報を提供します。

- 分散トレース

  次の項目を確認します。

  ・XA 関数実行時の順序、内容、実行結果を確認します。

  ・EADS API 実行時の順序、内容、実行結果を確認します。なお、トライアル版では確認不要です。
- 標準出力

  エラーメッセージ、またはスタックトレースが出力されているかどうかを確認します。

統計情報確認

通信遅延、プロセススローダウンなどの遅延が発生した場合、ブランチ XID の情報だけでは原因の特定（例えば、通信量やディスク IO が競合したなど）が難しいため、次の情報から、遅延発生時間帯やその時間帯に動いていたブランチ XID を特定します。

- メトリクス

  gRPC 通信量、GC 回数、CPU 情報を取得します。
- 分散トレース

  通信やファイルアクセスで遅延が発生し始めた時間を特定します。

10.7　システム運用に必要となる機能の障害対策

ここでは、HMP-PCTO のシステム運用に必要となる機能機能で障害が発生した場合の対策について説明します。

HMP-PCTO のライブラリ、またはライブラリ以外でシステム運用に必要となる機能で使用する Pod・コンテナで障害が発生した場合についても記載します。

10.7.1　障害の原因および対策

障害の種類と原因、対策について説明します。

(1)　メッセージログ

メッセージログに関する障害の種類と対策を次の表に示します。

表 10–56　メッセージログに関する障害の種類と対策

項番	障害の種類	現象	主な原因	対策
1	メッセージが出力されない	標準出力に HMP-PCTO のメッセージが出力されない（または、Kibana などのログ可視化ツールで HMP-PCTO のメッセージが出力されない）	HMP-PCTO のアプリケーションが動作していない	HMP-PCTO の Pod が動作していることを確認してください。
			誤って削除したなどの要因で、既定のディレクトリに製品提供の logback-spring.xml がない	製品提供の logback-spring.xml を既定のディレクトリに配置してください。
			コンテナの Logging Driver の障害	コンテナの設定で Logging Driver がデフォルトの「json-file」であることを確認してください。
			• Filebeat の Logstash への送信に失敗している • Logstash から Elasticsearch への送信に失敗している	Filebeat、Logstash、Elasticsearch の状態を確認し、再起動してください。
			可視化ツールの設定ミス	可視化ツールの設定を見直してください。
			可視化ツールへアクセスする URL が誤っている	URL が正しいかどうか確認してください。
			Elasticsearch の容量が不足している	各コンポーネントを一旦停止し、Elasticsearch の容量を拡大し、再度各コンポーネントを開始してください。
		イベントリソースに HMP-PCTO のメッセージが出力されない（ライ	ライフサイクルイベントハンドラで HMP-PCTO のシェルスクリプトが実行されていない（シェルスクリプトが	livenessProbe、および readinessProbe が正しく設定されていることを確認してください。

項番	障害の種類	現象	主な原因	対策
		フサイクルイベントハンドラで実行された HMP-PCTO のシェルスクリプトが出力する）	実行されている場合、KFSG82102-I がイベントリソースに出力される）	
2	プロセスダウン	KFSG74103-E が出力された（「ロガーがメッセージプロパティファイルを取得できませんでした。」）	誤って削除したなどの要因で、既定のディレクトリに製品提供の「hmppcto_messages.properties」がない	製品提供の「hmppcto_messages.properties」を既定のディレクトリに配置してください。

(2) 分散トレース

分散トレースに関する障害の種類と対策を次の表に示します。

表 10-57 分散トレースに関する障害の種類と対策

項番	障害の種類	現象	主な原因	対策
1	分散トレースの情報が出力されない	• Jaeger の Web UI に、HMP-PCTO の分散トレースの情報が出力されない • 標準出力に、「WARNING ～ Failed to export traces:～」というメッセージが出力される	HMP-PCTO のアプリケーションが動作していない	HMP-PCTO の Pod が動作していることを確認してください。
			HMP-PCTO の Pod から Jaeger-collector への送信が失敗している（左記の「WARNING～」が出力されている）	HMP-PCTO の、分散トレースの送信先の設定（「(1) jp.co.Hitachi.soft.hmppcto のパラメタ」の tracing.url）で指定する URL が、Jaeger-collector のホスト名・ポートと合っているか確認してください。
			Jaeger-collector から Elasticsearch への送信が失敗している	Jaeger-collector および Elasticsearch の状態を確認し、再起動してください。
			Jaeger-collector で内部キューあふれが発生し、分散トレース情報の drop が発生している	Jaeger-collector を一度アンデプロイします。その後、Jaeger-collector の内部キューの容量の設定（「(1) Helm チャートのパラメタ」の collector.queueSize）で内部キューのサイズを変更し、再度 Jaeger-collector をデプロイしてください。
			Jaeger-query の Pod が動作していない	Jaeger-query の状態を確認し、再起動してください。
			Jaeger の Web UI へアクセスする URL が誤っている	URL が正しいかどうか確認してください。

(3) メトリクス

メトリクスに関する障害の種類と対策を次の表に示します。

表 10-58 メトリクスに関する障害の種類と対策

項番	障害の種類	現象	主な原因	対策
1	メトリクスの情報が出力されない	Prometheus の Web UI に、HMP-PCTO のメトリクスの情報が出力されない	• HMP-PCTO のアプリケーションが動作していない • Prometheus-Server が動作していない	HMP-PCTO や Prometheus-Server の Pod が動作していることを確認してください。
			Prometheus-Server が、HMP-PCTO のメトリクス情報の収集に失敗している	Prometheus-Server の設定（「(1) Helm チャートのパラメタ」）を見直してください。
			Prometheus の Web UI へアクセスする URL が誤っている	URL が正しいかどうか確認してください。
			Prometheus-Server が何らかの要因で再起動した	Elasticsearch に登録されたメトリクスの情報を、Kibana などの可視化ツールで参照してください。
		Elasticsearch にある HMP-PCTO のメトリクスの情報が、Kibana などの可視化ツールで出力されない	• Prometheus-Server から Metricbeat への送信が失敗している • Metricbeat から Elasticsearch への送信が失敗している	Prometheus-Server、Metricbeat、Elasticsearch の状態を確認し、再起動してください。
			可視化ツールの設定ミス	可視化ツールの設定を見直してください。
			可視化ツールへアクセスする URL が誤っている	URL が正しいかどうか確認してください。

付録

付録 A　提供ファイル

HMP-PCTO の提供ファイルについて説明します。

HMP-PCTO は、HMP-PCTO 提供媒体内に各種提供ファイルを格納しています。

「2.　ユーザ責務の Kubernetes アプリケーション開発」および「3.　HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションのデプロイ」の手順に従い、提供媒体内ファイルを各環境に配置して使用してください。

付録 A.1　提供ファイル一覧

提供ファイル一覧を次に示します。

表 A-1　提供ファイル一覧

項番	ディレクトリ/ファイル 「¥」はディレクトリの区切り文字	説明
1	libs¥jp¥co¥Hitachi¥soft¥hmppcto	HMP-PCTO ライブラリ配置用ディレクトリ
2	hmppcto-common¥V.R.S	共通用ライブラリ配置ディレクトリ
3	hmppcto-common-V.R.S.jar	共通用ライブラリファイル
4	hmppcto-common-V.R.S.module	共通用 module ファイル
5	hmppcto-common-V.R.S.pom	共通用 pom ファイル
6	hmppcto-orchestrator¥V.R.S	Orchestrator 用ライブラリ配置ディレクトリ
7	hmppcto-orchestrator-V.R.S.jar	Orchestrator 用ライブラリファイル
8	hmppcto-orchestrator-V.R.S.module	Orchestrator 用 module ファイル
9	hmppcto-orchestrator-V.R.S.pom	Orchestrator 用 pom ファイル
10	hmppcto-entity-service¥V.R.S	Entity-service 用ライブラリ配置ディレクトリ
11	hmppcto-entity-service-V.R.S.jar	Entity-service 用ライブラリファイル
12	hmppcto-entity-service-V.R.S.module	Entity-service 用 module ファイル
13	hmppcto-entity-service-V.R.S.pom	Entity-service 用 pom ファイル
14	hmppcto-jdbc-proxy-driver¥V.R.S	JDBC Proxy Driver 用ライブラリ配置ディレクトリ
15	hmppcto-jdbc-proxy-driver-V.R.S.jar	JDBC Proxy Driver 用ライブラリファイル
16	hmppcto-jdbc-proxy-driver-V.R.S.module	JDBC Proxy Driver 用 module ファイル
17	hmppcto-jdbc-proxy-driver-V.R.S.pom	JDBC Proxy Driver 用 pom ファイル

項番	ディレクトリ/ファイル 「¥」はディレクトリの区切り文字	説明
18	hmppcto-propagation-rest-template¥V.R.S	RestTemplate 用ライブラリ配置ディレクトリ
19	hmppcto-propagation-rest-template-V.R.S.jar	RestTemplate 用ライブラリファイル
20	hmppcto-propagation-rest-template-V.R.S.module	RestTemplate 用 module ファイル
21	hmppcto-propagation-rest-template-V.R.S.pom	RestTemplate 用 pom ファイル
22	hmppcto-propagation-isc-imperative¥V.R.S	ISC コマンド用ライブラリ配置ディレクトリ
23	hmppcto-propagation-isc-imperative-V.R.S.jar	ISC コマンド用ライブラリファイル
24	hmppcto-propagation-isc-imperative-V.R.S.module	ISC コマンド用 module ファイル
25	hmppcto-propagation-isc-imperative-V.R.S.pom	ISC コマンド用 pom ファイル
26	hmppcto-mediator¥V.R.S	Mediator 用ライブラリ配置ディレクトリ
27	hmppcto-mediator-V.R.S.jar	Mediator 用ライブラリファイル
28	hmppcto-mediator-V.R.S.module	Mediator 用 module ファイル
29	hmppcto-mediator-V.R.S.pom	Mediator 用 pom ファイル
30	libhmppcto-mediator-V.R.S.so	Mediator 用のダイレクト IO 用ライブラリファイル
31	hmppcto-participant-common¥V.R.S	SQL-Participant/TCC-Participant 共通用ライブラリ配置ディレクトリ
32	hmppcto-participant-common-V.R.S.jar	SQL-Participant/TCC-Participant 共通用ライブラリファイル
33	hmppcto-participant-common-V.R.S.module	SQL-Participant/TCC-Participant 共通用 module ファイル
34	hmppcto-participant-common-V.R.S.pom	SQL-Participant/TCC-Participant 共通用 pom ファイル
35	hmppcto-participant-sql¥V.R.S	SQL-Participant 用ライブラリ配置ディレクトリ
36	hmppcto-participant-sql-V.R.S.jar	SQL-Participant 用ライブラリファイル
37	hmppcto-participant-sql-V.R.S.module	SQL-Participant 用 module ファイル
38	hmppcto-participant-sql-V.R.S.pom	SQL-Participant 用 pom ファイル
39	hmppcto-participant-tcc¥V.R.S	TCC-Participant 用ライブラリ配置ディレクトリ
40	hmppcto-participant-tcc-V.R.S.jar	TCC-Participant 用ライブラリファイル

項番	ディレクトリ/ファイル 「¥」はディレクトリの区切り文字	説明
41	hmppcto-participant-tcc-V.R.S.module	TCC-Participant 用 module ファイル
42	hmppcto-participant-tcc-V.R.S.pom	TCC-Participant 用 pom ファイル
43	hmppcto-lb¥V.R.S	ロードバランス用ライブラリ配置ディレクトリ
44	hmppcto-lb-V.R.S.jar	ロードバランス用ライブラリ
45	hmppcto-lb-V.R.S.module	ロードバランス用 module ファイル
46	hmppcto-lb-V.R.S.pom	ロードバランス用 pom ファイル
47	hmppcto-tx-paxos-manager¥V.R.S	トランザクション制御分散合意用ライブラリ配置ディレクトリ
48	hmppcto-tx-paxos-manager-V.R.S.jar	トランザクション制御分散合意用ライブラリ
49	hmppcto-tx-paxos-manager-V.R.S.module	トランザクション制御分散合意用 module ファイル
50	hmppcto-tx-paxos-manager-V.R.S.pom	トランザクション制御分散合意用 pom ファイル
51	hmppcto-participant-adapter¥V.R.S	Participant アダプタ用ライブラリ配置ディレクトリ
52	hmppcto-participant-adapter-V.R.S.jar	Participant アダプタ用ライブラリ
53	hmppcto-participant-adapter-V.R.S.module	Participant アダプタ用 module ファイル
54	hmppcto-participant-adapter-V.R.S.pom	Participant アダプタ用 pom ファイル
55	hmppcto-tp1-bridge ¥V.R.S	TP1-Bridge 用ライブラリ配置ディレクトリ
56	hmppcto-tp1-bridge-V.R.S.jar	TP1-Bridge 用ライブラリ
57	hmppcto-tp1-bridge-V.R.S.module	TP1-Bridge 用 module ファイル
58	hmppcto-tp1-bridge-V.R.S.pom	TP1-Bridge 用 pom ファイル
59	java-projects	HMP-PCTO Java プロジェクト配置用ディレクトリ
60	mediator	Mediator 用 Java プロジェクトディレクトリ
61	src¥main¥java¥jp¥co¥Hitachi¥soft¥hmppcto¥Application.java	Mediator 用 Java ソースファイル
62	build.gradle	Mediator 用 build.gradle ファイル
63	settings.gradle	Mediator 用 settings.gradle ファイル
64	gradle.properties	Mediator 用 gradle.properties ファイル
65	sql-participant	SQL-Participant 用 Java プロジェクトディレクトリ

項番	ディレクトリ/ファイル 「¥」はディレクトリの区切り文字	説明
66	src¥main¥java¥jp¥co¥Hitachi¥soft¥hmppcto¥Application.java	SQL-Participant 用 Java ソースファイル
67	build.gradle	SQL-Participant 用 build.gradle ファイル
68	settings.gradle	SQL-Participant 用 settings.gradle ファイル
69	gradle.properties	SQL-Participant 用 gradle.properties ファイル
70	tp1-bridge	TP1-Bridge 用 Java プロジェクトディレクトリ
71	src¥main¥java¥jp¥co¥Hitachi¥soft¥hmppcto¥Application.java	TP1-Bridge 用 Java ソースファイル
72	build.gradle	TP1-Bridge 用 build.gradle ファイル
73	settings.gradle	TP1-Bridge 用 settings.gradle ファイル
74	gradle.properties	TP1-Bridge 用 gradle.properties ファイル
75	libs	TP1-Bridge 用外部ライブラリ格納ディレクトリ
76	scripts	HMP-PCTO スクリプトファイル配置用ディレクトリ
77	dependency_check.sh	デプロイ依存関係チェックスクリプト
78	health_check.sh	トランザクションヘルスチェックスクリプト
79	wait_transactions.sh	未決着トランザクション待機スクリプト
80	templates	HMP-PCTO テンプレートファイル配置用ディレクトリ
81	container-files	Containerfile 配置ディレクトリ
82	tp1-bridge	TP1-Bridge 用 Containerfile 配置ディレクトリ
83	normal	TP1-Bridge 用 Containerfile 配置ディレクトリ（uCosminexus Application Runtime with Java for Spring Boot の修正パッチがない場合）
84	Containerfile	TP1-Bridge 用 Containerfile（uCosminexus Application Runtime with Java for Spring Boot の修正パッチがない場合）
85	patch	TP1-Bridge 用 Containerfile 配置ディレクトリ（uCosminexus Application Runtime with Java for Spring Boot の修正パッチがある場合）
86	Containerfile	TP1-Bridge 用 Containerfile（uCosminexus Application Runtime with Java for Spring Boot の修正パッチがある場合）
87	mediator	Mediator 用 Containerfile 配置ディレクトリ
88	normal	Mediator 用 Containerfile 配置ディレクトリ（uCosminexus Application Runtime with Java for Spring Boot の修正パッチがない場合）

項番	ディレクトリ/ファイル 「¥」はディレクトリの区切り文字	説明
89	Containerfile	Mediator 用 Containerfile（uCosminexus Application Runtime with Java for Spring Boot の修正パッチがない場合）
90	patch	Mediator 用 Containerfile 配置ディレクトリ（uCosminexus Application Runtime with Java for Spring Boot の修正パッチがある場合）
91	Containerfile	Mediator 用 Containerfile（uCosminexus Application Runtime with Java for Spring Boot の修正パッチがある場合）
92	dependency-checker	Dependency-Checker 用 Containerfile 配置ディレクトリ
93	Containerfile	Dependency-Checker 用 Containerfile
94	helm-charts	HMP-PCTOHelm チャート配置ディレクトリ
95	mediator-V.R.S.tgz	Mediator 用 Helm チャートパッケージファイル
96	tp1-bridge-V.R.S.tgz	TP1-Bridge 用 Helm チャートパッケージファイル
97	filebeat-V.R.S.tgz	Filebeat の Helm チャートのパッケージファイル
98	jaeger-query-V.R.S.tgz	Jaeger-query の Helm チャートのパッケージファイル
99	jaeger-collector-V.R.S.tgz	Jaeger-collector の Helm チャートのパッケージファイル
100	prometheus-V.R.S.tgz	Prometheus の Helm チャートのパッケージファイル
101	metricbeat-V.R.S.tgz	Metricbeat の Helm チャートのパッケージファイル
102	elasticsearch-V.R.S.tgz	Elasticsearch の Helm チャートのパッケージファイル
103	logstash-V.R.S.tgz	Logstash の Helm チャートのパッケージファイル
104	bundled-softwares	同梱ソフトウェア配置ディレクトリ
105	EADS-client	Hitachi Elastic Application Data Store Client for Java インストーラ配置ディレクトリ
106	X64LIN	Hitachi Elastic Application Data Store Client for Java インストーラ
107	PACK_TAR.Z	Hitachi Elastic Application Data Store Client for Java の修正パッチ。修正パッチがない場合はこのファイルは提供物に含まない。
108	TP1-Converter	TP1-Converter 用ファイル配置ディレクトリ
109	lib	TP1-Converter 用ライブラリ配置ディレクトリ
110	tp1converter.war	TP1-Converter 用アプリケーションライブラリ
111	usrconf	TP1-Converter 用 J2EE サーバ定義ファイル配置ディレクトリ

項番	ディレクトリ/ファイル 「¥」はディレクトリの区切り文字			説明
112			usrconf.cfg	J2EE サーバ用オプション定義ファイルのテンプレート
113			usrconf.properties	J2EE サーバ用ユーザプロパティファイルのテンプレート

V.R.S には、バージョンを設定します。S は数値に変換します。

（例）

Ver 01-00 開発時：1.0.0

Ver 01-02-/A 開発時：1.2.1

付録 B　HMP-PCTO のメトリクス取得項目

ここでは、HMP-PCTO で取得するメトリクス項目について説明します。

表 B-1　メトリクスの Meter 種別

Meter 種別	説明
Timer	多数の短時間実行イベント
Counter	単調に増加する値
Gauge	上下する値
MultiGauge	複数の Gauge

Prometheus サーバの標準の Web UI で次のメトリクス項目名をクエリとして入力することでメトリクス情報が画面に出力されます。

表 B-2　HMP-PCTO で取得するメトリクス項目

メトリクス取得項目	説明	Meter 種別	単位
system_cpu_usage	システム（Kubernetes の場合、コンテナ環境）の CPU 使用率です。 使用率は 0.0～1.0(double 型)の範囲で表されます。	Gauge	-
process_cpu_usage	Java VM プロセスの CPU 使用率です。 使用率は 0.0～1.0(double 型)の範囲で表されます。	Gauge	-
jvm_memory_max_bytes	Java VM で使用できるメモリの最大量です。	Gauge	bytes
jvm_memory_used_bytes	Java VM の使用メモリ量です。	Gauge	bytes
process_files_max_files	ファイル記述子の最大数です。	Gauge	files
process_files_open_files	開いているファイル記述子の数です。	Gauge	files
jvm_gc_memory_allocated_bytes_total	ある GC 実施後から、次の GC に実施されたの間での、Java ヒープ New 世代領域のメモリプールサイズの増加量です。	Counter	bytes
jvm_gc_memory_promoted_bytes_total	ある GC 実施前と GC 実施後での、Java ヒープ Old 世代領域のメモリプールサイズの増加量です。	Counter	bytes
executor_active_threads	アクティブにタスクを実行しているユーザのスレッドのおおよその数です。	Gauge	tasks
executor_queued_tasks	実行待ちのタスクのおおよその数です。	Gauge	tasks
grpc_server_processing_duration_seconds	gRPC サーバが呼び出し完了に掛かった時間	Timer	seconds
grpc_server_processing_duration_seconds_count			

メトリクス取得項目	説明	Meter 種別	単位
grpc_server_processing_duration_seconds_max			
grpc_server_processing_duration_seconds_sum			
grpc_client_processing_duration_seconds_count	gRPC クライアントが通信完了するまでに掛かった合計時間	Timer	seconds
grpc_client_processing_duration_seconds_max			
grpc_client_processing_duration_seconds_sum			
hmppcto_participant_transactions_inProcess_operations	処理中のトランザクションブランチ数	MultiGauge	operations
hmppcto_participant_transactions_terminated_operations	完了したトランザクションブランチ数	MultiGauge	operations
hmppcto_participant_transaction_duration_seconds	完了したトランザクションブランチの処理時間の統計情報	Timer	seconds
hmppcto_participant_transaction_duration_seconds_count			
hmppcto_participant_transaction_duration_seconds_max			
hmppcto_participant_transaction_duration_seconds_sum			
hmppcto_participant_transaction_duration_seconds_bucket			
hmppcto_asynctask_duration_seconds	HMP-PCTO が内部で非同期で送信する処理の所要時間	Timer	seconds
hmppcto_asynctask_duration_seconds_count			
hmppcto_asynctask_duration_seconds_max			
hmppcto_asynctask_duration_seconds_sum			
hmppcto_asynctask_operations_total	HMP-PCTO が内部で非同期で送信する処理の数	MultiGauge	operations
hmppcto_asynctask_queued_operations	キュー内にあり、まだ実行されていない非同期タスクの数	MultiGauge	operations
hmppcto_mediator_consensus_duration_seconds	ConsensusLog アクセス所要時間	Timer	seconds

メトリクス取得項目	説明	Meter 種別	単位
hmppcto_mediator_consensus_duration_seconds_bucket			
hmppcto_mediator_consensus_duration_seconds_count			
hmppcto_mediator_consensus_duration_seconds_max			
hmppcto_mediator_consensus_duration_seconds_sum			
hmppcto_participant_eads_duration_seconds	EADS API の所要時間（トライアル版では出力できません）	Timer	seconds
hmppcto_participant_eads_duration_seconds_bucket			
hmppcto_participant_eads_duration_seconds_count			
hmppcto_participant_eads_duration_seconds_max			
hmppcto_participant_eads_duration_seconds_sum			

付録 C　このマニュアルの参考情報

付録 C.1　関連ドキュメント

このマニュアルの関連ドキュメントを次に示します。必要に応じてお読みください。

(1)　取扱説明書

取扱説明書「Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator 01-02　テクニカルガイド」(062469)

次の内容について説明しています。

- システムの概要
- Orchestrator/Entity-Service の機能
- SQL-Participant の機能
- TCC-Participant の機能
- TP1-Bridge の機能
- コンポーネント共通機能

なお、このマニュアルでは、取扱説明書名について次のように省略して表記しています。

取扱説明書名	表記
Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator 01-02　テクニカルガイド	HMP-PCTO テクニカルガイド

(2)　マニュアル

- OpenTP1 Version 7 分散トランザクション処理機能　OpenTP1　システム定義（3000-3-D52）
- OpenTP1 Version 7 分散トランザクション処理機能　OpenTP1　運用と操作（3000-3-D53）
- OpenTP1 Version 7 分散トランザクション処理機能　OpenTP1　クライアント使用の手引　TP1/Client/J 編（3000-3-D59）
- インメモリデータグリッド　Hitachi Elastic Application Data Store ユーザーズガイド（3020-3-V22）
- ノンストップデータベース　HiRDB Version 10 システム定義（UNIX(R)用）（3020-6-554）
- ノンストップデータベース　HiRDB Version 10 システム定義（Windows(R)用）（3020-6-555）
- ノンストップデータベース　HiRDB Version 10 システム運用ガイド（UNIX(R)用）（3020-6-556）
- ノンストップデータベース　HiRDB Version 10 システム運用ガイド（Windows(R)用）（3020-6-557）
- ノンストップデータベース　HiRDB Version 10 コマンドリファレンス（UNIX(R)用）（3020-6-558）

- ノンストップデータベース　HiRDB Version 10 コマンドリファレンス（Windows(R)用）（3020-6-559）
- ノンストップデータベース　HiRDB Version 10 UAP 開発ガイド（3020-6-560）
- Cosminexus V11 アプリケーションサーバ 機能解説 運用／監視／連携編（3021-3-J10）
- Cosminexus V11 アプリケーションサーバ リファレンス コマンド編（3021-3-J15）
- Cosminexus V11 アプリケーションサーバ リファレンス 定義編(サーバ定義)（3021-3-J16）
- Cosminexus V11 BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編（3021-3-J43）
- Cosminexus V11 BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド（3021-3-J45）
- uCosminexus Application Runtime for Spring Boot ユーザーズガイド（3021-3-K03）

このマニュアルでは、関連マニュアルを次のように省略して表記しています。

マニュアル名	表記
OpenTP1 Version 7 分散トランザクション処理機能 OpenTP1　システム定義	OpenTP1 システム定義
OpenTP1 Version 7 分散トランザクション処理機能 OpenTP1　運用と操作	OpenTP1 運用と操作
OpenTP1 Version 7 分散トランザクション処理機能 OpenTP1　クライアント使用の手引　TP1/Client/J 編	OpenTP1 クライアント使用の手引 TP1/Client/J 編
インメモリデータグリッド　Hitachi Elastic Application Data Store ユーザーズガイド	Elastic Application Data Store ユーザーズガイド
ノンストップデータベース　HiRDB Version 10 システム定義（UNIX(R)用）	HiRDB システム定義
ノンストップデータベース　HiRDB Version 10 システム定義（Windows(R)用）	
ノンストップデータベース　HiRDB Version 10 システム運用ガイド（UNIX(R)用）	HiRDB システム運用ガイド
ノンストップデータベース　HiRDB Version 10 システム運用ガイド（Windows(R)用）	
ノンストップデータベース　HiRDB Version 10 コマンドリファレンス（UNIX(R)用）	HiRDB コマンドリファレンス
ノンストップデータベース　HiRDB Version 10 コマンドリファレンス（Windows(R)用）	
ノンストップデータベース　HiRDB Version 10 UAP 開発ガイド	HiRDB UAP 開発ガイド
Cosminexus V11 アプリケーションサーバ 機能解説 運用／監視／連携編	Cosminexus アプリケーションサーバ 機能解説 運用／監視／連携編

マニュアル名	表記
Cosminexus V11 アプリケーションサーバ リファレンス コマンド編	Cosminexus アプリケーションサーバ リファレンス コマンド編
Cosminexus V11 アプリケーションサーバ リファレンス 定義編(サーバ定義)	Cosminexus アプリケーションサーバ リファレンス 定義編(サーバ定義)
Cosminexus V11 BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編	Cosminexus BPM/ESB 基盤 サービスプラットフォーム 開発ガイド 基本開発編
Cosminexus V11 BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド	Cosminexus BPM/ESB 基盤 サービスプラットフォーム システム構築・運用ガイド

付録 C.2　マニュアルで使用している略語

このマニュアルでの表記と正式名称を次の表に示します。

表記	製品名
Amazon EBS	Amazon Elastic Block Store
Amazon EFS	Amazon Elastic File System
EADS	Hitachi Elastic Application Data Store
HMP-ADIF	Hitachi Microservices Platform - Application Distributed Integration Framework
HMP-PCTO	Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator
RHEL/UBI 8	Red Hat(R) Universal Base Image 8
Tomcat	Apache Tomcat(R)

付録 C.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
ACID	Atomicity Consistency Isolation Durability
AP	Application Program
API	Application Programming Interface
BLOB	Binary Large OBject
CA	Certification Authority
CLI	Command Line Interface
CN	Common Name

英略語	英字での表記
CPU	Central Processing Unit
DB	Data Base
DBMS	Database Management System
DDL	Data Definition Language
DI	Dependency Injection
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
GC	Garbage Collection
HTTP	Hyper Text Transfer Protocol
ID	Identifier
ILM	Information Life cycle Management
IP	Internet Protocol
ISC	Inter-Service Communication
JAR	Java Archive
Java SE	Java Platform, Standard Edition
Java VM	Java Virtual Machine
JDBC	Java Database Connectivity
JDK	Java Development Kit
JKS	Java Keystore
JSON	JavaScript Object Notation
KVS	Key-Value store
NFS	Network File System
OS	Operating System
PEM	Privacy Enhanced Mail
PV	Persistent Volume
PVC	Persistent Volume Claim
RDBMS	Relational DataBase Management System
REST	Representational State Transfer
RPC	Remote Procedure Call
RPM	RPM Package Manager
SCC	Security Context Constraints

英略語	英字での表記
SPP	Service Providing Program
TCC	Try、Confirm、Cancel
TCP	Transmission Control Protocol
TLS	Transport Layer Security

付録 C.4　KB（キロバイト）などの単位表記について

1KB（キロバイト）、1MB（メガバイト）、1GB（ギガバイト）、1TB（テラバイト）はそれぞれ 1,024 バイト、$1{,}024^2$ バイト、$1{,}024^3$ バイト、$1{,}024^4$ バイトです。

付録 D　HMP-PCTO で使用する用語

ここでは、HMP-PCTO で使用する用語について説明します。

表 D-1　用語解説

用語	説明
3/4 合意	• ThreeQuarters 合意のこと •「よんぶんのさんごうい」と呼ぶ
BranchQualifier	• トランザクション内のブランチを一意に識別するための識別子 • X/Open トランザクション識別子 XID 構造の Java マッピングの BranchQualifier
Consensus Acceptor	• リソース更新可否の決着結果の永続化機能のこと • Termination Judge から通知された投票結果を永続化したあと、投票結果を、Consensus Learner に通知する機能を持つ
Consensus Learner	• 分散合意習得（機能）のこと • 複数の Mediator の合意を習得し（受信し）、一貫した合意結果を得る機能 • Participant 内の機能の一つ
ConsensusId	• 分散合意処理のレイヤでトランザクションを一意にするための識別子 • RootXID を文字列化して生成する
ConsensusLog	Mediator で保持しているリソース更新可否の決着結果
DB Adaptor	• データベースアダプタのこと • アプリケーションの要求に従ってリソースへの SQL 実行や X/Open で規定する XA 仕様に則った XA インタフェース実行を行う機能を持つ
EADS	• Hitachi Elastic Application Data Store のこと • 分散 KVS • paxos commit では補助合意に使用する
Entity-Module	TCC-Participant は Entity-Service の機能の一部を利用しており、TCC-Participant 内で利用されている Entity-Service の機能を Entity-Module と呼ぶ
Entity-Service	データベースなどのリソースの参照、更新業務を行うアプリケーション、およびアプリケーションに組み込んで Participant と連携する HMP-PCTO のライブラリの総称
Ext Cons	• 補助合意のこと • External Consensus の略
External Consensus	補助合意のこと
HMP-ADIF	• Hitachi Microservices Platform-Application Distributed Integration Framework の略称 • GlobalLogic 社の Microservices Accelerator を導入した製品 • マイクロサービスアーキテクチャを用いたシステムのアプリケーション開発を容易にするフレームワーク
HMP-PCTO	• Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator の略称 • ミッションクリティカルシステムにおける分散トランザクション制御を、マイクロサービスアーキテクチャを用いたシステムで容易に実現するフレームワーク

用語	説明
Java アプリケーション	• HMP-PCTO のライブラリを使用し java 言語で作成するユーザアプリケーション • ユーザが作成したアプリケーションで、ユーザ任意の方法で JAR ファイルにビルドしたもの
JDBC Proxy Driver	• アプリケーションやフレームワークからの JDBC Driver のメソッドの呼び出しを、Participant と連携して透過的にベンダネイティブな JDBC Driver に伝える Driver • Entity-Service の HMP-PCTO のライブラリの機能
Kubernetes アプリケーション	• Kubernetes クラスタにデプロイして実行するコンテナ化されたアプリケーション • Pod、Deployment、StatefulSet などのワークロードリソースとして作成したもの
Learner	Consensus Learner のこと
Mediator	• Paxos Commit で、トランザクション制御を行うための構成要素の一つであり、Consensus Acceptor および Termination Judge の機能を持つ • Consensus Acceptor および Termination Judge は、Participant の Termination Proposer および Consensus Learner と連動してクウォーラム型分散合意を実現する
Orchestrator	Client システムからの要求を受け付けて、Entity-Service を利用して業務を実行するアプリケーション、およびアプリケーションに組み込んで Mediator、Entity-Service と連携する HMP-PCTO のライブラリの総称
Participant	• Paxos Commit で、トランザクション制御を行うための構成要素の一つであり、Resource Adapter、Termination Proposer および Consensus Learner の機能を持つ • Termination Proposer および Consensus Learner は、Mediator の Consensus Acceptor および Termination Judge と連動してクウォーラム型分散合意を実現する • SQL-Participant、TCC-Participant、および TP1-Bridge のことを指す
PrivilegedValue 合意	• 分散合意習得方式の一つ • PrivilegedValue パラメタ指定値になるように合意する分散合意習得方式
PV 合意	PrivilegedValue 合意のこと
RAS	• RASIS（Reliability Availability Serviceability Integrity Security）の省略形。RASIS とは、コンピュータシステムの信頼性を評価するときチェックする項目である Reliability（信頼性）、Availability（可用性）、Serviceability（保守性）、Integrity（保全性）、Security（機密性）の頭文字で構成された用語 • HMP-PCTO のシステム運用に必要となる機能のこと。障害対策のための情報や HMP-PCTO の性能を評価するための情報を取得する
SQL-Adapter	• JDBC Driver を介して SQL を実行する • XA インタフェース（xa_start、xa_end、xa_prepare、xa_commit、xa_rollback、および xa_recover）を使用して RDBMS に接続し、リソースを管理する
SQL-Participant	• X/Open の XA 仕様に準拠した Participant • SQL-Adapter および Tx+Paxos-Manager によって構成される
TCC	• REST API サービスに対する分散トランザクションの設計パターンの一つ • Try、Confirm、Cancel の機能を持つ。これらの頭文字から TCC と呼ばれる
TCC-Participant	• TCC パターンを利用した Participant • Try、Confirm、Cancel、Recover に対応したインタフェースを提供し、ユーザが実装することでリソースの管理を行う

用語	説明
Termination Judge	• ターミネーションジャッジ（トランザクション終了審査者）のこと • グローバルトランザクションに含まれる Participant から通知された投票内容を記録し、すべての Participant が Commit 投票（リソースを更新できる）であれば Commit に、1 つでも Rollback 投票（リソースを更新できない）の Participant があれば Rollback に、投票結果を集約する機能を持つ • 集約した投票結果は、Consensus Acceptor に通知する
Termination Proposer	• ターミネーションプロポーザ（トランザクション終了提案者）のこと • リソースに対する SQL 実行の更新可否を確認するため、X/Open が規定する XA 仕様に則って Prepare を実行し、その結果に応じた投票（Commit 投票/Rollback 投票）を Mediator に通知する機能を持つ
ThreeQuarters 合意	• 分散合意習得方式の一つ • 1/4 の Mediator の故障を許容する分散合意習得方式
TP1-Acceptor	• HMP-PCTO 上の AP から、OpenTP1 の SPP に対するリクエストを受け付けるサービス
TP1-Adapter	• TP1/Client/J を介して OpenTP1 のサービスを呼び出す機能 • XA インタフェース（xa_start、xa_end、xa_prepare、xa_commit、xa_rollback、および xa_recover）を使用して OpenTP1 に接続し、トランザクションを制御する
TP1-Bridge	• HMP-PCTO 上の AP から OpenTP1 のサービスの呼び出しを橋渡しする機能 • Entity-Service、TP1-Acceptor、TP1-Converter、TP1-Adapter、および Tx+Paxos-Manager によって構成される
TP1-Converter	• HMP-PCTO 上の AP が扱う JSON 形式のデータと、OpenTP1 の AP が扱うバイナリ形式のデータの相互変換をする • uCosminexus Service Platform 上で動作するサーブレット
Tx+Paxos-Manager	次の機能を持つ SQL-Participant および TP1-Bridge の構成要素 • X/Open の XA 仕様に準拠したトランザクションマネージャの機能 • Participant の機能
XID	• トランザクションを一意に識別するための識別子 • HMP-PCTO でのデータ構造は、X/Open トランザクション識別子 XID 構造の Java マッピングで表現する
コンテナ化されたアプリケーション	Java アプリケーションや HMP-PCTO および関連するソフトウェアのファイルなどを、コンテナイメージとしてビルドしたもの
ルート XID（RootXID）	• XID のうち、グローバルトランザクションの基底となる識別子 • X/Open トランザクション識別子 XID 構造の Java マッピングの GlobalTransactionId に相当する
高速合意	補助合意不要で一貫した合意結果を獲得すること
分散合意（機能）	• paxos commit では複数の Mediator と複数の Participant（Consensus Learner）で実現する • トランザクションの状態を複数の Mediator に分散冗長化し、一部の Mediator の故障を許容した上で（可用性向上）、複数の Mediator から得られるトランザクションの状態を、1 以上の Participant（Consensus Learner）で、一貫した合意結果として獲得し共有する機能

用語	説明
分散合意習得（機能）	Consensus Learner のこと
分散合意習得方式	• Consensus Learner で一意となる合意結果を得るための方式 • paxos commit では ThreeQuarters 合意と PrivilegedValue 合意がある
補助合意	• Consensus Learner で、複数の Mediator の合意に不一致がある場合に Consensus Learner 間で合意結果が一貫するように合意調整する機能 • Consensus Learner で、次の条件をすべて満たした場合に、合意結果が一貫するように合意調整する機能 ・一部の Mediator の合意が受信できない ・受信できた合意だけでは合意結果の一貫性が担保できない
補助合意試行間隔タイマ	Consensus Learner の補助合意の試行（補助合意可否の判定および補助合意リトライ）間隔タイマ

索引

A

absolute(int row)メソッド〔PctoResultSet クラス〕 557

add.class.path パラメタ〔uCosminexus Application Server の usrconf.cfg〕 742

addBatch()メソッド〔PctoPreparedStatement クラス〕 505

addBatch(java.lang.String sql)メソッド〔PctoStatement クラス〕 487

affinity パラメタ〔Elasticsearch〕 728

applicable(java.lang.String communicationType)メソッド〔TxPropagationIscImperativePlugin クラス〕 459

atomic_update オペランド〔OpenTP1 システムサービス定義の HMP-PCTO のトランザクションに参加する SPP のユーザサービス定義〕 739

available()メソッド〔PctoInputStream クラス〕 613

B

branch.qualifier.list パラメタ〔Entity-Service/Entity-Module〕 656

branch.qualifier.list パラメタ〔Orchestrator〕 654

branch.qualifier.propagation-order パラメタ〔Entity-Service/Entity-Module〕 656

branch.qualifier.propagation-order パラメタ〔Orchestrator〕 654

C

client.proposer.type パラメタ〔Entity-Service/Entity-Module〕 656

close()メソッド〔PctoConnection クラス〕 473

close()メソッド〔PctoInputStream クラス〕 613

close()メソッド〔PctoReader クラス〕 621

close()メソッド〔PctoResultSet クラス〕 558

close()メソッド〔PctoStatement クラス〕 488

closeOnCompletion()メソッド〔PctoStatement クラス〕 489

collector.numShards パラメタ〔Jaeger-collector〕 717

collector.numWorkers パラメタ〔Jaeger-collector〕 716

collector.queueSize パラメタ〔Jaeger-collector〕 716

consensusLog.io.blockSize パラメタ〔Mediator〕 679

consensusLog.io.queueSize パラメタ〔Mediator〕 678

consensusLog.io.threadPoolSize パラメタ〔Mediator〕 678

consensusLog.io.watchInSeconds パラメタ〔Mediator〕 679

consensusLog.size パラメタ〔Mediator〕 678

consensusLog.storageClassName パラメタ〔Mediator〕 679

converter.connectTime パラメタ〔TP1-Bridge〕 688

converter.host パラメタ〔TP1-Bridge〕 687

converter.port パラメタ〔TP1-Bridge〕 687

converter.watchTime パラメタ〔TP1-Bridge〕 688

createArrayOf(java.lang.String typeName, java.lang.Object[] elements)メソッド〔PctoConnection クラス〕 473

createSQLXML()メソッド〔PctoConnection クラス〕 474

createStatement()メソッド〔PctoConnection クラス〕 475

createStatement(int resultSetType, int resultSetConcurrency)メソッド〔PctoConnection クラス〕 476

D

data.storageClassName パラメタ〔Elasticsearch〕 729

data.storageSize パラメタ〔Elasticsearch〕 730

dependencyCheck.connectTimeoutSeconds パラメタ〔Ext-Cons〕 707

dependencyCheck.connectTimeoutSeconds パラメタ〔Filebeat〕 711
dependencyCheck.connectTimeoutSeconds パラメタ〔Jaeger-collector〕 717
dependencyCheck.connectTimeoutSeconds パラメタ〔Jaeger-query〕 714
dependencyCheck.connectTimeoutSeconds パラメタ〔Logstash〕 734
dependencyCheck.connectTimeoutSeconds パラメタ〔Mediator〕 674
dependencyCheck.connectTimeoutSeconds パラメタ〔Metricbeat〕 726
dependencyCheck.connectTimeoutSeconds パラメタ〔Prometheus〕 723
dependencyCheck.connectTimeoutSeconds パラメタ〔TP1-Bridge〕 685
dependencyCheck.enabled パラメタ〔Filebeat〕 710
dependencyCheck.enabled パラメタ〔Jaeger-collector〕 717
dependencyCheck.enabled パラメタ〔Jaeger-query〕 713
dependencyCheck.enabled パラメタ〔Logstash〕 734
dependencyCheck.enabled パラメタ〔Metricbeat〕 725
dependencyCheck.enabled パラメタ〔Prometheus〕 723
dependencyCheck.imagePullPolicy パラメタ〔Ext-Cons〕 706
dependencyCheck.imagePullPolicy パラメタ〔Filebeat〕 710
dependencyCheck.imagePullPolicy パラメタ〔Jaeger-collector〕 717
dependencyCheck.imagePullPolicy パラメタ〔Jaeger-query〕 713
dependencyCheck.imagePullPolicy パラメタ〔Logstash〕 734
dependencyCheck.imagePullPolicy パラメタ〔Mediator〕 674
dependencyCheck.imagePullPolicy パラメタ〔Metricbeat〕 726
dependencyCheck.imagePullPolicy パラメタ〔Prometheus〕 723
dependencyCheck.imagePullPolicy パラメタ〔TP1-Bridge〕 684
dependencyCheck.image パラメタ〔Ext-Cons〕 706
dependencyCheck.image パラメタ〔Filebeat〕 710
dependencyCheck.image パラメタ〔Jaeger-collector〕 717
dependencyCheck.image パラメタ〔Jaeger-query〕 713
dependencyCheck.image パラメタ〔Logstash〕 734
dependencyCheck.image パラメタ〔Mediator〕 674
dependencyCheck.image パラメタ〔Metricbeat〕 725
dependencyCheck.image パラメタ〔Prometheus〕 723
dependencyCheck.image パラメタ〔TP1-Bridge〕 684
dependencyCheck.readTimeoutSeconds パラメタ〔Ext-Cons〕 707
dependencyCheck.readTimeoutSeconds パラメタ〔Filebeat〕 711
dependencyCheck.readTimeoutSeconds パラメタ〔Jaeger-collector〕 717
dependencyCheck.readTimeoutSeconds パラメタ〔Jaeger-query〕 714
dependencyCheck.readTimeoutSeconds パラメタ〔Logstash〕 734
dependencyCheck.readTimeoutSeconds パラメタ〔Mediator〕 674
dependencyCheck.readTimeoutSeconds パラメタ〔Metricbeat〕 726
dependencyCheck.readTimeoutSeconds パラメタ〔Prometheus〕 723
dependencyCheck.readTimeoutSeconds パラメタ〔TP1-Bridge〕 685
dependencyCheck.retryCount パラメタ〔Ext-Cons〕 706

dependencyCheck.retryCount パラメタ〔Filebeat〕 710
dependencyCheck.retryCount パラメタ〔Jaeger-collector〕 717
dependencyCheck.retryCount パラメタ〔Jaeger-query〕 713
dependencyCheck.retryCount パラメタ〔Logstash〕 734
dependencyCheck.retryCount パラメタ〔Mediator〕 674
dependencyCheck.retryCount パラメタ〔Metricbeat〕 726
dependencyCheck.retryCount パラメタ〔Prometheus〕 723
dependencyCheck.retryCount パラメタ〔TP1-Bridge〕 684
dependencyCheck.retryIntervalSeconds パラメタ〔Ext-Cons〕 707
dependencyCheck.retryIntervalSeconds パラメタ〔Filebeat〕 710
dependencyCheck.retryIntervalSeconds パラメタ〔Jaeger-collector〕 717
dependencyCheck.retryIntervalSeconds パラメタ〔Jaeger-query〕 714
dependencyCheck.retryIntervalSeconds パラメタ〔Logstash〕 734
dependencyCheck.retryIntervalSeconds パラメタ〔Mediator〕 674
dependencyCheck.retryIntervalSeconds パラメタ〔Metricbeat〕 726
dependencyCheck.retryIntervalSeconds パラメタ〔Prometheus〕 723
dependencyCheck.retryIntervalSeconds パラメタ〔TP1-Bridge〕 684
dependencyCheck.targets パラメタ〔Ext-Cons〕 707
dependencyCheck.targets パラメタ〔Mediator〕 674
dependencyCheck.targets パラメタ〔TP1-Bridge〕 685
driver-class-name パラメタ〔Entity-Service/Entity-Module〕 657

E

eads.cache.disk.filenum パラメタ〔EADS〕 704
eads.cache.disk.filesize パラメタ〔EADS〕 704
eads.cache.eviction.keyCount パラメタ〔EADS〕 704
eads.cache.eviction.policy パラメタ〔EADS〕 704
eads.cache.key.maxsize パラメタ〔EADS〕 703
eads.cache.keyCount パラメタ〔EADS〕 703
eads.cache.type パラメタ〔EADS〕 704
eads.client.＜接続先 EADS サーバ＞.address パラメタ〔EADS〕 705
eads.client.＜接続先 EADS サーバ＞.port パラメタ〔EADS〕 705
eads.client.connect.servers パラメタ〔EADS〕 705
eads.client.connectionPool.poolsize パラメタ〔EADS〕 705
eads.client.logger.dir パラメタ〔EADS〕 705
eads.client.properties パラメタ〔TP1-Bridge〕 697
eads.failureDetector.connection.timeout パラメタ〔EADS〕 702
eads.failureDetector.heartbeat.timeout パラメタ〔EADS〕 702
eads.failureDetector.read.timeout パラメタ〔EADS〕 702
eads.java.external.heapsize パラメタ〔EADS〕 703
eads.java.heapsize パラメタ〔EADS〕 703
eads.java.nio.maxCachedBufferSize パラメタ〔EADS〕 702
eads.logger.message.console.enable パラメタ〔EADS〕 702
eads.node.＜ EADS サーバ ID ＞.address パラメタ〔EADS〕 704
eads.replication.external.heapsize パラメタ〔EADS〕 703
eads.replication.factor パラメタ〔EADS〕 703
eads.server.cache.maxExecuteThreads パラメタ〔EADS〕 702

eads.server.connection.keepAlive.timeout パラメタ〔EADS〕 702
eads.server.maxConnections パラメタ〔EADS〕 702
ejbserver.application.userlog.CJLogHandler.conf1.level パラメタ〔uCosminexus Application Server の usrconf.properties〕 741
ejbserver.application.userlog.Logger.jp.co.Hitachi.soft.hmppcto.converter.handlers パラメタ〔uCosminexus Application Server の usrconf.properties〕 741
ejbserver.application.userlog.Logger.jp.co.Hitachi.soft.hmppcto.converter.level パラメタ〔uCosminexus Application Server の usrconf.properties〕 742
ejbserver.application.userlog.loggers パラメタ〔uCosminexus Application Server の usrconf.properties〕 741
elasticUserSecretName パラメタ〔Elasticsearch〕 731
elasticUserSecretName パラメタ〔Jaeger-collector〕 718
elasticUserSecretName パラメタ〔Jaeger-query〕 714
elasticUserSecretName パラメタ〔Logstash〕 735
elasticUserSecretName パラメタ〔Metricbeat〕 726
execute()メソッド〔PctoPreparedStatement クラス〕 506
execute(java.lang.String sql, int autoGeneratedKeys)メソッド〔PctoStatement クラス〕 491
execute(java.lang.String sql)メソッド〔PctoStatement クラス〕 490
executeBatch()メソッド〔PctoStatement クラス〕 492
executeCancel(TccResponse<O> response)メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 643
executeConfirm(TccResponse<O> response)メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 642
executeQuery()メソッド〔PctoPreparedStatement クラス〕 507
executeQuery(java.lang.String sql)メソッド〔PctoStatement クラス〕 494
executeRecover()メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 643
executeTry(I input)メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 641
executeUpdate()メソッド〔PctoPreparedStatement クラス〕 508
executeUpdate(java.lang.String sql)メソッド〔PctoStatement クラス〕 495
externalConsensus.eads.cache パラメタ〔TP1-Bridge〕 690
externalConsensus.intervalInMillis パラメタ〔TP1-Bridge〕 690
external-consensus.eads.cache パラメタ〔SQL-Participant〕 659
external-consensus.eads.cache パラメタ〔TCC-Participant〕 670
external-consensus.eads.properties パラメタ〔SQL-Participant〕 659
external-consensus.eads.properties パラメタ〔TCC-Participant〕 670
external-consensus.interval-in-millis パラメタ〔SQL-Participant〕 658
external-consensus.interval-in-millis パラメタ〔TCC-Participant〕 670
external-consensus.type パラメタ〔SQL-Participant〕 664
external-consensus.type パラメタ〔TCC-Participant〕 670

F

filebeat.imagePullPolicy パラメタ〔TP1-Bridge〕 685
filebeat.image パラメタ〔TP1-Bridge〕 685
filebeat.livenessProbe.failureThreshold パラメタ〔TP1-Bridge〕 685
filebeat.livenessProbe.initialDelaySeconds パラメタ〔TP1-Bridge〕 685

filebeat.livenessProbe.periodSeconds パラメタ〔TP1-Bridge〕 685
filebeat.livenessProbe.timeoutSeconds パラメタ〔TP1-Bridge〕 685
filebeat.output.logstash.namespace パラメタ〔TP1-Bridge〕 685
filebeat.output.maxRetries パラメタ〔Filebeat〕 711
filebeat.queue.maxSize パラメタ〔Filebeat〕 711
filebeat.queue.readAhead パラメタ〔Filebeat〕 711
filebeat.readinessProbe.failureThreshold パラメタ〔TP1-Bridge〕 686
filebeat.readinessProbe.initialDelaySeconds パラメタ〔TP1-Bridge〕 685
filebeat.readinessProbe.periodSeconds パラメタ〔TP1-Bridge〕 685
filebeat.readinessProbe.timeoutSeconds パラメタ〔TP1-Bridge〕 686
free()メソッド〔PctoArray クラス〕 629
free()メソッド〔PctoSQLXML クラス〕 604

G

getArray()メソッド〔PctoArray クラス〕 630
getArray(int columnIndex)メソッド〔PctoResultSet クラス〕 559
getArray(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 533
getArray(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 559
getBigDecimal(int columnIndex)メソッド〔PctoResultSet クラス〕 560
getBigDecimal(int parameterIndex, int scale)メソッド〔PctoCallableStatement クラス〕 534
getBigDecimal(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 561
getBinaryStream()メソッド〔PctoBlob クラス〕 607
getBlob(int columnIndex)メソッド〔PctoResultSet クラス〕 562
getBlob(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 535
getBlob(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 563
getBoolean(int columnIndex)メソッド〔PctoResultSet クラス〕 564
getBoolean(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 565
getByte(int columnIndex)メソッド〔PctoResultSet クラス〕 566
getByte(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 567
getBytes(int columnIndex)メソッド〔PctoResultSet クラス〕 567
getBytes(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 536
getBytes(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 568
getBytes(long pos, int length)メソッド〔PctoBlob クラス〕 608
getCharacterStream()メソッド〔PctoClob クラス〕 610
getClob(int columnIndex)メソッド〔PctoResultSet クラス〕 569
getClob(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 536
getClob(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 570
getColumnClassName(int column)メソッド〔PctoResultSetMetadata クラス〕 600
getColumnCount()メソッド〔PctoResultSetMetadata クラス〕 601
getColumnLabel(int column)メソッド〔PctoResultSetMetadata クラス〕 601
getColumnName(int column)メソッド〔PctoResultSetMetadata クラス〕 602
getColumnType(int column)メソッド〔PctoResultSetMetadata クラス〕 603
getConnection()メソッド〔PctoDataSource クラス〕 470
getConnection()メソッド〔PctoStatement クラス〕 496
getData()メソッド〔TccResponse<T extends java.io.Serializable>クラス〕 640

getDatabaseProductName()メソッド〔PctoDatabaseMetaData クラス〕 598
getDate(int columnIndex)メソッド〔PctoResultSet クラス〕 571
getDate(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 537
getDate(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 572
getDouble(int columnIndex)メソッド〔PctoResultSet クラス〕 573
getDouble(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 538
getDouble(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 573
getFloat(int columnIndex)メソッド〔PctoResultSet クラス〕 574
getFloat(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 539
getFloat(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 575
getGeneratedKeys()メソッド〔PctoStatement クラス〕 496
getInt(int columnIndex)メソッド〔PctoResultSet クラス〕 576
getInt(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 540
getInt(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 577
getLong(int columnIndex)メソッド〔PctoResultSet クラス〕 578
getLong(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 541
getLong(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 578
getMetaData()メソッド〔PctoConnection クラス〕 477
getMetaData()メソッド〔PctoResultSet クラス〕 579
getMoreResults()メソッド〔PctoStatement クラス〕 497
getNString(int columnIndex)メソッド〔PctoResultSet クラス〕 580
getNString(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 542
getNString(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 581
getObject(int columnIndex, Class<T> type)メソッド〔PctoResultSet クラス〕 584
getObject(int columnIndex)メソッド〔PctoResultSet クラス〕 582
getObject(int parameterIndex, Class<T> type)メソッド〔PctoCallableStatement クラス〕 544
getObject(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 543
getObject(java.lang.String columnLabel, Class<T> type)メソッド〔PctoResultSet クラス〕 585
getObject(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 583
getOperationId()メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 644
getOrder()メソッド〔TxPropagationIscImperativePlugin クラス〕 459
getQueryTimeout()メソッド〔PctoStatement クラス〕 498
getResultSet()メソッド〔PctoStatement クラス〕 498
getShort(int columnIndex)メソッド〔PctoResultSet クラス〕 586
getShort(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 545
getShort(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 587
getSQLXML(int columnIndex)メソッド〔PctoResultSet クラス〕 587
getSQLXML(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 546
getSQLXML(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 588
getString()メソッド〔PctoSQLXML クラス〕 605
getString(int columnIndex)メソッド〔PctoResultSet クラス〕 589
getString(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 546

getString(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 590
getSubString(long pos, int length)メソッド〔PctoClob クラス〕 611
getTime(int columnIndex)メソッド〔PctoResultSet クラス〕 591
getTime(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 547
getTime(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 592
getTimestamp(int columnIndex)メソッド〔PctoResultSet クラス〕 593
getTimestamp(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 548
getTimestamp(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 593
getTransactionKey()メソッド〔TccResponse<T extends java.io.Serializable>クラス〕 640
getType()メソッド〔PctoResultSet クラス〕 594
getUpdateCount()メソッド〔PctoStatement クラス〕 499
getWarnings()メソッド〔PctoStatement クラス〕 500
grpc.keepalive.interval パラメタ〔Mediator〕680
grpc.keepalive. interval パラメタ〔Orchestrator〕655
grpc.keepalive. interval パラメタ〔SQL-Participant〕 666
grpc.keepalive. interval パラメタ〔TCC-Participant〕 672
grpc.keepalive.interval パラメタ〔TP1-Bridge〕695
grpc.keepalive.timeout パラメタ〔Mediator〕681
grpc.keepalive.timeout パラメタ〔Orchestrator〕655
grpc.keepalive.timeout パラメタ〔SQL-Participant〕 666
grpc.keepalive.timeout パラメタ〔TCC-Participant〕 672
grpc.keepalive.timeout パラメタ〔TP1-Bridge〕695
grpc.port パラメタ〔Mediator〕 680
grpc.port パラメタ〔TP1-Bridge〕 694
grpc.server.queueSize パラメタ〔Mediator〕 677
grpc.server.threadPoolSize パラメタ〔Mediator〕677
grpc.tls.client.client-auth.certificate パラメタ〔Orchestrator〕 654
grpc.tls.client.client-auth.certificate パラメタ〔SQL-Participant〕 665
grpc.tls.client.client-auth.certificate パラメタ〔TCC-Participant〕 671
grpc.tls.client.client-auth.key パラメタ〔Orchestrator〕 654
grpc.tls.client.client-auth.key パラメタ〔SQL-Participant〕 665
grpc.tls.client.client-auth.key パラメタ〔TCC-Participant〕 671
grpc.tls.client.server-auth.certificate パラメタ〔Orchestrator〕 654
grpc.tls.client.server-auth.certificate パラメタ〔SQL-Participant〕 665
grpc.tls.client.server-auth.certificate パラメタ〔TCC-Participant〕 671
grpc.tls.clientAuth.enabled パラメタ〔Mediator〕680
grpc.tls.clientAuth.enabled パラメタ〔TP1-Bridge〕 695
grpc.tls.secretName パラメタ〔Mediator〕 680
grpc.tls.secretName パラメタ〔TP1-Bridge〕 694
grpc.tls.server.client-auth.password パラメタ〔SQL-Participant〕 665
grpc.tls.server.client-auth.password パラメタ〔TCC-Participant〕 671
grpc.tls.server.client-auth.store パラメタ〔SQL-Participant〕 665
grpc.tls.server.client-auth.store パラメタ〔TCC-Participant〕 671
grpc.tls.server.server-auth.certificate パラメタ〔SQL-Participant〕 665
grpc.tls.server.server-auth.certificate パラメタ〔TCC-Participant〕 671

grpc.tls.server.server-auth.key パラメタ〔SQL-Participant〕 665
grpc.tls.server.server-auth.key パラメタ〔TCC-Participant〕 671
grpc.tls.session.cacheSize パラメタ〔Mediator〕 680
grpc.tls.session.cacheSize パラメタ〔TP1-Bridge〕 695
grpc.tls.session.cache-size パラメタ〔Orchestrator〕 655
grpc.tls.session.cache-size パラメタ〔SQL-Participant〕 666
grpc.tls.session.cache-size パラメタ〔TCC-Participant〕 672
grpc.tls.session.timeout パラメタ〔Mediator〕 680
grpc.tls.session.timeout パラメタ〔Orchestrator〕 655
grpc.tls.session.timeout パラメタ〔SQL-Participant〕 666
grpc.tls.session.timeout パラメタ〔TCC-Participant〕 672
grpc.tls.session.timeout パラメタ〔TP1-Bridge〕 695
grpc.useTls パラメタ〔Mediator〕 680
grpc.useTls パラメタ〔TP1-Bridge〕 694
grpc.use-tls パラメタ〔Orchestrator〕 654
grpc.use-tls パラメタ〔SQL-Participant〕 664
grpc.use-tls パラメタ〔TCC-Participant〕 670
grpc.watchtime パラメタ〔Entity-Service/Entity-Module〕 656
grpc.watchtime パラメタ〔Mediator〕 681
grpc.watchtime パラメタ〔Orchestrator〕 655
grpc.watchtime パラメタ〔SQL-Participant〕 666
grpc.watchtime パラメタ〔TCC-Participant〕 672
grpc.watchtime パラメタ〔TP1-Bridge〕 695

H

hmpPcto.namespaces パラメタ〔Filebeat〕 710
http.port パラメタ〔Elasticsearch〕 729
http.port パラメタ〔Ext-Cons〕 708
http.port パラメタ〔Jaeger-query〕 713
http.port パラメタ〔Mediator〕 679
http.port パラメタ〔Prometheus〕 720
http.port パラメタ〔TP1-Bridge〕 694

I

idle_in_transaction_session_timeout パラメタ〔PostgreSQL〕 736
imagePullPolicy パラメタ〔Elasticsearch〕 728
imagePullPolicy パラメタ〔Ext-Cons〕 706
imagePullPolicy パラメタ〔Filebeat〕 709
imagePullPolicy パラメタ〔Jaeger-collector〕 716
imagePullPolicy パラメタ〔Jaeger-query〕 712
imagePullPolicy パラメタ〔Logstash〕 732
imagePullPolicy パラメタ〔Mediator〕 673
imagePullPolicy パラメタ〔Metricbeat〕 724
imagePullPolicy パラメタ〔Prometheus〕 719
imagePullPolicy パラメタ〔TP1-Bridge〕 684
imagePullSecrets パラメタ〔Elasticsearch〕 727
imagePullSecrets パラメタ〔Ext-Cons〕 706
imagePullSecrets パラメタ〔Filebeat〕 709
imagePullSecrets パラメタ〔Jaeger-collector〕 715
imagePullSecrets パラメタ〔Jaeger-query〕 712
imagePullSecrets パラメタ〔Logstash〕 732
imagePullSecrets パラメタ〔Mediator〕 673
imagePullSecrets パラメタ〔Metricbeat〕 724
imagePullSecrets パラメタ〔Prometheus〕 719
imagePullSecrets パラメタ〔TP1-Bridge〕 684
image パラメタ〔Elasticsearch〕 727
image パラメタ〔Ext-Cons〕 706
image パラメタ〔Filebeat〕 709
image パラメタ〔Jaeger-collector〕 715
image パラメタ〔Jaeger-query〕 712
image パラメタ〔Logstash〕 732
image パラメタ〔Mediator〕 673
image パラメタ〔Metricbeat〕 724
image パラメタ〔Prometheus〕 719
image パラメタ〔TP1-Bridge〕 684

init.mode パラメタ〔Entity-Service/Entity-Module〕 657
instance-id パラメタ〔Orchestrator〕 652
intercept(org.springframework.http.HttpRequest request, byte[] body, org.springframework.http.client.ClientHttpRequestExecution execution)メソッド〔TxPropagationRestTemplateInterceptor クラス〕 462
isClosed()メソッド〔PctoResultSet クラス〕 595
isWrapperFor(Class<?> iface)メソッド〔PctoDataSource クラス〕 471

J

jaegerQuery.maxSpanAge パラメタ〔Jaeger-query〕 713
java.options パラメタ〔Ext-Cons〕 708
java.options パラメタ〔Mediator〕 682
java.options パラメタ〔TP1-Bridge〕 696

L

lb.list-timeout-in-seconds パラメタ〔Orchestrator〕 655
learner.consensus-type パラメタ〔Orchestrator〕 652
length()メソッド〔PctoBlob クラス〕 609
limits.cpu パラメタ〔Elasticsearch〕 730
limits.cpu パラメタ〔Filebeat〕 709
limits.cpu パラメタ〔Jaeger-collector〕 715
limits.cpu パラメタ〔Jaeger-query〕 712
limits.cpu パラメタ〔Logstash〕 732
limits.cpu パラメタ〔Metricbeat〕 724
limits.cpu パラメタ〔Prometheus〕 719
limits.memory パラメタ〔Elasticsearch〕 730
limits.memory パラメタ〔Filebeat〕 709
limits.memory パラメタ〔Jaeger-collector〕 715
limits.memory パラメタ〔Jaeger-query〕 712
limits.memory パラメタ〔Logstash〕 732
limits.memory パラメタ〔Metricbeat〕 724
limits.memory パラメタ〔Prometheus〕 719
livenessProbe.failureThreshold パラメタ〔Elasticsearch〕 728
livenessProbe.failureThreshold パラメタ〔Filebeat〕 709
livenessProbe.failureThreshold パラメタ〔Jaeger-collector〕 716
livenessProbe.failureThreshold パラメタ〔Jaeger-query〕 712
livenessProbe.failureThreshold パラメタ〔Logstash〕 733
livenessProbe.failureThreshold パラメタ〔Mediator〕 675
livenessProbe.failureThreshold パラメタ〔Metricbeat〕 725
livenessProbe.failureThreshold パラメタ〔Prometheus〕 719
livenessProbe.failureThreshold パラメタ〔TP1-Bridge〕 686
livenessProbe.healthCheck.connectTimeoutSeconds パラメタ〔Mediator〕 675
livenessProbe.healthCheck.connectTimeoutSeconds パラメタ〔TP1-Bridge〕 686
livenessProbe.healthCheck.readTimeoutSeconds パラメタ〔Mediator〕 675
livenessProbe.healthCheck.readTimeoutSeconds パラメタ〔TP1-Bridge〕 686
livenessProbe.initialDelaySeconds パラメタ〔Filebeat〕 709
livenessProbe.initialDelaySeconds パラメタ〔Elasticsearch〕 728
livenessProbe.initialDelaySeconds パラメタ〔Jaeger-collector〕 716
livenessProbe.initialDelaySeconds パラメタ〔Jaeger-query〕 712
livenessProbe.initialDelaySeconds パラメタ〔Logstash〕 733
livenessProbe.initialDelaySeconds パラメタ〔Mediator〕 675
livenessProbe.initialDelaySeconds パラメタ〔Metricbeat〕 724
livenessProbe.initialDelaySeconds パラメタ〔Prometheus〕 719

livenessProbe.initialDelaySeconds パラメタ〔TP1-Bridge〕 686
livenessProbe.periodSeconds パラメタ〔Elasticsearch〕 728
livenessProbe.periodSeconds パラメタ〔Filebeat〕 709
livenessProbe.periodSeconds パラメタ〔Jaeger-collector〕 716
livenessProbe.periodSeconds パラメタ〔Jaeger-query〕 712
livenessProbe.periodSeconds パラメタ〔Logstash〕 733
livenessProbe.periodSeconds パラメタ〔Mediator〕 675
livenessProbe.periodSeconds パラメタ〔Metricbeat〕 725
livenessProbe.periodSeconds パラメタ〔Prometheus〕 719
livenessProbe.periodSeconds パラメタ〔TP1-Bridge〕 686
livenessProbe.successThreshold パラメタ〔Elasticsearch〕 728
livenessProbe.successThreshold パラメタ〔Filebeat〕 709
livenessProbe.successThreshold パラメタ〔Jaeger-collector〕 716
livenessProbe. successThreshold パラメタ〔Jaeger-query〕 713
livenessProbe.successThreshold パラメタ〔Logstash〕 733
livenessProbe.successThreshold パラメタ〔Metricbeat〕 725
livenessProbe.successThreshold パラメタ〔Prometheus〕 719
livenessProbe.timeoutSeconds パラメタ〔Elasticsearch〕 728
livenessProbe.timeoutSeconds パラメタ〔Filebeat〕 709
livenessProbe.timeoutSeconds パラメタ〔Jaeger-collector〕 716
livenessProbe.timeoutSeconds パラメタ〔Jaeger-query〕 712
livenessProbe.timeoutSeconds パラメタ〔Logstash〕 733
livenessProbe.timeoutSeconds パラメタ〔Mediator〕 675
livenessProbe.timeoutSeconds パラメタ〔Metricbeat〕 725
livenessProbe.timeoutSeconds パラメタ〔Prometheus〕 719
livenessProbe.timeoutSeconds パラメタ〔TP1-Bridge〕 686
logstash.pipeline.batchSize パラメタ〔Logstash〕 732
logstash.queue.byte.enabled パラメタ〔Logstash〕 733
logstash.queue.byte.maxBytes パラメタ〔Logstash〕 733
logstash.queue.event.enabled パラメタ〔Logstash〕 732
logstash.queue.event.maxEvents パラメタ〔Logstash〕 732

M

mark(int readAheadLimit)メソッド〔PctoReader クラス〕 622
mark(int readlimit)メソッド〔PctoInputStream クラス〕 614
markSupported()メソッド〔PctoInputStream クラス〕 615
markSupported()メソッド〔PctoReader クラス〕 622
max_connections パラメタ〔PostgreSQL〕 736
max_prepared_transactions パラメタ〔PostgreSQL〕 736
mediator.discovery-interval-in-millis パラメタ〔Orchestrator〕 653
mediator.discovery-retry-count パラメタ〔Orchestrator〕 653
mediator.judge.proposalLogLimitNum パラメタ〔Mediator〕 681
mediator.judge.queueSize パラメタ〔Mediator〕 677

mediator.judge.threadPoolSize パラメタ〔Mediator〕 677
mediator.minimum-start-size パラメタ〔Orchestrator〕 653
mediator.nameSuffix パラメタ〔TP1-Bridge〕 696
mediator.name パラメタ〔Orchestrator〕 652
mediator.name パラメタ〔SQL-Participant〕 666
mediator.name パラメタ〔TCC-Participant〕 670
mediator.name パラメタ〔TP1-Bridge〕 696
mediator.name-suffix パラメタ〔Orchestrator〕 652
mediator.name-suffix パラメタ〔SQL-Participant〕 666
mediator.name-suffix パラメタ〔TCC-Participant〕 670
mediator.port パラメタ〔Orchestrator〕 652
mediator.port パラメタ〔SQL-Participant〕 666
mediator.port パラメタ〔TCC-Participant〕 670
mediator.port パラメタ〔TP1-Bridge〕 696
mediator.size パラメタ〔Orchestrator〕 653
mediator.termination-thread-pool-size パラメタ〔Orchestrator〕 653
mediatorCommunication.queueSize パラメタ〔TP1-Bridge〕 694
mediatorCommunication.threadPoolSize パラメタ〔TP1-Bridge〕 694
mediatorProcessManagement.threadPoolSize パラメタ〔TP1-Bridge〕 694
mediator-communication.queue-size パラメタ〔SQL-Participant〕 664
mediator-communication.queue-size パラメタ〔TCC-Participant〕 669
mediator-communication.thread-pool-size パラメタ〔SQL-Participant〕 664
mediator-communication.thread-pool-size パラメタ〔TCC-Participant〕 669
mediator-process-management.thread-pool-size パラメタ〔SQL-Participant〕 664
mediator-process-management.thread-pool-size パラメタ〔TCC-Participant〕 669
metricbeat.output.maxRetries パラメタ〔Metricbeat〕 725
metricbeat.queue.maxSize パラメタ〔Metricbeat〕 725
metricbeat.queue.readAhead パラメタ〔Metricbeat〕 725

N

namespace パラメタ〔Elasticsearch〕 727
namespace パラメタ〔Ext-Cons〕 706
namespace パラメタ〔Filebeat〕 709
namespace パラメタ〔Jaeger-collector〕 715
namespace パラメタ〔Jaeger-query〕 712
namespace パラメタ〔Logstash〕 731
namespace パラメタ〔Mediator〕 673
namespace パラメタ〔Metricbeat〕 724
namespace パラメタ〔Prometheus〕 719
namespace パラメタ〔TP1-Bridge〕 683
next()メソッド〔PctoResultSet クラス〕 596
nodes.hostnames パラメタ〔Filebeat〕 710

P

participant.sender.queueSize パラメタ〔Mediator〕 676
participant.sender.threadPoolSize パラメタ〔Mediator〕 676
password パラメタ〔Entity-Service/Entity-Module〕 657
pd_max_users オペランド〔HiRDB〕 737
port パラメタ〔SQL-Participant〕 658
port パラメタ〔TCC-Participant〕 667
postExecute(IscImperativeCommandExecutionPluginContext pluginContext, IscImperativeCommand command)メソッド〔TxPropagationIscImperativePlugin クラス〕 460
postStart.healthCheck.connectTimeoutSeconds パラメタ〔Mediator〕 675
postStart.healthCheck.connectTimeoutSeconds パラメタ〔TP1-Bridge〕 686
postStart.healthCheck.readTimeoutSeconds パラメタ〔Mediator〕 675
postStart.healthCheck.readTimeoutSeconds パラメタ〔TP1-Bridge〕 686

postStart.healthCheck.retryCount パラメタ〔Mediator〕 675
postStart.healthCheck.retryCount パラメタ〔TP1-Bridge〕 686
postStart.healthCheck.retryIntervalSeconds パラメタ〔Mediator〕 675
postStart.healthCheck.retryIntervalSeconds パラメタ〔TP1-Bridge〕 686
preExecute(IscImperativeCommandExecutionPluginContext pluginContext, IscImperativeCommand command)メソッド〔TxPropagationIscImperativePlugin クラス〕 460
prepareCall(java.lang.String sql, int resultSetType, int resultSetConcurrency)メソッド〔PctoConnection クラス〕 479
prepareCall(java.lang.String sql)メソッド〔PctoConnection クラス〕 478
prepareStatement(java.lang.String sql, int autoGeneratedKeys)メソッド〔PctoConnection クラス〕 481
prepareStatement(java.lang.String sql, int resultSetType, int resultSetConcurrency)メソッド〔PctoConnection クラス〕 483
prepareStatement(java.lang.String sql, java.lang.String[] columnNames)メソッド〔PctoConnection クラス〕 484
prepareStatement(java.lang.String sql)メソッド〔PctoConnection クラス〕 480
preStop.waitTransactions.connectTimeoutSeconds パラメタ〔Mediator〕 676
preStop.waitTransactions.connectTimeoutSeconds パラメタ〔TP1-Bridge〕 687
preStop.waitTransactions.readTimeoutSeconds パラメタ〔Mediator〕 676
preStop.waitTransactions.readTimeoutSeconds パラメタ〔TP1-Bridge〕 687
preStop.waitTransactions.retryCount パラメタ〔Mediator〕 675
preStop.waitTransactions.retryCount パラメタ〔TP1-Bridge〕 686
preStop.waitTransactions.retryIntervalSeconds パラメタ〔Mediator〕 676
preStop.waitTransactions.retryIntervalSeconds パラメタ〔TP1-Bridge〕 687
prometheus.remoteWrite.queue.capacity パラメタ〔Prometheus〕 722
prometheus.remoteWrite.queue.maxSamplesPerSend パラメタ〔Prometheus〕 722
prometheus.remoteWrite.queue.maxShards パラメタ〔Prometheus〕 722
prometheus.scrape.interval パラメタ〔Prometheus〕 722
prometheus.storage.retentionTime パラメタ〔Prometheus〕 722

R

rap_connection_assign_type オペランド〔OpenTP1 システムサービス定義の rap リスナーサービス定義〕 738
rap_listen_port オペランド〔OpenTP1 システムサービス定義の rap リスナーサービス定義〕 738
rap_parallel_server オペランド〔OpenTP1 システムサービス定義の rap リスナーサービス定義〕 738
rap_recovery_serve オペランド〔OpenTP1 システムサービス定義の rap リスナーサービス定義〕 738
read()メソッド〔PctoInputStream クラス〕 616
read()メソッド〔PctoReader クラス〕 623
read(byte[] b, int off, int len)メソッド〔PctoInputStream クラス〕 617
read(byte[] b)メソッド〔PctoInputStream クラス〕 616
read(char[] cbuf, int off, int len)メソッド〔PctoReader クラス〕 624
read(char[] cbuf)メソッド〔PctoReader クラス〕 624
read(CharBuffer target)メソッド〔PctoReader クラス〕 625
readinessProbe.failureThreshold パラメタ〔Elasticsearch〕 728
readinessProbe.failureThreshold パラメタ〔Filebeat〕 710
readinessProbe.failureThreshold パラメタ〔Jaeger-collector〕 716

readinessProbe.failureThreshold パラメタ〔Jaeger-query〕 713
readinessProbe.failureThreshold パラメタ〔Logstash〕 733
readinessProbe.failureThreshold パラメタ〔Metricbeat〕 725
readinessProbe.failureThreshold パラメタ〔Prometheus〕 720
readinessProbe.initialDelaySeconds パラメタ〔Elasticsearch〕 728
readinessProbe.initialDelaySeconds パラメタ〔Filebeat〕 710
readinessProbe.initialDelaySeconds パラメタ〔Jaeger-collector〕 716
readinessProbe.initialDelaySeconds パラメタ〔Jaeger-query〕 713
readinessProbe.initialDelaySeconds パラメタ〔Logstash〕 733
readinessProbe.initialDelaySeconds パラメタ〔Metricbeat〕 725
readinessProbe.initialDelaySeconds パラメタ〔Prometheus〕 719
readinessProbe.periodSeconds パラメタ〔Elasticsearch〕 728
readinessProbe.periodSeconds パラメタ〔Filebeat〕 710
readinessProbe.periodSeconds パラメタ〔Jaeger-collector〕 716
readinessProbe.periodSeconds パラメタ〔Jaeger-query〕 713
readinessProbe.periodSeconds パラメタ〔Logstash〕 733
readinessProbe.periodSeconds パラメタ〔Metricbeat〕 725
readinessProbe.periodSeconds パラメタ〔Prometheus〕 720
readinessProbe.successThreshold パラメタ〔Elasticsearch〕 728
readinessProbe.successThreshold パラメタ〔Filebeat〕 710
readinessProbe.successThreshold パラメタ〔Jaeger-collector〕 716
readinessProbe.successThreshold パラメタ〔Jaeger-query〕 713
readinessProbe.successThreshold パラメタ〔Logstash〕 733
readinessProbe.successThreshold パラメタ〔Metricbeat〕 725
readinessProbe.successThreshold パラメタ〔Prometheus〕 720
readinessProbe.timeoutSeconds パラメタ〔Elasticsearch〕 728
readinessProbe.timeoutSeconds パラメタ〔Filebeat〕 710
readinessProbe.timeoutSeconds パラメタ〔Jaeger-collector〕 716
readinessProbe.timeoutSeconds パラメタ〔Jaeger-query〕 713
readinessProbe.timeoutSeconds パラメタ〔Logstash〕 733
readinessProbe.timeoutSeconds パラメタ〔Metricbeat〕 725
readinessProbe.timeoutSeconds パラメタ〔Prometheus〕 720
ready()メソッド〔PctoReader クラス〕 626
registerOutParameter(int parameterIndex, int sqlType, int scale)メソッド〔PctoCallableStatement クラス〕 550
registerOutParameter(int parameterIndex, int sqlType, java.lang.String typeName)メソッド〔PctoCallableStatement クラス〕 552
registerOutParameter(int parameterIndex, int sqlType)メソッド〔PctoCallableStatement クラス〕 549
replicas パラメタ〔Elasticsearch〕 727
replicas パラメタ〔Jaeger-collector〕 715
replicas パラメタ〔Logstash〕 731
replicas パラメタ〔Mediator〕 673
replicas パラメタ〔Metricbeat〕 724
repo.storageClassName パラメタ〔Elasticsearch〕 730
repo.storageSize パラメタ〔Elasticsearch〕 730
requests.cpu パラメタ〔Elasticsearch〕 730
requests.cpu パラメタ〔Filebeat〕 709

requests.cpu パラメタ〔Jaeger-collector〕 715
requests.cpu パラメタ〔Jaeger-query〕 712
requests.cpu パラメタ〔Logstash〕 732
requests.cpu パラメタ〔Metricbeat〕 724
requests.cpu パラメタ〔Prometheus〕 719
requests.memory パラメタ〔Elasticsearch〕 730
requests.memory パラメタ〔Filebeat〕 709
requests.memory パラメタ〔Jaeger-collector〕 715
requests.memory パラメタ〔Jaeger-query〕 712
requests.memory パラメタ〔Logstash〕 732
requests.memory パラメタ〔Metricbeat〕 724
requests.memory パラメタ〔Prometheus〕 719
reset()メソッド〔PctoInputStream クラス〕 619
reset()メソッド〔PctoReader クラス〕 627
resources パラメタ〔Mediator〕 673
resources パラメタ〔TP1-Bridge〕 684
role パラメタ〔Elasticsearch〕 728
rpc_max_message_size オペランド〔OpenTP1 システムサービス定義のシステム共通定義〕 739

S

scrapeNamespace.entity パラメタ〔Prometheus〕 720
scrapeNamespace. jaegercollector パラメタ〔Prometheus〕 721
scrapeNamespace.mediator パラメタ〔Prometheus〕 721
scrapeNamespace.orchestrator パラメタ〔Prometheus〕 720
scrapeNamespace.sqlparticipant パラメタ〔Prometheus〕 720
scrapeNamespace.tccparticipant パラメタ〔Prometheus〕 720
scrapeNamespace.tp1bridge パラメタ〔Prometheus〕 720
scrapePort.entity パラメタ〔Prometheus〕 722
scrapePort.mediator パラメタ〔Prometheus〕 722
scrapePort.orchestrator パラメタ〔Prometheus〕 721
scrapePort.sqlparticipant パラメタ〔Prometheus〕 721
scrapePort.tccparticipant パラメタ〔Prometheus〕 721
scrapePort.tp1bridge パラメタ〔Prometheus〕 721
server.tomcat.maxSwallowSize パラメタ〔TP1-Bridge〕 687
server.tomcat.threads.max パラメタ〔TP1-Bridge〕 687
setArray(int parameterIndex,java.sql.Array x)メソッド〔PctoPreparedStatement クラス〕 509
setBigDecimal(int parameterIndex, java.math.BigDecimal x)メソッド〔PctoPreparedStatement クラス〕 510
setBinaryStream(int parameterIndex, java.io.InputStream x, int length)メソッド〔PctoPreparedStatement クラス〕 511
setBlob(int parameterIndex, java.io.InputStream inputStream)メソッド〔PctoPreparedStatement クラス〕 512
setBoolean(int parameterIndex, boolean x)メソッド〔PctoPreparedStatement クラス〕 513
setByte(int parameterIndex, byte x)メソッド〔PctoPreparedStatement クラス〕 514
setBytes(int parameterIndex, byte[] x)メソッド〔PctoPreparedStatement クラス〕 515
setCharacterStream(int parameterIndex, java.io.Reader reader, int length)メソッド〔PctoPreparedStatement クラス〕 516
setClob(int parameterIndex, java.io.Reader reader)メソッド〔PctoPreparedStatement クラス〕 517
setDate(int parameterIndex, java.sql.Date x)メソッド〔PctoPreparedStatement クラス〕 518
setDouble(int parameterIndex, double x)メソッド〔PctoPreparedStatement クラス〕 519
setEscapeProcessing(boolean enable)メソッド〔PctoStatement クラス〕 501
setFetchSize(int rows)メソッド〔PctoStatement クラス〕 502
setFloat(int parameterIndex, float x)メソッド〔PctoPreparedStatement クラス〕 520

setInt(int parameterIndex, int x)メソッド〔PctoPreparedStatement クラス〕 521
setLong(int parameterIndex, long x)メソッド〔PctoPreparedStatement クラス〕 521
setNString(int parameterIndex, java.lang.String value)メソッド〔PctoPreparedStatement クラス〕 522
setNull(int parameterIndex, int sqlType)メソッド〔PctoPreparedStatement クラス〕 523
setObject(int parameterIndex, java.lang.Object x, int targetSqlType)メソッド〔PctoPreparedStatement クラス〕 526
setObject(int parameterIndex, java.lang.Object x)メソッド〔PctoPreparedStatement クラス〕 524
setQueryTimeout(int seconds)メソッド〔PctoStatement クラス〕 503
setShort(int parameterIndex, short x)メソッド〔PctoPreparedStatement クラス〕 527
setString(int parameterIndex, java.lang.String x)メソッド〔PctoPreparedStatement クラス〕 528
setString(java.lang.String value)メソッド〔PctoSQLXML クラス〕 606
setTime(int parameterIndex, java.sql.Time x)メソッド〔PctoPreparedStatement クラス〕 529
setTimestamp(int parameterIndex, java.sql.Timestamp x)メソッド〔PctoPreparedStatement クラス〕 529
setTransactionIsolation(int level)メソッド〔PctoConnection クラス〕 485
skip(long n)メソッド〔PctoInputStream クラス〕 619
skip(long n)メソッド〔PctoReader クラス〕 628
sql.app.port パラメタ〔SQL-Participant〕 659
sql.datasource.class-name パラメタ〔SQL-Participant〕 660
sql.datasource.property.delimiter パラメタ〔SQL-Participant〕 661
sql.datasource.property.values パラメタ〔SQL-Participant〕 660
sql.db-type パラメタ〔SQL-Participant〕 664
sql.jdbc.driver.path パラメタ〔SQL-Participant〕 660
sql.mediator.port パラメタ〔SQL-Participant〕 660
sql.participant-processing.queue-size パラメタ〔SQL-Participant〕 663
sql.participant-processing.thread-pool-size パラメタ〔SQL-Participant〕 663
sql.pod.service-name パラメタ〔SQL-Participant〕 659
sql.pool.allocation-wait-time-in-millis パラメタ〔SQL-Participant〕 659
sql.pool.max-pool-size パラメタ〔SQL-Participant〕 661
sql.pool.transaction-timeout-in-sec パラメタ〔SQL-Participant〕 662
sql.transaction-termination.consensus.replay-interval-in-millis パラメタ〔SQL-Participant〕 663
sql.transaction-termination.timeout-in-millis パラメタ〔SQL-Participant〕 662
sql.xa-server-name パラメタ〔SQL-Participant〕 659
supportsMultipleResultSets()メソッド〔PctoDatabaseMetaData クラス〕 599
sysctlVmMaxMapCount パラメタ〔Elasticsearch〕 730

T

tcc.consensus.interval-in-millis パラメタ〔TCC-Participant〕 668
tcc.mediator.port パラメタ〔TCC-Participant〕 667
tcc.participant-processing.queue-size パラメタ〔TCC-Participant〕 669
tcc.participant-processing.thread-pool-size パラメタ〔TCC-Participant〕 669
tcc.pod.service-name パラメタ〔TCC-Participant〕 667
tcc.recover.enabled パラメタ〔TCC-Participant〕 667
tcc.recover.order パラメタ〔TCC-Participant〕 667
tcc.trn-termination.timeout-in-millis パラメタ〔TCC-Participant〕 668

TccResponse(final T data)メソッド〔TccResponse<T extends java.io.Serializable>クラス〕 639
terminationGracePeriodSeconds パラメタ〔Elasticsearch〕 728
terminationGracePeriodSeconds パラメタ〔Filebeat〕 710
terminationGracePeriodSeconds パラメタ〔Jaeger-collector〕 716
terminationGracePeriodSeconds パラメタ〔Jaeger-query〕 713
terminationGracePeriodSeconds パラメタ〔Logstash〕 734
terminationGracePeriodSeconds パラメタ〔Mediator〕 676
terminationGracePeriodSeconds パラメタ〔Metricbeat〕 725
terminationGracePeriodSeconds パラメタ〔Prometheus〕 720
terminationGracePeriodSeconds パラメタ〔TP1-Bridge〕 687
timezone パラメタ〔Ext-Cons〕 708
timezone パラメタ〔Mediator〕 682
timezone パラメタ〔TP1-Bridge〕 696
tls.enabled パラメタ〔Elasticsearch〕 730
tls.enabled パラメタ〔Jaeger-collector〕 717
tls.enabled パラメタ〔Jaeger-query〕 714
tls.enabled パラメタ〔Logstash〕 734
tls.enabled パラメタ〔Metricbeat〕 726
tls.secretName パラメタ〔Elasticsearch〕 730
tls.secretName パラメタ〔Jaeger-collector〕 717
tls.secretName パラメタ〔Jaeger-query〕 714
tls.secretName パラメタ〔Logstash〕 734
tls.secretName パラメタ〔Metricbeat〕 726
tp1.cltj.dccltdelay パラメタ〔TP1-Bridge〕 688
tp1.cltj.dccltrecvbuffsize パラメタ〔TP1-Bridge〕 689
tp1.cltj.dccltrpcmaxmsgsize パラメタ〔TP1-Bridge〕 689
tp1.cltj.dccltsendbuffsize パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcdatatracemaxsize パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcdatatracesize パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcdatatrace パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcerrtracesize パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcerrtrace パラメタ〔TP1-Bridge〕 689
tp1.cltj.dchost パラメタ〔TP1-Bridge〕 688
tp1.cltj.dcmethodtracesize パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcmethodtrace パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcrapport パラメタ〔TP1-Bridge〕 688
tp1.cltj.dcuaptracesize パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcuaptrace パラメタ〔TP1-Bridge〕 689
tp1.cltj.dcwatchtiminherit パラメタ〔TP1-Bridge〕 688
tp1.cltj.dcwatchtimrpcinherit パラメタ〔TP1-Bridge〕 688
tp1.cltj.dcwatchtim パラメタ〔TP1-Bridge〕 688
tp1.data-conversion.definition-file.dir パラメタ〔uCosminexus Application Server の usrconf.properties〕 740
tp1.mediator.port パラメタ〔TP1-Bridge〕 691
tp1.participantProcessing.queueSize パラメタ〔TP1-Bridge〕 693
tp1.participantProcessing.threadPoolSize パラメタ〔TP1-Bridge〕 693
tp1.pod.serviceName パラメタ〔TP1-Bridge〕 691
tp1.pool.allocationWaitTimeInMillis パラメタ〔TP1-Bridge〕 690
tp1.pool.maxPoolSize パラメタ〔TP1-Bridge〕 691
tp1.transactionTermination.consensus.replayIntervalInMillis パラメタ〔TP1-Bridge〕 692
tp1.transactionTermination.timeoutInMillis パラメタ〔TP1-Bridge〕 692
tp1.xaServerName パラメタ〔TP1-Bridge〕 690
tracing.batchSize パラメタ〔Mediator〕 682

tracing.batchSize パラメタ〔TP1-Bridge〕 696
tracing.batch-size パラメタ〔共通〕 651
tracing.delayTime パラメタ〔Mediator〕 682
tracing.delayTime パラメタ〔TP1-Bridge〕 696
tracing.delay-time パラメタ〔共通〕 651
tracing.instanceId パラメタ〔Mediator〕 682
tracing.instanceId パラメタ〔TP1-Bridge〕 696
tracing.instance-id パラメタ〔共通〕 649
tracing.queueSize パラメタ〔Mediator〕 682
tracing.queueSize パラメタ〔TP1-Bridge〕 696
tracing.queue-size パラメタ〔共通〕 650
tracing.traceUse パラメタ〔Mediator〕 682
tracing.traceUse パラメタ〔TP1-Bridge〕 696
tracing.trace-use パラメタ〔共通〕 650
tracing.url パラメタ〔Mediator〕 682
tracing.url パラメタ〔TP1-Bridge〕 696
tracing.url パラメタ〔共通〕 649
trn_xar_use オペランド〔OpenTP1 システムサービス定義のトランザクションサービス定義〕 739
type パラメタ〔Entity-Service/Entity-Module〕 657

U

ucars.enabled パラメタ〔Mediator〕 682
ucars.snapshots.claimName パラメタ〔Mediator〕 683
ucars.snapshots.claimName パラメタ〔TP1-Bridge〕 697
url パラメタ〔Entity-Service/Entity-Module〕 657
user connections オプション〔SQL Server〕 737
username パラメタ〔Entity-Service/Entity-Module〕 657

W

wasNull()メソッド〔PctoCallableStatement クラス〕 553
wasNull()メソッド〔PctoResultSet クラス〕 597
webserver.connector.nio_http.keep_alive.max_requests パラメタ〔uCosminexus Application Server の usrconf.properties〕 741
webserver.connector.nio_http.limit.max_request_body パラメタ〔uCosminexus Application Server の usrconf.properties〕 741
webserver.connector.nio_http.max_servlet_execute_threads パラメタ〔ucosminexusapplicationserver の usrconf.properties〕 741
webserver.connector.nio_http.max_threads パラメタ〔uCosminexus Application Server の usrconf.properties〕 740
webserver.connector.nio_http.port パラメタ〔uCosminexus Application Server の usrconf.properties〕 741

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